MODELING AND CONTROL STUDIES FOR A REACTIVE BATCH DISTILLATION COLUMN

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

BY

ALMILA BAHAR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
CHEMICAL ENGINEERING

MAY 2007

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 Doctor of Philosophy.

Prof. Dr. Nurcan Baç
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 Doctor of Philosophy.

Prof. Dr. Canan Özgen
Supervisor

Examining Committee Members

Prof. Dr. Türker Gürkan (METU,CHE)

Prof. Dr. Canan Özgen (METU,CHE)

Prof. Dr. Kemal Leblebicioğlu (METU,EE)

Prof. Dr. Timur Doğu (METU,CHE)

Prof. Dr. Rıdvan Berber (AU,CHE)

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Almıla Bahar

Signature :

ABSTRACT

MODELING AND CONTROL STUDIES FOR A REACTIVE BATCH DISTILLATION COLUMN

Bahar, Almıla

Ph.D., Department of Chemical Engineering

Supervisor: Prof. Dr. Canan Özgen

May 2007, 162 pages

Modeling and inferential control studies are carried out on a reactive batch distillation system for the esterification reaction of ethanol with acetic acid to produce ethyl acetate. A dynamic model is developed based on a previous study done on a batch distillation column. The column is modified for a reactive system where Artificial Neural Network Estimator is used instead of Extended Kalman Filter for the estimation of compositions of polar compounds for control purposes.

The results of the developed dynamic model of the column is verified theoretically with the results of a similar study. Also, in order to check the model experimentally, a lab scale column (40 cm height, 5 cm inner diameter with 8 trays) is used and it is found that experimental data is not in good agreement with the models'. Therefore, the model developed is improved by using different rate expressions and thermodynamic models ($\phi-\phi$, combination of equations of state (EOS) and excess Gibbs free energy (EOS-G^{ex}), $\gamma-\phi$) with different equations of states (Peng Robinson (PR) / Peng Robinson - Stryjek-Vera (PRSV)), mixing rules (van der Waals / Huron Vidal (HV) / Huron Vidal Original (HVO) / Orbey Sandler Modification of HVO (HVOS)) and activity coefficient

models (NRTL / Wilson / UNIQUAC). The $\gamma - \phi$ method with PR-EOS together with van der Waals mixing rule and NRTL activity coefficient model is selected as the best relationships which fits the experimental data. The thermodynamic models; EOS, mixing rules and activity coefficient models, all are found to have very crucial roles in modeling studies.

A nonlinear optimization problem is also carried out to find the optimal operation of the distillation column for an optimal reflux ratio profile where the maximization of the capacity factor is selected as the objective function.

In control studies, to operate the distillation system with the optimal reflux ratio profile, a control system is designed with an Artificial Neural Network (ANN) Estimator which is used to predict the product composition values of the system from temperature measurements. The network used is an Elman network with two hidden layers. The performance of the designed network is tested first in open-loop and then in closed-loop in a feedback inferential control algorithm. It is found that, the control of the product compositions with the help of an ANN estimator with error refinement can be done considering optimal reflux ratio profile.

Keywords: Reactive Distillation, Batch Column, Mathematical Modeling, State Estimation, Artificial Neural Networks, Optimization

ÖZ

BİR TEPKİMELİ KESİKLİ DAMITMA KOLONUNDA MODELLEME VE DENETİM ÇALIŞMALARI

Bahar, Almıla
Doktora, Kimya Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Canan Özgen

Mayıs 2007, 162 sayfa

Etanolün asetik asit ile etil asetat üretimi esterleşme reaksiyonu için bir reaktif damıtma sisteminde modelleme ve algısal denetim çalışmaları yapılmıştır. Kesikli bir damıtma kolonunda daha önce yapılmış bir çalışmaya dayanan dinamik bir model geliştirilmiştir. Kolon, denetim çalışmaları için, polar maddelerin derişimlerini tahmin etmede Kalman Filtre yerine Sinir Ağları Tahmin Edicisi kullanan tepkimeli bir sistem için değiştirilmiştir.

Kolonun geliştirilen dinamik modelinin sonuçlarının doğruluğu, benzer bir çalışmanın sonuçları ile teorik olarak kanıtlanmıştır. Ayrıca, modeli deneysel olarak test etmek için, laboratuar ölçekli bir kolon (40 cm yükseklikte, 5 cm iç çapında, 8 tepsili) kullanılmıştır ve deneysel verilerin model ile uyuşmadığı görülmüştür. Bu yüzden, model, farklı hız ifadeleri ve farklı durum denklemleri (Peng Robinson (PR) / Peng Robinson - Stryjek-Vera (PRSV)), karışma kuralları (van der Waals / Huron Vidal (HV) / Huron Vidal Orijinal (HVO) / HVO'nun Orbey Sandler Modifikasyonu (HVOS)) ve aktivite modelleri (NRTL / Wilson / UNIQUAC) kullanan $\phi-\phi$, durum denklemleri-Gibbs serbest enerji (EOS-G^{ex}), $\gamma-\phi$ gibi

farklı termodinamik modeller araştırılmıştır. PR durum denklemi, van der Waals karışma kuralı ve NRTL aktivite modeli kullanan $\gamma - \phi$ metodu, en iyi ilişki olarak seçilmiştir. Termodinamik modellerin; durum denklemi, karışma kuralları ve aktivite modellerinin hepsinin modelleme çalışmalarında çok kritik etkilerinin olduğu bulunmuştur.

Damıtma kolonunun optimum geri akış oranı profilinde çalışmasını sağlamak için, kapasite faktörünün maksimum olarak seçildiği amaç fonksiyonunu kullanan, doğrusal olmayan bir optimizasyon problemi çözülmüştür.

Denetim çalışmalarında, damıtma sistemini optimum geri akış oranı profile ile çalıştırmak için, sistemin ürün derişimlerini sıcaklık ölçümlerinden tahmin eden Yapay Sinir Ağı (YSA) Tahmin Edicisi kullanan bir denetim sistemi tasarlanmıştır. YSA, iki katmanlı bir Elman ağıdır. Tasarlanan tahmin edicinin performansı, önce açık devrede ve sonra geri beslemeli algısal denetim algoritmasında kapalı devrede test edilmiştir. Ürün derişimlerinin denetiminin, optimum geri akış oranı profili ile YSA tahmin edicisi kullanılarak (gerektiğinde hata düzeltmesi ile) yapılabildiği bulunmuştur.

Anahtar Kelimeler: Tepkimeli Damıtma, Kesikli Kolon, Matematiksel Modelleme, Durum Tahmini, Yapay Sinir Ağları, Optimizasyon.

To my beloved family,

ACKNOWLEDGEMENTS

When I look back over the years that have passed during my PhD, I would like to thank many people. First of all, I would like to express my deepest appreciation to my supervisor Prof. Dr. Canan Özgen for her guidance and suggestions, for her time and effort taking for me, and especially for her lovely and kindly attitude not only relating with my thesis but in every aspect along the way of my graduate study. Without her encouragement and motivation, it would have been difficult to come to the end of this thesis.

I would like to extend my thanks to Prof. Dr. Kemal Leblebicioğlu and Prof. Dr. Timur Doğu for their valuable comments and discussions during the progress of this study. I would like to thank also to my thesis examining committee members Prof. Dr. Türker Gürkan and Prof. Dr. Rıdvan Berber.

I would like to thank especially to my colleague, Uğur Yıldız, for his assistance to this work and for being available, whenever I need help, suggesting his best solutions to my problems. Thanks also to other control team members; Salih Obut, Hatice Ceylan, Oluş Özbek and Özgecan Dervişoğlu for their helpful and enjoyable friendships throughout this study. I would like to thank to my friends Dilek Varışlı and Zeynep Obalı for helping me when I need something especially in the experimental part of the study and thanks also to Değer Şen, Ceren Oktar Doğanay, Eda Çelik, Sezin İslamoğlu, Işıl Işık and to my friends that I could not write them all down here, for their friendships all the way along this study.

I am also very grateful to my parents for their love, trust and sacrifices for me. And lastly my special thanks is for İsmail Doğan, for being beside me with his endless love, support, encouragement and belief in me throughout this long, self-sacrificing but somewhat enjoyable period of my life.

TABLE OF CONTENTS

ABSTRACT iv

ÖZ vi

DEDICATION viii

ACKNOWLEDGEMENTS ix

TABLE OF CONTENTS x

LIST OF TABLES xiii

LIST OF FIGURES xiv

LIST OF SYMBOLS xvi

CHAPTER

1. INTRODUCTION 1

2. LITERATURE SURVEY 5

2.1 Modeling and Optimization Studies 5

2.2 State Estimation Studies 8

2.2.1 State Estimation Studies for Continuous Distillation Column 8

2.2.2 State Estimation Studies for Batch Distillation Columns 9

2.2.3 State Estimation Studies for Reactive Distillation Columns 10

2.3 Control of Reactive Distillation Columns 11

3. EXPERIMENTAL 15

3.1 Experimental Setup 15

3.2 Experimental Procedure 16

4. REACTIVE BATCH DISTILLATION OPERATION MODELING 19

4.1 Calculation of V_{NT+1} 22

4.2 Holdup Calculations 24

4.3 Algebraic Equations 24

4.4 Physical Parameters Calculation 25

4.5 Initial Conditions 26

4.6 Kinetic Rate Expressions 26
4.7 Vapor Liquid Equilibrium (VLE) Calculations 27
4.7.1 Model-I: Phase Equilibrium Calculation Using the VLE data from Literature 28
4.7.2 Model-II: Phase Equilibrium Calculation Using $\phi-\phi$ Approach 28
4.7.3 Model-III: Phase Equilibrium Calculation Using the Combination of EOS Models with Excess Free Energy Models (EOS-G^{ex} Approach) 35
4.7.4 Model-IV: Phase Equilibrium Calculation Using $\gamma-\phi$ Approach 38
4.8 Summary of the Modeling Chapter 38
5. OPERATION AND NONLINEAR OPTIMIZATION OF THE REACTIVE BATCH DISTILLATION COLUMN 41
5.1 Operational Characteristics of a Multi-Component Batch Distillation Column 41
5.2 Nonlinear Optimization of the Reactive Batch Distillation Column 43
6. INFERENTIAL CONTROL AND ARTIFICIAL NEURAL NETWORK STATE ESTIMATOR 46
6.1 Inferential Control 46
6.2 Observability Criteria and Selection of Measurements 48
6.3 Artificial Neural Networks 49
6.3.1 Historical Development 49
6.3.2 Features of Artificial Neural Networks 50
6.3.3 Biological Neurons 51
6.3.4 Artificial Neurons 53
6.3.5 Types of Artificial Neural Networks 54
6.3.6 ANN Architecture 61
6.3.7 Applications of Artificial Neural Networks 61
7. SIMULATION CODE AND ALGORITHM 62
7.1 Main Simulation Code 62
7.2 Thermodynamic Library Code 66
7.3 ANN Estimator Code 66
8. RESULTS AND DISCUSSION 68
8.1 Modeling Studies 68
8.1.1 Simulation Results 68
8.1.2 Experimental Results 69
8.1.3 Comparison of Experimental Data and Simulation Results 72
i) Model-I 72
ii) Model-II 74

iii) Model-III 74
iv) Model-IV 78
v) Summary of Thermodynamic Models 84
vi) VLE Data Check 84
8.2 Nonlinear Optimization 87
8.3 Artificial Neural Network State Estimator 88
8.3.1 Selection of Measurement Points 89
8.3.2 Range of Variables 89
8.3.3 ANN Architecture 90
8.3.4 Normalization 91
8.3.5 Estimator Performance 91
8.4 Control Studies with the Designed ANN Estimator 97
8.4.1 Control Studies with Actual Composition Values 97
8.4.2 Control Studies with Estimated Composition Values 98
8.4.3 Control Studies with Estimated Composition Values (With Error Refinement) 100
9. CONCLUSIONS 103
REFERENCES 105
APPENDICES 111
A PROPERTIES OF THE COMPONENTS 111
B CALIBRATION OF GAS CHROMATOGRAPH 112
B.1 Correction Factor Calculation for Acetic Acid 112
B.2 Correction Factor Calculation for Water 113
B.3 Correction Factor Calculation for Ethyl Acetate 114
C SOURCE CODE 115
C.1 Main Program Code 115
C.2 Thermodynamic Library MATLAB Interface Code 135
C.3 Thermodynamic Library FORTRAN dll Code 136
C.4 ANN Estimator Code 154

LIST OF TABLES

Table 3.1 Experimental Column Parameters and Operating Conditions 17

Table 3.2 The Method and Conditions for GC Analysis 18

Table 3.3 Calibration Results 18

Table 4.1 Reaction Constants (litre/gmol min) 27

Table 4.2 Vapor-Liquid Equilibrium Data 28

Table 4.3 Binary Interaction Parameters, k_{ij} (Burgos-Solarzano et. al., 2004) .. 30

Table 4.4 NRTL Model Parameters 32

Table 4.5 Wilson Model Parameters 33

Table 4.6 UNIQUAC Model Parameters 35

Table 4.7 PRSV EOS Parameters, κ_1 36

Table 4.8 Multi-Component Batch Distillation Model Equations 39

Table 5.1 Optimization Problem 45

Table 7.1 Overall Structure of the Simulation Code 63

Table 8.1 Column Parameters 69

Table 8.2 Summary of Thermodynamic Models 85

Table 8.3 Comparison of Thermodynamic Model with Experimental VLE Data ... 85

Table 8.4 Optimum Reflux Ratio Profile 88

Table 8.5 Reflux Ratio Values Used in ANN Training 90

Table A.1 Physical Parameters of the Components 111

Table A.2 Heat Capacity Constants of the Components[1] 111

LIST OF FIGURES

Figure 3.1. Experimental Reactive Batch Distillation Column 16

Figure 5.2. The Schematic of a Multi-Component Batch Distillation Column 42

Figure 6.1 The Structure of Inferential Control Configuration..... 47

Figure 6.2 Structure of a Biological Neuron..... 52

Figure 6.3 Basic Structure of an Artificial Neuron..... 53

Figure 6.4 Basic Structure of a Feed-Forward Network 55

Figure 6.5 Basic Structure of an Elman Network..... 57

Figure 7.1 Flow Diagram of the Main Program Algorithm 64

Figure 8.1. Distillate Compositions at Total Reflux (a) Results of Monroy-Loperena and Alvarez-Ramirez (2000) (b) Results of the Simulation in This Study 70

Figure 8.2. Distillate Compositions at R=0.95 (a) Results of Monroy-Loperena and Alvarez-Ramirez (2000) (b) Results of the Simulation in This Study 70

Figure 8.3. Experimental Results..... 71

Figure 8.4. Experimental and Simulation Results (with Model-I) for Distillate and Reboiler Compositions at Total Reflux..... 73

Figure 8.5. Experimental and Simulation (with Model-II) Results for Distillate and Reboiler Compositions at Total Reflux..... 75

Figure 8.6. Experimental and Simulation (with Model-III-A) Results for Distillate and Reboiler Compositions at Reflux Ratio of 5.72 after Total Reflux 76

Figure 8.7. Experimental and Simulation (with Model-III-B) Results for Distillate and Reboiler Compositions at Reflux Ratio of 5.72 after Total Reflux 77

Figure 8.8. Experimental and Simulation (with Model-III-C) Results for Distillate and Reboiler Compositions at Reflux Ratio of 5.72 after Total Reflux 79

Figure 8.9. Experimental and Simulation (with Model-III-D) Results for Distillate and Reboiler Compositions at Reflux Ratio of 5.72 after Total Reflux 80

Figure 8.10 Experimental and Simulation (with Model-III-E) Results for Distillate and Reboiler Compositions at Reflux Ratio of 5.72 after Total Reflux 81

Figure 8.11. Experimental and Simulation (with Model-IV-A) Results for Distillate and Reboiler Compositions at Reflux Ratio of 5.72 after Total Reflux 82

Figure 8.12. Experimental and Simulation (with Model-IV-B) Results for Distillate and Reboiler Compositions at Reflux Ratio of 5.72 after Total Reflux 83

Figure 8.13. Actual and Estimated Distillate Compositions with Total Reflux Followed by a Reflux Ratio of 0.7 92

Figure 8.14. Actual and Estimated Distillate Compositions with Total Reflux Followed by a Reflux Ratio of 0.9 93

Figure 8.15. Actual and Estimated Distillate Compositions with Optimal Reflux Profile 93

Figure 8.16. Actual and Estimated Distillate Compositions with Total Reflux Followed by a Reflux Ratio of 0.75 94

Figure 8.17. Actual and Estimated Distillate Compositions with Total Reflux Followed by a Reflux Ratio of 0.83 95

Figure 8.18. Actual and Estimated Distillate Compositions with Total Reflux Followed by a Reflux Ratio of 0.895..... 95

Figure 8.19. Actual and Estimated Distillate Compositions with 10% Increased Optimal Reflux Profile 96

Figure 8.20. Actual and Estimated Distillate Compositions with 5% Decreased Optimal Reflux Profile..... 96

Figure 8.21. Block Diagram of the Control Scheme 97

Figure 8.22. Distillate Compositions Change with respect to Time Using Actual Composition Values 98

Figure 8.23. Distillate Compositions with Estimated Composition Feedback 99

Figure 8.24. Distillate Compositions with Estimated Compositions Feedback with Two Error Refinement..... 100

Figure 8.25. Distillate Compositions with Estimated Compositions Feedback with an Error Tolerance Level for Error Refinement..... 102

LIST OF SYMBOLS

a	Heat capacity constant, Peng-Robinson parameter, Activation in ANN
A	Area of component i obtained from GC
b	Heat capacity constant, Peng-Robinson parameter
c	Heat capacity constant
c_i	Concentration of component i (mol/lt)
C_p	Heat capacity coefficient
d	Heat capacity constant
D	Distillate flow rate (mol/h)
e	Error between the output of the network and its target value
E	Activation energy (kJ/kmol)
f	Fugacity
G	NRTL model parameter
h	Liquid mixture enthalpy (J/h)
H	Vapor mixture enthalpy (J/mol)
k_1	Forward reaction rate constant (lt/mol min)
k_2	Backward reaction rate constant (lt/mol min)
k_{ij}	Binary interaction parameter
K	Equilibrium constant
l	UNIQUAC model parameter
L	Liquid molar flow rate (mol/h)
M	Molar liquid holdup (mol)
P	Pressure (Pa)
P_i	Amounts of products obtained in tank i
Q	Heat load (J/mol h)
q	Surface area parameter in UNIQUAC model
r	Reaction rate (lt/mol min), Volume parameter in UNIQUAC model
R	Reflux ratio (L/D), Gas constant (J/mol K), Reaction rate (mol/h)
R_p	L/V ratio in the column
T	Temperature (K)
t	Time (h)

t_F	Total batch time (h)
u	Inputs to ANN
u_{ij}	Average interaction energy for a species i-species j interaction
V	Vapor molar flow rate (mol/h)
x	Liquid mole fraction (mol/mol), State
y	Vapor mole fraction (mol/mol)
y_d	Desired value of the network output
w	Acentric factor, Weight in ANN
Z	Compressibility factor

Greek Letters

α	Peng-Robinson parameter, NRTL model parameter
β	Correction factor in the gas chromatography calibration
ε	Stoichiometric coefficient of the components
$\underline{G}_{\gamma}^{ex}$	Molar excess Gibbs free energy
ρ	Liquid density (kg/m^3)
ν	Volumetric holdup (m^3)
κ	Peng-Robinson parameter
κ_1	PRSV EOS parameter
η	Learning factor
ϕ	Fugacity coefficient, Segment/volume fraction in UNIQUAC model
θ	Area fraction in UNIQUAC model, Threshold in ANN
γ	Liquid activity coefficient
τ	NRTL model parameter, UNIQUAC model parameter
Λ	Wilson model parameter

Subscripts

i	i^{th} stage, i^{th} component
j	j^{th} component
c	Critical value
boil	Boiling value
t	Total

Superscripts:

avg	Average

Abbreviations:

AcAc	Acetic Acid
ANN	Artificial Neural Network
ART	Adaptive Resonance Theory

CAP	Capacity Factor
EOS	Equation of State
$Eff_{Murphree}$	Murphree tray efficiency
EtAc	Ethyl Acetate
EtOH	Ethanol
EKF	Extended Kalman Filter
ETBE	Ethyl Tertiary Butyl Ether
GA	Genetic Algorithm
GC	Gas Chromatography
H_2O	Water
HVO	Huron-Vidal Original
HVOS	Orbey-Sandler Modification of Huron-Vidal
IAE	Integral Absolute Error
IGM	Ideal gas mixture
ITAE	Integral Time Absolute Error
lr	Learning rate
MGS	Model Gain Scheduling
MPC	Model Predictive Control
Mw	Molecular weight (kg/mol)
NC	Number of components
NIR	Near-Infrared Spectroscopy
NN	Neural Network
NPI	Nonlinear PI
NRTL	Non-Random Two Liquid
NT	Number of trays
PP	Pattern-Based Predictor
PPC	Pattern-Based Predictive Control
PR	Peng-Robinson
PRSV	Peng-Robinson Stryjek and Vera
r.h.s.	Right hand side
l.h.s.	Left hand side
RBF	Radial Basis Function
SVD	Singular Value Decomposition
TAME	Tertiary Amyl Methyl Ether
UNIQUAC	Universal Quasichemical
VLE	Vapor Liquid Equilibrium

CHAPTER 1

INTRODUCTION

Reactive distillation operation is a combination of reaction and separation operations in a single unit. It has been somehow used in industry for many years, but interest in its research and application has increased significantly in the last decade (Al-Arfaj and Luyben, 2002d; Wang et al., 2003). The main advantage of using reactive distillation is the reduction of capital and operating costs due to elimination of equipment, as a result of the operation of the combination of the reaction and the separation phases in a single unit. Also, the overall reactant conversion increases with the constant recycling of reactants and removal of products. Reactive distillation also increases energy efficiency due to direct utilization of reaction heat, makes temperature control of reaction easy, reacts away azeotropes and simplifies separation. It is particularly effective for reversible reactions with low equilibrium constants (Sneesby et al., 1999; Al-Arfaj and Luyben, 2002c; Wang et al., 2003).

Modeling and control of reactive distillation is a challenging task because of its complex dynamics resulting from the integration of reaction and separation. Its behavior is highly nolinear due to changing sign and direction of the process gain. Control problems arise due to the complex interactions between vapor-liquid equilibrium (VLE), chemical kinetics, vapor-liquid mass transfer, and diffusion inside the particle catalyst (Sneesby et al., 1999; Bisowarno et al., 2004). Computer simulation is important for deciding the optimum operation of the column, the optimum feed location, the number of separation trays in case of continuous column, and the size of the catalyst packed sections in case of reactive distillation with a catalyst.

The reaction studied in this work is an esterification process where ethanol reacts with acetic acid to produce ethyl acetate and water. The working temperature of this endothermic, second order and reversible reaction is around 70^0C and atmospheric pressure is used (Bakker et al., 2001). In practice the equilibrium is often forced towards the ester by azeotropic water removal. This reaction system is one of the most frequently used system in reactive distillation studies due to the available reaction rate data. In this quaternary system, ethanol forms azeotrope with water, ethyl acetate forms azeotropes with water (8.2 wt% water, boiling point 70.4^0C) and with ethanol (30.8 wt% ethanol, boiling point 71.8^0C). A ternary azeotrope between ethyl acetate-water-ethanol is also formed (7.8 wt% water, 9.0 wt% ethanol, boiling point 70.3^0C) (Ullmann, 1996). The complexity of the VLE and reversibility of the reaction makes the system very complicated. Therefore, modeling and testing the model by simulation studies for this system is very challenging.

There are many studies in the literature dealing with this esterification reaction of ethanol and acetic acid. Most of these studies considered the numerical methods of solution (Suzuki et al., 1971; Komatsu and Holland, 1977; Chang and Seader, 1988; Bogacki et al., 1989; Simandl and Svrcek, 1991) and phase equilibrium (Bock et al., 1997; Okur and Bayramoğlu, 2001; Park et al., 2006). However, all these studies used simplified models in simulation. Some of them assumed ideal plates with constant molar holdup and some others neglected the tray hydrodynamics. Most importantly, all of these studies were carried out under steady state conditions.

The dynamic simulation of a reactive distillation column for the ethyl acetate system in the presence of a catalyst is studied first by Alejski and Duprat (1996). Tang et al. (2003) showed that the NRTL model parameters predict the vapor-liquid equilibrium data of this four component system very well. Both of these dynamic studies were focused on a continuous distillation column. Mujtaba and Macchietto (1997) developed an optimization algorithm and Monroy-Loperena and Alvarez-Ramirez (2000) developed an output-feedback control algorithm for the ethyl acetate system in a reactive batch distillation column. However, in all these studies, in the dynamic simulation, simplified models were used.

In reactive distillation columns, always a high conversion is expected with a satisfactory purity which obviously, depends on high performance closed-loop control of both conversion and purity (Tade and Tian, 2000). Unfortunately,

either the conversion or the purity cannot be economically and reliably measured on-line. The on-line measurement of compositions is a typical problem in the industry (Bahar et al. 2004, Kano et al. 2000, Baratti et al. 1995). In the product compositions control systems, on-line measurements of the product compositions can be possible with direct composition analyzers such as gas chromatographs and NIR (Near-Infra Red) analyzers. However, these composition analyzers may introduce high investment and maintenance costs. Furthermore, since composition analyzers introduce large time delays to the system, designing an effective feedback control system by the use of measurements by analyzers can in many cases bring stability problems. Thus, instead of composition, temperature control loops can be used in the industry aiming to set product compositions at their desired values. In distillation columns, the compositions are strong functions of temperatures. However, especially in multi-component distillation, the temperature control may not always be adequate for composition control, since the tray temperatures do not correspond exactly to the product compositions in the face of disturbances (Kano et al. 2000, Patke et al. 1982). Therefore, it is important to be able to infer compositions from secondary measurements like temperatures, flows, pressures, etc. An estimator that utilizes temperature measurements can be used for this purpose. Then, these estimations are used for control purposes. This scheme is called as the inferential control.

Most of the works relating with the state estimation in distillation columns are based on continuous distillation columns. Unlike continuous columns, very few studies deal with the state estimation of batch columns which is widely used in the production of fine chemicals. Monitoring and control of composition also play an essential role in these units. Batch distillation is more complex, and highly nonlinear system compared to continuous distillation. Also it is an intrinsically dynamic process which makes the state estimation a more challenging task. As stated by Mujtaba and Macchietto (1996) and Oisiovici and Cruz (2000), composition profiles and operating conditions may change over a wide range of values during the entire operation and the state estimators must be designed to deal with the time-varying nature of the batch columns. Furthermore, the batch distillation is an attractive choice in reactive distillation as given by Wajge and Reklaitis (1999), when the reaction is slow and a large resident time is required to attain high conversion and when the reaction is so fast that a significant reaction may occur before the continuous column reaches steady state.

The identification and control of complex systems with unknown and uncertain dynamics has become a topic of considerable importance in the last decades and several control strategies have been developed for this purpose. One popular strategy among them is the Artificial Neural Networks (ANN) method. An ANN can be viewed as a nonlinear empirical model that is especially useful in representing input-output data, making predictions in time, classifying data, and recognizing patterns from an engineering viewpoint (Himmelblau, 2000). The reasons for the use of ANNs is that, they

- have the ability to evolve good process models by learning from available input-output data,
- require little or no priori knowledge of the system,
- can solve complex, highly nonlinear problems that cannot satisfactorily be handled by some traditional methods.

In the applications of ANNs, two general approaches are used. In one ANNs are used as the process controller; where the network is trained to identify the inverse dynamics of the controlled process and then directly used to control the process. In the other approach, the developed ANN model of the process is used for some type of model-based-control such as model predictive control. The latter is the more commonly used application of neural networks for control of chemical processes.

The objective of this study is to develop a mathematical model for the esterification reaction of ethanol and acetic acid in a reactive batch distillation column using first principles model and then to find an optimal operation policy for the column and finally, to design a state estimator using ANN method that can estimate the product compositions from temperature measurements to be used in the column control algorithm.

The reactive distillation column that is used in experimental and simulation studies are given in Chapter 3. The studies relating with the modeling of the column are discussed in Chapter 4. The optimization algorithm and the ANN estimator are presented in Chapter 5 and Chapter 6, respectively. The simulation code used for modeling, optimization, and estimator design is explained in Chapter 7. Finally, the results and discussions of this study are given in Chapter 8.

CHAPTER 2

LITERATURE SURVEY

In this chapter, previous studies done on reactive batch distillation operation are given. The chapter is organized in three subsequent sections as; optimization and modeling studies of esterification reaction of ethanol with acetic acid in a distillation column, studies on state estimation for the continuous, batch and reactive distillation column and lastly previous work on control studies of reactive distillation columns.

2.1 Modeling and Optimization Studies

Esterification reaction of ethanol and acetic acid to produce ethyl acetate and water is the most frequently considered reactive system in the literature considering reactive distillation. The modeling studies for this system goes back to 1970s. First studies focus especially on numerical solution methods.

Suzuki et al. (1971) used modified Muller's method for the convergence of temperature and tridiagonal matrix algorithm for the solution of the linearized material balance equations in a continuous distillation column at steady state. In their simulation, they used only temperature dependent VLE constant and Antoine's correlation for vapor pressure calculations.

Another method for convergence, which is called multi $\theta - \eta$ method, is developed by Komatsu and Holland (1977) for the same esterification system in continuous distillation column. Their VLE constant, K, depends on both temperature and liquid composition. In simulation, they used a very simplified model for the distillation column.

Chang and Seader (1988) applied a robust homotopy-continuation method to solve simultaneous nonlinear equations in modeling of the reactive distillation column at steady state. They utilized Antoine's correlation for vapor pressure calculations and Margules activity coefficients for the phase equilibrium. They compared their results with that of Suzuki et al. (1971) using a different reaction rate expression. In their simulations, they obtained a low conversion and a low purity ethyl acetate in the distillate.

Bogacki et al. (1989) proposed the Adam-Moulton method for simulation of the continuous reactive distillation under steady and unsteady state conditions. They used the same phase equilibrium data as Komatsu and Holland (1977) with a temperature independent rate expression and compared the results with their experimental data. They proposed that the differences between the results they observed might be due to inadequate precision of the VLE and kinetic data or may be due to the model simplifications which neglect the column hydraulics, plate efficiencies and the heat balance.

Simandl and Svrcek (1991) compared the simultaneous solution method result with that of the inside-outside tearing method in continuous reactive distillation at steady state conditions. They used temperature dependent reaction rate and Wilson activity coefficient model for the phase equilibrium.

A dynamic simulation of a continuous reactive distillation column for the esterification of ethanol with acetic acid with a homogeneous catalyst of sulphuric acid is proposed by Alejski and Duprat (1996). They used the same phase equilibrium data of Komatsu and Holland (1977) and temperature dependent rate expression. Comparison of simulation results with the experimental results showed that the concentration results are not very accurate due to the large disturbances imposed, simplifications of the mathematical model, inaccuracy of the kinetic and vapor-liquid equilibrium description and due to the precision of experimental measurements.

Bock et al. (1997) analyzed the continuous reactive distillation column for the same esterification reaction under steady state conditions. They compared the phase equilibrium data of Suzuki (1971) and that obtained from NRTL with the experimental data of Komatsu and Holland (1977) and showed that the phase equilibrium data of Suzuki (1971) is not suitable.

The first study on modeling of a batch distillation column is done by Mujtaba and Macchietto (1997). They developed a mathematical model and a nonlinear optimization algorithm for maximum profit problem for the same reactive system in a *batch* distillation column. In modeling, they used steady state energy balances by assuming adiabatic plates and fast energy dynamics. Furthermore, they used the temperature independent rate expression and phase equilibrium data of Simandl and Svrcek (1991).

Another study that considered the batch reactive distillation column for the same esterification system is that of Monroy-Loperena and Alvarez-Ramirez (2000). They designed an output-feedback control for this system by using an approximate unsteady state model and a reduced-order observer to estimate the modeling error. In modeling, they used temperature independent rate expression and the phase equilibrium data of Mujtaba and Macchietto (1997).

Okur and Bayramoğlu (2001) compared the liquid activity coefficient models, UNIQUAC, UNIFAC, and Margules, on the simulation of continuous reactive distillation with esterification reaction of ethanol and acetic acid. They used steady state modeling and temperature dependent rate equation.

Giessler et al. (2001) solved the optimization problem for the reactive batch distillation column for different types of models and objective functions. They used the same esterification reaction of ethanol and acetic acid. In their optimization algorithm, the reflux ratio and the heat duty are selected as the optimization variables and they are assumed to be piecewise constant. They investigated the effect of the number of time periods, reaction on the trays, holdup dynamics, model preciseness, and the difference in the objective function.

Tang et al. (2003) established suitable NRTL model parameters for the calculation of liquid activity coefficients. The compositions and temperatures of the four azeotropes in the system were predicted well. Vapor association of the acetic acid due to dimerization has also been included in their study. They obtained high purity ethyl acetate product from the overall system which includes two continuous columns (one being reactive distillation column) one decanter, and two recycle streams. They also found the optimum operating condition of the overall system in order to minimize the total operating cost of the system.

Another steady state model for the production of ethyl acetate with an acid catalyst through a continuous reactive distillation process is developed by Park et al. (2006). They used NRTL model for the phase equilibrium calculations and obtained high purity ethyl acetate production after further purification of the top product using a common distillation column. The comparison of simulation results with that of experiments showed that the results are in good agreement.

On the other hand, relating to dynamic modeling of a reactive batch distillation operation, the first study belongs to Wajge and Reklaitis (1999). A maximum conversion problem for the optimum operation of the ethyl acetate production in a batch distillation column is provided in this study. However, the study does not consider the hydrodynamics on the trays and chemical reactions in the vapor phase.

2.2 State Estimation Studies

In this section, the previous studies on state estimation for continuous, batch, and reactive distillation columns are presented.

2.2.1 State Estimation Studies for Continuous Distillation Column

Most of the works related to the state estimation in distillation columns are based on continuous distillation columns. Starting in 1972, Weber and Brosilow presented a method for designing a static estimator which predicts product quality from a linear combination of process input and output measurements. Since then, many studies are done and given in the M.Sc. study of Bahar (2003).

Bahar et al. (2004) developed an inferential control methodology that utilized an ANN estimator in a model predictive controller algorithm for an industrial multi-component distillation column. Singular Value Decomposition (SVD) technique was used for the selection of the temperature measurement points and a moving window neural network estimator was used in order to incorporate the system dynamics into account. The performance of the estimator in the open-loop was found satisfactorily. Furthermore, the performance of the Model Predictive Controller (MPC) that used the state variables obtained from the estimator was also found to be satisfactory for set-point tracking and disturbance rejection problems.

2.2.2 State Estimation Studies for Batch Distillation Columns

Unlike continuous columns, very few studies were done on the state estimation of batch columns which is widely used in the production of fine chemicals. However, monitoring and control of composition of products play a very essential role in batch columns. Batch distillation is a very complex, nonlinear and a high-order system. It is also an intrinsically dynamic process which makes the state estimation really a challenging task. Composition profiles and operating conditions may change over a wide range of values during the entire operation, and the state estimators must be designed to deal with the time-varying nature of the batch columns (Mujtaba and Macchietto, 1996; Oisiovici and Cruz, 2000).

Quintero-Marmol and Luyben (1992) presented model-based inferential control for multi-component batch distillation systems. They studied the two different inferential model-based control schemes: the rigorous steady-state and the quasi-dynamic non-linear estimator to estimate the distillate compositions of a multi-component distillation column. It was seen that, both estimators provide good results using only one-temperature measurement. Some drawbacks of the steady-state model are also given in the paper such as; the thermocouple has to be located at the end of the column for good results and sometimes another thermocouple is needed to predict the end of the batch in the steady-state model.

Oisiovici and Cruz (2000) developed and tested a discrete nonlinear Extended Kalman Filter (EKF) estimator for binary and multi-component batch distillation columns in order to infer the instantaneous product compositions and the composition profile along the column from temperature measurements. They calculated and updated on-line the gain of the EKF. They also addressed the important issues such as the number of sensors, the presence of temperature noise and the sampling rate.

Fileti et al. (2000) presented a predictive control strategy based on a non-linear nonparametric dynamic system model, the dynamic neural network. A neural-network-based identifier provided n-step-ahead top composition predictions for the optimization of the control action. The basic process variables that are required to predict the top composition are the running top and bottom compositions and the reflux ratio. The network has an input moving window with dynamic characteristics. The performance of the control strategy was first tested

through rigorous simulations and then on a batch distillation column for a ternary system of n-hexane-benzene-toluene. It was observed that ANN offers very good predictions of the nonlinear process behavior.

Zamprogna et al. (2001) developed a virtual sensor based on a recurrent ANN for a middle-vessel batch distillation column to estimate the product compositions. It was shown that the estimated compositions are in good agreement with the actual values. The effects of sensor location, model initialization, and temperature measurement noise on the performance of the soft sensor were investigated.

Yıldız et al. (2005) designed an Extended Kalman Filter (EKF) state estimator to infer the product composition in a multi-component batch distillation column. The EKF parameters which are the diagonal elements of the process noise covariance matrix and those of measurement model noise covariance matrix are selected in the range where the estimator is stable and selection is based on the smallest IAE scores for the reflux-drum and the reboiler composition estimates. They found that, although NC-1 temperature measurements are sufficient, using NC (number of components) measurements improve the performance of the estimator, but increasing the number of temperature measurements further does not result in a better performance. They used the designed EKF estimator successfully in the composition-feedback inferential control of the column operated under variable reflux-ratio policy.

2.2.3 State Estimation Studies for Reactive Distillation Columns

Tade and Tian (2000) inferred the reactant conversion from multiple process temperatures in a 10-stage pilot plant ethyl tertiary butyl ether (ETBE) reactive distillation column. They developed a third-order, two-variable nonlinear inferential model by employing the regression method. The two temperatures they used in this model were the bottom reactive section temperature and the reboiler temperature taken from the simulation of the process.

Dadhe (2004) used nonlinear models, the neural networks and the support vector machines for the nonlinear calibration from the near-infrared spectroscopy (NIR) for the on-line estimation of methyl acetate mole fraction in a reactive distillation column. One NIR probe was at the top of the column directly underneath the condenser and another one was located at the reboiler.

For the calculation of prediction intervals, the bootstrap method known from the computational statistics was used. The support vector machine showed slightly better prediction than the neural network. Although nonlinear methods improved the prediction of methyl acetate mole fraction over the linear model, in all of these cases, the variance of the prediction interval was too large to consider the prediction as a reliable estimate. More calibration data can be gathered or an on-line adjustment with gas chromatographic measurements can be done to reduce the prediction error.

Venkateswarlu and Kumar (2006) designed an Extended Kalman Filter in order to estimate the compositions in a reactive batch distillation column for the esterification reaction of ethanol and acetic acid. They used a simulation based on data of Mujtaba and Macchietto (1997) using the same column specifications, vapor-liquid equilibrium and kinetic data.

2.3 Control of Reactive Distillation Columns

Although the dynamics and behavior of reactive distillation column (steady-state design, open-loop dynamics and multiplicity) have been investigated extensively, there are only few studies on the closed-loop control of reactive distillation columns. Therefore, the control of reactive distillation is still an open research area. The studies related to the control of reactive distillation columns are given below.

A dynamic simulation of the ETBE reactive distillation column was developed by Sneesby et al. (1997) using a dynamic process simulator and this model was used for determining the transient open-loop responses of the column and also used for control purposes. Control performance was tested by step increases in feed rate and feed composition, and for a set-point change. Controlled variable was selected as a temperature at the middle of the stripping section. Simple PI controllers were used. Dynamic simulations were used to evaluate several control configurations but the LV and LB configurations were recommended.

Sneesby et al. (1999) proposed a two-point control scheme which used simple, linear PI controllers to control both the product composition and the reactant conversion. They used dynamic simulations to test this control scheme and showed that the two-point control scheme is effective and better than one-point

control scheme especially for feed rate disturbances and set-point changes. For feed composition disturbances, there was a significant offset in the two-point control scheme, however, the ether purity deviated less in the two-point control scheme according to one-point control scheme.

Al-Arfaj and Luyben (2000) studied an ideal two-reactant two-product reactive distillation system. In a later study, Al-Arfaj and Luyben (2002d) dealed with the control of a methyl acetate (two-reactant and two-product) reactive distillation system and compared this chemical system with the ideal one. Afterwards, Al-Arfaj and Luyben (2002a) studied single-feed and double-feed designs of the ETBE column and its control schemes in which there were two reactants, one product and one inert. A multiple reaction case, the ethylene glycol system which has two feeds but only one product, was also studied by Al-Arfaj and Luyben (2002b). For this system, a simple PI control scheme in which a temperature in the stripping section was controlled by the heat input was found to be effective. The stoichiometric balancing of the reactants was achieved and the product purity was maintained within reasonable bounds. Al-Arfaj and Luyben (2002c) found effective the control scheme for olefin metathesis case, which have one-reactant and two-products, in which a temperature in the stripping section was controlled by the heat input and another temperature in the rectifying section was controlled by the reflux rate. In the study of Al-Arfaj and Luyben (2002d), a plant wide flow sheet consisting of one reactor, one reactive column, two conventional columns, and two recycles was developed for the production of tertiary amyl methyl ether (TAME). The control structure that was applied to this flow sheet was such that a temperature in the stripping section and a methanol composition in the reactive zone were controlled. The other two columns were controlled by simple temperature controllers since there was no reaction. The fresh feed of methanol was manipulated to maintain the overall methanol balance in the flow sheet.

Balasubramhanya and Doyle III (2000) developed a low order nonlinear model by using traveling wave phenomena. To test the proposed procedure, they used a simulated batch reactive distillation column consisting of eight trays for the production of ethyl acetate. They used the reduced model of the column in a MPC algorithm employing a nonlinear process model to control the temperature on the second tray. They used a Nonlinear Quadratic Dynamic Matrix Control with State Estimation approach. They used the reduced wave model to predict outputs into the future and used a nonlinear optimization routine to calculate the

input moves. They showed that the performance of the nonlinear MPC using the reduced order model was as good as that of the controller using the detailed column model, with an advantage of reduced computational effort.

Engell and Fernholz (2003) studied the conventional control structures and nonlinear MPC for the methyl acetate production in a semi-batch reactive distillation column. They used a neural net model of the process in the nonlinear predictive control scheme. A static network with external recurrence was used because of simpler training of the network. The radial-basis function (RBF) networks were used as one-step-ahead predictor of the process. The past and present process inputs and past process outputs were the inputs of the net and the prediction of the plant output at the next time step was the output of the net. The outputs of the net were externally fed back to the input for the generation of long-range predictions. A rigorous nonlinear model simulation was used to collect the training data. They tested the controller for set-point tracking and for disturbance rejection. In case of set-point changes, the nonlinear controller reduced the rise time significantly compared to the linear controller. For disturbance rejection case, the nonlinear controller gave better performance than the linear one.

Tian et al. (2003) developed a pattern-based predictive control (PPC) algorithm for the control of ETBE purity at the bottom of the reactive distillation column. The temperature of stage 7 was used as the indicator of the bottoms product purity by manipulating the reboiler heat duty. Although the reactive distillation process is highly non-linear, the degree of process non-linearity was reduced by fixing the reflux flow rate since this study considered one-point control (only the bottom product purity). Therefore, direct PI control also showed acceptable performance. However, in order to improve the performance, they incorporated a pattern-based predictor (PP) with a conventional PI controller. PI controller was tuned for set-point tracking for the ITAE index. The PPC system increased the performance significantly over the direct PI control system for both set-point tracking and disturbance rejection cases.

Bisowarno et al. (2003) developed a model gain scheduling control for one-point control of an ETBE reactive distillation column. They derived simplified input-output first-order models which identified relevant operating conditions which cope with nonlinear characteristics. A switching scheme was used for providing a smooth transition between the simple models. The performance of the proposed

control scheme outperforms that of the standard PI controller for both set-point tracking and disturbance rejection cases.

Bisowarno et al. (2004) investigated two adaptive PI control strategies, a non-linear PI (NPI) and a model gain-scheduling (MGS) for ETBE reactive distillation column. The LV configuration was used for the control of ETBE purity. The primary manipulated variable, the reboiler heat duty was used to control the temperature of stage 7. The second manipulated variable, the reflux rate was kept constant to achieve high isobutylene conversion. Both adaptive control systems were based on a PI controller integrated with a tuning method. For the NPI, the controller gain was allowed to vary in order to accommodate the directionality in the process gain. In case of both set-point tracking and disturbance rejection, it was shown that the performances of the NPI and MGS were better than a standard PI control with fixed parameters.

CHAPTER 3

EXPERIMENTAL

In order to check the results obtained from the simulation studies, experimental studies are carried out in a lab-scale batch distillation column. The experimental setup and the experimental procedure used for the reactive batch distillation column are given below.

3.1 Experimental Setup

The batch distillation column that is used in this work has an inner diameter of 5 cm, a height of 40 cm and has 8 sieve plates with a plate spacing of 5 cm. The feed tank, having a 20 L volume, is made from corrosion resistant stainless steel and incorporated with two electrically heated cartridge type heating elements. Power to the heaters can be continuously varied using a regulator and can directly be read from the wattmeter which is calibrated between 0-2 kW. A level sensor is situated on the top of the feed tank to prevent the heating elements from overheating in case of reboiler run dry. Thermocouples are located at the top and bottom of the column (T2 and T3, respectively), at the inlet and outlet of the cooling water used in condenser (T6 and T7, respectively), at the reflux line (T5), at the condensate (T4) and at the reboiler (T1) to measure the temperatures. However, thermocouples cannot be placed along the column to measure the temperatures of plates. Different reflux ratio values can be manually set by using two electronic timers, which proportion to the position of the reflux divider. The reboiler, the condenser, and the column are insulated to reduce any heat losses. The schematic representation of the column is given in Figure 3.1.

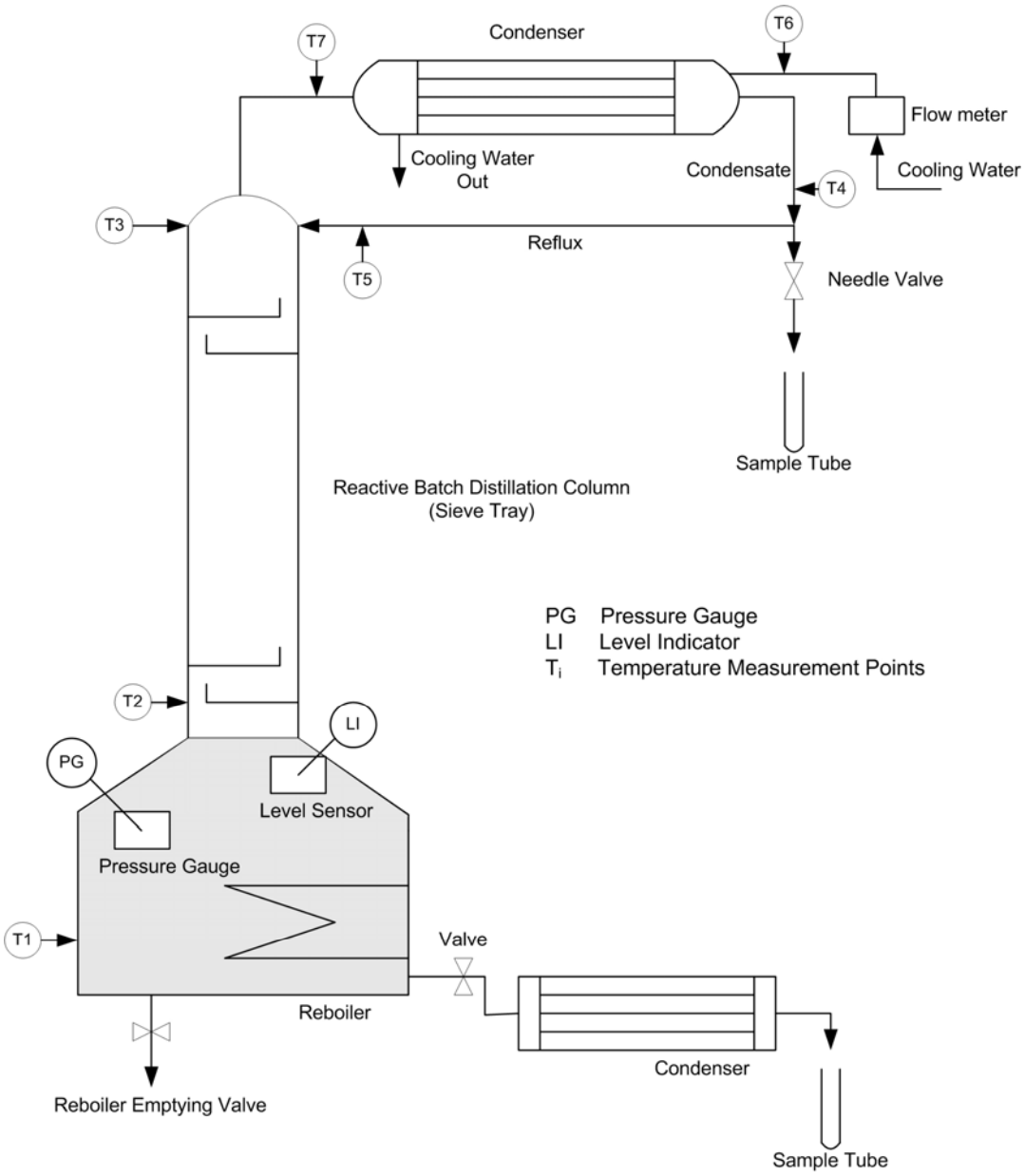


Figure 3.1. Experimental Reactive Batch Distillation Column

3.2 Experimental Procedure

In the experiments, the reboiler is first charged with an equimolar mixture of ethanol and acetic acid. Ethanol (Merck grade, ≥99.99% w/w purity) and acetic acid (Merck grade, ≥99.8% w/w) are used. The cooling water flow rate is adjusted to a constant value during the experiments. At the beginning of the experiments, the reflux ratio control is set for total reflux and heat is adjusted to its maximum value. After a certain time, the heater is adjusted to give a gentle bubbling on the trays. This steady state value of boilup rate and the reboiler

temperature at this value is approximately 0.56 kW and 90^0C, respectively. The overall column parameters and operating conditions are given in Table 3.1.

Table 3.1 Experimental Column Parameters and Operating Conditions

No. of stages (including reboiler and total condenser)	10
Total fresh feed, mol	311.67
Feed composition (ethyl acetate, ethanol, water, acetic acid), mole fraction	0.0, 0.5, 0.0, 0.5
Column holdup, mol	
condenser+drum	30
internal plates	0.779
Reboiler heat duty, J/h	2.016×10^6
Column pressure, bar	1.013
Cooling water flow rate, lt/min	1.0

In the experiments, the column is first operated at total reflux condition. At this period, samples are taken in every 30-minute intervals at the beginning and later in 60-minute intervals. After steady state is reached, reflux ratio is set to a predefined value and samples are continued to be taken from both the reflux drum and from the reboiler in every 60-minute intervals into small bottles. Analyses of the collected samples, which are secured in dry ice to stop the course of the reaction, are done through Varian CP-3800 Gas Chromatography (GC) with Porapak-T packed column and TCD detector. As the carrying gas, helium is used. The method and conditions of the GC are given in Table 3.2.

Calibration of the GC must be done prior to experimental evaluation of compositions of the components. Detailed calculations for the calibration of GC to find the compositions of the components are given in Appendix B. The areas for each component are obtained from GC and the mole fractions of the components are calculated by using Equation 3.1 where x_A represents the mole fraction of component A, A_i represents the area under the curve of component i

and β_i represents the correction factor for the i[th] component according to the base component. The results of the calibration, i.e., the correction factors are given in Table 3.3.

$$x_A = \frac{A_A \beta_A}{A_A \beta_A + A_B \beta_B + A_C \beta_C + A_D \beta_D} \tag{3.1}$$

Table 3.2 The Method and Conditions for GC Analysis

(a)

Temperature (^{0}C)	Rate (^{0}C/min)	Hold (min)	Total (min)
75		0.1	0.1
175	40	10	12.60

(b)

Detector Temperature	225 ^{0}C
Injector Temperature	200 ^{0}C
Carrier Gas	Helium

Table 3.3 Calibration Results

Component, i	Correction Factor, β_i
Ethyl Acetate	0.658
Ethanol	1.0
Water	2.259
Acetic Acid	0.93

CHAPTER 4

REACTIVE BATCH DISTILLATION OPERATION MODELING

The unsteady state model developed in this study for the reactive batch distillation operation is based on the modeling study done by Yıldız et al. (2005). The assumptions that are used in the modeling studies are negligible vapor holdup, constant volume of tray liquid holdup, constant liquid molar holdup in the reflux-drum, total condenser, negligible fluid dynamic lags, linear pressure drop profile, Murphree tray efficiency, approximated enthalpy derivatives and adiabatic operation.

The total molar mass balance, the component mass balance and the energy balance for the reactive batch distillation column are given below for reboiler, trays and reflux-drum-condenser.

Reboiler:

$$\frac{dM_1}{dt} = L_2 - V_1 + \varepsilon_t R_1 M_1 \tag{4.1}$$

$$\frac{dM_1 x_{1j}}{dt} = L_2 x_{2j} - V_1 y_{1j} + \varepsilon_j R_1 M_1 \quad \text{j = 1,...,NC} \tag{4.2}$$

$$\frac{d(M_1 h_1)}{dt} = L_2 h_2 - V_1 H_1 + Q_1 \tag{4.3}$$

Trays: i = 2,...,NT+1; j = 1,...,NC

$$\frac{dM_i}{dt} = L_{i+1} + V_{i-1} - L_i - V_i + \varepsilon_t R_i M_i \tag{4.4}$$

$$\frac{d(M_i x_{ij})}{dt} = L_{i+1} x_{i+1,j} + V_{i-1} y_{i-1,j} - L_i x_{ij} - V_i y_{ij} + \varepsilon_j R_i M_i \tag{4.5}$$

$$\frac{d(M_i h_i)}{dt} = L_{i+1} h_{i+1} + V_{i-1} H_{i-1} - L_i h_i - V_i H_i \tag{4.6}$$

Reflux-drum-condenser system: j = 1,…,NC

$$\frac{dM_{NT+2}}{dt} = V_{NT+1} - L_{NT+2} - D + \varepsilon_t R_{NT+2} M_{NT+2} \tag{4.7}$$

$$\frac{d(M_{NT+2} x_{NT+2,j})}{dt} = V_{NT+1} y_{NT+1,j} - L_{NT+2} x_{NT+2,j} - D x_{NT+2,j} + \varepsilon_j R_{NT+2} M_{NT+2} \tag{4.8}$$

$$\frac{d(M_{NT+2} h_{NT+2})}{dt} = V_{NT+1} H_{NT+2} - L_{NT+2} h_{NT+2} - D h_{NT+2} - Q_{NT+2} \tag{4.9}$$

where *x* and *y* are liquid and vapor fractions (*mol/mol*); *M*, molar liquid holdup (*mol*); *L* and *V*, liquid and vapor molar flow rates (*mol/h*); *h* and *H*, liquid and vapor mixture enthalpy (*J/mol*); Q_1 and Q_{NT+2}, reboiler and condenser heat loads (*J/mol.h*); *D*, distillate flow rate (*mol/h*) and the subscripts *i* and *j* are for stage and component numbers; NT and NC are number of trays and number of components, respectively (*i.e. i* = 1 for reboiler, *i* = 2,...,*NT* + 1 for trays and *i* = *NT* + 2 for reflux-drum-condenser unit); R_i is the reaction rate at i^{th} stage in mol/h defined as in Equation 4.10, where the definition of the rate expression r_i is explained in Section 4.6.

$$R_i = r_i \rho_i / MW_i \qquad \text{for i=1,…,NT+2} \tag{4.10}$$

In the energy balance equations (Equation 4.3, 4.6 and 4.9), no additional term for the heat of reaction is included because when the enthalpies are referred to their elemental state. Thus, the heat of reaction is accounted automatically and no separate term is needed (Taylor and Krishna, 2002; Mujtaba and Macchietto, 1997; Monroy-Loperena and Alvarez-Ramirez, 2000).

Using Equations 4.1 and 4.2, the time derivative of the compositions in the reboiler can be obtained as

$$\frac{dx_{1j}}{dt} = \frac{L_2(x_{2j} - x_{1j}) - V_1(y_{1j} - x_{1j})}{M_1} + \varepsilon_j R_1 - \varepsilon_t R_1 x_{1j} \qquad (4.11)$$

Similarly, combining Equation 4.4 and Equation 4.5 gives the time derivative of the compositions on the trays as

$$\frac{dx_{ij}}{dt} = \frac{V_{i-1}(y_{i-1,j} - x_{ij}) + L_{i+1}(x_{i+1,j} - x_{1j}) - V_i(y_{ij} - x_{ij})}{M_i} + \varepsilon_j R_i - \varepsilon_t R_i x_{ij} \qquad (4.12)$$

If the assumption of constant molar liquid holdup in the reflux-drum, is employed, Equation 4.7 reduces to

$$V_{NT+1} = L_{NT+2} + D - \varepsilon_t R_{NT+2} M_{NT+2} \qquad (4.13)$$

and inserting Equation 4.13 to Equation 4.8 gives

$$\frac{dx_{NT+2,j}}{dt} = \frac{V_{NT+1}(y_{NT+1,j} - x_{NT+2,j})}{M_{NT+2}} + \varepsilon_j R_{NT+2} - \varepsilon_t R_{NT+2} x_{NT+2,j} \qquad (4.14)$$

Therefore the time derivatives of the compositions throughout the column, which are the *state equations* of the column, are obtained (Equations 4.11, 4.12, and 4.14).

In order to solve these equations, the vapor and liquid flow rates V_i and L_i are required. Extracting l.h.s. of Equation 4.6 and inserting dM_i / dt from Equation 4.4 as

$$\frac{d(M_i h_i)}{dt} = M_i \frac{dh_i}{dt} + h_i \frac{dM_i}{dt} = M_i \frac{dh_i}{dt} + h_i \left[L_{i+1} + V_{i-1} - L_i - V_i + \varepsilon_t R_i M_i \right] \qquad (4.15)$$

and equating l.h.s. of Equation 4.6 and Equation 4.15 yield the vapor flow entering the i^{th} tray as

$$V_{i-1} = \frac{V_i(H_i - h_i) + L_{i+1}(h_i - h_{i+1}) + M_i \frac{dh_i}{dt} + h_i \varepsilon_t R_i M_i}{H_{i-1} - h_i} \qquad (4.16)$$

Moreover, solving Equation 4.4 for L_i gives

$$L_i = V_{i-1} + L_{i+1} - V_i - \frac{dM_i}{dt} + \varepsilon_t R_i M_i \qquad (4.17)$$

Starting from i = NT+1 and solving Equations 4.15 and 4.17 can yield all the flowrates except the vapor flow entering the condenser, V_{NT+1} and the reflux flow entering the top tray, L_{NT+2}. These flow rates can be found depending upon the reflux ratio.

4.1 Calculation of V_{NT+1}

Finite Reflux Ratio Case:

If the reflux ratio, R, is defined as

$$R = \frac{L_{NT+2}}{D} \qquad (4.18)$$

Equation 4.13 becomes

$$V_{NT+1} = D(R+1) - \varepsilon_t R_{NT+2} M_{NT+2} \qquad (4.19)$$

Employing the assumptions of constant holdup and total condenser and expanding l.h.s. of Equation 4.9 give

$$M_{NT+2} \frac{dh_{NT+2}}{dt} = V_{NT+1}(H_{NT+1} - h_{NT+2}) - Q_{NT+2} \qquad (4.20)$$

Rearranging

$$Q_{NT+2} = V_{NT+1}(H_{NT+1} - h_{NT+2}) - M_{NT+2} \frac{dh_{NT+2}}{dt} \qquad (4.21)$$

Inserting Equation 4.19 in Equation 4.21, to eliminate V_{NT+1}, results in Equation 4.22.

$$Q_{NT+2} = \left[D(R+1) - \varepsilon_t R_{NT+2} M_{NT+2}\right](H_{NT+1} - h_{NT+2}) - M_{NT+2}\frac{dh_{NT+2}}{dt} \qquad (4.22)$$

When the energy balance is applied around the overall column, the following equation is obtained.

$$Q_1 - Q_{NT+2} - Dh_{NT+2} = \sum_{n=1}^{NT+2}\frac{d(M_n h_n)}{dt} \qquad (4.23)$$

Using Equation 4.22, Q_{NT+2} can be eliminated from Equation 4.23 to find the distillate rate, D in the form of

$$D = \frac{Q_1 - \sum_{n=1}^{NT+1}\frac{d(M_n h_n)}{dt} + \varepsilon_t R_{NT+2} M_{NT+2}(H_{NT+1} - h_{NT+2})}{(R+1)H_{NT+1} - Rh_{NT+2}} \qquad (4.24)$$

As a result, by using the distillate rate, D, from Equation 4.24, the required flow rates, V_{NT+1} and L_{NT+2} can be calculated from Equation 4.19 and 4.18, respectively.

Total Reflux Case:

In case of total reflux operation, the overhead flow rates, V_{NT+1} and L_{NT+2} cannot be obtained from the equations derived in finite reflux ratio case, because they become undefined for infinite reflux ratio for total reflux operation. Therefore, the flow rates must be found from another formulation.

At total reflux condition, no product is withdrawn from the column (D = 0). Thus,

$$V_{NT+1} = L_{NT+2} - \varepsilon_t R_{NT+2} M_{NT+2} \qquad (4.25)$$

Therefore, if the value of the reflux flow, L_{NT+2} is found, the vapor flow rate of the stream entering the condenser can be obtained.

If the energy balance is applied to the total system of all the trays and the reboiler, the following equation is obtained by using Equation 4.25

$$\sum_{n=2}^{NT+1} \frac{d(M_n h_n)}{dt} + \frac{d(M_1 h_1)}{dt} = Q_1 + L_{NT+2}(h_{NT+2} - H_{NT+1}) + \varepsilon_t R_{NT+2} M_{NT+2} H_{NT+1} \qquad (4.26)$$

and arranging Equation 4.26 gives L_{NT+2} as

$$L_{NT+2} = \frac{Q_1 - \frac{d(M_1 h_1)}{dt} - \sum_{n=2}^{NT+1} \frac{d(M_n h_n)}{dt} + \varepsilon_t R_{NT+2} M_{NT+2} H_{NT+1}}{H_{NT+1} - h_{NT+2}} \qquad (4.27)$$

4.2 Holdup Calculations

The molar holdups on trays can be calculated by utilizing constant volume assumption of liquid holdups, as

$$M_i = \frac{\rho_i^{avg}}{Mw_i^{avg}} v_i \qquad (4.28)$$

where ρ_i^{avg} is the average density of the mixture on the ith tray, Mw_i^{avg} is the average molecular weight of the mixture on the ith tray, v_i is the volume of the liquid tray holdup.

The reboiler holdup at any time, t, is calculated from an algebraic equation given as

$$M_1 = M_f^0 - \sum_{n=2}^{NT+2} M_n - \int_0^t D(\tau) d\tau \qquad (4.29)$$

where M_f^0 is the molar amount of feed initially charged to the column.

4.3 Algebraic Equations

The mole fraction sum for liquid and vapor phases are stated as

$$\sum_{n=1}^{NC} x_n = 1; \sum_{n=1}^{NC} y_n = 1 \tag{4.30}$$

Using the linear pressure drop assumption, the pressure profile can be written as

$$P_i = P_1 - i(P_1 - P_{NT+2})/NT \tag{4.31}$$

where P_i is the pressure in i^{th} tray, P_1, the pressure in the reboiler and P_{NT+2}, the pressure in the reflux drum.

The effects of non-equilibrium between the liquid and the vapor phases on a tray are incorporated to the model by Murphree tray efficiency formulation specifed as

$$y_{ij} = y_{i-1,j} + Eff_{Murphree}(y_{ij}^{*} - y_{i-1,j}) \tag{4.32}$$

where y_{ij}^{*} is the composition of vapor in phase equilibrium with liquid on i^{th} tray with composition x_{ij}; y_{ij}, the actual composition of vapor leaving i^{th} tray; $y_{i-1,j}$, the actual composition of vapor entering i^{th} tray.

4.4 Physical Parameters Calculation

The representative functions for the physical properties of a mixture are expressed as a function of composition, temperature and pressure. Peng-Robinson EOS is used for the calculation of average mixture densities and average molecular weights of a mixture.

Enthalpy departure functions of the vapor and the liquid phases are also calculated using the same equation of state. The enthalpy departure of a mixture from the ideal gas mixture using the Peng-Robinson EOS is as follows:

$$\underline{H}(T,P,x_i) - \underline{H}^{IGM}(T,P,x_i) = RT(Z_m - 1) + \frac{T\left(\frac{da_m}{dT}\right) - a_m}{2\sqrt{2}b_m} \ln\left[\frac{Z_m + (\sqrt{2}+1)B_m}{Z_m - (\sqrt{2}-1)B_m}\right] \tag{4.33}$$

where the subscript m denotes a mixture property and $\underline{H}^{IGM}(T, P, x_i)$ is the ideal gas mixture enthalpy at the conditions of interest (Sandler, 1999). This equation can be used for both vapor and liquid phases, using the appropriate parameters for the terms with the subscript m.

The parameters of critical temperature, T_j^c, critical pressure, P_j^c, boiling temperature, T_j^{boil}, molecular weight, Mw_j, acentric factor, w_j and heat capacity coefficients, $Cp_j^a, Cp_j^b, Cp_j^c, Cp_j^d$ for each component used in the simulation studies are given in Tables A.1 and A.2 (Perry and Green, 1984).

4.5 Initial Conditions

The initialization of differential equations in the model is necessary due to the difficulties of the simulation of highly transient process. Therefore, all the compositions throughout the column (plate compositions and reflux drum compositions) are initialized by the feed compositions, and expressed as

$$x_{ij} = x_j^{feed} \quad for\ i = 1, ..., NT+2 \quad j = 1, ..., NC \tag{4.34}$$

The initial flow rates are taken as

$$V_i = \frac{Q_1}{(H_1 - h_2)} \quad for\ i = 2, ..., NT+2 \tag{4.35}$$

$$L_i = V_{NT+2} \quad for\ i = 1, ..., NT+1 \tag{4.36}$$

4.6 Kinetic Rate Expressions

The reaction kinetics without catalyst is given by Equation 4.37 and the values of the constants k_1 and k_2 for the forward and backward reaction rate constants are given in Table 4.1 (Alejski and Duprat, 1996). The components are numbered as follows: ethyl acetate [1], ethanol [2], water [3], acetic acid [4] and used in subscripts for the mole fractions of liquid (x_i) and vapor (y_i) phases.

$$r = k_1 x_2 x_4 - k_2 x_3 x_1 \tag{4.37}$$

Table 4.1 Reaction Constants (litre/gmol min)

k_1	29100 exp(-7190/T(K))
k_2	7380 exp(-7190/T(K))

4.7 Vapor Liquid Equilibrium (VLE) Calculations

In the modeling of reactive batch distillation column operation, one must estimate the compositions of the liquid and vapor mixtures in equilibrium. The equilibrium temperature and the composition of vapor phase at equilibrium with the liquid phase is represented by

$$[T_i, y_{ij}^*] = f[(x_{ik}, k = 1...NC), T_i^{guess}, P_i] \tag{4.38}$$

where T_i^{guess} is initial temperature guess for trial-error calculation of equilibrium temperature, T_i. In this estimation, thermodynamic modeling of phase equilibrium is very important and the selection of proper thermodynamic model, which affects the estimation of compositions highly, is very crucial. The starting point for all VLE calculations is given below as;

$$\overline{f}_i^L(T, P, x_i) = \overline{f}_i^V(T, P, y_i) \tag{4.39}$$

where the superscripts L and V represent the liquid and vapor phases, respectively and x_i and y_i represent the mole fraction of species *i* in the liquid and vapor, respectively. The fugacity of a species in a liquid and vapor can be computed by using an equation of state (EOS), which is referred as the $\phi - \phi$ method. On the other hand, if an activity coefficient (excess Gibbs free energy) model is used for the liquid fugacity calculation this approach is referred as the $\gamma - \phi$ method. (Sandler, 1999). In the simulation studies, four different models

are used for the calculation of phase equilibrium. The models are explained below in detail.

4.7.1 Model-I: Phase Equilibrium Calculation Using the VLE data from Literature

Vapor-liquid equilibrium data for the ethyl acetate-ethanol-water-acetic acid system which is available in the literature (Suzuki et al., 1971) is given in Table 4.2. This data is utilized in the simulation as a preliminary check.

Table 4.2 Vapor-Liquid Equilibrium Data

Ethyl Acetate	$\log K = -2.3 \times 10^3/T + 6.742$
Ethanol	$\log K = -2.3 \times 10^3/T + 6.588$
Water	$\log K = -2.3 \times 10^3/T + 6.484$
Acetic Acid	$K = (2.25 \times 10^{-2})/T - 7.812$ for $T > 347.6$ K
	$K = 0.001$ for $T \leq 347.6$ K

4.7.2 Model-II: Phase Equilibrium Calculation Using $\phi - \phi$ Approach

In this approach, an EOS is used to calculate the fugacity of species for both liquid and vapor phases. There are many equations of states given in the literature such as Peng Robinson, Redlich-Kwong, Redlick-Kwong-Soave, etc. In this study, the Peng Robinson EOS is used. The generalized form of the Peng Robinson EOS is given in Equations 4.40-4.44.

$$P = \frac{RT}{\underline{V} - b} - \frac{a(T)}{\underline{V}(\underline{V} + b) + b(\underline{V} - b)} \tag{4.40}$$

with

$$a(T) = 0.45724 \frac{R^2 T_c^{\,2}}{P_c} \alpha(T) \tag{4.41}$$

$$b = 0.07780 \frac{RT_c}{P_c} \tag{4.42}$$

$$\sqrt{\alpha} = 1 + \kappa \left(1 - \sqrt{\frac{T}{T_c}} \right) \tag{4.43}$$

$$\kappa = 0.37464 + 1.5422w - 0.26992w^2 \tag{4.44}$$

where w is the acentric factor. However, Equation 4.40 is written for single component vapor-liquid systems. In order to apply the Peng Robinson EOS to a multi-component mixture, the parameters "a" and "b" in Equation 4.40 must be modified by introducing a compositional dependence. In order to obtain these mixture parameters, first of all pure component "a" and "b" values are evaluated using Equations 4.41 and 4.42. Then, "a" and "b" values for the mixture are obtained by making use of the mixing and combining rules (Sandler, 1999). One of the most commonly used mixing rules is the van der Waals one-fluid mixing rule as given in Equations 4.45 and 4.46.

$$a = \sum_{i=1}^{N} \sum_{j=1}^{N} x_i x_j a_{ij} \tag{4.45}$$

$$b = \sum_{i=1}^{N} x_i b_i \tag{4.46}$$

where a_{ii} and b are the parameters for pure component i, and the cross coefficients a_{ij} is obtained by the combining rules given in Equation 4.47.

$$a_{ij} = \sqrt{a_{ii} a_{jj}} (1 - k_{ij}) = a_{ji} \tag{4.47}$$

The binary interaction parameters, k_{ij} which are "introduced to obtained better agreement in mixture equation of state calculations" (Sandler, 1999), for the

ethyl acetate production are given in Table 4.3. The fugacity coefficient of species "i" in a mixture is obtained from the Peng Robinson EOS using van der Waals one-fluid mixing rule for vapor (V) and liquid (L) phases as given below:

$$\ln\frac{\overline{f}_i^V(T,P,y_i)}{y_iP}=\frac{b_i}{b}(Z^V-1)-\ln(Z^V-B)$$
$$-\frac{A}{2\sqrt{2}B}\left[\frac{2\sum_j y_i a_{ij}}{a}-\frac{b_i}{b}\right]\ln\left[\frac{Z^V+(\sqrt{2}+1)B}{Z^V-(\sqrt{2}-1)B}\right] \quad (4.48)$$

$$\ln\frac{\overline{f}_i^L(T,P,x_i)}{x_iP}=\frac{b_i}{b}(Z^L-1)-\ln(Z^L-B)$$
$$-\frac{A}{2\sqrt{2}B}\left[\frac{2\sum_j x_i a_{ij}}{a}-\frac{b_i}{b}\right]\ln\left[\frac{Z^L+(\sqrt{2}+1)B}{Z^L-(\sqrt{2}-1)B}\right] \quad (4.49)$$

where

$$A=\frac{aP}{(RT^2)} \quad (4.50)$$

$$B=\frac{bP}{RT} \quad (4.51)$$

Compressibility factors, Z^L and Z^V, are computed from the EOS.

Table 4.3 Binary Interaction Parameters, k_{ij} (Burgos-Solarzano et. al., 2004)

k_{ij}	Ethyl acetate	Ethanol	Water	Acetic Acid
Ethyl acetate	0.0	0.022	-0.280	-0.226
Ethanol	0.022	0.0	-0.935	-0.0436
Water	-0.280	-0.935	0.0	-0.144
Acetic Acid	-0.226	-0.0436	-0.144	0.0

In Model-III and Model-IV, activity coefficients must be utilized. In thermodynamic VLE calculations, activity coefficient models can be used in two ways. In one way, they are used in the traditional $\gamma-\phi$ approach, which is generally used to correlate and predict VLE behavior at low to moderate pressures. In the second way, they are incorporated into the EOS which is called EOS-G^{ex} method. This second approach is used for the description of the VLE of non-ideal mixtures at high pressures. The problem here is to decide on the activity coefficient model and on the values of the model parameters. There are many activity coefficient models such as van Laar, Margules, Wilson, NRTL, UNIFAC, and UNIQUAC. Van Laar and two-constant Margules models depend on the overall space-averaged composition of the solution. Orbey and Sandler (1998) suggested that "with the same number of adjustable parameters, Wilson, UNIQUAC, and NRTL usually represent the properties of the non-ideal mixtures better than the models based on the overall composition". Therefore, in this study, NRTL, Wilson, and UNIQUAC models are used and the performances for the system under consideration are compared. The adjustable parameters of these models can be obtained either from data compilations such as DECHEMA Chemistry Data Series or by fitting experimental activity coefficient to phase equilibrium data. Three different activity coefficient models are given below in detail.

1. The Non-Random-Two-Liquid (NRTL) Activity Coefficient Model

The multi-component NRTL equation is given in Equations 4.52-4.57 (Sandler, 1999).

$$\frac{G_{\gamma}^{ex}}{RT}=\sum_{i=1}^{N}x_i\frac{\sum_{j=1}^{N}\tau_{ji}G_{ji}x_j}{\sum_{i=1}^{N}G_{ji}x_j} \tag{4.52}$$

with

$$G_{ij}=\exp(-\alpha_{ij}\tau_{ij})\,, \tag{4.53}$$

$$\tau_{ij}=a_{ij}+b_{ij}/T(K)\,, \tag{4.54}$$

$$\alpha_{ij} = \alpha_{ji}, \tag{4.55}$$

and

$$\tau_{ii} = 0 \tag{4.56}$$

for which

$$\ln \gamma_i = \frac{\sum_{j=1}^{N} \tau_{ji} G_{ji} x_j}{\sum_{j=1}^{N} G_{ji} x_j} + \sum_{j=1}^{N} \frac{x_j G_{ij}}{\sum_{k=1}^{N} x_k G_{kj}} \left[\tau_{ij} - \frac{\sum_{k=1}^{N} x_k \tau_{kj} G_{kj}}{\sum_{k=1}^{N} x_k G_{kj}} \right] \tag{4.57}$$

NRTL model has three parameters, $\tau_{ij}, \tau_{ji}, \alpha_{ij}$, for each pair of components in the multi-component mixture. Their values are obtained from Tang et al. (2003) and are given in Table 4.4.

Table 4.4 NRTL Model Parameters

Comp. i Comp. j	AcAc EtOH	AcAc EtAc	AcAc H_2O	EtOH EtAc	EtOH H_2O	EtAc H_2O
a_{ij}	0	0	-1.9763	1.817306	0.806535	-2.34561
a_{ji}	0	0	3.3293	-4.41293	0.514285	3.853826
b_{ij}	-252.482	-235.279	609.8886	-421.289	-266.533	1290.464
b_{ji}	225.4756	515.8212	-723.888	1614.287	444.8857	-4.42868
α_{ij}	0.3	0.3	0.3	0.1	0.4	0.364313

2. Wilson Activity Coefficient Model

The multi-component form of the Wilson equation is given in Equations 4.58 and 4.59 (Sandler, 1999).

$$\frac{G_{\gamma}^{ex}}{RT} = -\sum_{i=1}^{N} x_i \ln\left(\sum_{j=1}^{N} \Lambda_{ij} x_j\right) \tag{4.58}$$

for which

$$\ln \gamma_i = 1 - \ln\left(\sum_{j=1}^{N} x_j \Lambda_{ij}\right) - \sum_{j=1}^{N} \frac{x_j \Lambda_{ji}}{\sum_{k=1}^{N} x_k \Lambda_{jk}} \tag{4.59}$$

Since $\Lambda_{ii} = 1$, there are two parameters, $\Lambda_{ij}, \Lambda_{ji}$, for each binary pair of components in the multi-component mixture. The parameters of this model are given in Table 4.5 obtained from Suzuki et al. (1970).

Table 4.5 Wilson Model Parameters

System	Λ_{12}	Λ_{21}
AcAc – EtOH	0.27558	2.28180
AcAc – H_2O	0.26838	1.22642
AcAc – EtAc	0.61790	0.89277
EtOH – H_2O	0.15347	0.92038
EtAc – EtOH	0.55046	0.76670
EtAc – H_2O	0.12353	0.14907

3. Universal Quasichemical (UNIQUAC) Activity Coefficient Model

The UNIQUAC model is based on statistical mechanical theory. In this model, for a system of N components, the activity coefficient of the i^{th} component is given by

$$\ln \gamma_i = \ln \gamma_i(combinatorial) + \ln \gamma_i(residual) \tag{4.60}$$

In this equation, the combinatorial term accounts for molecular size and shape differences, and the second term accounts largely for energy differences, and expressed as in Equations 4.61-4.65.

$$\ln \gamma_i(combinatorial) = \ln \frac{\phi_i}{x_i} + \frac{z}{2} q_i \ln \frac{\theta_i}{\phi_i} + l_i - \frac{\phi_i}{x_i} \sum_{j=1}^{N} x_j l_j$$

$$\ln \gamma_i(residual) = q_i \left[1 - \ln \left(\sum_{j=1}^{N} \theta_j \tau_{ji} \right) - \sum_{j=1}^{N} \frac{\theta_j \tau_{ij}}{\sum_{j=1}^{N} \theta_k \tau_{kj}} \right] \tag{4.61}$$

with

$$l_i = 5(r_i - q_i) - (r_i - 1) \tag{4.62}$$

$$\theta_i = \frac{x_i q_i}{\sum_{j=1}^{N} x_j q_j} \tag{4.63}$$

$$\phi_i = \frac{x_i r_i}{\sum_{j=1}^{N} x_j r_j} \tag{4.64}$$

$$\ln \tau_{ij} = -\frac{u_{ij}}{RT} \tag{4.65}$$

where r_i is the volume parameter for species i, q_i is the surface area parameter for species i, θ_i is the area fraction of species i, ϕ_i is the segment or volume fraction of species i, and u_{ij} is the average interaction energy for a species i – species j interaction. The UNIQUAC model parameters used in this study are given in Table 4.6 (Okur and Bayramoglu, 2001; Kang et al., 1992).

Table 4.6 UNIQUAC Model Parameters

	Ethanol	Acetic acid	Ethyl acetate	Water
Q	1.972	2.092	3.116	1.365
R	2.1055	2.2024	3.4786	0.90

System	u_{ij} /R (K)	u_{ji} /R (K)
AcAc – EtOH	268.54	-225.62
AcAc – H_2O	398.51	-255.84
AcAc – EtAc	-112.33	219.41
EtOH – H_2O	-126.91	467.04
EtOH – EtAc	-173.91	500.68
H_2O - EtAc	-36.18	638.60

4.7.3 Model-III: Phase Equilibrium Calculation Using the Combination of EOS Models with Excess Free Energy Models (EOS-G^{ex} Approach)

Polar fluids exhibit peculiar behavior of high non-linearity and the modeling of such systems are difficult. There are two important issues for polar fluids. One of them is the EOS and the other is the mixing rule selection. The generalization of the Peng-Robinson EOS parameters is especially useful for hydrocarbons and inorganic gases. However, "for polar fluids (water, organic acids, alcohols, etc.), this simple generalization is not accurate, especially at low temperatures and pressures" (Sandler, 1999).

One of the alternate procedures for the EOS that have been suggested for polar fluids is the Peng-Robinson-Stryjek-Vera (PRSV) equation developed by Stryjek and Vera (1986) in which Equation 4.44 is replaced with

$$\kappa = \kappa_0 + \kappa_1(1+T_r^{0.5})(0.7-T_r) \tag{4.66}$$

where

$$\kappa_0 = 0.378893 + 1.4897153w + 0.17131848w^2 + 0.0196554w^3 \tag{4.67}$$

Here κ_1 is a parameter specific to each pure compound that is optimized to accurately fit low-temperature vapor-pressure data. The Stryjek-Vera modification of α term takes care of the inaccuracies in temperature dependence of the "a" term at low temperatures. κ_1 parameters for the components used in this study which are obtained from the study Stryjek and Vera (1986) are given in Table 4.7.

Table 4.7 PRSV EOS Parameters, κ_1

Components	κ_1
Ethyl acetate	0.0693
Ethanol	-0.03374
Water	-0.06635
Acetic acid	-0.19724

In the selection of the mixing rule, if fitting VLE data with the van der Waals one-fluid mixing rule is used, it was found that the binary interaction parameter is approximately zero for relatively simple mixtures, such as alkane mixtures. However, for non-ideal mixtures, it is not only nonzero but will also change in value with temperature. Therefore, accurate correlation of VLE is not possible by van der Waals one-fluid mixing rule. There are many studies in the literature for which the van der Waals one-fluid mixing rules give false predictions of liquid-liquid splits. Difficulties are also encountered with water and alcohol mixtures (Orbey and Sandler, 1998). "One is led to expect that the combination of a cubic EOS with the van der Waals mixing rules can only represent those mixtures that are describable by augmented regular solution theory. This excludes polar and hydrogen-bonding fluids." (Orbey and Sandler, 1998).

One method which can be utilized for the mixing rules is to combine an EOS with activity coefficient models (Orbey and Sandler, 1998) such as Huron-Vidal (Original) Mixing Rule (HVO). In this mixing rule the mixture EOS parameters are given as

$$b = \sum_{i=1}^{N} x_i b_i \tag{4.68}$$

which is identical to the mixing rule for the b parameter in van der Waals rule, and

$$a = b\left[\sum_{i=1}^{N} x_i \frac{a_i}{b_i} + \frac{\underline{G}_{\gamma}^{ex}(T, x_i)}{C^*}\right] \tag{4.69}$$

where C* is the EOS-dependent constant, and for the Peng-Robinson equation $C^* = \ln(\sqrt{2}-1)/\sqrt{2} = -0.62323$. $\underline{G}_{\gamma}^{ex}$, the molar excess Gibbs free-energy obtained from any excess free-energy model, is a function of temperature and composition only.

In this case, the fugacity coefficient of species i in the mixture become

$$\begin{aligned} \ln \bar{\phi}_i = {} & \frac{b_i}{b}(Z-1) - \ln(Z-B) \\ & - \frac{1}{2\sqrt{2}}\left[\frac{a_i}{b_i RT} + \frac{\ln \gamma_i}{C^*}\right] \ln\left[\frac{Z + (\sqrt{2}+1)B}{Z - (\sqrt{2}-1)B}\right] \end{aligned} \tag{4.70}$$

where again the compressibility factor is computed from the EOS.

Another mixing rule in this category is the Orbey-Sandler modification of the Huron-Vidal mixing rule (HVOS). Again "b" is same as in Equation 4.68 and the a parameter and the fugacity coefficient of species "i" in the mixture becomes as given in Equation 4.71 and Equation 4.72, respectively.

$$a = bRT\left[\sum_{i=1}^{N} x_i \frac{a_i}{b_i RT} + \frac{1}{C^*}\left[\frac{\underline{G}_{\gamma}^{ex}(T, x_i)}{RT} + \sum_i x_i \ln\left(\frac{b}{b_i}\right)\right]\right] \tag{4.71}$$

$$\begin{aligned} \ln \bar{\phi}_i = {} & \frac{b_i}{b}(Z-1) - \ln(Z-B) \\ & - \frac{1}{2\sqrt{2}}\left[\frac{a_i}{b_i RT} + \frac{\ln \gamma_i}{C^*} + \frac{1}{C^*}\ln\left(\frac{b}{b_i}\right) + \frac{1}{C^*}\left(\frac{b_i}{b} - 1\right)\right] \ln\left[\frac{Z + (\sqrt{2}+1)B}{Z - (\sqrt{2}-1)B}\right] \end{aligned} \tag{4.72}$$

4.7.4 Model-IV: Phase Equilibrium Calculation Using $\gamma - \phi$ Approach

In VLE descriptions with the $\gamma - \phi$ approach as given before, an activity coefficient model is used for the liquid phase and an EOS is used for the vapor phase. At low to moderate pressures, omitting the Ponyting correction factor, the equality of the fugacities becomes

$$\overline{f}_i^L(T,P,x_i) = x_i\gamma_i(T,P,x_i)P_i^{sat}(T) = \overline{f}_i^V(T,P,y_i) = y_iP\overline{\phi}_i^V(T,P,y_i) \qquad (4.73)$$

Very non-ideal mixtures can be described with $\gamma - \phi$ method since an activity coefficient model can give very large excess Gibbs free energies of mixing if suitable values of parameters are used (Orbey and Sandler, 1998). However, there are important disadvantages of this method. Since different methods are used for the liquid and the vapor phases, the properties of these two phases cannot be identical and therefore it is not useful in description of the critical region behavior. Furthermore, in $\gamma - \phi$ method unlike $\phi - \phi$ method, other thermodynamic properties such as densities and enthalpies cannot be computed from the same model.

4.8 Summary of the Modeling Chapter

The model equations are summarized in Table 4.8.

Table 4.8 Multi-Component Batch Distillation Model Equations

Compositions and Holdups (Section 4.3)

Reboiler Dynamics

$$M_1 = M_f^0 - \sum_{n=2}^{NT+2} M_n - \int_0^t D(\tau)d\tau$$

$$\frac{dx_{1j}}{dt} = \frac{L_2(x_{2j} - x_{1j}) - V_1(y_{1j} - x_{1j})}{M_1} + \varepsilon_j R_1 - \varepsilon_t R_1 x_{1j}$$

$$j = 1 \ldots NC$$

Tray Dynamics

$$M_i = (\rho_i^{avg} / Mw_i^{avg})\upsilon_i$$

$$\frac{dx_{ij}}{dt} = \frac{V_{i-1}(y_{i-1,j} - x_{ij}) + L_{i+1}(x_{i+1,j} - x_{1j}) - V_i(y_{ij} - x_{ij})}{M_i} + \varepsilon_j R_i - \varepsilon_t R_i x_{ij}$$

$$i = 2 \ldots NT+1, \quad j = 1 \ldots NC$$

Reflux-Drum Dynamics

$$\frac{dx_{NT+2,j}}{dt} = \frac{V_{NT+1}(y_{NT+1,j} - x_{NT+2,j})}{M_{NT+2}} + \varepsilon_j R_{NT+2} - \varepsilon_t R_{NT+2} x_{NT+2,j}$$

$$j = 1 \ldots NC$$

Composition Sums (Section 4.3)

$$\sum_1^{NC} x_l = 1, \quad \sum_1^{NC} y_l = 1$$

Flowrates for given R and Q_1 (Section 4.1)

Overhead Flowrates for Total Reflux

$$D = 0$$

$$V_{NT+1} = L_{NT+2} - \varepsilon_t R_{NT+2} M_{NT+2}$$

$$L_{NT+2} = \frac{Q_1 - \frac{d(M_1 h_1)}{dt} - \sum_{n=2}^{NT+1} \frac{d(M_n h_n)}{dt} + \varepsilon_t R_{NT+2} M_{NT+2} H_{NT+1}}{H_{NT+1} - h_{NT+2}}$$

Overhead Flowrates for Finite Reflux Ratio

$$D = \frac{Q_1 - \sum_{n=1}^{NT+1} \frac{d(M_n h_n)}{dt} + \varepsilon_t R_{NT+2} M_{NT+2}(H_{NT+1} - h_{NT+2})}{(R+1)H_{NT+1} - Rh_{NT+2}}$$

$$V_{NT+1} = D(R+1) - \varepsilon_t R_{NT+2} M_{NT+2}$$

$$L_{NT+2} = RD$$

Table 4.8 Multi-Component Batch Distillation Model Equations (continued)

Trays
$$V_{i-1} = \frac{V_i(H_i - h_i) + L_{i+1}(h_i - h_{i+1}) + M_i \frac{dh_i}{dt} + h_i \varepsilon_t R_i M_i}{H_{i-1} - h_i}$$
$$L_i = V_{i-1} + L_{i+1} - V_i - \frac{dM_i}{dt} + \varepsilon_t R_i M_i$$
$$i = NT + 1 \ldots 2$$
Pressure Drop Profile (Section 4.3)
$$P_i = P_1 - i(P_1 - P_{NT+2}) / NT$$
Thermodynamic Models
VLE Calculation (Section 4.7)
$$[T_i, y_{ij}^*] = f[(x_{ik}, k = 1 \ldots NC), T_i^{guess}, P_i]$$
Murphree Tray Efficiency (Section 4.3)
$$y_{i,j} = y_{i-1,j} + eff_{Murphree}(y_{i,j}^* - y_{i-1,j})$$
Enthalpy Calculations (Section 4.4)
$$h_i = f((x_{i,k}, k = 1 \ldots NC), T_i, P_i)$$
$$H_i = f((y_{i,k}, k = 1 \ldots NC), T_i, P_i)$$
Physical Properties (Section 4.4)
$$\rho_i^{avg} = f((x_{i,k}, k = 1 \ldots NC), T_i, P_i)$$
$$Mw_i^{avg} = f(x_{i,k}, k = 1 \ldots NC)$$
Kinetic Rate Expression (Section 4.6)
$r = k_1 x_2 x_4 - k_2 x_3 x_1$ (1) EtAc, (2) EtOH, (3) H_2O, (4) AcAc

CHAPTER 5

OPERATION AND NONLINEAR OPTIMIZATION OF THE REACTIVE BATCH DISTILLATION COLUMN

In this chapter, the operational characteristics of a multi-component batch distillation column are given in the first section. In the second section of the chapter, the nonlinear optimization problem for the multi-component reactive batch distillation column for ethyl acetate production is outlined.

5.1 Operational Characteristics of a Multi-Component Batch Distillation Column

In batch distillation, the composition of the product depends on the initial still-pot feed composition, the number of plates in the column and on the reflux ratio used. In batch distillation operations, products are specified as follows; *slop-cuts* which are the off-spec products and p*roduct-cuts* which are the products of specified purities (Luyben, 1988). In the batch column operation, there are a number of operational stages; start-up period, distillation at total-reflux, withdrawal of the lightest product, removal of a slop-cut, withdrawal of the next heaviest product, removal of a second slop-cut and so on. "If the aim of production is to separate each compound from the feed mixture at the specified purity levels, the number of product-cuts is NC in the feed (*i.e. NC*: number of compounds) and the number of the maximum possible slop-cuts is NC-1" (Yıldız et al., 2005). In Figure 5.2, a schematic of a multi-component batch distillation column system is shown which consists of a reboiler, trays, a condenser, a reflux drum, product and slop-cut storage tanks.

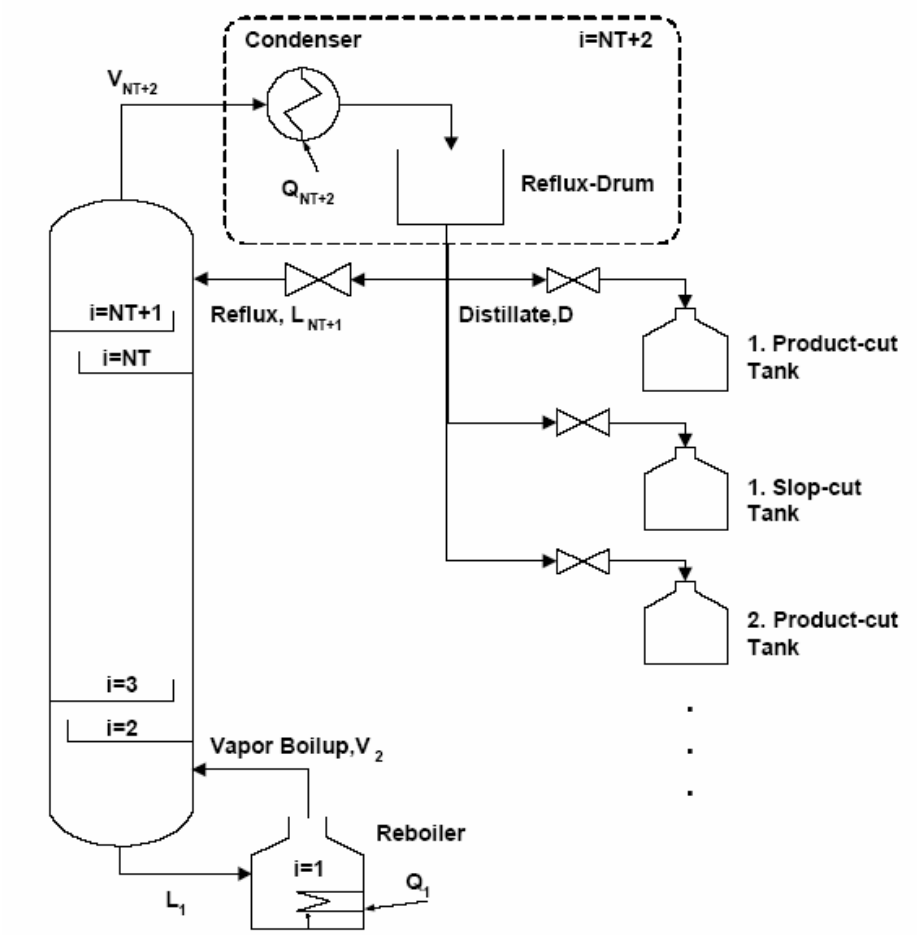


Figure 5.2. The Schematic of a Multi-Component Batch Distillation Column (Yıldız et. al, 2005)

In the operation, the column is first run with total reflux until the distillate composition of the lightest component reaches its desired purity, which is same as its steady state value. Then, the reflux-ratio is set to a pre-specified value. At the same time, the distillate stream is transferred to first product-cut storage tank and the distillate product is withdrawn. In time, the degree of purity of the lightest component in the first product-cut tank drops. There is a period of time in which the distillate will be off-spec of the lightest component until the next heavier component to be used for that product reaches its specified composition level. Therefore, a slop-cut must be withdrawn until the next heaviest component reaches its specified purity. When another specification level is reached, then the distillate is again diverted to another product-cut tank. This cyclic operation between the product-cut and the slop-cut withdrawal continues until all the intermediate compounds are separated which leaves the content of the reboiler as the final product-cut rich in the heaviest component. The

distillation is continued until the component in the reboiler reaches the desired concentration.

5.2 Nonlinear Optimization of the Reactive Batch Distillation Column

As can be understood from the explanation above, during the operation of a batch distillation column, the first operation is at total reflux. Then the reflux ratio changes to a specified value. In the operation, there are two policies for the reflux ratio. Either the reflux ratio can be taken as constant after the total reflux period or can be adjusted in time to different values. In constant reflux ratio policy, only the first product-cut is obtained at the desired purity and the rest cannot be attainable. This reflux ratio policy can be applicable in cases where the only objective is to obtain the lightest or the heaviest product solely. However, if aim is to separate the products from each other, the variable reflux ratio policy must be considered.

The variable reflux ratio policy can only be applied after a study on the optimal reflux ratio policy is carried out. The aim in the optimization is multifold, depending upon the objective function.

In a batch distillation column reflux ratio optimization problem, either the purity of the products/the amounts of the products/the energy requirement or the costs of the raw materials can be considered in the objective function and/or in the constraints. Generally, three different formulations of the optimization problem are given in the literature:

1) Maximum distillate problem, where the amount of distillate with a specified final concentration for a specified batch time is maximized.

2) Minimum time problem, where the batch time required to produce a prescribed amount of distillate of a specified concentration is minimized.

3) Maximum profit problem, where a profit function is maximized.

The optimal operation policy of a batch distillation process can be calculated off-line if there is reliable process model. In the reactive batch distillation column under study, the heat input to the reboiler is constant. Therefore, the only

manipulated variable is the reflux ratio, and since it is a function of time, a dynamic optimization problem must be solved. A low reflux ratio may lead to an excessive withdrawal of the reactants which can affect the conversion inversely. On the other hand, a high reflux ratio may result in an undesirable side reaction giving undesirable by-products. For this reason, the performance of a reactive distillation operation is related with the operation policy in a higher extend than that of batch distillation operation without chemical reaction.

In the reactive distillation process for ethyl acetate production, three binary azeotropes (ethanol-water, ethyl acetate-ethanol, and ethyl acetate-water) and one ternary azeotrope (ethyl acetate-water-ethanol) is formed as explained in Chapter 1. Furthermore, due to the close boiling points between ethanol and ethyl acetate (351.7 K and 350.2 K, respectively), the reflux ratio profile needs to be adjusted carefully so as to withdraw ethyl acetate selectively over ethanol.

The three optimization problem formulations given above are considered together in the study of Luyben (1988) where maximization of the Capacity Factor (CAP) is selected as the performance criteria in the dynamic optimization of the reactive batch distillation process. The CAP of a batch distillation is defined as the ratio between the total amount of the specified products and the total batch time. If P_1, P_2, P_3, and P_4 are the amounts of products obtained and the t_F is the total batch time, the CAP formulation can be given as in Equation 5.1. The total batch time includes the time at total reflux and the time producing the products and slop cuts.

$$CAP = \frac{P_1 + P_2 + P_3 + P_4}{t_F} \tag{5.1}$$

In this optimization problem, there are functional constraints given by the model equations and constraints on the reflux ratio itself. There are also constraints on the purities of the components in the product-cut tanks since this optimization problem is subject to the desired purities of the products collected in the product-cut tanks. The desired purity of the ethyl acetate collected in the first product-cut tank is selected as 0.52, which is the maximum purity of ethyl acetate obtained after total reflux operation. Other desired purities of the components collected in the product-cut tanks are selected considering their maximum possible purities.

The nonlinear optimization problem explained above can be summarized as given in Table 5.1.

Table 5.1 Optimization Problem

Objective Function:
$\max_{R}$ Capacity Factor
subject to:
1. Model equations 2. Purity constraints on the product cuts: Desired purity of EtAC in product-cut tank 1 = 0.52 Desired purity of EtOH in product-cut tank 2 = 0.50 Desired purity of H2O in product-cut tank 3 = 0.65 Desired purity of AcAc in reboiler = 0.999 3. Constraints on the reflux ratio: $0 < R_p = L/V \leq 1$

CHAPTER 6

INFERENTIAL CONTROL AND ARTIFICIAL NEURAL NETWORK STATE ESTIMATOR

In this study, an inferential control methodology that uses Artificial Neural Network (ANN) estimator to infer the product compositions from temperature measurements and provides a feedback control by using the optimal reflux profile of the column is developed. In this chapter, the theoretical background of the inferential control strategy, the observability criteria and the ANN estimator are explained.

6.1 Inferential Control

When the controlled output of the process cannot be conveniently measured on-line, measurements of a secondary variable can be used to estimate the controlled variable in time. If the disturbances can be measured and an adequate process model is available, then feed-forward control can be used to keep the unmeasured output at its desired value instead of inferential control. However, in the presence of unmeasured disturbances, inferential control is the only solution that can be used to control an unmeasured process output.

The control of multi-component distillation columns is a typical example of this type of process control problem. Here, the major input disturbances are variations in the feed composition, temperature, and flow rate. Controlled quantities are the product compositions. Composition analyzers such as gas chromatographs can be used to measure the product composition but these are expensive, difficult to maintain and introduce undesirable time delays in the feedback control loop. In industry, to maintain a constant temperature on one of

the trays close to the product withdrawal location using feedback control is a method commonly used (Joseph and Brosilow, 1978). For a binary distillation column, the Gibbs phase rule indicates that there is a unique relation between composition and temperature if pressure is constant. Therefore, a thermodynamic equation could be employed to relate the temperature of the tray at one end of the column to the corresponding product composition. However, for the separation of multi-component mixtures, the tray temperature does not correspond exactly to the product composition. Therefore, approximate methods must be used to estimate the compositions (Bahar, 2003).

Inferential control system uses measurements of secondary process outputs to infer the effect of immeasurable disturbances on primary process outputs. The estimator uses the values of the available measured outputs, together with the material and energy balances that govern the process, to estimate the values of the unmeasured controlled variables. These estimates, in turn, are used by the controller to adjust the values of the manipulated variables. The structure of inferential control configuration used in this study is given in Figure 6.1 (adapted from Stephanopoulos, 1984).

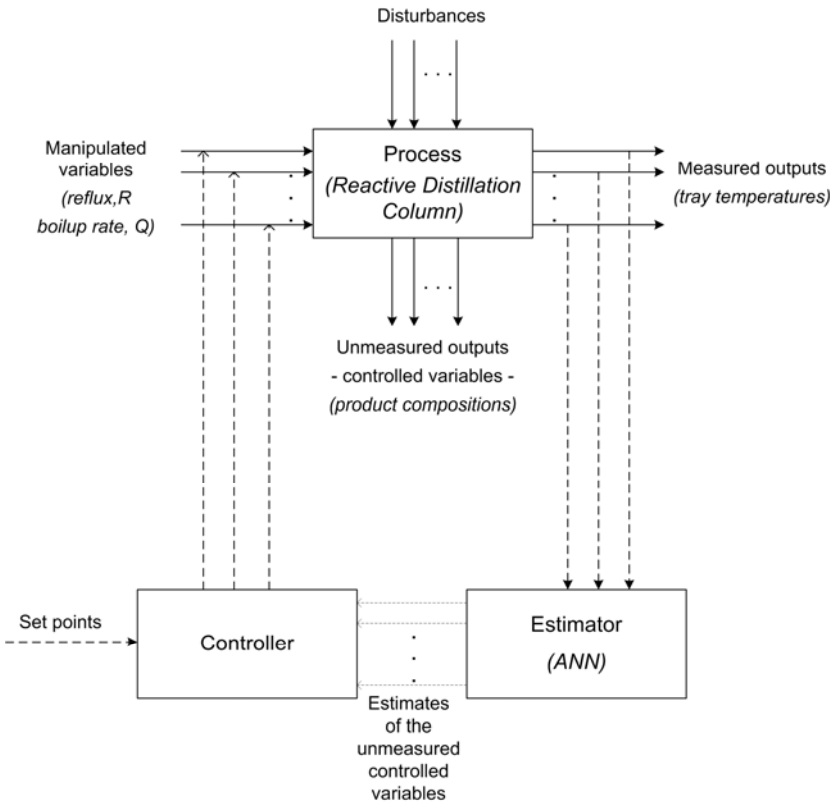


Figure 6.1 The Structure of Inferential Control Configuration

6.2 Observability Criteria and Selection of Measurements

Most control laws used in the batch distillation are feedback laws and the state-space description of dynamics realizes that the information required for feedback control is the state of the system (Jacobs, 1974). As in the case of batch distillation, in most real controlled processes, the system state (*i.e.* compositions in the batch distillation) is not identical with the observable outputs (*i.e.* temperatures in the distillation) but there is a time-varying relationship between the states, *X* and the outputs, *z* as given in Equation 6.1.

$$\underline{z}(t_k) = h(\underline{X}(t_k); t_k) \tag{6.1}$$

Therefore, the question arises whether or not it is possible to evaluate the state form observations of the output (*i.e.* measurements). The observability criteria is to be satisfied for solving the problem of inferring immeasurable state variables from measurements in the minimum possible length of time (Yıldız et al., 2005).

Since the estimated compositions are used in the control of the column operation, the temperature measurements that are used as inputs to the estimator must be suitably selected in order to provide accurate estimation of the compositions because the measurement locations have significant effects on the performance.

It is not suitable to use all the available temperature measurements as inputs to the estimator because of the measurement redundancy. As Venkateswarlu and Kumar (2006) stated, inappropriate use of measurements may lead to numerical problems such as singularity, over-parameterization, and reduction of estimation accuracy.

The observability concept plays an important role in the design of control systems in state space. Although most physical systems are observable, corresponding mathematical models may not be observable. For this reason, it is necessary to know the conditions under which a system is observable. "A system is said to be observable at time t_0 if, with the system in state X (t_0), it is possible to determine this state from the observation of the output over a finite time interval" (Ogata, 1997). Employing a degree-of-freedom concept, Yu *et al.* (1987) found that a distillation column is observable if the number of

measurements is at least (NC - 1). The study for multi-component batch distillation column of Quintero-Marmol *et al.* (1991) and Yıldız et al. (2005), dealing with the design of an Extended Luenberger Observer and Extended Kalman Filter, respectively, concluded that, even though the linear observer in theory needs only (NC - 1) temperature measurements to be observable, the nonlinear observer needed at least (NC) thermocouples to be effective.

Furthermore, Yıldız et al. (2005) showed that increasing the number of temperature measurements above NC does not result in better performance. Venkateswarlu and Kumar (2006) found in their study that the reboiler and the top tray are the most sensitive temperature measurement locations for a multi-component batch distillation column. Similarly, Yıldız et al. (2005) concluded that the temperature measurement locations should be spread throughout the column homogeneously and should include the reboiler and the top tray.

6.3 Artificial Neural Networks

"A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experimental knowledge and making it available for use" (Haykin, S., 1999).

6.3.1 Historical Development

The neural networks (NNs) modern era began with the work of McCulloch and Pitts (1943). They describe in their study, a logical calculus of NNs that united the studies of neurophysiology and mathematical logic. "Their formal model of a neuron was assumed to follow an all-or-none law. With a sufficient number of such simple units, and synaptic connections set properly and operating synchronously, they showed that a network so constituted would, in principle compute any computable function. This was a very significant results and with it, it is generally agreed that the disciplines of NN and of artificial intelligence were born" (Haykin, S., 1999).

The Hebb's book (The Organization of Behavior) in 1949 was the next major development in NNs. In this book, an explicit statement of a physiological learning rule for synaptic modification was presented for the first time. The important problem of designing a reliable network with neurons that may be

viewed as unreliable components was solved by von Neumann (1956) using the idea of redundancy. This motivated Winograd and Cowan (1963) to suggest the use of a distributed redundant representation for NNs. After 15 years from the paper of McCulloch and Pitt, a new approach to the pattern recognition was introduced by Rosenblatt (1958) in his work on the perceptron, a novel method of supervised learning.

Some technological, psychological and financial factors (high computing labor) contributed to the decrease of interest in NNs in the 1970s. Many of the researchers, except those in psychology and neurosciences, moved away from the field during that decade. Only after 1980s some contributions to the theory and design of NNs were made, and the interest in NNs again increased. The paper by Hopfield in 1982 and the two-volume book by Rumelhart and McLelland in 1986 were the most important publications in the 1980s. Today, NNs is found to be a subject in the neurosciences, psychology, mathematics, physical sciences and engineering (Haykin, S., 1999).

6.3.2 Features of Artificial Neural Networks

There are many advantages of Artificial Neural Networks (ANNs) over first principles models or other empirical models. ANNs can be highly nonlinear. Their structure can be more complex and therefore they are more representative than most other empirical models. They are quite flexible models and the structure does not have to be pre-specified.

ANNs learn with input-output data. One of the most important characteristics of neural networks is the ability to utilize the data and to organize the information into a form that is useful. Typically, this form constitutes a model that represents the relationship between the input and the output variables.

A neural network (NN) memory is distributed since the information is spread among all of the weights that have been adjusted in the training process. These connection weights are the memory units of neural networks and the values of the weights represent the current state of the knowledge of the network. Therefore, each individual unit of knowledge is distributed across all the memory units in the network. Furthermore, it shares these memory units with all other items of information stored in the network. The NN is also associative. Because, if the trained network is presented with a partial input, the network choose the

closest match in the memory to that input and generate an output that corresponds to a full input.

NNs are fault-tolerant, since the information storage is distributed over all weights. Even when a large number of the weights are destroyed, the performance of the NN degrades gradually. While the performance suffers, the system does not fail catastrophically since the information is not contained in just one place but instead, it is distributed throughout the network.

NNs are also capable of pattern recognition, which requires the NN to match large amounts of input information simultaneously and generate a categorical or generalized output with a reasonable response to noisy or incomplete data. For a complex system with many sensors and possible fault types, real-time response is a difficult challenge to both human operators and expert systems. However; while the training time for a neural network may be long, once it has been trained to recognize the various conditions or states of a complex system, it only takes one cycle of the neural network to detect or identify a specific condition or state (Tsoukalas and Uhrig 1997).

6.3.3 Biological Neurons

ANNs are based on the modeling of the behavior of neurons found in the human brain. The biological neural network consists of nerve cells (neurons) as shown in Figure 6.2. In the cell body (soma of the neuron), which includes the neuron's nucleus most of the neural computation occurs. The signals generated in soma are transmitted to other neurons through an extension on the cell body called axon or nerve fibres. Another extension on the cell body is dendrites. They are like bushy tree and are responsible from receiving the incoming signals generated by other neurons. The axon is separated into several branches and at the very end the axon enlarges and forms terminal buttons. Terminal buttons are placed in special structures called the synapses which are the junctions transmitting signals from one neuron to another.

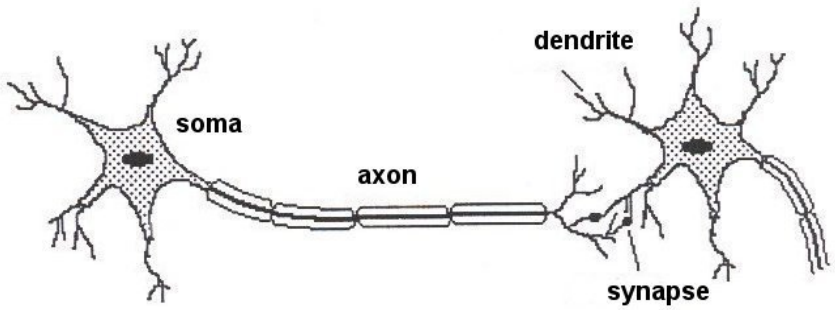


Figure 6.2 Structure of a Biological Neuron

All interconnections are not equally weighted. Some have a higher priority (a higher weight) than others do. Also, some are excitory and some are inhibitory. These differences are due to the differences in chemistry, the existence of chemical transmitter and modulating substances inside and near the neurons, the axons, and in the synaptic junction. This nature of interconnection between neurons and weighting of messages is also fundamental to ANNs.

In terminal buttons, the synaptic vesicles which hold several thousand molecules of chemical transmitters take place. When a nerve impulse arrives at the synapse, some of these chemical transmitters are discharged into synaptic cleft, (the narrow gap between the terminal button of the neuron transmitting the signal and the membrane of the neuron receiving it). The membrane of the post-synaptic cell gathers the chemical transmitters. This cause either decrease or increase in the efficiency of the local sodium and potassium pumps depending on the type of the chemicals released into the synaptic cleft. The synapses, whose activation decreasing the efficiency of the pumps cause depolarization of the resting potential. On the other hand, the synapses increasing the efficiency of pumps results in hyper-polarization. The first kind of synapses which encourage depolarization is called excitatory and the others which discourage it are called inhibitory synapses. If the decrease in the polarization is adequate to exceed a threshold, then the post-synaptic neuron fires.

6.3.4 Artificial Neurons

A NN resembles the brain since knowledge is acquired by the network from its environment through a learning process, and synaptic weights are used to store the acquired knowledge. Engineering systems are considerably less complex than the brain, hence from an engineering viewpoint, ANN can be viewed as nonlinear empirical models that are especially useful in representing input-output data, making predictions in time, classifying data, and recognizing patterns.

Figure 6.3 shows the basic structure of a neuron model. A neuron receives one or more input signals. The artificial neuron (processing element) given in this Figure 6.3 has N inputs (u_1, u_2,…,u_N) and each input is weighted according to the value wj (w1, w_2, …, w_N), which is called a weight. These weights in the artificial model are similar to the synaptic strength between two connected neurons in the human brain. A negative value for a weight indicates an inhibitory connection while a positive value indicating excitatory one. The weighted signals are summed and the resulting signal called the activation, a, given by the formula as in Equation 6.2, is sent to a transfer function, f, which can be any type of mathematical function. θ represents the threshold in artificial neuron, and it may be assigned a positive value in artificial neurons unlike biological neurons. If θ is positive, it is usually referred as bias (Bahar, 2003).

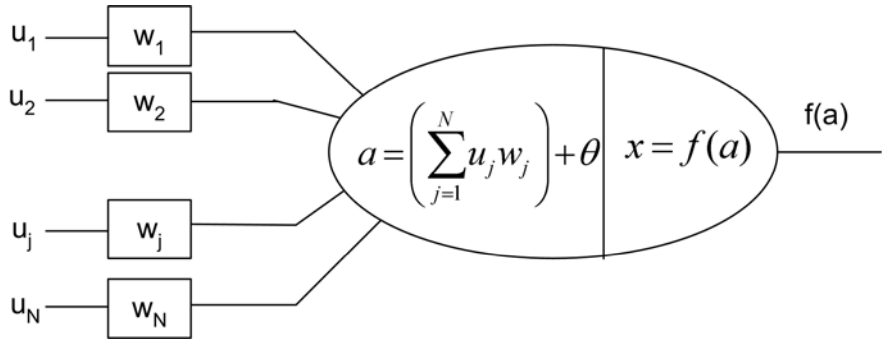


Figure 6.3 Basic Structure of an Artificial Neuron

$$a=\left(\sum_{j=1}^{N} w_j u_j\right)+\theta \qquad (6.2)$$

Originally, the neuron output function, f(a), in McCulloch-Pitts model proposed as threshold function, however, linear, ramp, and sigmoid functions are also widely used output functions.

A collection of neurons connected to each other forms the ANN. Connection weights of neurons are represented in the form of matrix w, element w_{ij} of which is the connection weight between i^{th} and j^{th} neurons.

6.3.5 Types of Artificial Neural Networks

The arrangement of the interconnections between the NNs and the nature of the connections determine the structure of a NN. How the strengths of the connections are adjusted or trained to achieve a desired overall behavior of the network is governed by its learning algorithm. Neural networks can be classified according to their structures and learning algorithms.

Basic Structures of Neural Networks

Neural networks can have two different structures as feed-forward networks and recurrent networks.

Feed-forward neural networks: The neurons are organized in the form of layers. Figure 6.4 is a structure of a feed-forward (layered) neural network. A group of neurons, called the input layer, receives a signal from some external source and passes this information to the network. In general, this input layer does not process the signal unless it needs scaling. Another group of neurons, called the output layer, return signals to the external environment. The remaining groups of neurons in the network are called hidden layers since they do not receive signals from or send a signal to an external source. A typical NN consists also 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 input is connected to each of the hidden and output neurons in a network. Signals flow from the input layer through the output layer in one direction. The neurons are connected from one layer to the next (input to hidden, hidden to hidden, hidden to output), but not within the same layer or to the previous layer (Bahar, 2003).

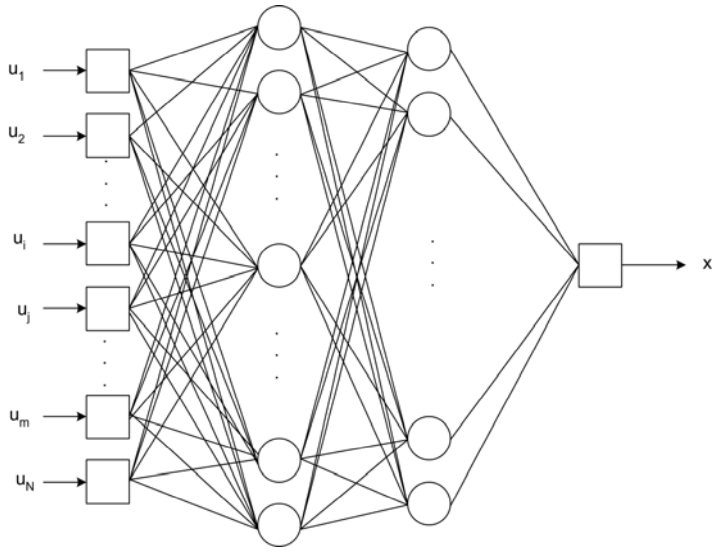


Figure 6.4 Basic Structure of a Feed-Forward Network

In a feed forward neural network, if there is only the layer of input neurons and a single layer of neurons constituting the output layer, then they are called single layer network. If there are one or more hidden layers, such networks are called multilayer networks.

Examples of feed-forward networks include the multilayer perceptron (MLP), the learning vector quantization (LVQ) network, and the group-method of data handling (GMDH) network (Bulsari, 1995).

A feed-forward network doesn't have a dynamic memory. In order to use it for a dynamic system, generally moving window method is used. This method employs both the current and the past inputs and outputs of the system as inputs to the network and has been used satisfactorily for representing a dynamic system (Bahar et al., 2004, Jazayeri-Rad, 2004). One of the drawbacks of this method is its slow computation due to the large number of inputs. The large number of inputs also makes the identifier highly susceptible to external noise. Another drawback is that, the training structure is different from the recall structure and frequently the networks seem to have been trained well with the training structure but have poor performance with the recall structure. Thus, to obtain a satisfactory independent simulation of a dynamic system is not easy (Pham and Liu, 1995). In feed-forward networks, the output at any instant is dependent only on the inputs and the weights at that instant; therefore, these networks have no dynamic memory.

Recurrent networks: In a recurrent network, connections may be made between neurons in nonadjacent layers or within the same layer or feedback connections from a neuron in one layer to a neuron in a previous layer. Thus, signals can flow in both forward and backward directions and the output of that neuron becomes a function of both inputs from the previous layer at time t and its own output that existed at an earlier time. Because of this property, recurrent networks have a dynamic memory. Examples of recurrent networks include the Hopfield network, the Elman network, and the Jordan network. The Elman network is one of the simplest type that can be trained using the backpropagation learning algorithm and is therefore used in this study.

In an Elman network, in addition to the input units, hidden units, and output units, there are also context units. The input and output units have an interaction with the outside environment, however the hidden and context units have not. The input units only pass the signals without changing them. The output units sum the signals fed to them. The hidden units can have linear or nonlinear activation functions. The context units are used only to memorize the previous activations of the hidden units. The Elman network is only partially recurrent since the recurrent connections are fixed and the feed-forward connections are modifiable. At a specific time, k, the previous activations of the hidden units (at time k-1) and the current inputs (at time k) are used as inputs to the network. These inputs are propagated forward to produce the outputs. The standard backpropagation learning rule is then employed to train the network. After this training step, the activations of the hidden units at time k are sent back through the recurrent links to the context units and saved there for the next training step (time k+1).

In Figure 6.5, a sample structure for an Elman network is given. The external inputs to the network is denoted by U(k-1), the network outputs is denoted by Y(k), the activations of the hidden units and the outputs of the context units are denoted by X(k) and $X^c(k)$, respectively. These can be represented as in Equation 6.3 when the weight matrices are denoted as W^{xc}, W^{xu}, W^{yx}. F is the nonlinear vector function.

$$\begin{aligned} X(k) &= F\{W^{xc}X^c(k), W^{xu}U(k-1)\} \\ X^c(k) &= X(k-1) \\ Y(k) &= W^{yx}X(k) \end{aligned} \tag{6.3}$$

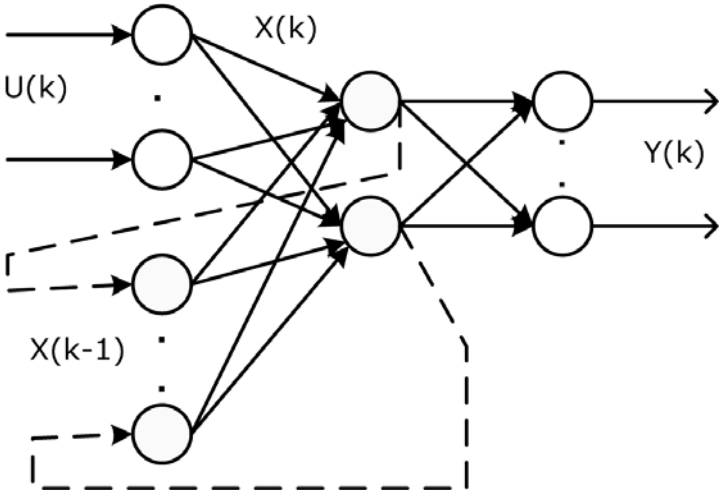


Figure 6.5 Basic Structure of an Elman Network

In particular, when linear hidden units are adopted and the biases of the hidden and outputs units are assumed to be zero. Standard state space descriptions of dynamic systems can be written as in Equation 6.4.

$$\begin{aligned} X(k) &= W^{xc} X^{c}(k) + W^{xu} U(k-1) \\ X^{c}(k) &= X(k-1) \\ Y(k) &= W^{yx} X(k) \end{aligned} \tag{6.4}$$

The order of the models depends on the number of states, which is also the number of hidden units. Theoretically, an Elman network is able to model an nth order dynamic system (where n is the number of units in the hidden layer) if it can be trained to do so. An Elman network will be significantly smaller in structure than a feed-forward network when n is large (Pham and Liu, 1995).

Learning in Neural Networks

When NN is used for modeling, generally, there is no direct analytical method of calculating the values of the weights. Instead, the NN must be trained with a set of data of the process to be modeled. The procedure of estimating the values of the weights and establishing the NN structure is called "training" and the algorithm used to do this is called a "learning algorithm". Learning and recall are the two major functions that NN perform. In the learning process, the connection weights of ANN are adapted to produce the desired output vector in response to a stimulus vector presented to the input buffer. The learning algorithm is a type

of optimization algorithm. Recall is the process of accepting an input stimulus and producing an output response in accordance with the network weight structure. Recall is an integral part of the learning process since a desired response to the network must be compared to the actual output to create an error function.

Two types of learning algorithms exist as supervised and unsupervised learning algorithms. In supervised learning, the ANN is trained to give the desired response to a specific input stimulus. Examples of supervised learning algorithms include the delta rule, the generalized delta rule or backpropagation algorithm and the LVQ algorithm. The vast majority of learning in engineering applications involves supervised learning. A stimulus is presented at the input layer representing the input vector and another stimulus is presented at the output buffer representing the desired response to the given input. The difference between actual output and desired response constitutes an error, which is used to adjust the connection weights. Reinforcement learning is a special case of supervised learning. Instead of using a teacher to give target outputs, a reinforcement learning algorithm employs a critic only to evaluate the goodness of the neural network output corresponding to a given input. An example of this type learning algorithm is the genetic algorithm (GA).

In unsupervised learning, the desired outputs are not known. Only the input patterns are presented to the NN which automatically adapts the weights of its connections to cluster the input patterns into groups with similar features. This type of learning is sometimes referred to as self-organizing learning, i.e., learning to classify without being taught. Examples of unsupervised algorithms include the Kohonen, and Carpenter-Grossberg Adaptive Resonance Theory (ART) competitive learning algorithms.

Backpropagation Training in Elman Networks

Regardless of the training algorithm used to calculate the values of the weights, all of the training methods go through the same general steps as:

1. For a specified ANN architecture, the values of the weights in the network are initialized as small random values;
2. The input and output data are scaled;

3. The inputs of the training set are sent to the network and the resulting outputs are calculated;
4. Some measure (an objective function) of the error between the outputs of the network and the known correct (target) values is calculated;
5. The gradients of the objective function with respect to each of the individual weights are calculated;
6. The weights are adjusted in such a way that minimizes the error, starting from the output layer and going backward to input layer;
7. The procedure is repeated from step 3 for each vector in the training set until the error for the set is lower than the required minimum error.

In this study, the backpropagation algorithm is used for training. The weights of the Elman network are estimated recursively and found that, the feedback depends on $X(k-1)$ which can be represented by $W_{k-1}^{xc}X^c(k-1)+W_{k-1}^{xu}U(k-2)$. $X^c(k-1)$ depends on $X(k-2)$ which is equal to $W_{k-2}^{xc}X^c(k-2)+W_{k-2}^{xu}U(k-3)$. For this reason, $X^c(k)$ depends on the weights of different previous time instants. When the backpropagation method is applied, the dependence of the $X^c(k)$ on the weights should also be taken into account. Assuming that, there is only one input unit and one output unit in the Elman network. The training data set is (u(k), y_d(k)), k = 1,2, …, N. Here, u denotes the input, and y_d denotes the desired value of the output. When one input-output data pair is presented to the network at time k, the squared error at the network output, e can be defined as:

$$e_k = \frac{1}{2}(y_d(k) - y(k))^2 \tag{6.5}$$

For the whole training data set, the summed squared error is:

$$e = \sum_{k=1}^{N} e_k \tag{6.6}$$

If pattern-based learning is conducted, the weights are modified at each time step k. For W^{yx}

$$\frac{\partial e_k}{\partial W^{yx}} = -(y_d(k) - y(k))\frac{\partial y(k)}{\partial W^{yx}} = -(y_d(k) - y(k))X^T(k) \tag{6.7}$$

For W^{xu} and W^{xc},

$$\frac{\partial e_k}{\partial W^{xu}} = -\frac{\partial e_k}{\partial y(k)}\frac{\partial y(k)}{\partial X(k)}\frac{\partial X(k)}{\partial W^{xu}} = -(y_d(k) - y(k))(W^{yx})^T u(k) \tag{6.8}$$

$$\frac{\partial e_k}{\partial w_i^{xc}} = -\frac{\partial e_k}{\partial y(k)}\frac{\partial y(k)}{\partial x_i(k)}\frac{\partial x_i(k)}{\partial w_i^{xc}} = -(y_d(k) - y(k))w_i^{yx}\frac{\partial x_i(k)}{\partial w_i^{xc}} \tag{6.9}$$

where $x_i(k)$ is the i^{th} element of X(k), w_i^{xc} is the i^{th} row of W^{xc}, w_i^{yx} is the i^{th} element of W^{yx}. As discussed previously, the internal feedback is dependent on W^{xc}. Therefore, from Equation 6.4,

$$\frac{\partial x_i(k)}{\partial w_i^{xc}} = (X^c)^T(k) + w_i^{xc}\frac{\partial X^c(k)}{\partial w_i^{xc}} = X^T(k-1) + w_i^{xc}\frac{\partial X(k-1)}{\partial w_i^{xc}} \tag{6.10}$$

This equation shows that there is a dynamic trace of the gradient. This is similar to standard backpropagation since the general expression for weight modification in the gradient descent algorithm is

$$\Delta W = -\eta\frac{\partial e_k}{\partial W} \tag{6.11}$$

where η is a positive constant, called learning factor.

If the dependence of X(k-1) on W^{xc} is ignored, the above algorithm degenerates to the standard backpropagation algorithm (Pham and Liu, 1995).

6.3.6 ANN Architecture

The input/output mapping of a network is established according to the weights and the activation functions of their neurons in input, hidden and output layers. The number of input neurons corresponds to the number of input variables in the NN, and the number of output neurons is the same as the number of desired output variables. The choice of the number of hidden layers and the neurons in the hidden layer(s) is not as straightforward as input and output layers since it depends on the network application. However, the number of hidden layers can be chosen based on the training of the network, using various configurations. The configuration with the fewest number of layers and neurons which still quickly and efficiently give the minimum root-mean-squares (RMS) error can be selected. In general, adding a second hidden layer improves the network's prediction capability. However, adding an extra hidden layer commonly gives prediction capabilities similar to those of two-hidden layer networks, but requires longer training times due to the more complex structures (Zilouchian and Jamshidi, 2001).

6.3.7 Applications of Artificial Neural Networks

NNs have been employed in a wide range of applications; including modeling and prediction, classification and pattern recognition, clustering, signal processing, and optimization. In this study, it is used for prediction of state variables.

CHAPTER 7

SIMULATION CODE AND ALGORITHM

The developed unsteady state dynamic model (Chapter 4) of the reactive batch distillation column and the designed ANN estimator (Chapter 6) are simulated utilizing *MATLAB Sofware*. The simulation code, the algorithm for the program, the thermodynamic library and the ANN estimator which are included in the main program are given below.

7.1 Main Simulation Code

The simulation codes are written as m-files and given in Appendix C. A method of Euler's integration is used for integration of ordinary differential equations of the model. The main simulation program consists of four m-files, *"Glob_Decs.m"*, *"Glob_Initial.m"*, *"Cont_Plant_Mfile.m"*, and *"PressureProfile.m"*. The overall structure of the simulation code is given in Table 7.1 and the overall flow diagram is shown in Figure 7.1. The algorithm given in Figure 7.1 is a modified version of what is given by Yıldız (2002). The modifications are in the column (normal versus reactive), in compounds (hydrocarbon versus polar) and in the estimator (EKF versus ANN).

In the program algorithm, the simulation begins with *"Glob.Decs.m"* and *"Glob_Initial.m"*, which are responsible for defining and initializing the operational parameters. Following the initialization, the main integration loop starts with *"thermo_LIBRARY.dll"*, in which physical properties (densities and molecular weights) and thermodynamical properties (equilibrium temperatures, vapor compositions and vapor/liquid enthalpies) of the mixtures are evaluated. After all the physical data are evaluated, vapor flow rates throughout the column

are calculated using the energy equations. The ordinary differential equations are then integrated and the current liquid flow rates are calculated. Then the integration loop goes to the first step of the algorithm. Pressure profile in the column is calculated by *"PressureProfile.m"*. The main file *"Cont_Plant_Mfile.m"*, where the simulation of whole plant is realized, executes all the m-files including the ANN estimator. Simulation codes for the thermodynamic library and for the ANN estimator will be briefly explained in the following sections.

Table 7.1 Overall Structure of the Simulation Code

Main Program Code
Glob_Decs.m Glob_Initial.m Cont_Plant_Mfile.m Pressure_Profile.m
Thermodynamic Library "thermo.LIBRARY.dll"
Thermodynamic Library MATLAB Interface Code
thermo_Init.m thermo_Equilibrium.m thermo_Enthalpy.m thermo_Density.m
Thermodynamic Library FORTRAN dll Code
thermo_LIBRARY.f thermo_LIBRARY.h common_plant.h parameter.h thermo_data.dat
ANN Estimator Code
training.m find_R_interval.m normalize_sim_input.m simulate.m unnormalize_sim_output.m

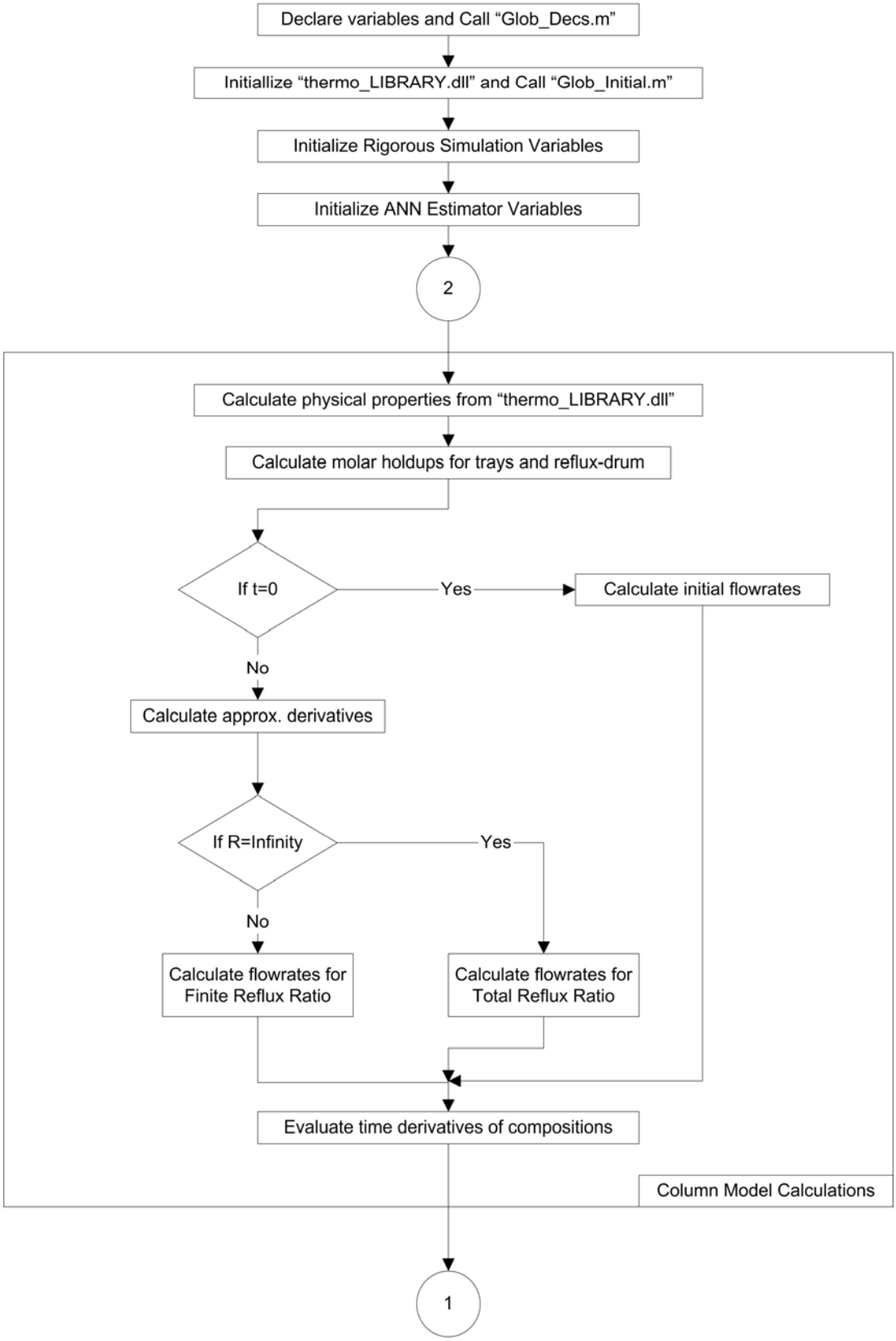


Figure 7.1 Flow Diagram of the Main Program Algorithm (adapted from Yıldız, 2002)

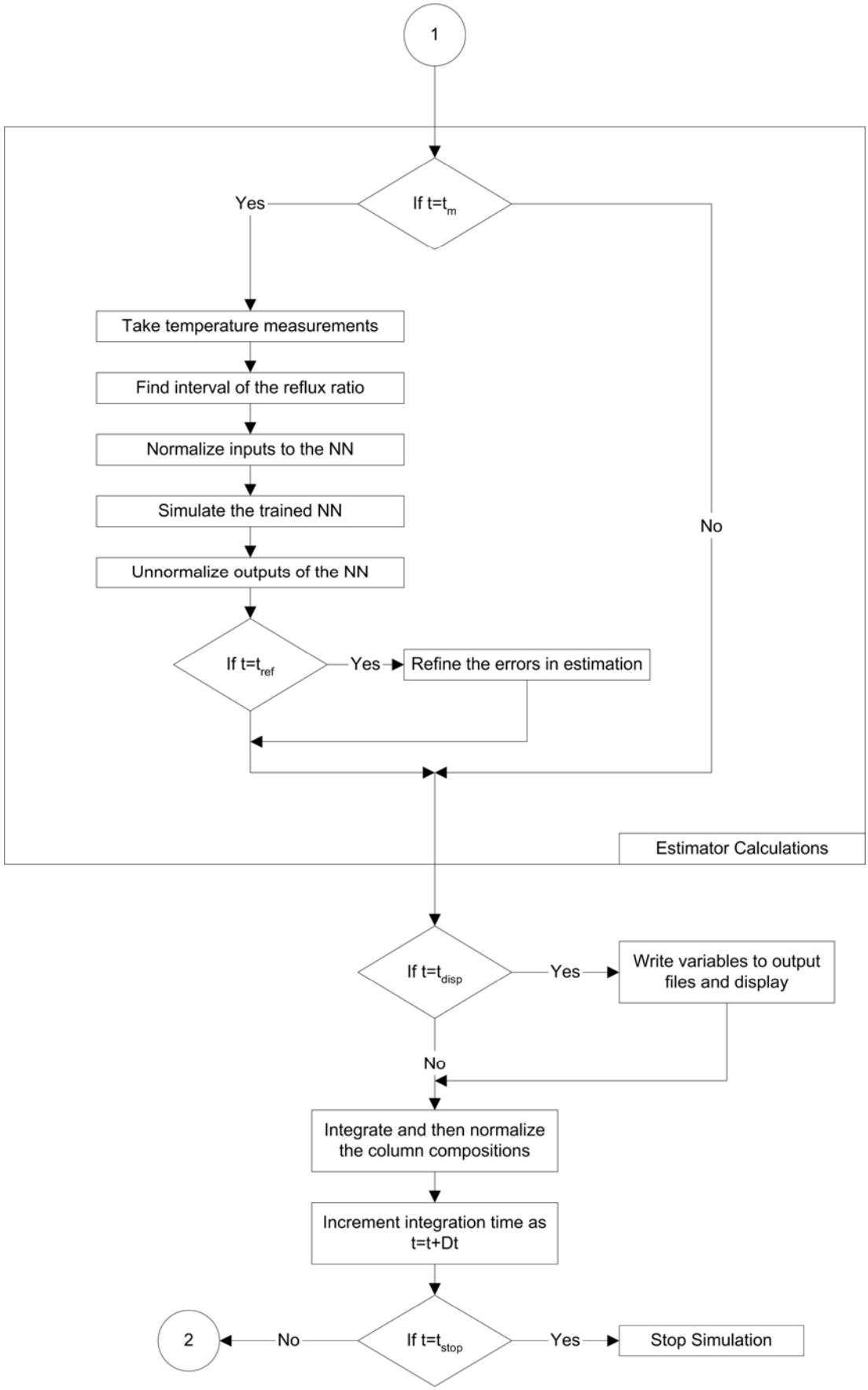


Figure 7.1 Flow Diagram of the Main Program Algorithm (continued)

7.2 Thermodynamic Library Code

The thermodynamic library has two parts, one part is written using *MATLAB Software* and the other part is written in *FORTRAN Programming Language*. The *MATLAB* part forms only an interface between the *MATLAB* and *FORTRAN*, since all the calculations are done in *FORTRAN*. The library has functions for phase equilibrium, enthalpy, density, and molecular weight calculations. In the *MATLAB* part, "*thermo_Init.m*" initializes the library. "*thermo_Equilibrium*" calls the function from *FORTRAN* which calculates the phase equilibrium temperature and vapor composition for a given liquid composition, pressure, and an initial guess for the equilibrium temperature. "*thermo_Enthalpy*" calls the function where the liquid and vapor phase enthalpies are evaluated from given parameters of liquid and vapor composition, temperature and pressure. "*thermo_Density*" calls the function that calculates the average liquid density and molecular weight for the mixture using its liquid composition, temperature, and pressure. All the functions in *FORTRAN* are coupled in a single *FORTRAN* project, "*thermo_LIBRARY.f*". *FORTRAN* "dll" codes also includes the files "*thermo.LIBRARY.h*", "*common_plant.h*", and "*parameter.h*", which provide the definitions of some algorithmic parameters. Through the input "*thermo_data.dat*" file, the properties of the components are supplied. The *MATLAB* interface codes and *FORTRAN* codes can be found in Appendix C.2 and Appendix C.3, respectively.

7.3 ANN Estimator Code

The m-files used for neural network is given in Appendix C.4. The neural network is trained by using the m-file "*training.m*". The first step in training a network is to create the network object. The function *newelm* creates an Elman network. It requires four inputs and returns the network object. The first input is an R by 2 matrix of minimum and maximum values for each of the R elements of the input vector. The second input is an array containing the sizes of each layer. The third input is a cell array containing the names of the transfer functions to be used in each layer. The final input contains the name of the training function to be used. When the network is created, the weights and biases of each layer are initialized and the network is ready for training.

The training process requires a set of examples of network inputs (temperatures at selected trays) and target outputs (distillate compositions). The m-file

"*training.m"* takes the examples for the distillation column from normalized data *"input_norm"* and *"output_norm"*. During training, the weights and biases of the network are iteratively adjusted to minimize the network performance function. The default performance function for Elman network is mean square error (mse), the average squared error between the network outputs and the target outputs. The program uses a back-propagation training algorithm, *trainbfg*, to adjust the weights of the network in order to minimize the sum-squared error of the network. This is done by continually changing the values of the network weights in the direction of steepest descent with respect to the error. The change in each weight is proportional to that element's effect on the sum-squared error of the network. The changes to the weights and biases of the network are obtained by multiplying the learning rate (*lr*) times the negative of the gradient. The larger the learning rate, the bigger the step. If the learning rate is made too large, the algorithm will become unstable. If the learning rate is set too small, the algorithm will take a long time to converge. The training status will be displayed every *show* iterations of the algorithm. Other parameters like *epochs*, *goal*, and *time* determine when training is stopped. The training will stop if the number of iterations exceeds *epochs*, if the performance function drops below *goal*, or if the training time is longer than *time* seconds. In this study; the performance goal, the learning rate, and the epochs are chosen as 1×10^{-7}, 0.0001 and 500, respectively.

Once the network is trained, it is saved and used for online estimation. At each estimation time interval, as given in Table 7.1, *"find_R_interval.m"* finds the interval of the reflux ratio and also finds the maximum and minimum values of the input and output variables by interpolation. Each input element to the network is normalized to a value between -1 and 1 by using "*normalize_sim_input.m"*. The network outputs are obtained in *"simulate.m"* by calling the function *sim* that takes the network input and the network object *net* that was created earlier. The network output is then converted to its original value through *"unnormalize_sim_output.m"*.

CHAPTER 8

RESULTS AND DISCUSSION

The results of the modeling and the control studies of the reactive batch distillation tray column are discussed separately in three sections given below; considering modeling, experimentation, optimization and state estimation using ANN for control.

8.1 Modeling Studies

Modeling studies are carried in three steps. In the first step, simulation studies are done and then the results are checked with the simulation data obtained from the literature. In the second step, experimental studies are done and data is collected at total and different reflux ratios. In the third step, the experimental data and the simulation results are compared and the dynamic model is improved by selecting the appropriate thermodynamic model for the VLE calculations.

8.1.1 Simulation Results

Modeling studies carried in this study are explained in Chapter 4 in detail. The proposed model is checked first with similar studies found from the literature. (Mujtaba and Macchietto, 1997 and Monroy-Loperena and Alvarez-Ramirez, 2000). The detailed column parameters used by for comparison are given in Table 8.1.

Table 8.1 Column Parameters

No. of stages (including reboiler and total condenser)	10
Total fresh feed, kmol	5.0
Feed composition (ethyl acetate, ethanol, water, acetic acid), mole fraction	0.0, 0.45, 0.1, 0.45
Column holdup, kmol	
condenser	0.1
internal plates	0.0125
Condenser vapor load, kmol/h	2.5
Column pressure, bar	1.013

In the comparison test studies, the VLE data of Model-I which is given in Section 4.7.1 and the rate expression given in Equation 8.1 are used.

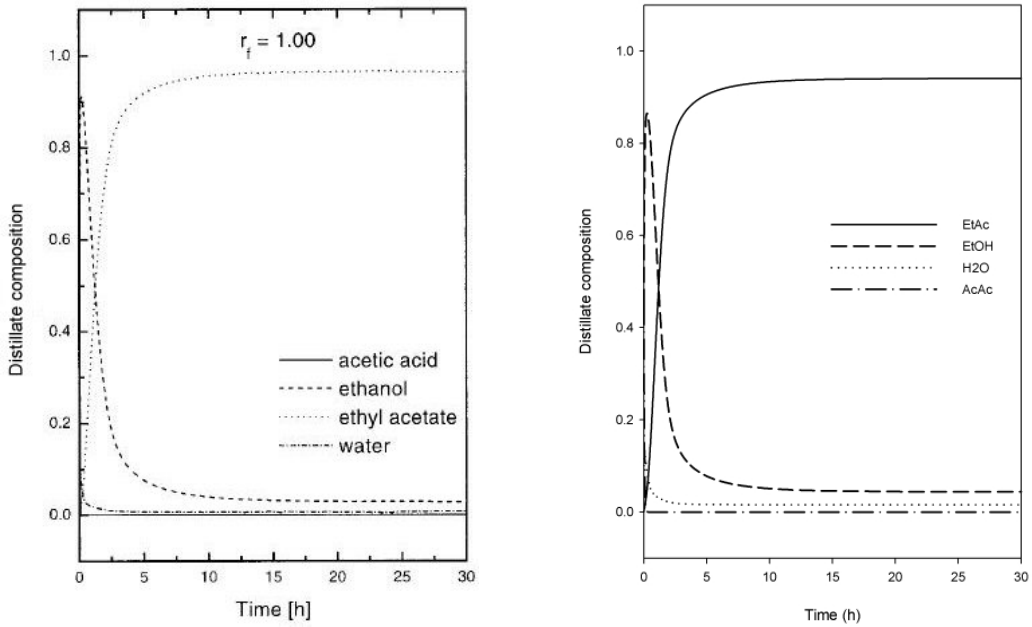
$$r\,[gmol/(L\,\min)] = k_1 c_1 c_2 - k_2 c_3 c_4 \tag{8.1}$$

where k1 = 4.76 x 10-4 and k2 = 1.63 x 10-4 and ci represents the concentration in terms of gmol/L for the i^{th} component.

The comparison of simulation studies at total reflux and at reflux ratio of 0.95 are given in Figures 8.1 and 8.2, respectively. As can be seen from the figure the results are almost the same. This indicates that the developed dynamic model in this study is as good as the previous models to represent this non-linear and complex problem of reactive batch distillation column behavior.

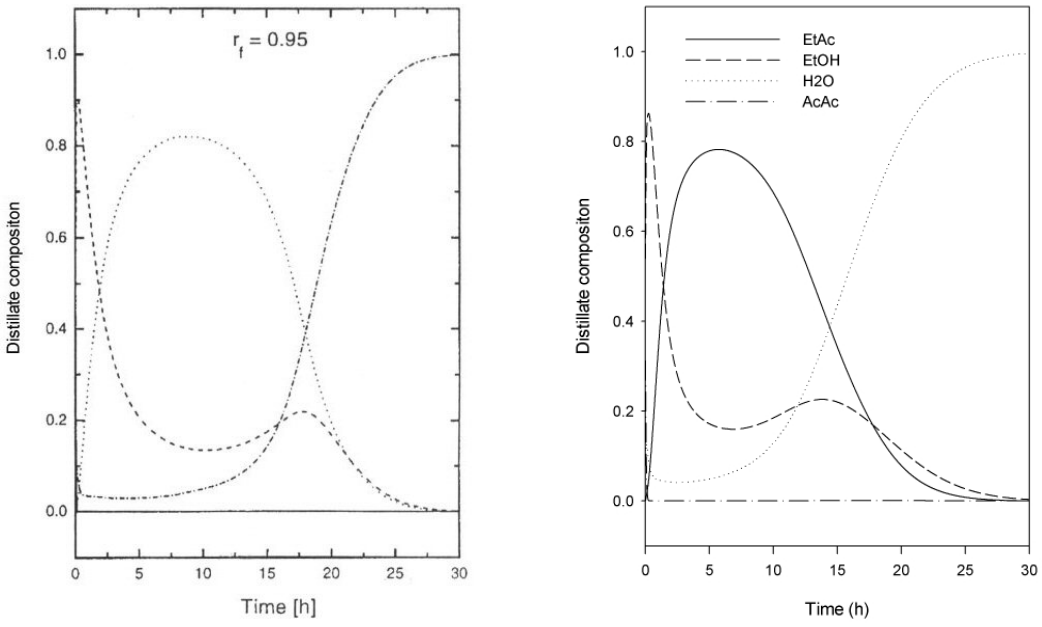
8.1.2 Experimental Results

After seeing that the simulation results are similar to that of the literature, experiments (Chapter 3) are performed in a lab-scale distillation column in order to check the model results with the experimental data. The experiments are done at total reflux for 8.5 hours until the first steady state is reached. After the steady state is reached, the operation is continued with an arbitrary reflux ratio of 5.72. The compositions of distillate and the reboiler data are collected with respect to time and are shown in Figure 8.3.



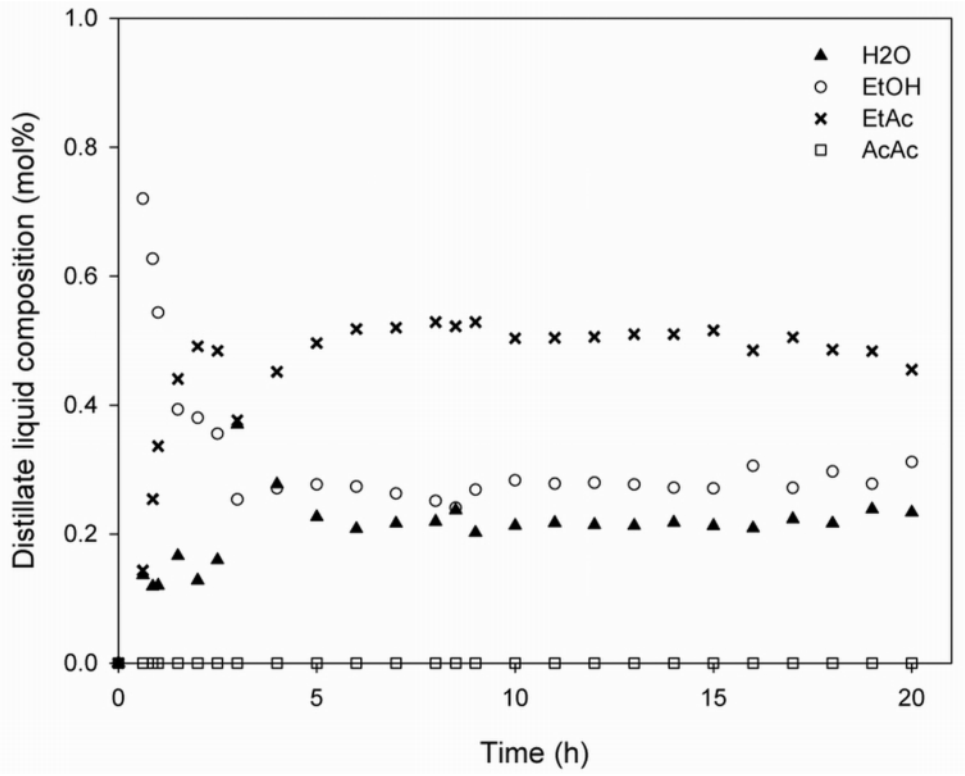
(a) (b)

Figure 8.1. Distillate Compositions at Total Reflux (a) Results of Monroy-Loperena and Alvarez-Ramirez (2000) (b) Results of the Simulation in This Study

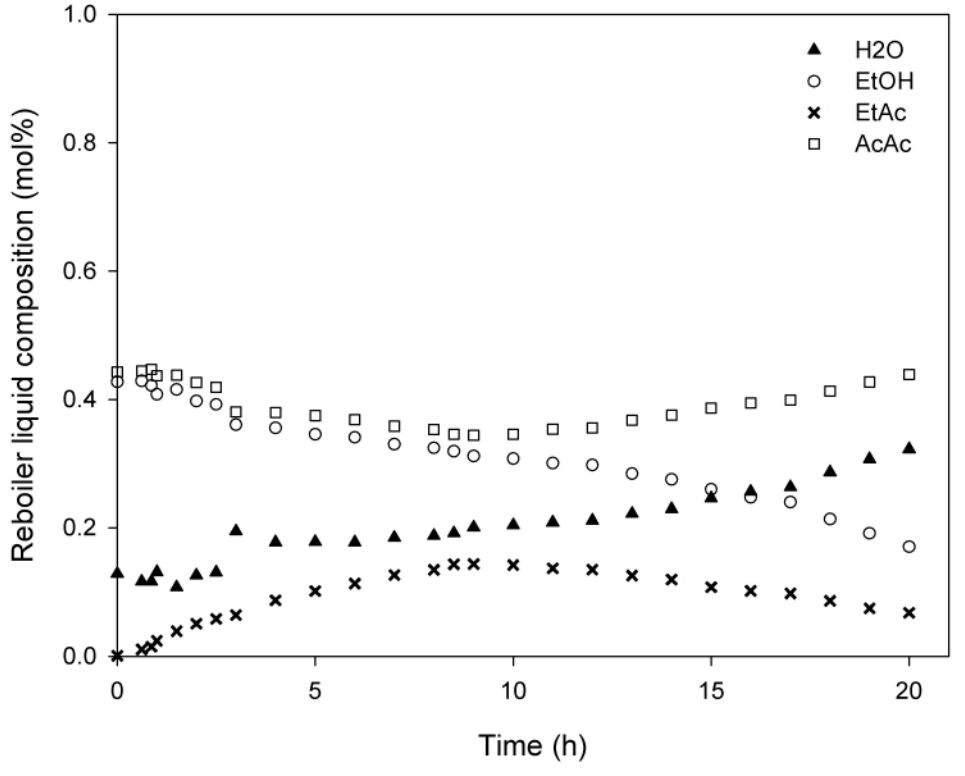


(a) (b)

Figure 8.2. Distillate Compositions at R=0.95 (a) Results of Monroy-Loperena and Alvarez-Ramirez (2000) (b) Results of the Simulation in This Study



(a)



(b)

Figure 8.3. Experimental Results

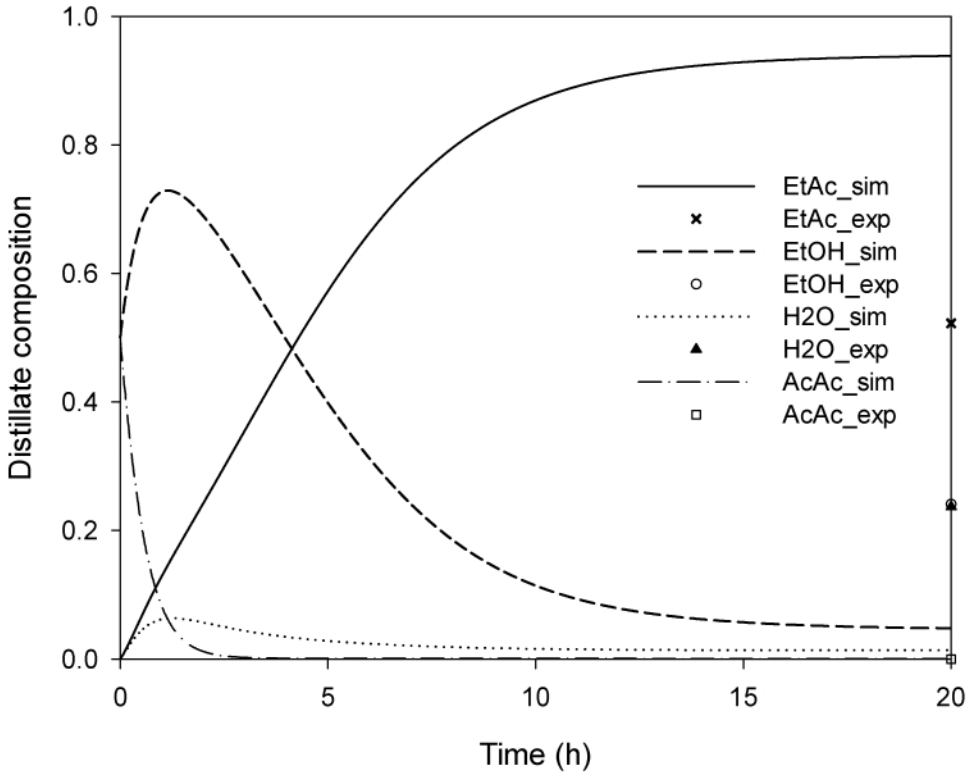
8.1.3 Comparison of Experimental Data and Simulation Results

There is a difference in the starting conditions of the experiments and simulation. As it was explained in Chapter 4, the simulation code is written by taking the concentrations across the column as same with the reboiler feed compositions. Therefore, although the trends of the profiles of the compositions of the components are similar, they cannot be compared up to the steady state point. Therefore, the dynamic comparison can only be done after the match between the two steady state results is obtained. The dynamic change of the compositions of the distillate and the reboiler of the experiment are compared at the given reflux ratio with that of simulation results using different VLE models and rate expressions as given in Table 8.2.

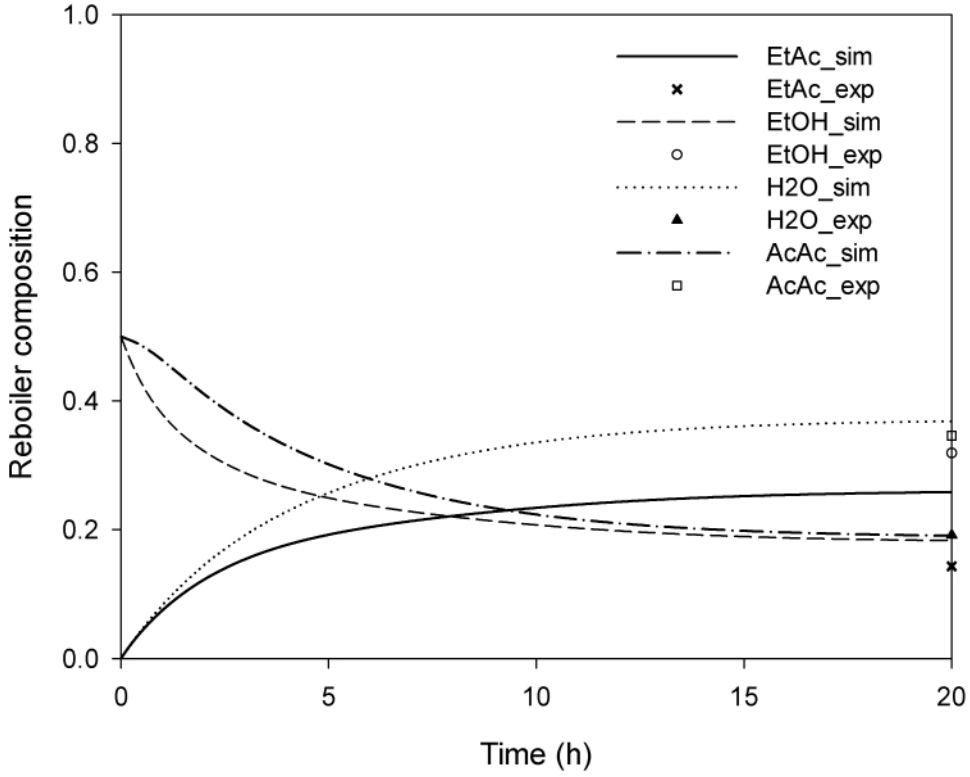
i) Model-I

The results of the simulation study which uses Model-I (Table 8.2) is compared with the experimental data both for distillate and reboiler compositions of the components at steady state reached at total reflux conditions in Figure 8.4. It can be seen from the figure that the steady state values (t = 20 hour) are too different from each other. Thus, there is no need to check the dynamic behavior with a finite reflux ratio when the steady state values are so different.

The comparison of the results of the theoretical studies found from literature with the results of the theoretical studies done revealed that, the model used is as good as others. However, the results of the previous studies taken from the literature are not in good agreement with the results of the experiments in terms of steady state composition values reached under total reflux conditions. The differences seen can be due to either VLE expression or the rate expression used. Therefore, studies are carried out to find the source of the error.



(a)



(b)

Figure 8.4. Experimental and Simulation Results (with Model-I) for Distillate and Reboiler Compositions at Total Reflux

ii) Model-II

In Model-II (Table 8.2), $\phi - \phi$ method with Peng-Robinson EOS and van der Waals mixing rule are used. The results of this simulation and the experiments are given in Figure 8.5 for total reflux operation. It is found that there are great differences in the attained steady state points. Thus, there is no need to check the dynamic trends. Therefore, another model must be checked.

iii) Model-III

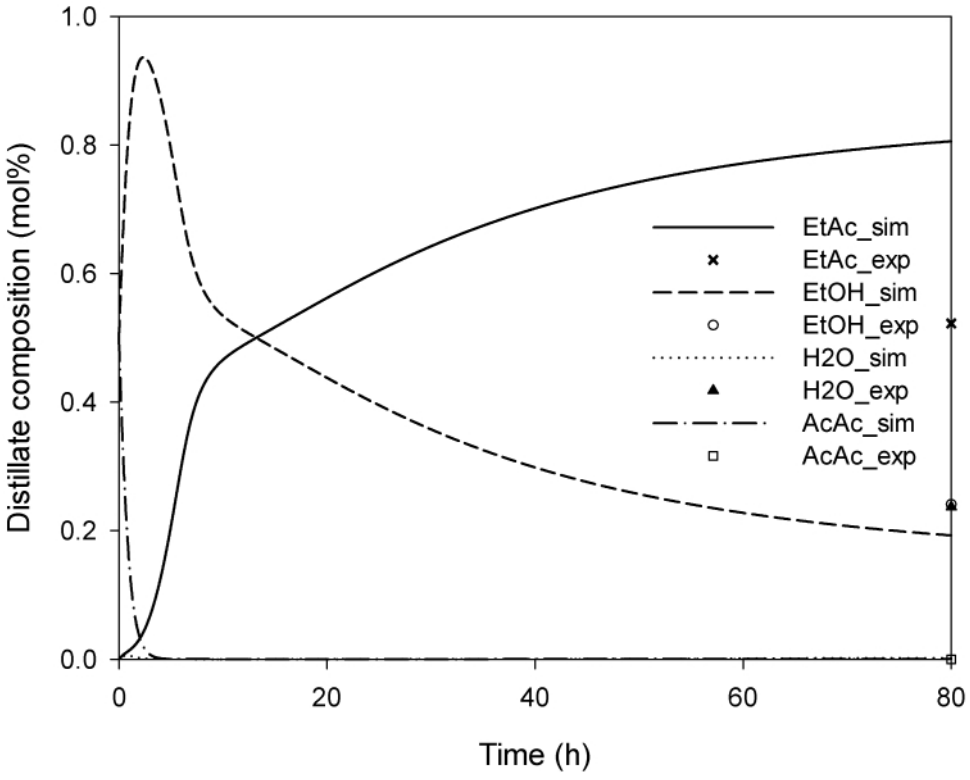
In Model-III, six different model approaches are tried. These are related to the use of different models for EOS, mixing rule and activity coefficient. These model approaches are given in Table 8.2.

Model-III-A

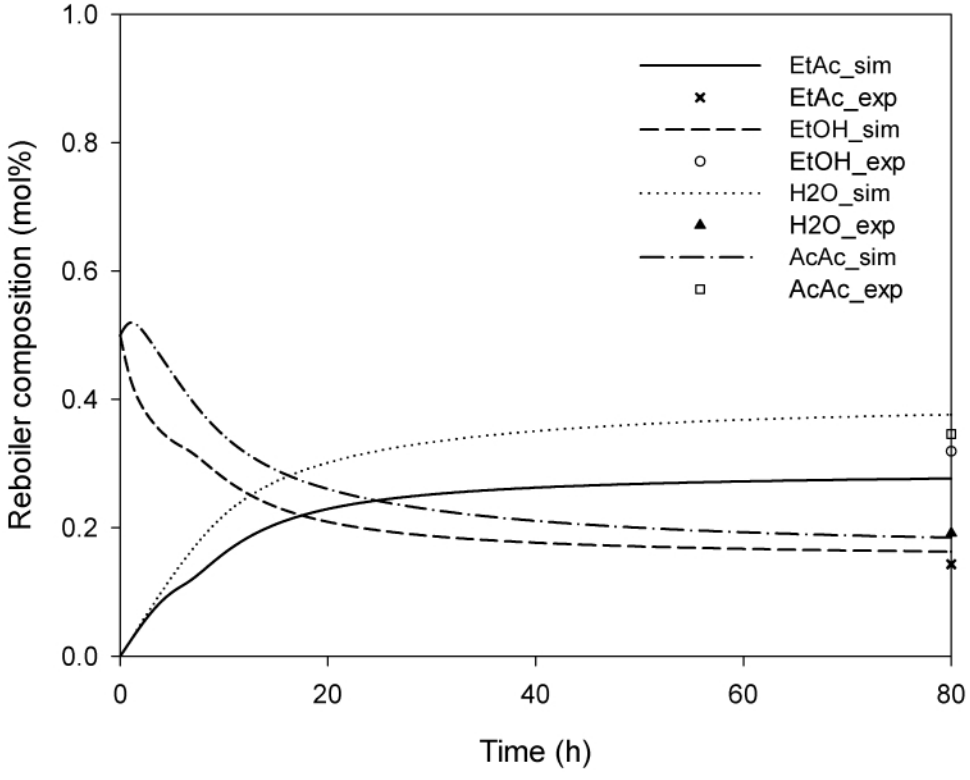
The dynamic results of the simulation and the experiments are given in Figure 8.6 for a constant reflux ratio of 5.72. The time at which steady state is reached at total reflux is shown as zero. It is found that the steady state values reached at total reflux are in a better agreement with experimental data compared to Model-I and Model-II. The checks that will be done from this point on will be based on the data taken after steady state is reached. It can be seen from Figure 8.6 that the results are somewhat improved especially for the reboiler compositions. However, the dynamic trends for distillate compositions deteriorate and it can be understood that either the PR EOS or HVO mixing rule is not appropriate for this system.

Model-III-B

In Model-III-B, EOS is changed to PRSV with the same mixing rule and the activity coefficient model. The performance of the system with this model is given in Figure 8.7 and the IAE scores with this VLE model are given in Table 8.2. As it can be seen, distillate compositions are much better than in the Model-III-A which uses PR. However, the reboiler compositions become worse in this case and therefore this model also does not give satisfactory results.

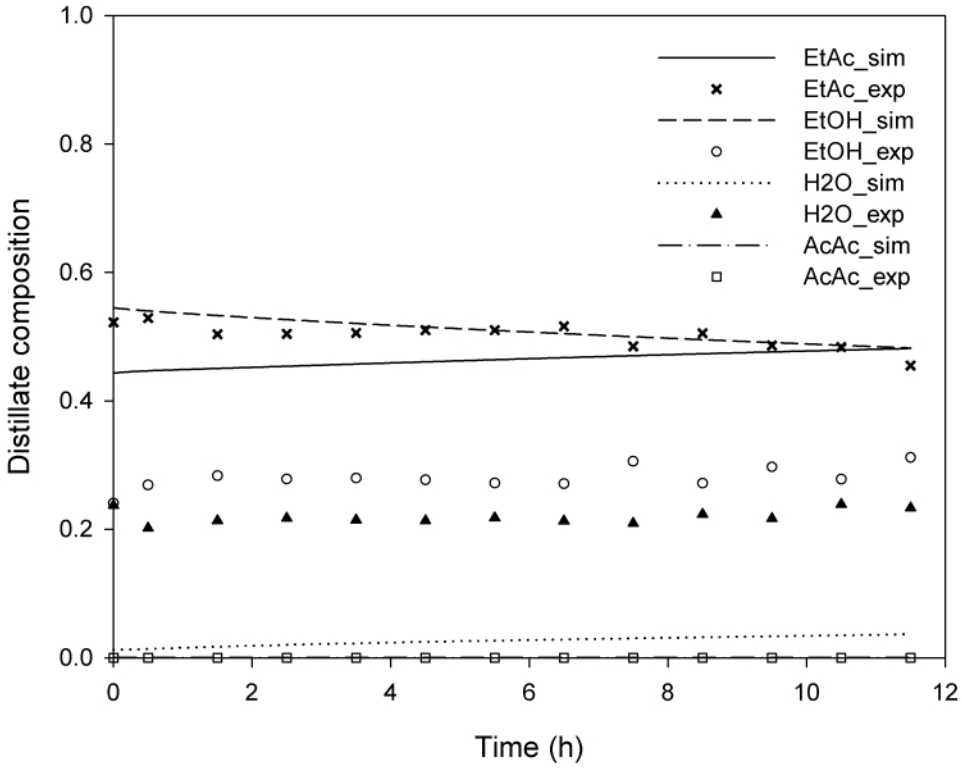


(a)

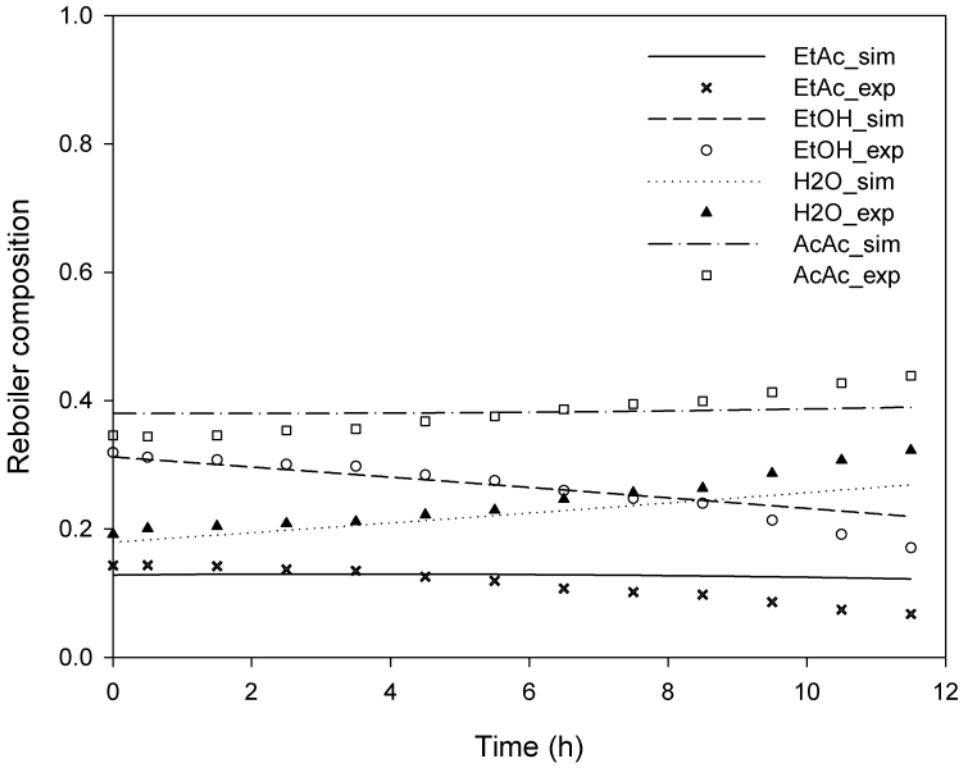


(b)

Figure 8.5. Experimental and Simulation (with Model-II) Results for Distillate and Reboiler Compositions at Total Reflux

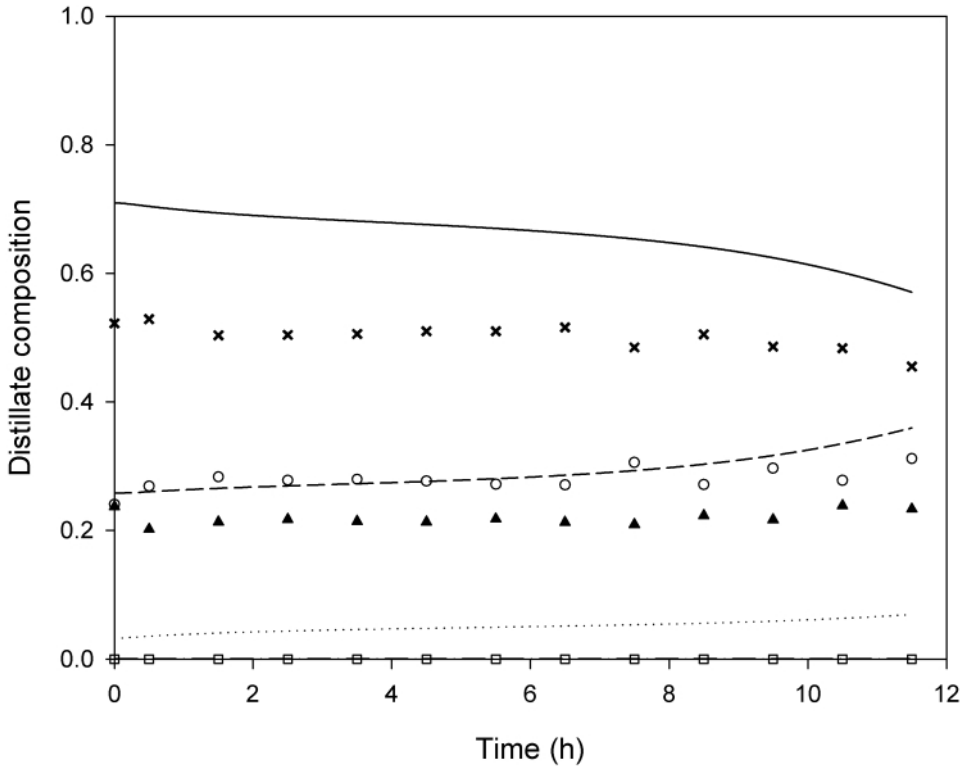


(a)

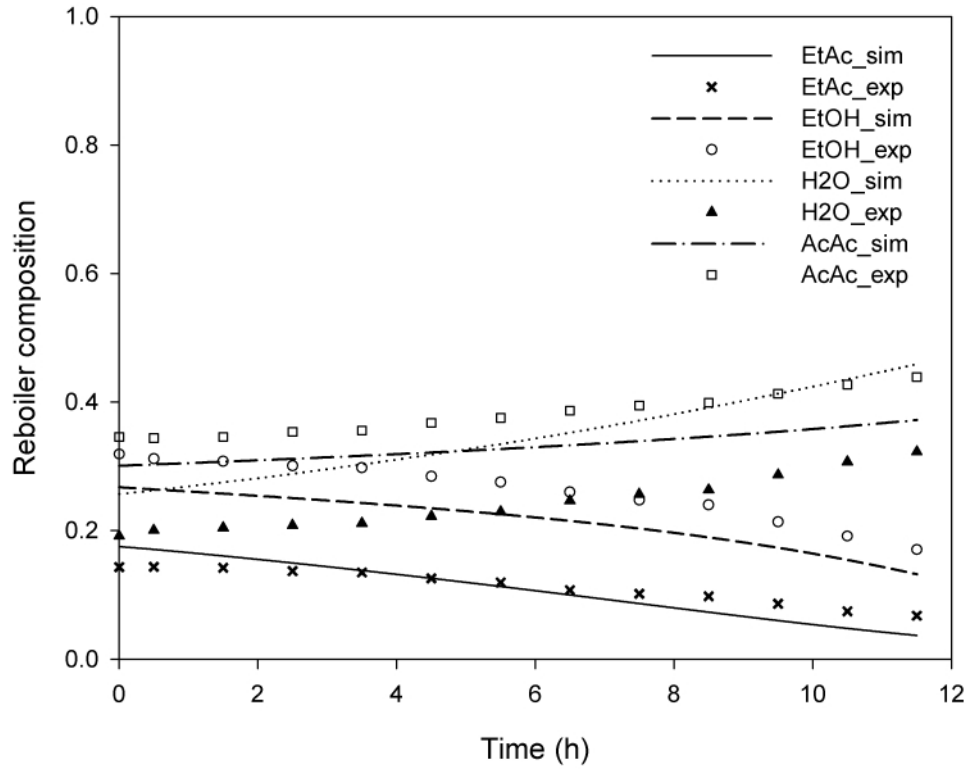


(b)

Figure 8.6. Experimental and Simulation (with Model-III-A) Results for Distillate and Reboiler Compositions at Reflux Ratio of 5.72 after Total Reflux



(a)



(b)

Figure 8.7. Experimental and Simulation (with Model-III-B) Results for Distillate and Reboiler Compositions at Reflux Ratio of 5.72 after Total Reflux

Model-III-C

In this model, mixing rule is changed to HVOS and it is used together with PRSV EOS and NRTL activity coefficient model. The results are given in Figure 8.8 and Table 8.2. The results for both distillate and reboiler compositions are improved significantly with this thermodynamic model.

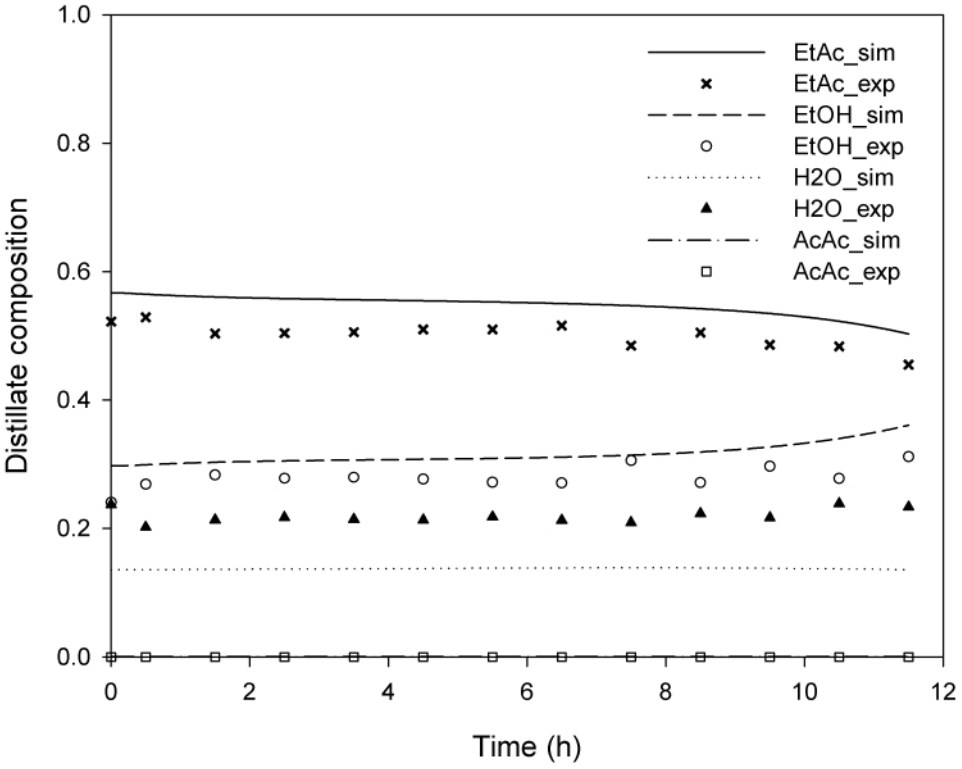
Model-III-D and Model-II-E

In order to see the effects of different activity coefficient models, Wilson and UNIQUAC activity coefficient models are used in EOS-G^{ex} approach. The distillate and reboiler liquid compositions with Wilson model (Model-III-D) and UNIQUAC model (Model-III-E) are given in Figures 8.9 and 8.10, respectively. The IAE scores with these models are given in Table 8.2. It can be seen from the figures that while the NRTL and Wilson models give similar results, UNIQUAC performs poorly. Since NRTL model gives slightly better results than Wilson model, NRTL model is selected to be the most proper activity coefficient model for this system. This activity coefficient model will be used also in Model-IV.

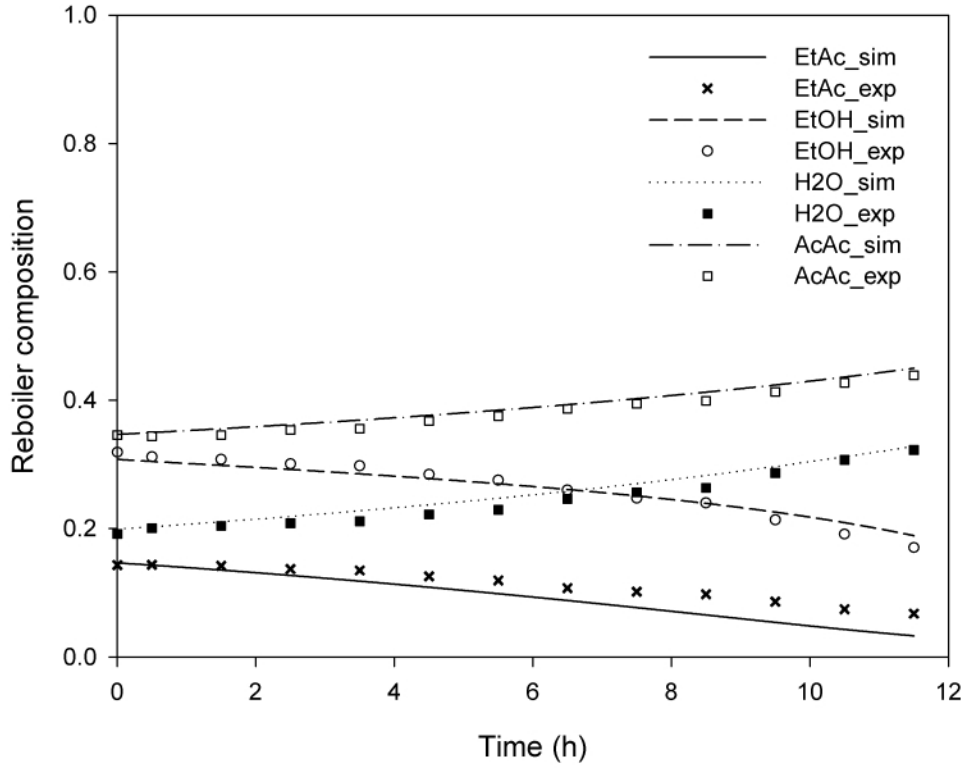
iv) Model-IV

In this model, NRTL activity coefficient model is used for the liquid phase and PRSV-EOS with the van der Waals mixing rule (Model-IV-A) and PR-EOS (Model-IV-B) is used for the vapor phase.

It can be seen from Figure 8.11 and from the IAE scores given in Table 8.2, that the distillate compositions are improved compared to the Model-III. Although the reboiler compositions become a little worse, they are in an acceptable range. Unlike Model-III, PR-EOS also gives similar results with PRSV in Model-IV as can be seen in Figure 8.12 and IAE scores given in Table 8.2.

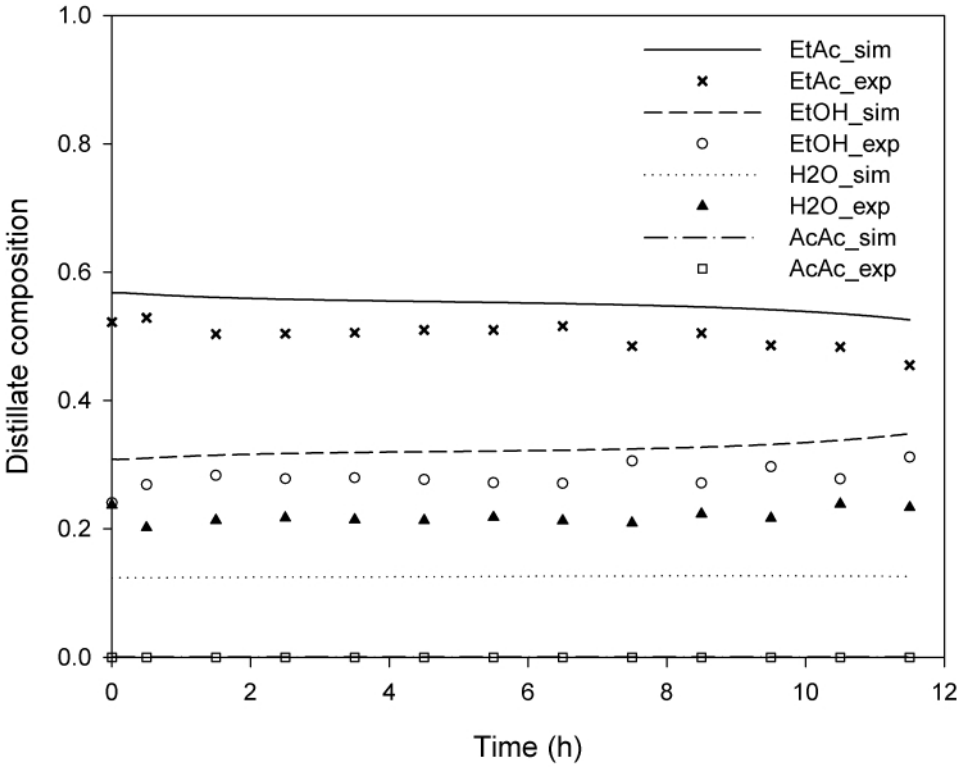


(a)

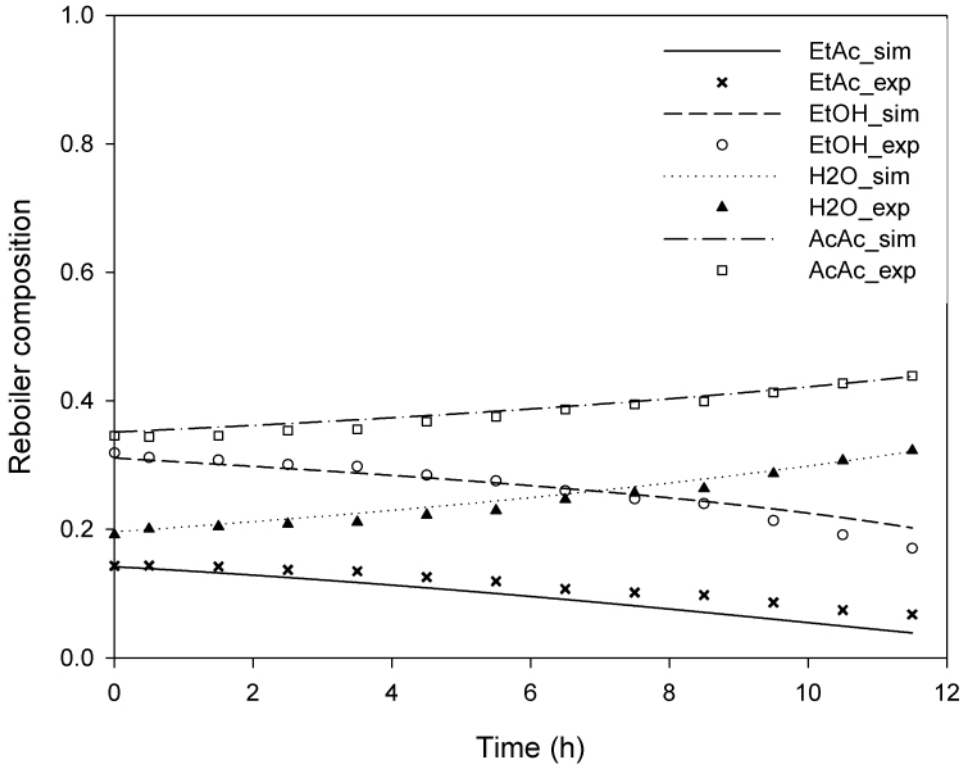


(b)

Figure 8.8. Experimental and Simulation (with Model-III-C) Results for Distillate and Reboiler Compositions at Reflux Ratio of 5.72 after Total Reflux

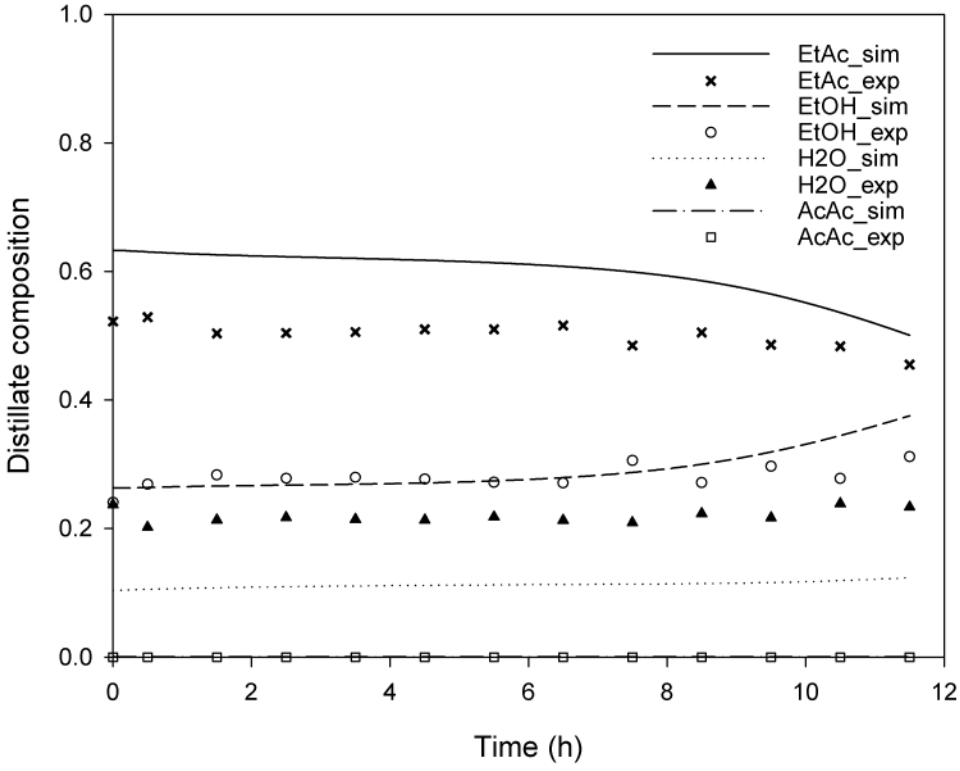


(a)

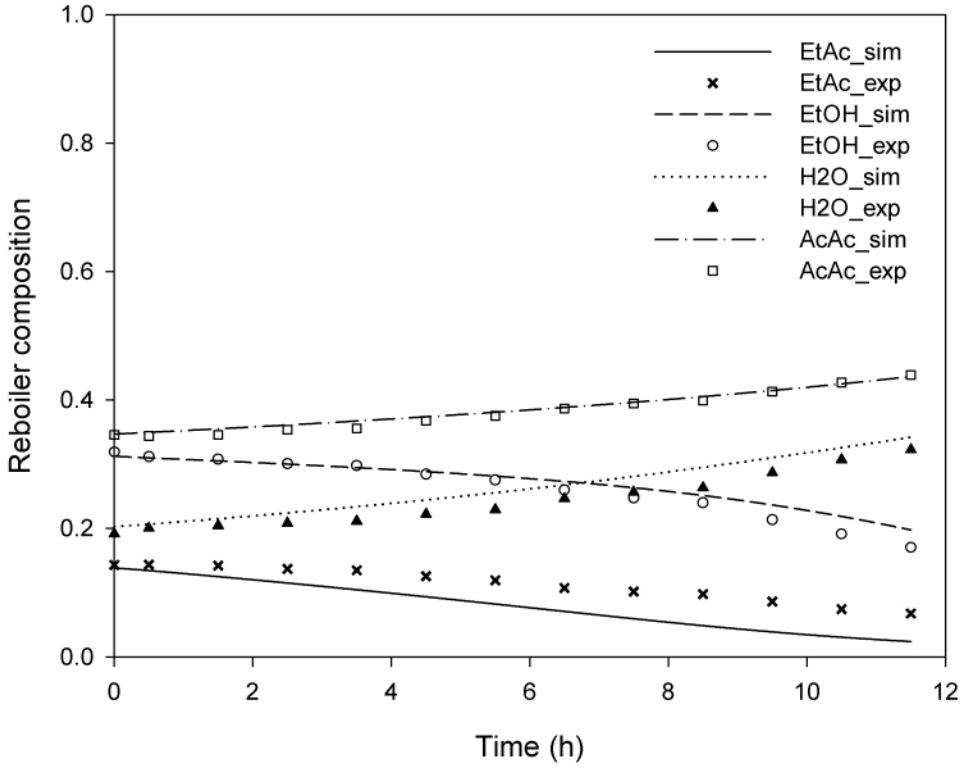


(b)

Figure 8.9. Experimental and Simulation (with Model-III-D) Results for Distillate and Reboiler Compositions at Reflux Ratio of 5.72 after Total Reflux

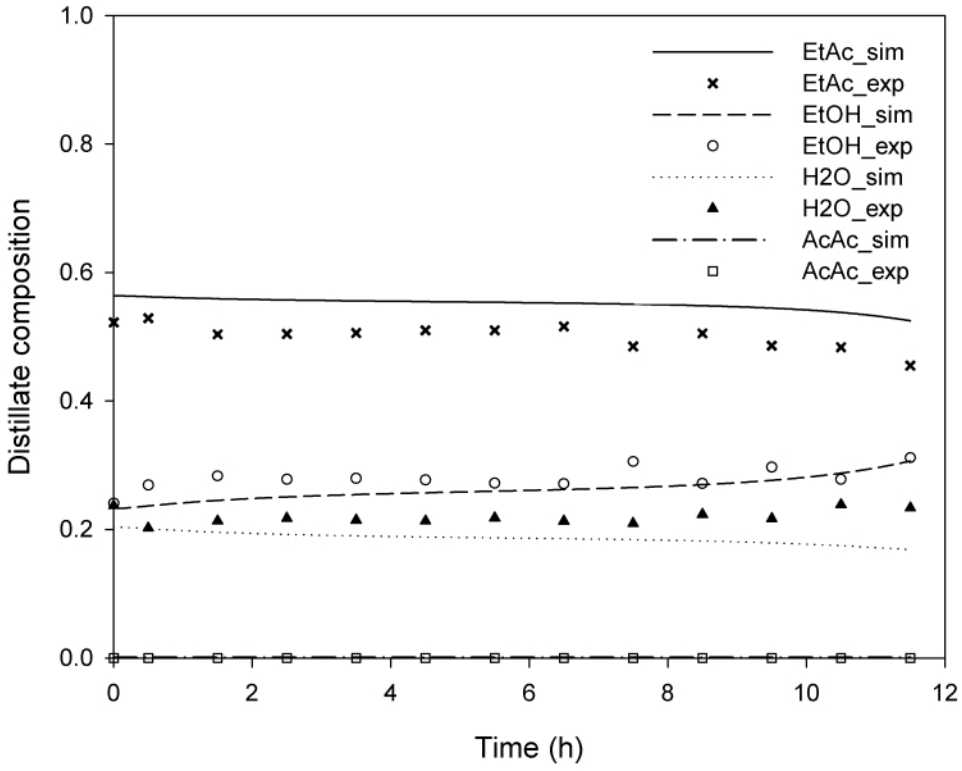


(a)

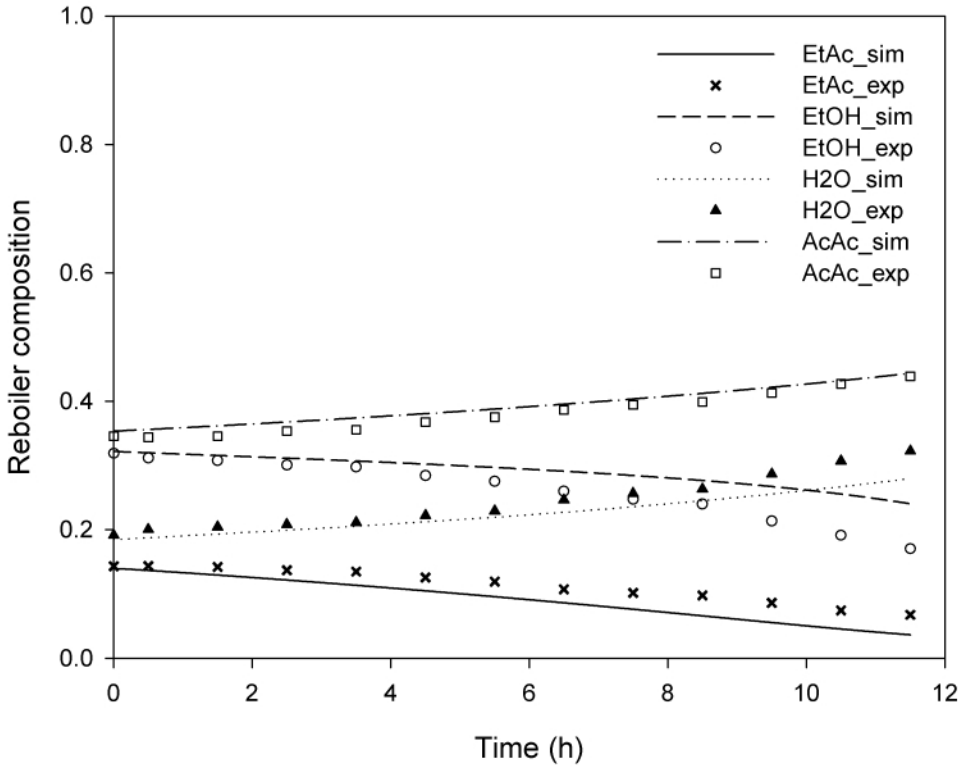


(b)

Figure 8.10 Experimental and Simulation (with Model-III-E) Results for Distillate and Reboiler Compositions at Reflux Ratio of 5.72 after Total Reflux

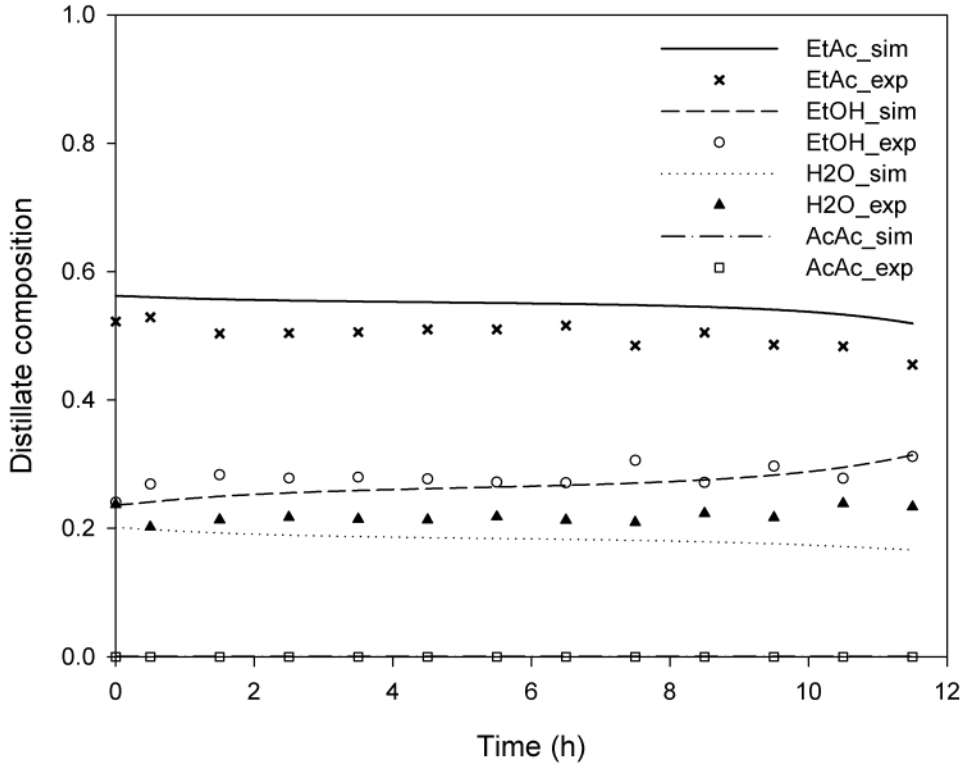


(a)

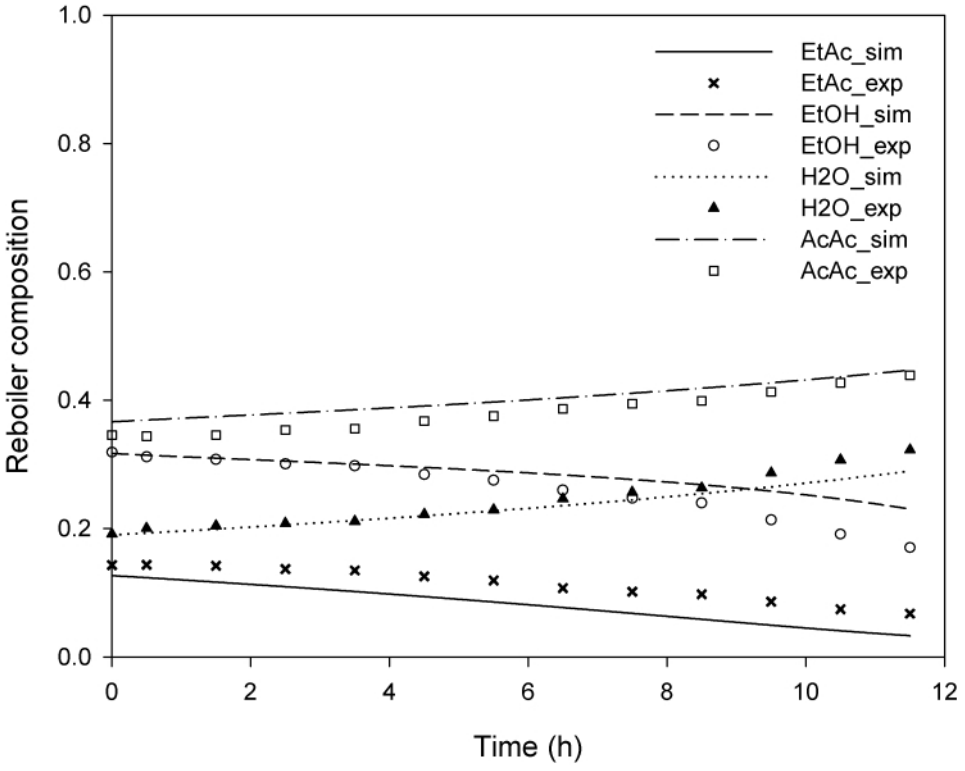


(b)

Figure 8.11. Experimental and Simulation (with Model-IV-A) Results for Distillate and Reboiler Compositions at Reflux Ratio of 5.72 after Total Reflux



(a)



(b)

Figure 8.12. Experimental and Simulation (with Model-IV-B) Results for Distillate and Reboiler Compositions at Reflux Ratio of 5.72 after Total Reflux

v) Summary of Thermodynamic Models

The summary of the descriptions and the IAE scores for different thermodynamic models that are used in the study are given in Table 8.2. Comparing the IAE scores for distillate and reboiler compositions, it can be concluded that, Model-IV-A which uses the traditional $\gamma - \phi$ approach with NRTL activity coefficient model for the liquid phase and the PRSV equation of state for the vapor phase, is the best model for the quaternary ethanol-acetic acid-ethyl acetate-water system. Nevertheless, Model-IV-B which uses the traditional $\gamma - \phi$ approach with NRTL activity coefficient model for the liquid phase and the PR-EOS for the vapor phase, which is simple to use also gives similar result with a very close IAE score. Thus, both methods can be recommended to be used in the simulation of ethanol esterification reaction with acetic acid in a batch distillation column.

vi) VLE Data Check

VLE experimental data for the system ethanol-acetic acid-ethyl acetate-water is available in the literature. Therefore, the best selected thermodynamic model performs very well when used in the reactive distillation column modeling, which is tested with the experimental VLE data taken from the literature. Table 8.3-b gives the equilibrium vapor compositions of the components (y_i) and the equilibrium temperatures, obtained from the simulation and from the study of Kang et al. (1992), for the liquid compositions (x_i) given in Table 8.3-a. From the comparison of the VLE Model-IV-A with the experimental VLE data, it can be said that this VLE model gives satisfactorily accurate results.

Table 8.2 Summary of Thermodynamic Models

Model	*Description*	*IAE Scores*		
		Distillate	*Reboiler*	*Overall*
Model-I	VLE data from literature			
Model-II	$\phi-\phi$ method with PR + van der Waals			
Model-III-A	EOS-G^{ex} method with PR + HVO + NRTL	6.080	1.049	7.129
Model-III-B	EOS-G^{ex} method with PRSV + HVO + NRTL	4.514	2.805	7.320
Model-III-C	EOS-G^{ex} method with PRSV + HVOS + NRTL	2.131	0.621	2.751
Model-III-D	EOS-G^{ex} method with PRSV + HVOS + Wilson	2.437	0.552	2.989
Model-III-E	EOS-G^{ex} method with PRSV + HVOS + UNIQUAC	2.915	0.877	3.791
Model-IV-A	$\gamma-\phi$ method with PRSV + van der Waals + NRTL	1.321	1.026	2.347
Model-IV-B	$\gamma-\phi$ method with PR + van der Waals + NRTL	1.279	1.072	2.351

Table 8.3 Comparison of Thermodynamic Model with Experimental VLE Data

(a) Liquid Compositions

X_{EtAc}	X_{EtOH}	X_{H2O}	X_{AcAc}
0.1164	0.0161	0.396	0.4715
0.3943	0.052	0.2155	0.3382
0.2498	0.6636	0.0786	0.008
0.0291	0.003	0.4276	0.5403
0.0478	0.0004	0.0731	0.8787

(b) Equilibrium Vapor Compositions and Temperatures

Experimental					*Theoretical*					*Absolute Error*				
y_{EtAc}	y_{EtOH}	y_{H2O}	y_{AcAc}	*T (K)*	y_{EtAc}	y_{EtOH}	y_{H2O}	y_{AcAc}	*T (K)*	y_{EtAc}	y_{EtOH}	y_{H2O}	y_{AcAc}	*T (K)*
0.3695	0.0312	0.4036	0.1957	366.85	0.353	0.0291	0.4207	0.1972	363.18	0.02	0.00	0.02	0.00	3.67
0.6145	0.0636	0.2456	0.0763	356.45	0.6238	0.0639	0.2262	0.0861	352.21	0.01	0.00	0.02	0.01	4.24
0.3544	0.5787	0.0666	0.0003	346.55	0.3632	0.5588	0.0768	0.0012	345.75	0.01	0.02	0.01	0.00	0.80
0.1213	0.0074	0.5126	0.3587	375.85	0.1192	0.0072	0.5553	0.32	371.325	0.00	0.00	0.04	0.04	4.52
0.1206	0.0013	0.1284	0.7497	384.55	0.1713	0.0011	0.1671	0.6605	380.81	0.05	0.00	0.04	0.09	3.74

8.2 Nonlinear Optimization

The objective function in the nonlinear optimization of the ethanol esterification system is selected as the maximization of the capacity factor as stated in Chapter 5. With this objective function, the maximum product amount at specified purities in minimum time is aimed to be obtained. In this optimization problem, the reflux ratio for each product-cut and slop-cut operations and their durations are the optimization variables. The total batch time, t_F is a dependent variable in the optimization algorithm. A piecewise constant reflux ratio profile is used since the reflux ratio is kept constant during each of the product-cut and slop-cut operations. The optimum reflux ratio value during the operation for each product-cut and slop-cut and the durations of these operations are obtained from the optimization algorithm.

For the solution of the optimization problem, *fmincon* algorithm, available in *MATLAB Optimization Toolbox* is used. *fmincon* attempts to find a constrained mimimum of a scalar function of several variables starting at an initial estimate. However, while using *fmincon*, the function to be mimimized and the constraints must both be continuous. In the optimization of the nonlinear problem in this study, fmincon also gives only local solutions. For this reason, the solution depends mostly on the initial estimate given. Therefore, the optimization run is done with different initial estimates within the range of constraints and a global optimum is found for each product-cut and slop-cut operations. The whole reactive batch distillation column model is simulated for each case with the specified desired purities in product-cut tanks. The reflux ratio which makes the capacity factor maximum is selected as the optimum value for that period of operation (i.e., 1^{st} product-cut, 1^{st} slop-cut, 2^{nd} product-cut, and so on).

The system studied is a four component system of ethanol-acetic acid-water-ethyl acetate, therefore, three slop-cuts are obtained in the operation of reactive batch distillation column. Table 8.4 shows the optimal reflux ratio profile obtained in order to give three product-cuts with the desired purities of 0.52, 0.50, and 0.65 from the reactive mixture of ethyl acetate, ethanol, water, and acetic acid with the initial compositions of 0.0, 0.5, 0.0, and 0.5. The heaviest component in the system, acetic acid, is collected in the reboiler with a desired purity of 0.999. With this optimal reflux ratio policy, 77.2% of the feed can be collected in product-cut tanks since the total amount of products collected in product cut tanks is 240.61 moles and that in slop-cut tanks is 71.06 moles.

The purities of ethyl acetate and ethanol in product-cut tanks (52% and 50%, respectively) are low despite the use of reactive distillation column. This is mainly due to the close boiling points of these two components as stated in Chapter 5. Another reason is that acetic acid is the highest component in the system and it remains mostly in the bottom of the column. Therefore, the production of ethyl acetate is limited by the lack of reactant acetic acid. Similar low values are also reported in the literature (Wajge and Reklaitis, 1999; Chang and Seader, 1988). In order to make use of the obtained slop-cuts, they can be further reprocessed through next operation. There are studies in the literature considering this issue like Bock et al. (1997) and Tang et al. (2003). The system under study is based on a single batch operation.

Table 8.4 Optimum Reflux Ratio Profile

Optimum Reflux Profile	
Time Interval (hour)	Reflux Ratio
0 – 9.15	Total reflux
9.15 – 27.75	8.21
27.75 – 29.43	2.08
29.43 – 32.33	3.26
32.33 – 35.12	1.94
35.12 – 36.21	3.98
36.21 – 38.17	21.78

8.3 Artificial Neural Network State Estimator

For the inferential control of the distillation column under study, in order to operate the system effectively and efficiently, the time duration of the periods for each reflux ratio must be faced to limit the product compositions at their maximum purities with the help of optimization program. However, this necessitates the information about the composition values of the products. Thus, composition measurements at distillate and reboiler levels must be done. However, there are some drawbacks for these measurements (Chapter 6) and the compositions must be estimated. Therefore, temperature measurements

throughout the column are used to estimate the distillate compositions by the use of ANN which is frequently used in the industry also.

The dynamic system under study is highly nonlinear and it is observed that the composition profiles in the column changes significantly with different reflux ratio values. Thus, forming only one neural network and training it with input-output data obtained for various reflux ratio values is not reasonable. Therefore, a separate network is formed for each reflux ratio values and the value of the reflux ratio is also given to the estimator as input. The output of the ANN estimator (distillate compositions) that corresponds to its inputs (temperatures and reflux ratio) is found by interpolating the two networks, in which reflux ratio falls into. For example, when the reflux ratio value, R is between R_1 and R_2, the output of the estimator is found as given in Equations 8.2 and 8.3 where d_i represents the distance between R and R_i.

$$\begin{aligned} d_1 &= \left|R - R_1\right| \\ d_2 &= \left|R_2 - R\right| \end{aligned} \tag{8.2}$$

$$estimator\ output = \frac{d_1}{d_1 + d_2} \times output\ of\ \ ANN_2 + \frac{d_2}{d_1 + d_2} \times output\ of\ \ ANN_1 \tag{8.3}$$

Each ANN is designed considering the network's architecture, normalization issue, and network performance with respect to verification and generalization tests.

8.3.1 Selection of Measurement Points

Considering the discussions given in Chapter 6, four temperature measurement locations equally spaced throughout the column; the reboiler, 2nd tray, 5th tray, and the top tray (8th tray); are used in the estimation by neural networks.

8.3.2 Range of Variables

The neural network is trained with temperature and composition data generated by the help of simulation, utilizing unsteady state responses for different reflux ratio values. Neural networks cannot make accurate estimations if the operating

input/output data are outside their training data range. Therefore, the training data set should include sufficient operational range within the maximum and minimum values for both input/output variables.

In this study, the reflux ratio (L/V) is changed between 0.5 and 1.0 for the constant reflux ratio period after the steady state is reached for total reflux. The lower values of the reflux ratio are found to be not suitable for separation in distillation. In order to extend the training range, the reflux ratio profiles, which are in the range of $\pm 20\%$ of the optimal reflux profile, are also used in the training of the network. The 13 values of reflux ratio used in training are given in Table 8.5.

Table 8.5 Reflux Ratio Values Used in ANN Training

Reflux Ratio Trend After Total Reflux	Values of Reflux Ratio (L/V)
Constant	0.5, 0.6, 0.7, 0.8, 0.85, 0.9, 0.95, 1.0
Variable with respect to a profile	$R_{optimal}$ $\pm 10\%$ $R_{optimal}$ $\pm 20\%$ $R_{optimal}$

In the training of the ANN, back-propagation training algorithm is used which is simple, easy to apply and successful in application. The training of the ANN is done by using the dynamic data which are collected using the values of reflux ratios given in Table 8.5 utilizing the MATLAB simulation program.

8.3.3 ANN Architecture

Considering the discussions given in Chapter 6, an Elman Network is used in this study. Since four composition values for the four components in the distillate are desired to be taken as outputs from the neural network, it must have four neurons in the output layer. Two hidden layers, with 20 neurons in the first hidden layer and 34 neurons in the second hidden layer, are used in the network

structure. The frequently preferred transfer functions in the network structure, i.e. *tan-sigmoid* transfer functions for the hidden layers and *purelin* transfer function for the output layer, are used.

8.3.4 Normalization

Neural networks require the same order of magnitude for their input and output data. If the input and the output variables are not of the same order of magnitude, some variables may appear to have more significance than they actually do. The training algorithm has to compensate for order-of-magnitude differences by adjusting the network weights, which is not very effective in many of the training algorithms such as backpropagation algorithm. For this reason, the collected training data are needed to be scaled. Each input and output parameter, p, is normalized to the range [-1….1] as p_n according to Equation 8.4 before being used in the neural network. The maximum and minimum values of data parameter p are p_{max} and p_{min}. Any future input-output data are normalized by the same method, and the network output is then converted back to its original values.

$$p_n = \frac{(1-(-1))}{(p_{\max} - p_{\min})} \times (p - p_{\min}) + (-1) = \frac{2(p - p_{\min})}{p_{\max} - p_{\min}} - 1 \qquad (8.4)$$

8.3.5 Estimator Performance

An important aspect of developing neural networks is determining how well the network performs once training is completed. Checking the performance of a trained network involves two main criteria:

- how well the neural network recalls the output vector from data sets used to train the network (called the *verification step*);

- how well the network predicts responses from data sets that were not used in the training phase (called the *recall* or *generalization step*).

After training the neural network, the ANN estimator performance must be found using the model for both the verification and the generalization tests.

Verification Tests

In the verification step, the network's performance is evaluated utilizing the specific initial input used in training. Thus, the previously used input data pattern is introduced to the trained network. The network then attempts to predict the corresponding output. In Figures 8.13 - 8.15, the responses of the distillate compositions for reflux ratios (L/V) of 0.7, 0.9 and for the optimum reflux ratio profile are shown respectively. It can be seen that, the network output will differ only slightly from the actual output data. It must be noticed that, in testing the network, the weight factors of ANN are not changed, they are frozen at their last values when training is ceased.

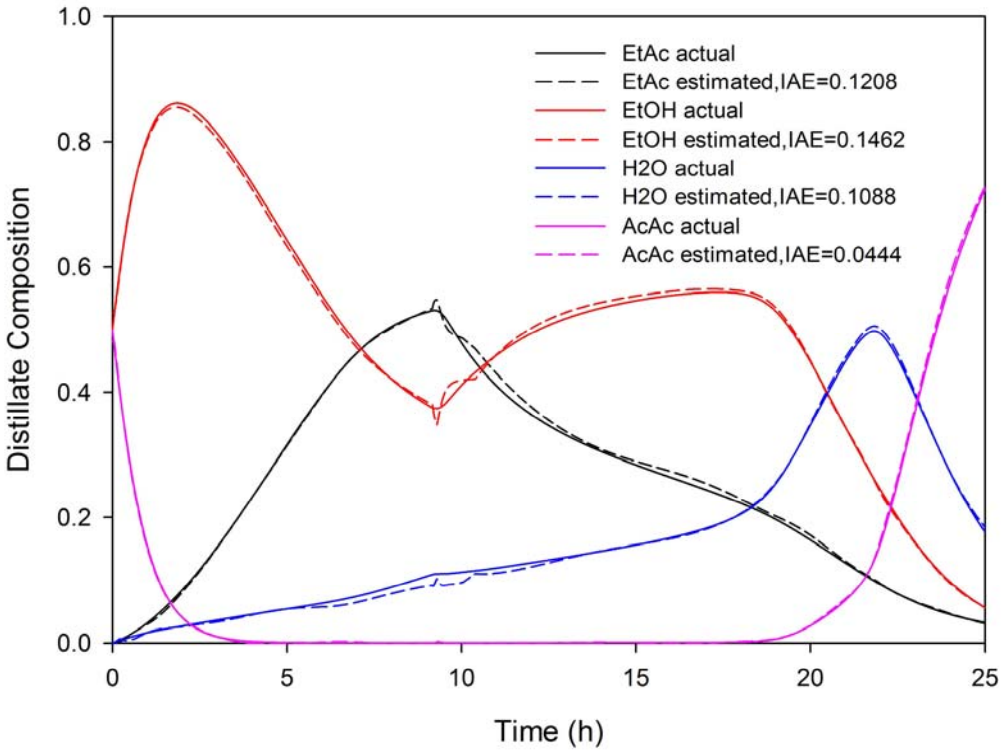


Figure 8.13. Actual and Estimated Distillate Compositions with Total Reflux Followed by a Reflux Ratio of 0.7

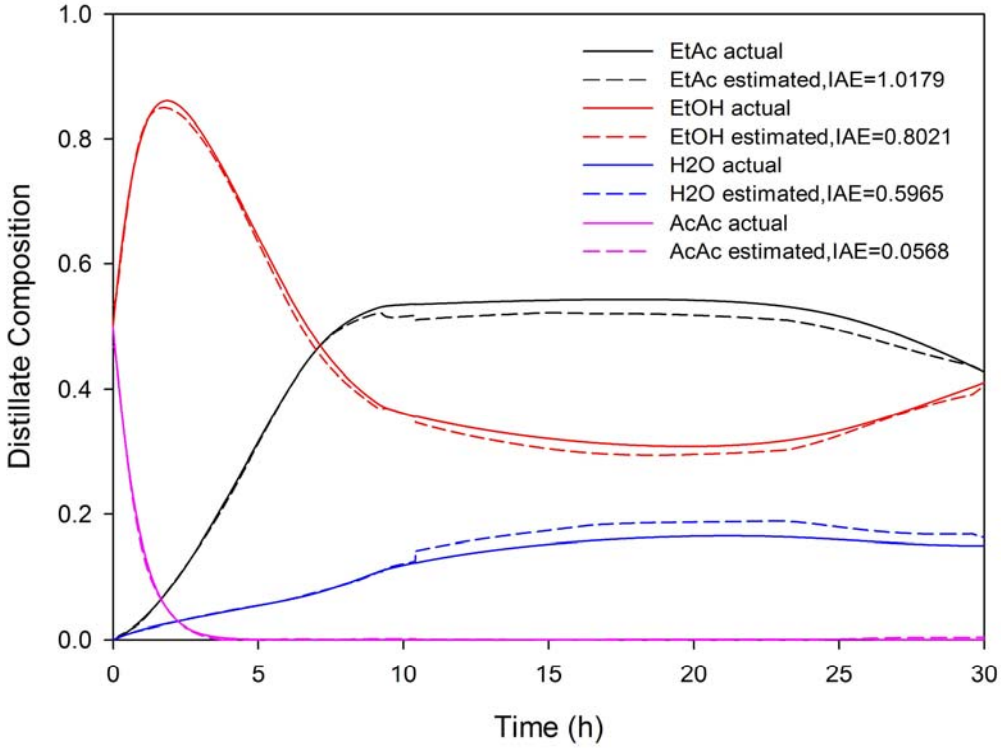


Figure 8.14. Actual and Estimated Distillate Compositions with Total Reflux Followed by a Reflux Ratio of 0.9

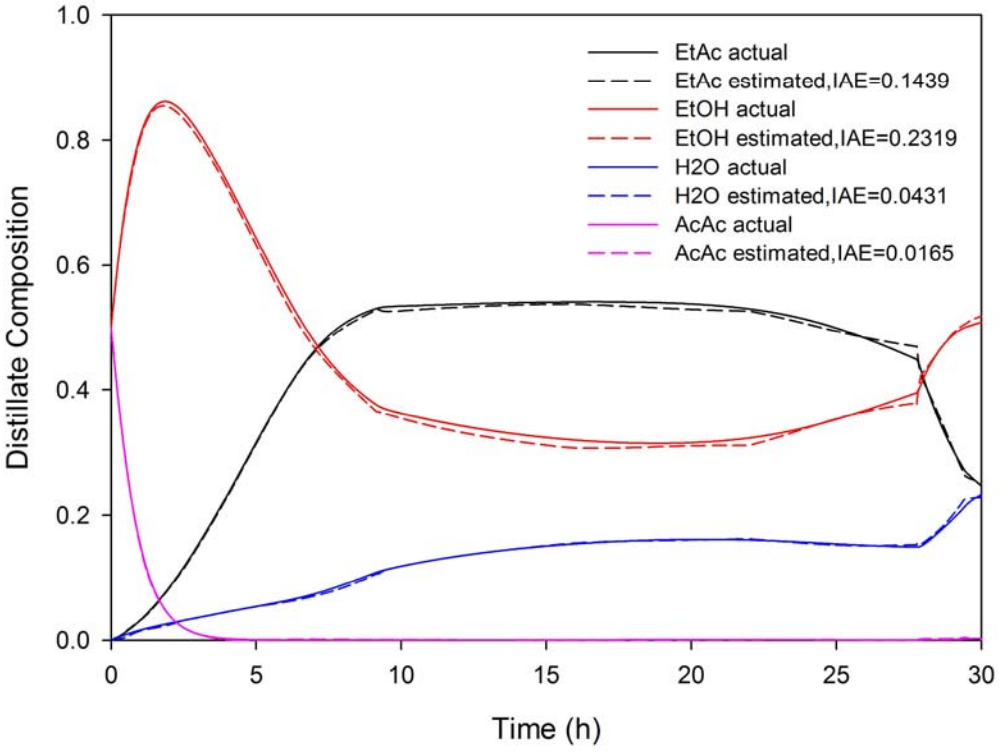


Figure 8.15. Actual and Estimated Distillate Compositions with Optimal Reflux Profile

Generalization Tests

Recall or generalization testing is conducted in the same manner as verification testing. However, in this case, input data with which the ANN is not trained, but which is in the range of the trained data set, is used. In this testing, the ANN measures how well the network can generalize what it has learned, and can form rules to make decisions about the data it has not previously learned. In the generalization step, new input patterns (whose results are known to us, but not to the network) are fed to the trained network. The responses of the column distillate compositions for a reflux ratio (L/V) of 0.75, 0.83, and 0.895 are shown in Figures 8.16 - 8.18, respectively. Furthermore, the responses of the column to a 10% increase and to a 5% decrease in the optimal reflux ratio profile are given in Figure 8.19 and 8.20, respectively.

It can be seen from the figures that, the network estimates the outputs by interpolation with a good accuracy. Observing the success of the designed ANN after verification and generalization tests, it is used in the control system as the next step.

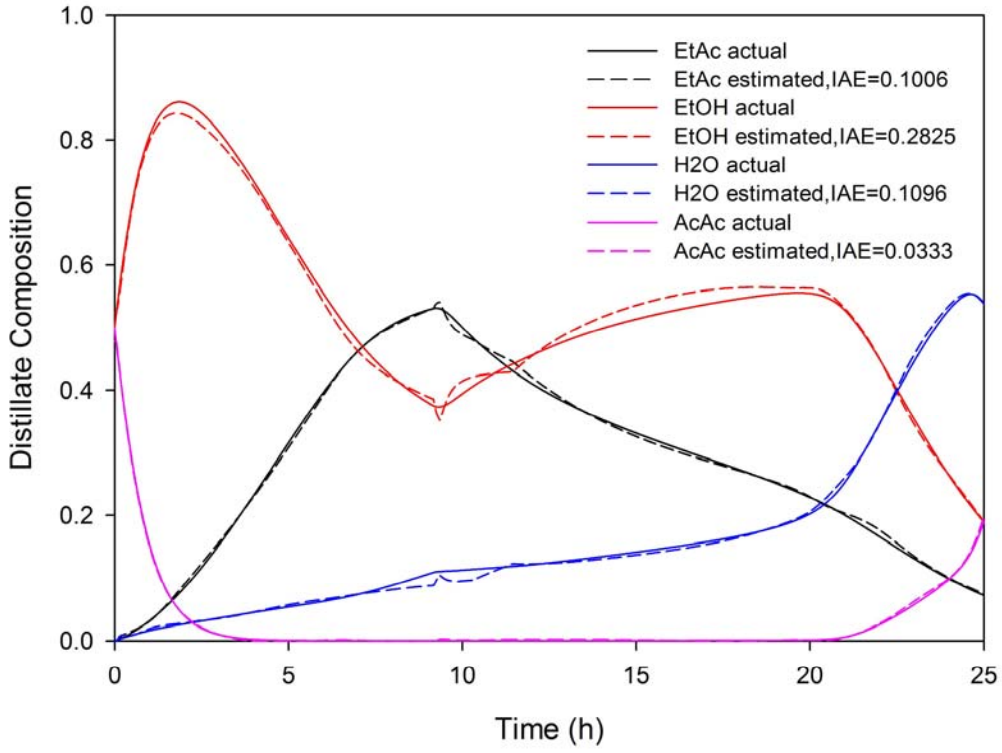


Figure 8.16. Actual and Estimated Distillate Compositions with Total Reflux Followed by a Reflux Ratio of 0.75

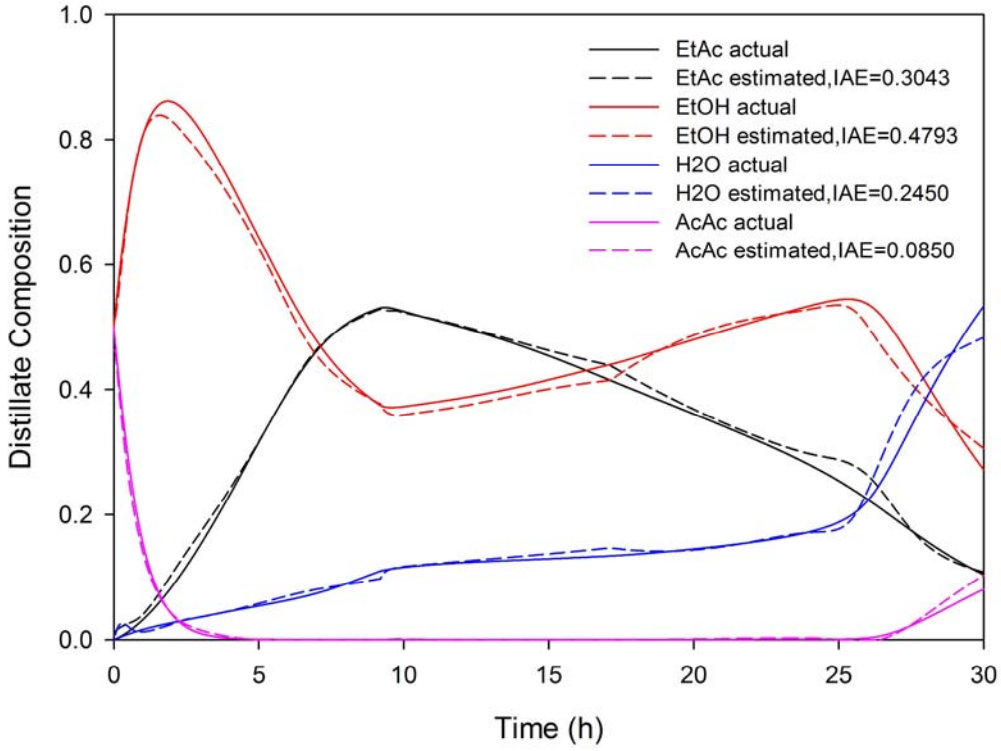


Figure 8.17. Actual and Estimated Distillate Compositions with Total Reflux Followed by a Reflux Ratio of 0.83

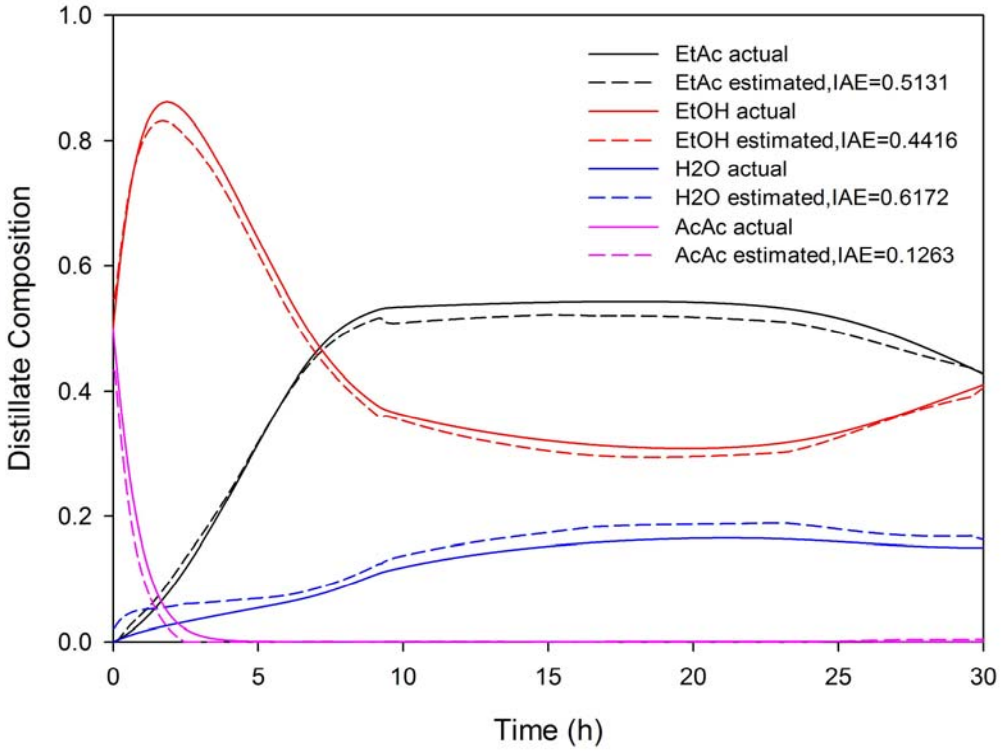


Figure 8.18. Actual and Estimated Distillate Compositions with Total Reflux Followed by a Reflux Ratio of 0.895

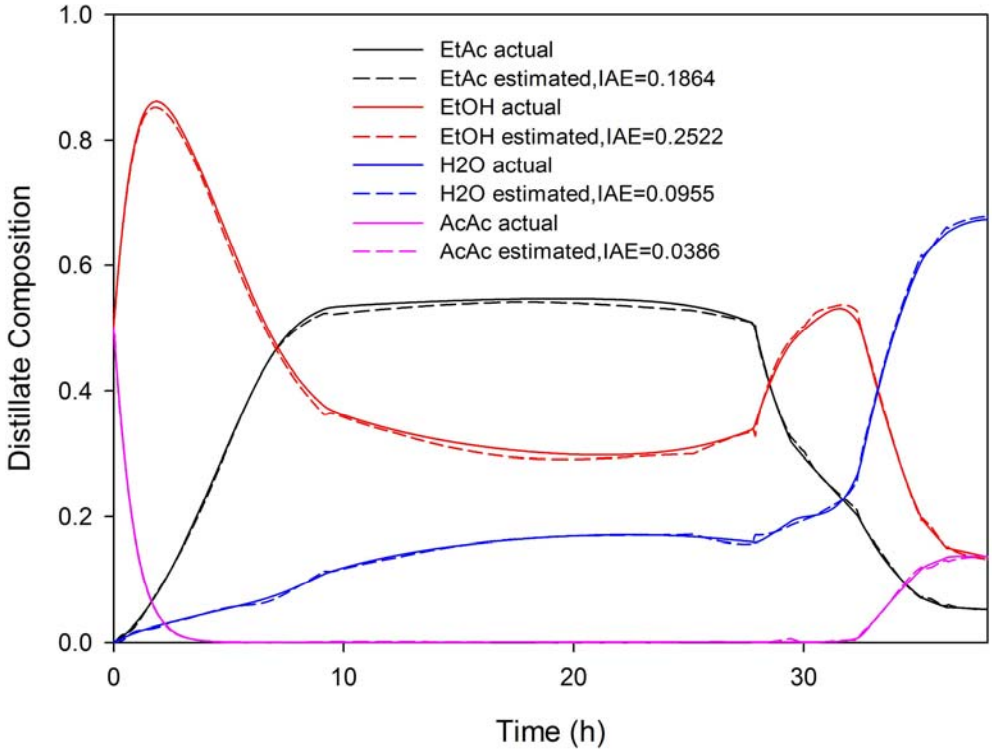


Figure 8.19. Actual and Estimated Distillate Compositions with 10% Increased Optimal Reflux Profile

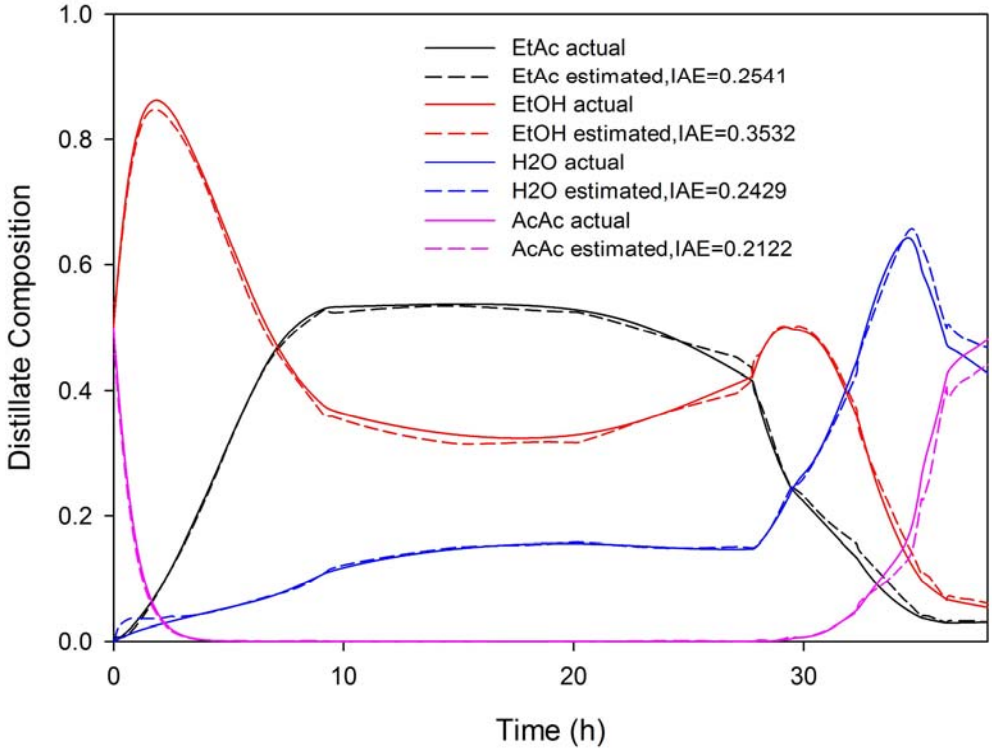


Figure 8.20. Actual and Estimated Distillate Compositions with 5% Decreased Optimal Reflux Profile

8.4 Control Studies with the Designed ANN Estimator

The ANN estimator is designed for utilizing it in the inferential control algorithm for composition control of the reactive batch distillation column. Thus, the performance of the ANN estimator is tested within this control algorithm. The control algorithm uses an actual scheduling policy explained in Chapter 4. The compositions in the reflux drum or in the product-cut tanks are the inputs to the controller as measured variables and the reflux ratio of the column is the manipulated variable. The optimal reflux ratio profile of the column is used as the pre-set reflux ratio values in the control algorithm. The tank, to which the distillate stream has to be diverted and the time for diversion (i.e. to change the reflux ratio to optimized pre-set value) are decided by the input compositions to the controller and by utilizing the actual scheduling policy. In the control structure, the compositions are supplied by the ANN estimator. The block diagram of this control scheme is shown in Figure 8.21. In control studies, the verified column simulation model is used to find the "actual" composition values to adjust the operation scheduling (i.e. the reflux ratio policy). The same procedure is repeated with the estimated values of the compositions using the ANN estimator. Thus, for performance measures, IAE scores in the estimation of the compositions, the capacity factor, and the total batch time are considered and used.

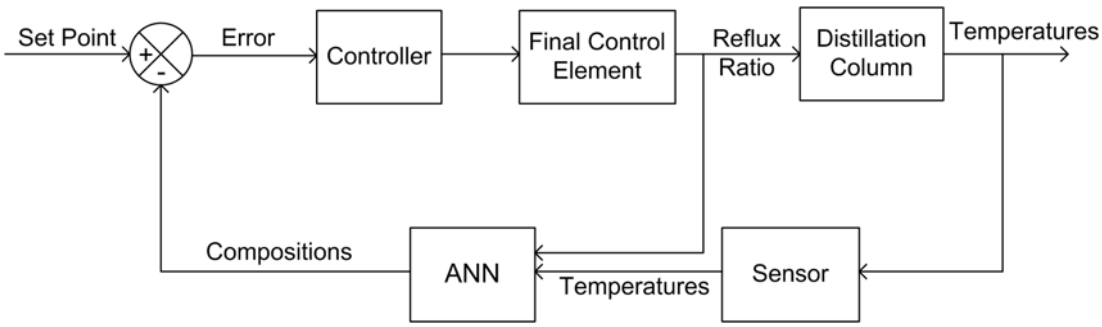


Figure 8.21. Block Diagram of the Control Scheme

8.4.1 Control Studies with Actual Composition Values

Control studies by simulation are carried out by taking the composition data directly from the column model as the feedback to the controller. The desired product purities in the product-cut tanks are the set points of the controller which are 0.52, 0.5, 0.65, and 0.999. The response of the column in terms of

distillate compositions is given in Figure 8.22. In the figure, reflux ratio values, which are adjusted in time by the controller, are also shown. This response resulted in the capacity factor and the total batch time of 5.35 mol/h and 38.17 h, respectively.

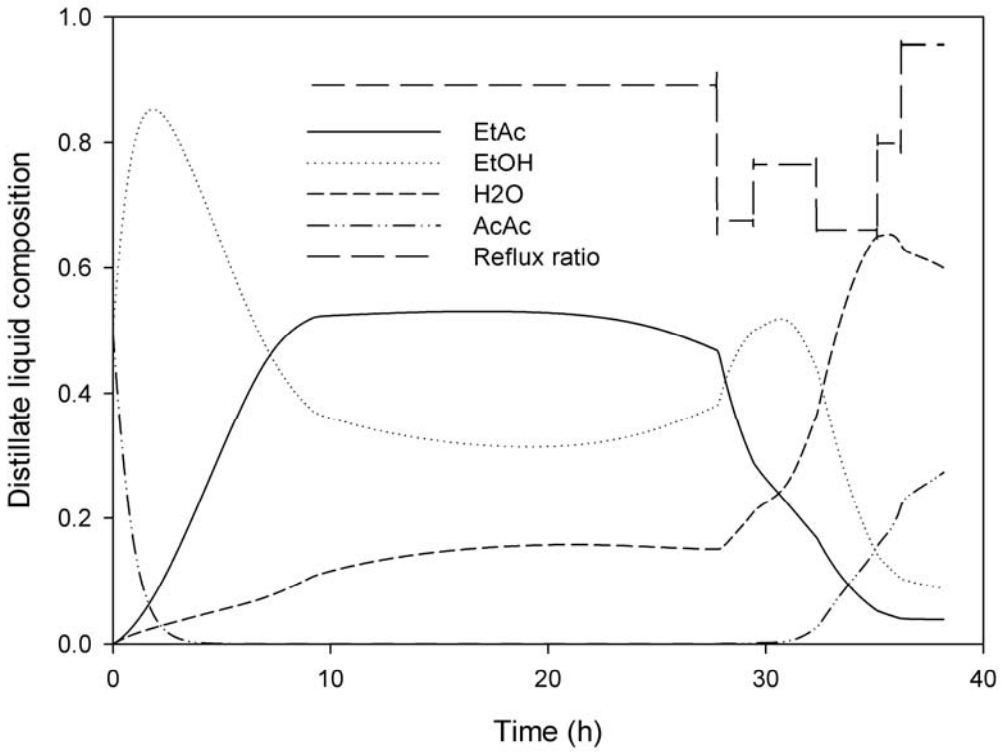


Figure 8.22. Distillate Compositions Change with respect to Time Using Actual Composition Values

8.4.2 Control Studies with Estimated Composition Values

In order to see the performance of the ANN estimator in the closed-loop control system, the same simulation is done by using the estimated compositions as the feedback to the controller. The response of the reflux drum compositions and the optimum reflux ratio values (adjusted in time by the controller) are given in Figure 8.23. As can be seen from the figure, the IAE scores of the compositions are high. Furthermore, errors in the capacity factor and the total batch time, which are in this case 6.36 mol/h and 34.59 h, are 19% and 9%, respectively. These differences in the performance measures are considered to be large. Therefore, in order to improve the performance of the ANN estimator in the closed-loop algorithm, it is considered to refine the error in the composition estimation by using the actual distillate composition measurements obtained

directly from the column in certain time intervals. In a real plant, composition measurements for such a system can be done in 15 minutes intervals. Therefore, the estimated compositions can be corrected in every 15 minutes interval. In Figure 8.23, it is observed that deviation in composition profiles start after total reflux period. Thus, a correction at this point is crucial. As a first trial, a correction during the total reflux period and one just after this period is considered.

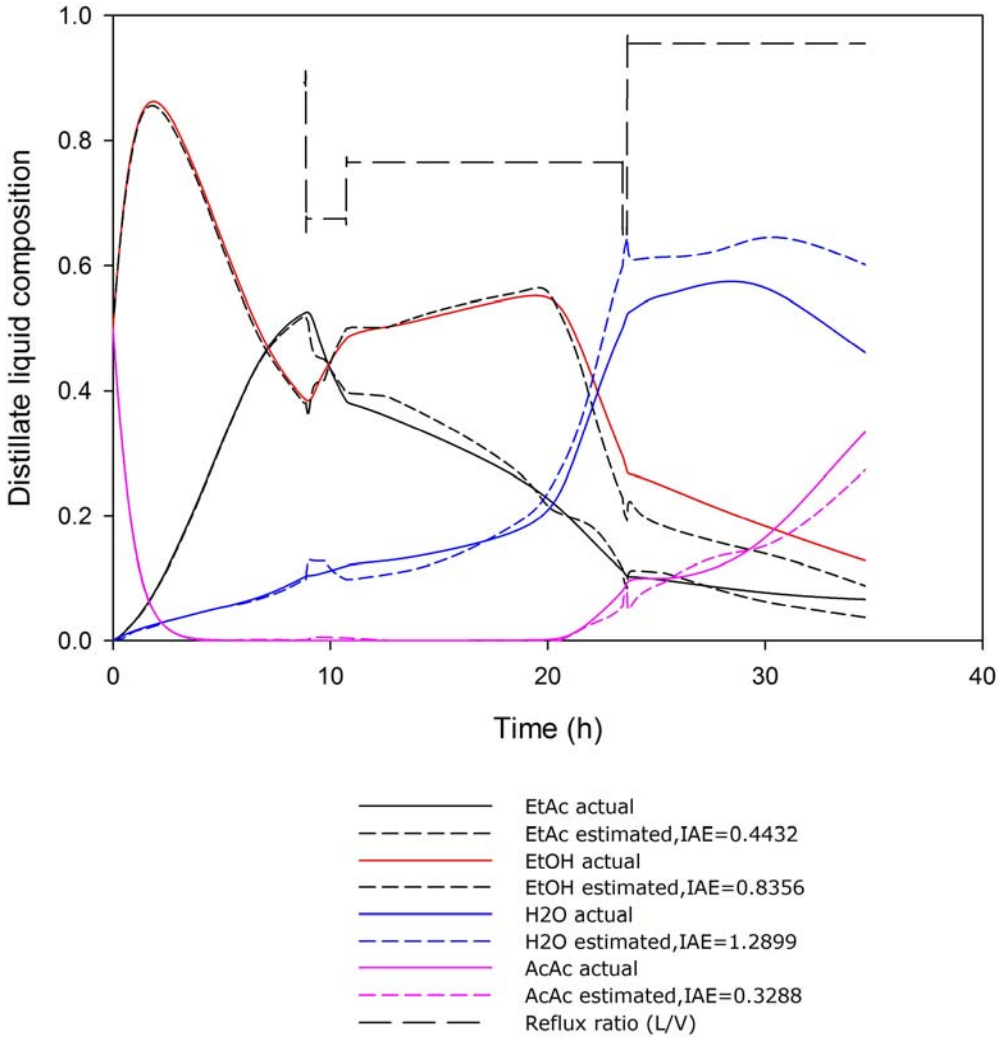


Figure 8.23. Distillate Compositions with Estimated Composition Feedback

8.4.3 Control Studies with Estimated Composition Values (With Error Refinement)

In Figure 8.24, response in the compositions in the reflux drum, in the feedback inferential control algorithm, with error refinement by using only two actual distillate composition values taken from the column are shown. One of these measurements is taken during total reflux operation and the other one is taken just after total reflux operation as explained above. The optimum reflux ratio values adjusted in time by the controller are shown with dashed line. The IAE scores in the estimations of four components in the reflux drum are remarkably reduced. In addition, the errors in the capacity factor and the total batch time (5.34 mol/h and 37.41 h) are also reduced to 5% and 2%, respectively.

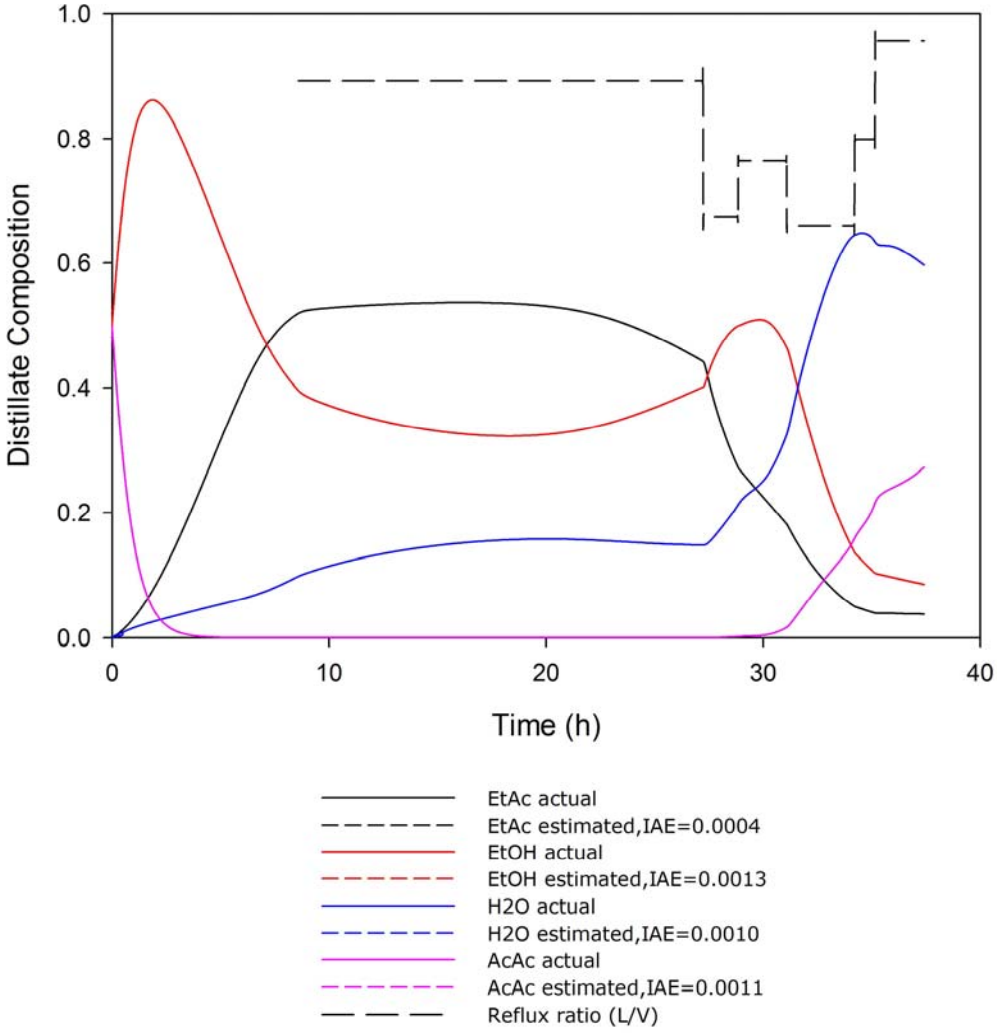


Figure 8.24. Distillate Compositions with Estimated Compositions Feedback with Two Error Refinement

Furthermore, in order to be sure about the error refinement time, the estimated values are compared with the ones obtained from simulation at discrete time intervals and the estimation errors are calculated. When the estimation errors are above their tolerance level, then the errors are refined. Until the time when the new product composition data are obtained, the estimation errors are assumed to be constant. When new composition values (measurements from real plant) are obtained, the estimation errors are also updated. Figure 8.25 shows the reflux drum liquid compositions obtained by using this procedure and the optimum reflux ratio values adjusted in time by the controller. The capacity factor and the total batch time in this case are 5.34 mol/h and 37.41 h, respectively. The IAE scores in the estimations and the errors in the capacity factor and the total batch time are almost same with the two measurement case.

As a result of this analysis, it can be said that the reactive batch distillation column can be controlled for variable reflux ratio policy by the use of the designed ANN estimator.

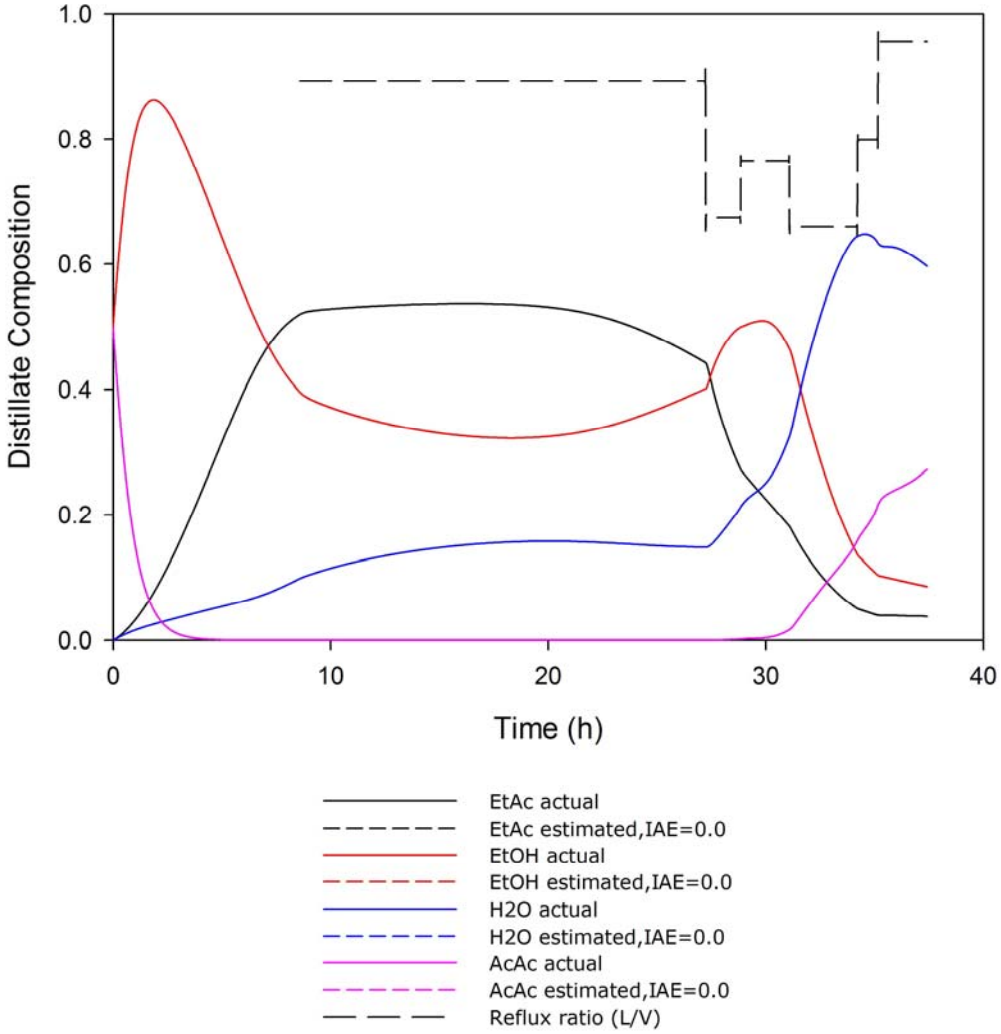


Figure 8.25. Distillate Compositions with Estimated Compositions Feedback with an Error Tolerance Level for Error Refinement

CHAPTER 9

CONCLUSIONS

In this work, the inferential control of a multi-component batch reactive distillation column is studied. The reaction studied is an esterification reaction where ethanol and acetic acid reacts to produce ethyl acetate and water. The estimator used in the inferential control algorithm is an ANN.

- A dynamic first principles model is developed for the reactive batch distillation column by modifying previously developed batch distillation column simulation (Yıldız et. al., 2005).

- In simulation studies, nearly same results are obtained with literature. However, when compared with the experimental data, it is found that, the model that uses the VLE and rate expressions given in the literature is not accurate.

- Among the three thermodynamic model approaches; $\phi-\phi, \gamma-\phi$, EOS-G^{ex}; which are used in combined with two different equation of states (PR and PRSV) and with three different mixing rules (van der Waals, HVO, HVOS), $\phi-\phi$ approach is shown to be inappropriate for this system. EOS-G^{ex} approach with PRSV and HVOS performs well, however this approach is not satisfactory with PR and/or HVO. Among the different activity coefficient models (NRTL, Wilson, UNIQUAC), NRTL gives the smallest IAE score for the distillate and reboiler compositions. $\gamma-\phi$ approach performs even better than EOS-G^{ex} method with NRTL model for the liquid phase and PRSV EOS for the vapor phase. The $\gamma-\phi$ approach, with PR EOS together with van der Waals

mixing rule and NRTL model, gives also very small errors and has simple structure and is used through the rest of the study.

- It can be said as a result of modeling studies that the thermodynamic part is very important in modeling studies.

- The optimal reflux ratio profile is obtained through a nonlinear optimization problem, where maximization of the capacity factor is selected as the objective function.

- An ANN estimator is designed to estimate the distillate composition values of the column from available four temperature measurements. The network used is an Elman network with two hidden layers. It has 20 neurons in the first hidden layer, 3 neurons in the second hidden layer and 4 neurons in the output layer.

- The performance of the designed neural network is found to be good in open-loop. In the closed-loop, the estimated compositions are given as inputs to the controller and a scheduling policy is used as the control law by using the optimal reflux ratio profile as pre-defined reflux ratio values required in the control law and it is found that the estimation accuracy must be increased in order to have a better composition control in the column. The actual composition values are used to refine the error in the estimation.

- It can be said as a result of estimation and control studies that, it is possible to control the compositions in this reactive distillation column by using the designed ANN estimator, by refining the errors in estimation whenever they pass their tolerance levels.

REFERENCES

Al-Arfaj, M., Luyben, W.L., (2000), "Comparison of Alternative Control Structures for an Ideal Two-Product Reactive Distillation Column", Ind. Eng. Chem. Res., Vol. 39, pp. 3298-3307.

Al-Arfaj, M., Luyben, W.L., (2002a), "Control Study of Ethyl tert-Butyl Ether Reactive Distillation ", Ind. Eng. Chem. Res., Vol. 41, pp. 3784-3796.

Al-Arfaj, M., Luyben, W.L., (2002b), "Control of Ethylene Glycol Reactive Distillation Column", AIChE Journal, Vol. 48, No. 4, pp. 905-908.

Al-Arfaj, M., Luyben, W.L., (2002c), "Design and Control of an Olefin Metathesis Reactive Distillation Column", Chemical Engineering Science, Vol. 57, pp. 715-733.

Al-Arfaj, M., Luyben, W.L., (2002d), "Comparative Control Study of Ideal and Methyl Acetate Reactive Distillation", Chemical Engineering Science, Vol. 57, pp. 5039-5050.

Alejski, K., Duprat, F., (1996), "Dynamic Simulation of the Multicomponent Reactive Distillation", Chemical Engineering Science, Vol. 51, No. 18, pp. 4237-4252.

Bahar, A., (2003), "An Artificial Neural Network Estimator Design for the Inferential Model Predictive Control of an Industrial Multi-Component Distillation Column", M.Sc. Thesis, Middle East Technical University, Department of Chemical Engineering, Ankara, Turkey.

Bahar, A., Özgen, C., Leblebicioğlu, K., Halıcı, U., (2004), "An Artificial Neural Network Estimator Design for the Inferential Model Predictive Control of an Industrial Distillation Column", Industrial and Engineering Chemistry Research, Vol. 43, No. 19, pp. 6102-6111.

Bakker, R., Hangx, G., Kwant, G., Maessen, H., Markusse, P., (2001), "Reaction and Catalyst Deactivation Kinetics of the Aldol Condensation of Acetone",

Intelligent Column Internals for Reactive Separations, Technical Report, Delivarable 21, pp. 2-8.

Balasubramhanya, L.S., Doyle III, F.J., (2000), "Nonlinear Model-Based Control of a Batch Reactive Distillation Column", Journal of Process Control, vol. 10, pp. 209-218.

Baratti, R., Bertucco, A., Rold, A.D., Morbidelli, M., (1995), "Development of a Composition Estimator for Binary Distillation Columns. Application to a Pilot Plant", Chemical Engineering Science, Vol. 50, No. 10, pp. 1541-1550.

Bisowarno, B.H., Tian, Y., Tade, M.O., (2004), "Adaptive Control of an ETBE Reactive Distillation Column", Journal of Chemical Engineering of Japan, Vol. 37, No. 2, pp. 210-216.

Bock, H., Jimoh, M., Wozny, G., (1997), "Analysis of Reactive Distillation Using the Esterification of Acetic Acid as an Example", Chem. Eng. Technol., Vol. 20, pp. 182-191.

Bogacki, M.B., Alejski, K., Szymanowski, J., (1989), "The Fast Method of the Solution of a Reacting Distillation Problem", Computers chem. Engng., Vol. 13, No. 9, pp. 1081-1085.

Bulsarı, A.B., (1995), "Neural Networks for Chemical Engineering", Elsevier.

Burgos-Solorzano, G.I., Brennecke, J.F., Stadtherr, M.A., (2004), "Validated Computing Approach for High-Pressure Chemical and Multiphase Equilibrium", Fluid Phase Equilibria, Vol. 219, Iss. 2, pp. 245-255.

Chang, Y.A., Seader, J.D., (1988), "Simulation of Continuous Reactive Distillation by a Homotopy-Continuation Method", Comput. chem. Engng., Vol. 12, No. 12, pp. 1243-1255.

Dadhe, K., (2004), "Nonlinear Calibration for Near-Infrared Spectroscopy", Chem. Eng. Technol., vol. 27, No. 9, pp. 946-950.

Engell, S., Fernholz, G., (2003), "Control of a Reactive Separation Process", Chemical Engineering and Processing", Vol. 42, pp. 201-210.

Fileti, A.M., Cruz, S.L., Pereira, J.A.F.R., (2000), "Control Strategies Analysis for a Batch Distillation Column with experimental Testing", Chemical Engineering and Processing, Vol. 39, pp. 121-128.

Giessler, S., Hasebe, S., Hashimoto, I., (2001), "Optimization Aspects for Reactive Batch Distillation", Journal of Chemical Engineering of Japan, Vol. 34, No. 3, pp. 312-318.

Haykin, S., (1999), Neural Networks – A Comprehensive Foundation, Second Edition, Prentice Hall Inc., America.

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

Jacobs, O.L.R., (1974), "Introduction to Control Theory", Clarendon Press, Oxford.

Jazayeri-Rad, H., (2004), "The Nonlinear Model Predictive Control of a Chemical Plant Using Multiple Neural Networks", Neural Computing and Applications, Vol. 13, Iss.1, pp. 2-15.

Joseph, B., Brosilow, C.B., (1978), "Inferential Control of Processes", AIChE Journal, Vol. 24, No. 3, pp. 485-502.

Kang, Y. W., Lee, Y. Y., Lee, W. K., (1992), "Vapor-Liquid Equilibria with Chemical Reaction Equilibrium - Systems Containing Acetic Acid, Ethly Alcohol, Water, and Ethyl Acetate", Journal of Chemical Engineering of Japan, Vol. 25, No. 6, pp. 649-655.

Kano, M., Miyazaki, K., Hasebe, S., Hashimoto, I., (2000), "Inferential Control System of Distillation Compositions Using Dynamic Partial Least Squares Regression", Journal of Process Control, Vol. 10, pp. 157-166

Komatsu H., Holland, C.D., (1977), "A New Method of Convergence for Solving Reacting Distillation Problems", Journal of Chemical Engineering of Japan, Vol. 10, No. 4, pp. 292-297.

Luyben, W.L., (1988), "Multicomponent Batch Distillation. 1. Ternary Systems with Slop Recycle", Ind. Eng. Chem. Res., Vol. 27, pp. 642-647.

Monroy-Loperena R., Alvarez-Ramirez, J., (2000), "Output-Feedback Control of Reactive Batch Distillation Columns", Ind. Eng. Chem. Res., Vol. 39, pp. 378-386.

Mujtaba, I.M., Macchietto, S., (1996), "Simultaneous optimization of design and operation of multicomponent batch distillation column—single and multiple separation duties", Journal of Process Control, Vol. 6, Iss. 1, pp. 27-36.

Mujtaba, I.M., Macchietto, S., (1997), "Efficient Optimization of Batch Distillation with Chemical Reaction Using Polynomial Curve Fitting Techniques", Ind. Eng. Chem. Res., Vol. 36, pp. 2287-2295.

Ogata, K., (1997), Modern Control Engineering, Third Edition, Prentice Hall.

Oisiovici, R.M, Cruz, S.L., (2000), "State Estimation of Batch Distillation Columns Using an Extended Kalman Filter", Chemical Engineering Science, Vol. 55, pp. 4667-4680.

Okur, H. and Bayramoglu, M., (2001), "The Effect of the Liquid-Phase Activity Model on the Simulation of Ethyl Acetate Production by Reactive Distillation", Ind. Eng. Chem. Res., Vol. 40, pp. 3639-3646.

Orbey H., Sandler S.I., (1998), "Modeling Vapor-Liquid Equilibria", Cambridge.

Park J., Lee, N., Park, S., Cho, J., (2006), "Experimental and Simulation Study on the Reactive Distillation Process for the Production of Ethyl Acetate", J. Ind. Eng. Chem., Vol. 12, No. 4, pp. 516-521.

Patke, N.G., Deshpande, P.B., Chou, A.C., (1982), "Evaluation of Inferential and Parallel Cascade Schemes for Distillation Control", Ind. Eng. Chem. Process Des. Dev., Vol. 21, No. 2, pp. 266-272.

Perry, R.H. and Green, D., (1984), "Perry's Chemical Engineers' Handbook", Sixth Edition, McGraw-Hill.

Pham, D.T. and Liu, X., (1995), "Neural Networks for Identification, Prediction and Control", Springer-Verlag London Limited.

Quintero-Marmol, E., Luyben, W.L., Georgakis, C. (1991), "Application of an Extended Luenberger Observer to the Control of Multicomponent Batch Distillation", Ind. Chem. Eng. Res., Vol. 30, pp. 1870-1880.

Quintero-Marmol, E., Luyben, W.L., (1992), "Inferential Model-Based Control of Multicomponent Batch Distillation", Chemical Engineering Science, Vol. 47, No. 4, pp. 887-898.

Sandler, S.I., (1999), "Chemical and Engineering Thermodynamics", Third Edition, Wiley.

Seborg, D.E., Edgar, T.F., Mellichamp, D.A., (1998), "Process Dynamics and Control", Wiley Inc.

Simandl, J., Svrcek, W.Y., (1991), "Extension of the Simultaneous-Solution and Inside-Outside Algorithms to Distillation with Chemical Reactions", Computers chem. Engng., Vol. 15, No. 5, pp. 337-348.

Sneesby, M.G., Tade, M. O., Datta, R., Smith, T. N., (1997), "ETBE synthesis via Reactive Distillation. 2. Dynamic Simulation and Control Aspects", Ind. Eng. Chem. Res., 36, 1870-1881.

Sneesby, M.G., Tade, M. O., Smith, T. N., (1999), "Two-point Control of a Reactive Distillation Column for Composititon and Conversion", Journal of Process Control, Vol. 9, pp. 19-31.

Suzuki, I., Yagi, H., Komatsu, H., Hirata. M., (1971), "Calculation of Multi-component Distillation Accompanied by a Chemical Reaction", Journal of Chemical Engineering of Japan, Vol. 4, No. 1, pp. 26-32.

Suzuki, I., Komatsu, H., Hirata. M., (1970), "Formulation and Prediction of Quaternary Vapor-Liquid Equilibria Accompanied by Esterification", Journal of Chemical Engineering of Japan, Vol. 3, No. 2, pp. 152-157.

Stephanopoulos, G., (1984), Chemical Process Control An Introduction to Theory and Practice, Prentice Hall.

Stryjek R., Vera J.H., (1986), "An Improved Peng-Robinson Equation of State for Pure Compounds and Mixtures", Vol. 64, No. 2, pp. 323-333.

Tang, Y. T., Huang, H., Chien, I., (2003), "Design of a Complete Ethyl Acetate Reactive Distillation System", Journal of Chemical Engineering of Japan, Vol. 36, No. 11, pp. 1352-1363.

Tade, M.O., Tian, Y., (2000), "Conversion Inference for ETBE Reactive Distillation", Separation and Purification Technology, Vol. 19, pp. 85-91.

Taylor, R., Krishna, R., (2002), "Modeling of Homogeneous and Heterogeneous Reactive Distillation Processes", Reactive Distillation. Edited by Sundmacher K., Kienle A., Wiley-VCH, Germany.

Tian, Y., Zhao, F., Bisowarno, B.H., Tade, M.O., (2003), "Pattern-Based Predictive Control for ETBE Reactive Distillation", Journal of Process Control, 13, 57-67.

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

Ullman's Encyclopedia of Industrial Chemistry Index, Volume A1 (1996).

Venkateswarlu, Ch., Kumar, B.J., (2006), "Composition Estimation of Multicomponent Reactive Batch Distillation with Optimal Sensor Configuration", Chemical Engineering Science", Vol. 61, pp. 5560-5574.

Wajge, R.M., Reklaitis, G.V., (1999), "RBDOPT: a general-purpose object-oriented module for distributed campaign optimization of reactive batch distillation", Chemical Engineering Journal, Vol. 75, pp. 57-68.

Wang, S.J., Wong, D.S.H., Lee, E.K., (2003), "Effect of Interaction Multiplicity on Control System Design for a MTBE Reactive Distillation Column", Journal of Process Control, Vol. 13, pp. 503-515.

Weber, R., Brosilow, C., (1972), "The Use of Secondary Measurements to Improve Control", AIChE Journal, Vol. 18, No. 3, pp. 614-622

Yıldız, U., (2002), "Multicomponent Batch Distillation Column Simulation and State Observer Design", M.Sc. Thesis, Middle East Technical University, Department of Chemical Engineering, Ankara, Turkey.

Yu, C., Luyben, W.L., (1987), "Control of Multicomponent Distilllation Columns Using Rigorous Composition Estimators", I.CHEM.E. Symposium Series, No. 104, pp. A29-A49.

Zamprogna, E., Barolo, M., Seborg, D.E., (2001), "Composition Estimations in a Middle-Vessel Batch Distillation Column Using Artificial Neural Networks", Trans IChemE, Vol. 79, Part A, pp. 689-696.

Zilouchian, A. and Jamshidi, M., (2001), "Intelligent Control Systems Using Soft Computing Methodologies", CRC Press.

APPENDIX A

PROPERTIES OF THE COMPONENTS

Table A.1 Physical Parameters of the Components

Component	Mw (kg/kmol)	Density (kg/lt)	T_c (K)	P_c (MPa)	w	T_{boil} (K)
Ethyl acetate	88.106	0.90	523.2	3.830	0.360	350.3
Ethanol	46.069	0.790	516.2	6.394	0.635	351.5
Water	18.015	1.0	647.3	22.129	0.344	373.2
Acetic acid	60.052	1.05	594.4	5.796	0.432	391.1

Table A.2 Heat Capacity Constants of the Components[1]

Component	a	b	c	d
Ethyl acetate	7.235	4.072e-1	-2.092e-4	2.855e-8
Ethanol	19.875	2.095e-1	-1.037e-4	2.004e-8
Water	50.811	2.129e-1	-6.310e-4	64.830e-8
Acetic acid	4.840	2.549e-1	-1.753e-4	4.949e-8

[1] $C_p = a + bT + cT^2 + dT^3$ (J/mol K) where T is in K.

APPENDIX B

CALIBRATION OF GAS CHROMATOGRAPH

In the experiments, pure ethanol (≥99.9%), pure acetic acid (≥99.8), pure ethyl acetate supplied by Merck, and pure water are used. The molecular weights and the densities of the substances used are given in Table A.1. For calibration, known amounts of binary mixtures from these substances are prepared and analyzed through GC. For binary mixtures, Equation 3.1 becomes as in Equation B.1. Thus, for known amounts of binary mixtures, the correction factors can be calculated from this equation by taking one of the components as base component. In this study, base component is selected as ethanol and thus its correction factor is taken to be equal to 1. Other correction factors are calculated by the analysis of the prepared binary mixtures and by the use of Equation B.1. The calculation of correction factors for the calibration of GC are given in Sections B.1-B.3.

$$
\begin{aligned}
x_A &= \frac{A_A\beta_A}{A_A\beta_A + A_B\beta_B} \\
A_A\beta_A &= x_A A_A\beta_A + x_A A_B\beta_B \\
(1-x_A)A_A\beta_A &= x_A A_B\beta_B \\
\frac{x_B}{x_A} &= \frac{A_B\beta_B}{A_A\beta_A}
\end{aligned}
\qquad \text{(B.1)}
$$

B.1 Correction Factor Calculation for Acetic Acid

50% EtOH – 50% AcAc mixture

$$\frac{0.5}{0.5} = \frac{(2479)(1)}{(2700)\beta_{AcAc}} \Rightarrow \beta_{AcAc} = 0.918$$

30% EtOH – 70% AcAc mixture

$$\frac{0.3}{0.7} = \frac{(1588)(1)}{(3819)\beta_{AcAc}} \Rightarrow \beta_{AcAc} = 0.970$$

60% EtOH – 40% AcAc mixture

$$\frac{0.6}{0.4} = \frac{(2863)(1)}{(2118)\beta_{AcAc}} \Rightarrow \beta_{AcAc} = 0.901$$

The correction factor for acetic acid does not change with composition, therefore average value of these three numbers, 0.930 is taken as the correction factor of acetic acid.

B.2 Correction Factor Calculation for Water

50% EtOH – 50% H_2O mixture

$$\frac{0.5}{0.5} = \frac{(3911)(1)}{(1624)\beta_{H_2O}} \Rightarrow \beta_{H_2O} = 2.408$$

30% EtOH – 70% H_2O mixture

$$\frac{0.3}{0.7} = \frac{(2937)(1)}{(3091)\beta_{H_2O}} \Rightarrow \beta_{H_2O} = 2.217$$

70% EtOH – 30% H_2O mixture

$$\frac{0.7}{0.3} = \frac{(4650)(1)}{(926)\beta_{H_2O}} \Rightarrow \beta_{H_2O} = 2.152$$

The correction factor for water does not change with composition, therefore average value of these three numbers, 2.259 is taken as the correction factor of water.

B.3 Correction Factor Calculation for Ethyl Acetate

50% EtOH – 50% EtAc mixture

$$\frac{0.5}{0.5} = \frac{(1850)(1)}{(3047)\beta_{EtAc}} \Rightarrow \beta_{EtAc} = 0.607$$

30% EtOH – 70% EtAc mixture

$$\frac{0.3}{0.7} = \frac{(1179)(1)}{(3762)\beta_{EtAc}} \Rightarrow \beta_{EtAc} = 0.731$$

70% EtOH – 30% EtAc mixture

$$\frac{0.7}{0.3} = \frac{(2961)(1)}{(1995)\beta_{EtAc}} \Rightarrow \beta_{EtAc} = 0.636$$

The correction factor for ethyl acetate does not change with composition, therefore average value of these three numbers, 0.658 is taken as the correction factor of ethyl acetate.

APPENDIX C

SOURCE CODE

C.1 Main Program Code

Glob_Decs.m

```
%----------------     Programming Definitions     ------------------------%
% ================================================
% Simulation Parameters
% ================================================
% Dummy variables
global Dummy1;  global Dummy2;  global Dummy3;  global Dummy4;
% Output Warnnings
global OUT_WARNNING;
% tolerance for the decision to make the component fraction zero
global zero_tolerance;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Output File ID
% Liquid Profile file
global FID_lprofile;
% Vapor Profile file
global FID_vprofile;
% Temperature Profile file
global FID_tprofile;
% Holdup Profile file
global FID_holdup;
% Liquid and Vapor Flowrate Profile file
global FID_lvflow;
% Controller Outputs file
global FID_control;
% Estimator Outputs file
global FID_estimator;
% Tank Outputs file
global FID_tank;
% Capacity factor file
global FID_cap;
% -------------------     End Programming Definitions     --------------- %
% -------------------     Simulation Definitions       ---------------------%
% ================================================
% Step time and Time Span
% ================================================
% Initial Time (hour)
global tstart;
% Integration time step (hour)
global DeltaT;
% Time Span of Simulation (hour)
global tfinal;
global num_step;
% Displaying time step (hour)
global disp_DeltaT;
```

```
% ================================================
% Time Step to Estimate and Control
% ================================================
% Time Step to Estimate (hour)
global estimate_DeltaT;
% ================================================
% Define Controller types and Initialize Type of Controller
% ================================================
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Type of Controllers
% The open-loop controller with using predefined switching times and
% corresponding distillate flowrates (or reflux ratios)
global Cont_OL;
% The closed-loop controller with the actual composition feed-back
global Cont_CL_ActualFB;
% The closed-loop Controller with the estimated composition feed-back
global Cont_CL_EstFB;
% Current Type of Controller
global Type_Controller;
% -----------------     End Simulation Definitions       ------------------%
% ----------------- Real Plant Simulation Parameters        -------------- %
% ================================================
% Physical System Definitions
% ================================================
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Column Specs.
% Number of Components - in the order of volatilities :
%                         1st is most volatile and last is least volatile
global NC;
% Number of Trays
global NT;
% Murphree Efficiency
global Eff_Murphree;
% Tray Efficiencies
global Eff;
% Tray(s) Volumetric holdups (m3)
global Tray_Vol_Holdup;
% Initial reflux drum liquid holdup (moles)
global Drum_M_Holdup_initial;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Feed Specs.
% Total amount of feed charged to the still pot (moles)
global M_Feed;
% Feed compositions (moles/moles)
global X_Feed;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Initial Operation Params.
% Initial Boiler load (J/hour)
global Q_Boiler_initial;
% Initial One Over Reflux ratio (D/L0) (ratio)
global R_Ratio_inv_initial;
% Initial Distillate Flow Rate (mol/hour)
global D_DistillRate_initial;
% Initial Amount of product distilled (moles)
global M_Distilled_initial;
% Initial still pot and reflux drum pressure (Pa)
global Press_Pot_initial;
global Press_Drum_initial;
% Stochiometric coefficients of reaction components
global epsilon;
% -------------  End Real Plant Simulation Parameters      ----------------%
% Indication of whether the system continues with constant reflux ratio
% after total reflux steady state or not
global constant_R_Ratio;
% Reflux Ratio
global R_Ratio;
% Percent change in R profile from the optimal profile when R isnot constant
% after total reflux steady state.
global Rpercent;
% Output Measurement Index
global EST_MeasurementOrder;
```

```
% Estimation time
global est_time;
% Correction interval multiplication
global corr_int;
%Estimation error
global est_error;
```

Glob_Initial.m

```
% ------------------     Programming Initialization  --------------------- %
% ==================================================
% Simulation Parameters Settings
% ==================================================
% Dummy variables
Dummy1=0.0; Dummy2=0.0; Dummy3=0.0; Dummy4=0.0;
% Output Warnnings
OUT_WARNNING = 1;
% tolerance for the decision to make the component fraction zero
zero_tolerance = 9e-180;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Output File ID Creation
fclose all;  Current_Directory = cd;
% Liquid Profile file
delete([Current_Directory '\' 'lprofile.txt']);
FID_lprofile = fopen('lprofile.txt','at');
% Vapor Profile file
delete([Current_Directory '\' 'vprofile.txt']);
FID_vprofile = fopen('vprofile.txt','at');
% Temperature Profile file
delete([Current_Directory '\' 'tprofile.txt']);
FID_tprofile = fopen('tprofile.txt','at');
% Holdup Profile file
delete([Current_Directory '\' 'holdup.txt']);
FID_holdup = fopen('holdup.txt','at');
% Liquid and Vapor Flowrate Profile file
delete([Current_Directory '\' 'lvflow.txt']);
FID_lvflow = fopen('lvflow.txt','at');
% Controller Outputs file
delete([Current_Directory '\' 'control.txt']);
FID_control = fopen('control.txt','at');
% Estimator Outputs file
delete([Current_Directory '\' 'estimator.txt']);
FID_estimator = fopen('estimator.txt','at');
% Tank Outputs file
delete([Current_Directory '\' 'tank.txt']);FID_tank=fopen('tank.txt','at');
% Capacity Factor file
delete([Current_Directory '\' 'cap.txt']);FID_cap = fopen('cap.txt','at');
% ----------------     End Programming Initialization  ------------------- %
% ------------------     Simulation Initialization    --------------------- %
% ==================================================
% Step time and Time Span
% ==================================================
% Initial Time (hour)
tstart = 0.0;
% Integration time step (hour)
DeltaT = 0.0003;
% Time Span of Simulation (hour)
tfinal = 38.1729;
%tfinal = 30.0;
num_step = round((tfinal - tstart)/DeltaT);
% Displaying time step (hour)
disp_DeltaT = 0.01;
% ==================================================
% Time Step to Estimate and Control
% ==================================================
% Time Step to Estimate (hour)
estimate_DeltaT = 1*DeltaT;  %1*DeltaT;  %round(3*(1/60)/DeltaT)*DeltaT;
```

```
% ================================================
% Define Controller types and Initialize Type of Controller
% ================================================
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Type of Controllers
% The open-loop controller with using predefined switching times and
% corresponding distillate flowrates (or reflux ratios)
Cont_OL = 1;
% The closed-loop controller with the actual composition feed-back
Cont_CL_ActualFB = 2;
% The closed-loop Controller with the estimated composition feed-back
Cont_CL_EstFB = 3;
% Current Type of Controller
Type_Controller = Cont_CL_EstFB;
% ---------------------- End Simulation Initialization ------------------- %
% --------------    Real Plant Simulation Initialization    -------------- %
% ================================================
% Physical System Settings
% ================================================
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Column Specs.
% Number of Components - in the order of volatilities :
%1st is most volatile and last is least volatile
NC = 4;
% Number of Trays
NT = 8;
% Murphree Efficiency
Eff_Murphree = 0.85;
% Tray Efficiencies
Eff = Eff_Murphree * ones(NT,NC);
% Tray(s) Volumetric holdups (m3)
%Tray_Vol_Holdup = 8.3999e-004; % paper data
Tray_Vol_Holdup = 5.5887e-005; % experimental data
% Initial reflux drum liquid holdup (moles)
%Drum_M_Holdup_initial = 100;   % paper data
Drum_M_Holdup_initial = 30;  % experimental data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Feed Specs.
% Total amount of feed charged to the still pot (moles)
%M_Feed = 5000;   % paper data
M_Feed = 311.67;   % experimental data
% Feed compositions (moles/moles)
X_Feed = [0.0; 0.5; 0.0; 0.5];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Initial Operation Params.
% Initial Boiler load (J/hour)
%Q_Boiler_initial = 1.066e+08;  % paper data
Q_Boiler_initial = 2.016e6;% exp. data
% Initial One Over Reflux ratio (D/L0) (ratio)
R_Ratio_inv_initial = 0.0;
% Initial Distillate Flow Rate (mol/hour)
D_DistillRate_initial = 0.0;
% Initial Amount of product distilled (moles)
M_Distilled_initial = 0.0;
% Initial still pot and reflux drum pressure (Pa)
Press_Pot_initial = 101325.0;
Press_Drum_initial = 101325.0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Reaction Params.
% Stoichiometric coefficients of components
epsilon = [+1 -1 +1 -1];
% Summation of stoic. coeff.
epsilon_t = sum(epsilon);
% -------------    End Real Plant Simulation Initialization -------------- %
% -----------------     Estimator Initialization  ----------------------- %
EST_X0 = zeros(NC*(NT+2),1);
% Output Measurement Index
EST_MeasurementOrder = [1, 1+2, 1+5, 1+NT];
% Number of States
EST_NumStates = size(EST_X0,1);
% Number of Measurements
EST_NumMeasurement = 4;
```

```
% constant reflux ratio after total reflux (1) or not (0)
constant_R_Ratio = 0;
% if it continues with constant R enter R_Ratio_p =L/V = R_Ratio/(1+R_Ratio)
R_Ratio_p = 0.83;
% if it continues with variable R enter percent change from optimal profile
Rpercent = 0;
% First measurement is taken at t = 0.25
est_time = 0.25;
% Initial correction interval multiplication
corr_int = 1;
% Initial estimation errors
est_error = [0.0 0.0 0.0 0.0];
% --------------     End Estimator Initialization  ---------------------- %
```

Cont_Plant_Mfile.m

```
function Cont_Plant_Mfile
% Clear command window
clc;
% Include all global variables
Glob_Decs;
% Initialize thermo_LIBRARY.dll
thermo_Init(0);
% Initialize all global variables
display('Global variables are initializing ...');
Glob_Initial;
display('Global variables have been initialized.');
% ================================================
% Step time and Time Span
% ================================================
% Current Simulation Time (hour)
t = tstart;
% Previuos integration step Time (hour)
t_prv = t;
% Current displaying time step (hour)
disp_t = tstart;
% ================================================
% Time Step to Estimate and Control
% ================================================
%% Time Step to Estimate (hour)
estimate_DeltaT = 1*DeltaT;  %1*DeltaT;  %round(3*(1/60)/DeltaT)*DeltaT;
% Current Estimation time (hour)
estimate_t = tstart;
% ================================================
% Initialize Real Plant
% ================================================
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Initial Operation Params.
% Boiler load (J/hour)
Q_Boiler = Q_Boiler_initial;
% One Over Reflux ratio (D/L0) (ratio)
R_Ratio_inv = R_Ratio_inv_initial;
% Distillate Flow Rate (mol/hour)
D_DistillRate = D_DistillRate_initial;
% Amount of product distilled (moles)
M_Distilled = M_Distilled_initial;
% Initial still pot, tray(s), reflux drum pressure (Pa)
Press(1,1) = Press_Pot_initial;
Press(NT+2,1) = Press_Drum_initial;
Press(2:NT+1,1) = PressureProfile(Press(1), Press(NT+2));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Initialize liquid compositions
% Instantaneous still pot,tray(s),reflux drum liquid comp. (moles/moles)
X_frac = zeros(NT+2,NC);
% Initially still pot,tray(s),reflux drum liquid comp. = that of the feed
for i=1:NT+2;  X_frac(i,:) = X_Feed.'; end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Define physical properties
% Instantaneous still pot, tray(s), reflux drum vapor comp. (moles/moles)
```

```
Y_frac = zeros(NT+2,NC);
% Instantaneous still pot, tray(s), reflux drum temperature (K)
Temp = 350.0 * ones(NT+2,1);
% Instantaneous still pot,tray(s), reflux drum liq. phase enthalpy (J/moles)
H_l_Enthalpy = zeros(NT+2,1);
% Instantaneous still pot,tray(s),reflux drum vapor phase enthalpy (J/moles)
H_v_Enthalpy = zeros(NT+2,1);
% Instantaneous still pot,tray(s),reflux drum liq.phase avg density (kg/m3)
Ro_l_Density = zeros(NT+2,1);
% Instantaneous still pot,tray(s),reflux drum avg molecular weight (kg/mol)
Mw_MolWeight = zeros(NT+2,1);
%%%%%%%%%%%%%%%%%%%%%%%%%   Define and Initialize liquid holdups
% Instantaneous still pot, tray(s), reflux drum total holdup amount (moles)
M_Holdup = zeros(NT+2,1);
% ---------------------------------------  Initialize Molar Holdups
% Initial reflux drum liquid holdup (moles)
M_Holdup(NT+2,1) = Drum_M_Holdup_initial;
%%%%%%%%%%%%%%%%%%%   Define and Initialize liquid and vapor flowrates
% Instantaneous liq. flow rates leaving tray(s) and reflux drum (moles/hour)
%  L_flow(1) : dummy
L_flow = zeros(NT+2,1);
% Instantaneous vapor flow rates leaving still pot and tray(s) (moles/hour)
% V_flow(NT+2) : dummy
V_flow = zeros(NT+2,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Define outputs
Out_Temp = zeros(NT+2,1);
%%%%%%%%%%%%%%%%%%%   Define and Initialize storage tank compositions
% Instantaneous compositions in storage tanks (moles/moles) (2*(NC-1)xNC)
Tank_X_frac = zeros(2*(NC-1), NC);
% Instantaneous total holdup amount in storage tanks (moles)
Tank_M_Holdup = zeros(2*(NC-1), 1);
% ================================================
% Initialize Reaction Parameters
% ================================================
% Forward reaction rate constant (m3/(mol.h))
k1 = zeros(NT+2,1);
% Backward reaction rate constant (m3/(mol.h))
k2 = zeros(NT+2,1);
% Reaction rate (m3/(mol.h))
rate = zeros(NT+2,1);
% Reaction rate (1/h) = r*Ro_l_Density/Mw_MolWeight
rate_ro_mw = zeros(NT+2,1);
% ================================================
% Initialize Estimator
% ================================================
 [outputs] = INIT_ESTIMATE;
% Measurement of Plant Outputs
sim_i = zeros(1,EST_NumMeasurement);
% Measurement of Plant Inputs
EST_u = zeros(2,1);
% Instantaneous compositions in storage tanks (moles/moles)
EST_Tank_X_frac = zeros(2*(NC-1), NC);
% Instantaneous total holdup amount in storage tanks (moles)
EST_Tank_M_Holdup = zeros(2*(NC-1), 1);
% Max. and min. values of the inputs and outputs used in the neural network
i_max = zeros(1,4);
i_min = zeros(1,4);
o_max = zeros(1,4);
o_min = zeros(1,4);
% Error in the estimated compositions
est_error = zeros(1,4);
% ================================================
% Initialize Controller
% ================================================
% CONT_SetPoints: Controller Set Points of Product Specifications
% CONT_Num_Oper_Stage   : Number of different operation stage
```

```
% CONT_DistillProfile   : Distillate Flow rate values of different operation
%  stages (Products and/or Slopcuts)
[CONT_SetPoints, CONT_Num_Oper_Stage, CONT_DistillProfile] = INIT_CONTROL;
% Define Controller Outputs
CONT_QBoiler = zeros(1,1);
CONT_RRatioinv = zeros(1,1);
% Initialize Activated Tank Index
Tank_Activated = 0;
% Initialize Current Stage Number (startup:0, )
CONT_Curr_Stage = 0;
% ================================================
% Integration Starts
% ================================================
for i=0:num_step;
%############ Find Current Information of Real Plant Starts ###########%
% Find current physical variables
[Y_frac, Temp, H_l_Enthalpy, H_v_Enthalpy, Ro_l_Density, Mw_MolWeight]=...
P_Calc_Phys_Vars(t, X_frac, Press, Temp);

%%%%%%%%%%%%%%%%%%%%%%%%%%% Calculate reaction rate
for i=1:NT+2;
k1(i) = 1740*exp(-7150/Temp(i));  % m3/(mol.h)
k2(i) = 442.8*exp(-7150/Temp(i)); % m3/(mol.h)
rate(i) = k1(i)*X_frac(i,2)*X_frac(i,4)-k2(i)*X_frac(i,1)*X_frac(i,3);
                                                        % m3/(mol.h)
rate_ro_mw(i) = rate(i)* Ro_l_Density(i)/Mw_MolWeight(i); % 1/h
end;
% Calculate current trays molar holdups
[M_Holdup]=P_Calc_Mol_Tray_Holdups(t, Ro_l_Density, Mw_MolWeight, M_Holdup);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate current Reflux-Drum molar holdup
[M_Holdup] = P_Calc_Mol_Drum_Holdup(t, M_Holdup);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if t<=0;  % For initialization of variables depending on physical variables
% not available in initialization section
% Calculate current distillate, vapor and liquid flow rates
% Use initial values
% -------------------------  Calculate intial vapor and liquid flow rates
% Initialize still pot vapor flow rate
V_flow(1) = Q_Boiler/(H_v_Enthalpy(1) - H_l_Enthalpy(2));
% Initialize tray(s) vapor flow ratesV_flow(NT+2) : dummy
for i=2:NT+1;V_flow(i) = V_flow(1); end;
% Initialize tray(s) and reflux drum liquid flow rates  L_flow(1) : dummy
for i=NT+2:-1:2;L_flow(i) = V_flow(1);  end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate Initial Still Pot Holdup
[M_Holdup] = P_Calc_Mol_Still_Holdup(t, M_Holdup, M_Distilled);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else
% Calculate current Approximated derivatives
[Del_M_Holdup, Del_H_l_Enthalpy, Del_M_Hl] = P_Calc_Apprx_Deriv(t,...
M_Holdup, H_l_Enthalpy, t_prv, M_Holdup_prv, H_l_Enthalpy_prv);

% Calculate current distillate, vapor and liquid flow rates
if R_Ratio_inv == 0; %% for total reflux condition
[D_DistillRate, L_flow, V_flow] = P_Calc_LV_for_Total_Reflux(t,...
H_l_Enthalpy, H_v_Enthalpy, M_Holdup, Del_M_Holdup, Del_H_l_Enthalpy,...
Del_M_Hl, Q_Boiler, R_Ratio_inv, rate_ro_mw, epsilon_t);
else %% for distillate withdrawal
[D_DistillRate, L_flow, V_flow] = P_Calc_LV_for_Finite_Reflux(t, ...
H_l_Enthalpy, H_v_Enthalpy, M_Holdup, Del_M_Holdup, Del_H_l_Enthalpy,...
Del_M_Hl, Q_Boiler, R_Ratio_inv, rate_ro_mw, epsilon_t);
end;
end;
% Find Derivatives
[DX_frac] = P_f(t, X_frac, Y_frac, Temp, M_Holdup, L_flow, V_flow, ...
Q_Boiler, D_DistillRate, R_Ratio_inv, rate_ro_mw, epsilon_t);
```

```
% Get Real Plant Outputs
Out_Temp = [Temp];
% Keep current parameters for future use
[t_prv, M_Holdup_prv, H_l_Enthalpy_prv] = P_Keep_Current_Vars(t, ...
M_Holdup, H_l_Enthalpy);
%########## Find Current Information of Real Plant Ends ############%

%########## Estimator and Controller Simulation Starts #############%
if (t <= 1.00001*estimate_t) & (t >= (2-1.00001)*estimate_t);
   % Take measurements @ t
   % Take Plant Outputs
   sim_i = Out_Temp(EST_MeasurementOrder(:)).';
   % Find Predicted System Outputs by using ANN
   % Find the reflux ratio interval and max. and min. values
[i_max, i_min, o_max, o_min]=find_R_interval(R_Ratio_p, ...
constant_R_Ratio, Rpercent);
   % Normalize the network inputs
   [sim_i_norm] = normalize_sim_input(sim_i, i_max, i_min);
   % Simulate the network that was created previously
 [sim_output] = simulate(R_Ratio_p, constant_R_Ratio, Rpercent, sim_i_norm);
   % Unnormalize the network outputs
   [sim_output] = unnormalize_sim_output(sim_output, o_max, o_min);
   % Calculate the estimation errors in distillate compositions
%  if the correction times are to be specified
%  [sim_output] = refine_error(t, NT, sim_output, X_frac);
%  if corrections will be done when the error is above its tolerance level
% [sim_output,est_time, flag] = refine_error_tol(t, NT, sim_output, X_frac,
flag, est_time, est_e);
if (0.25*corr_int <= t < 0.251*corr_int)
corr_int = corr_int + 1;
estim_error(1,:) = sim_output(1,:) - X_frac(NT+2,:);
if (abs(estim_error(1,1))|abs(estim_error(1,2))|abs(estim_error(1,3))...
|abs(estim_error(1,4)) >= 1.0000e-003)
est_error = estim_error;
end;
end;
   % Correct the estimated compositions
sim_output = sim_output - est_error;
EST_X = sim_output;

% Run controller (Find new real plant inputs)
 switch  Type_Controller
 case Cont_OL  % Run the open-loop system with using predefined switching
%times and corresponding distillate flowrates (or reflux ratios)
 [CONT_QBoiler, CONT_RRatioinv, Tank_Activated] = CONTROL(t, X_frac, ...
Q_Boiler, R_Ratio_inv, CONT_SetPoints, CONT_Num_Oper_Stage, ...
CONT_DistillProfile);
 case Cont_CL_ActualFB % Run the closed-loop system with the actual
%composition feed-back
 [CONT_QBoiler, CONT_RRatioinv, Tank_Activated, CONT_Curr_Stage] = ...
CONTROL_real(t, X_frac, Q_Boiler, R_Ratio_inv, CONT_SetPoints, ...
CONT_Num_Oper_Stage, CONT_DistillProfile, Tank_Activated, Tank_X_frac, ...
CONT_Curr_Stage);
 case Cont_CL_EstFB% Run the CL system with the estimated comp. feed-back
 [CONT_QBoiler, CONT_RRatioinv, Tank_Activated, CONT_Curr_Stage] = ...
CONTROL_real(t, EST_X, Q_Boiler, R_Ratio_inv, CONT_SetPoints, ...
CONT_Num_Oper_Stage, CONT_DistillProfile,Tank_Activated,EST_Tank_X_frac, ...
CONT_Curr_Stage);
 otherwise
 error('Type of controller doesn''t match [Cont_Plant_Mfile]');
 end
% Set next measurement (or update) time
estimate_t = estimate_t + estimate_DeltaT;
end;
%############## Estimator and Controller Simulation Ends ###########%

if t >= disp_t;
```

```
% Write plant data to Screen
write_plant_to_scr(t, X_frac);
% Write estimator and controller data to Screen
write_estcont_to_scr(t, EST_X);
% Write plant data to file
write_plant_to_file(t, X_frac, Y_frac, Temp, M_Holdup, L_flow, V_flow);
% Write estimator and controller data to file
write_estcont_to_file(t, CONT_QBoiler, CONT_RRatioinv, EST_X);
% Write tank compositions and holdups
%Tank_X_frac
EST_Tank_X_frac
write_tank_to_file(t, Tank_X_frac, Tank_M_Holdup, EST_Tank_X_frac, ...
EST_Tank_M_Holdup);
disp_t = disp_t + disp_DeltaT;
end;
% Stop simulation when the distillation finishes
if (Tank_Activated == -1)
break;
end;
%################### Real Plant Simulation Starts #############%
% Calculate current holdup amount and composition in storage tanks
[Tank_X_frac, Tank_M_Holdup] = P_Calc_Tanks(t, DeltaT, Tank_X_frac, ...
Tank_M_Holdup, X_frac(NT+2,:), D_DistillRate, Tank_Activated);
% Calculate current amount of product distilled
[M_Distilled] = P_Calc_Distilled_Amount(t,DeltaT,D_DistillRate,M_Distilled);
% Calculate current Still Pot Holdup
[M_Holdup] = P_Calc_Mol_Still_Holdup(t, M_Holdup, M_Distilled);
% Take Integration
[t_dummy, X_frac] = P_Int_Euler(t, DeltaT, X_frac, DX_frac);
% Normalizes Plant States
[X_frac] = P_Normalize_States(t, X_frac);
%################ Real Plant Simulation Ends ################%
%################### Estimator Starts ###################%
% Estimate current holdup amount and composition in storage tanks
[EST_Tank_X_frac, EST_Tank_M_Holdup] = EST_Tanks(t, DeltaT, ...
EST_Tank_X_frac, EST_Tank_M_Holdup,
EST_X(1,:),D_DistillRate,Tank_Activated);
% Normalize States
% [EST_X] = EST_Norm_States(t, EST_X);
%################ Estimator Ends ####################%

%################ Set Current Time ######################%
t = t + DeltaT;
%################ Set Current Time ######################%
% Manipulate real plant inputs by controller outputs
Q_Boiler = CONT_QBoiler;
R_Ratio_inv = CONT_RRatioinv;
end;
% Calculate the Capacity Factor
CAP = (Tank_M_Holdup(1,1)+Tank_M_Holdup(3,1)+Tank_M_Holdup(5,1)+ ...
M_Holdup(1,1))/t;
% Write Capacity Factor to file
write_cap_to_file(t,Tank_M_Holdup(1,1),Tank_M_Holdup(2,1), ...
Tank_M_Holdup(3,1),Tank_M_Holdup(4,1),Tank_M_Holdup(5,1), ...
Tank_M_Holdup(6,1),M_Holdup(1,1),CAP);
% ================================================
% Close Output Files
% ================================================
% Liquid Profile file
fclose(FID_lprofile);
% Vapor Profile file
fclose(FID_vprofile);
% Temperature Profile file
fclose(FID_tprofile);
% Holdup Profile file
fclose(FID_holdup);
% Liquid and Vapor Flowrate Profile file
```

```
fclose(FID_lvflow);
% Controller Outputs file
fclose(FID_control);
% Estimator Outputs file
fclose(FID_estimator);
% Tank Outputs file
fclose(FID_tank);
% Capacity Factor file
fclose(FID_cap);
% ================================================
% Simulation finishes
% ================================================
fprintf('Simulation finished successfully.\n\n');
% ----------------------------End of Main Program function Cont_Plant_Mfile

% -----------------Real Plant Simulation Functions   --------------------- %
%================================================
% P_Calc_Phys_Vars
% Return
%   Find current physical variables
% given
%   Time, t; Previous system variables (Temp at previous time step is for
%initial guess)
%================================================
function [Y_frac, Temp, H_l_Enthalpy, H_v_Enthalpy, Ro_l_Density, ...
Mw_MolWeight] = P_Calc_Phys_Vars(t, X_frac, Press, TempPr)
Glob_Decs;
%% Calculated only by using liquid composition and Pressure
% Set sizes
Y_frac = zeros(size(X_frac));
Temp = zeros(size(TempPr));
H_l_Enthalpy = zeros(size(TempPr));
H_v_Enthalpy = zeros(size(TempPr));
Ro_l_Density = zeros(size(TempPr));
Mw_MolWeight = zeros(size(TempPr));
%  ------------------------------------------  Buble Point Calculation
% Calculate still pot, tray(s),reflux drum temp.&vapor comps(moles/moles)
for i=1:NT+2;
[Temp(i), Dummy3] = thermo_Equilibrium(TempPr(i), Press(i), X_frac(i,:));
Y_frac(i,:) = Dummy3;
if (i>1) & (i<NT+2);
Y_frac(i,:) = Y_frac(i-1,:) + Eff(i-1,:).*(Y_frac(i,:) - Y_frac(i-1,:));
end;
end;
%  ----------------------------------  Calculate specific phase enthalpies
% Calculate still pot,tray(s),reflux drum liq.-vap.phase enthalpies(J/moles)
for i=1:NT+2;
[H_l_Enthalpy(i), H_v_Enthalpy(i)] = thermo_Enthalpy(Temp(i), Press(i),
X_frac(i,:), Y_frac(i,:));
end;
%  --------Caculate liquid phase average desity and average molecular weight
% Calculate still pot,tray(s),reflux drum liq. phase avrg. densities (kg/m3)
% Calculate still pot,tray(s),reflux drum avg. molecular weight (kg/mol)
for i=1:NT+2;
[Mw_MolWeight(i), Ro_l_Density(i)] = thermo_Density(Temp(i), Press(i),
X_frac(i,:));
end;
%end P_Calc_Phys_Vars

%================================================
% P_Calc_Mol_Tray_Holdups
% Return
%   Calculate current trays molar holdups
% given
%   Time, t; Avg. Density, Ro_l_Density; Avg. Molecular Weight,
Mw_MolWeight;
%================================================
```

```
function [M_Holdup] = P_Calc_Mol_Tray_Holdups(t, Ro_l_Density, ...
Mw_MolWeight, M_Holdup_pr)
Glob_Decs;
% Set size
M_Holdup = M_Holdup_pr;
% Calculate tray(s) molar holdups (mol)
for i=2:NT+1;
M_Holdup(i) = (Ro_l_Density(i) / Mw_MolWeight(i)) * Tray_Vol_Holdup;
end;
%end P_Calc_Mol_Tray_Holdups

%=====================================================
% P_Calc_Mol_Drum_Holdup
% Return
%   Calculate current Reflux-Drum molar holdup
% given
%   Time, t; Previous Holdups, M_Holdup_pr;
%=====================================================
function [M_Holdup] = P_Calc_Mol_Drum_Holdup(t, M_Holdup_pr)
Glob_Decs;
% Set size
M_Holdup = M_Holdup_pr;
% Reflux Drum Molar holdup is constant
M_Holdup(NT+2) = M_Holdup_pr(NT+2);
%end P_Calc_Mol_Drum_Holdup

%=====================================================
% P_Calc_Apprx_Deriv
% Return
%   Calculate Approximated derivatives
% given
%   Time, t;
%=====================================================
function [Del_M_Holdup,Del_H_l_Enthalpy,Del_M_Hl]= P_Calc_Apprx_Deriv(t, ...
M_Holdup, H_l_Enthalpy, t_prv, M_Holdup_prv, H_l_Enthalpy_prv)
Glob_Decs;
% Set sizes
Del_M_Holdup = zeros(size(M_Holdup));
Del_H_l_Enthalpy = zeros(size(H_l_Enthalpy));
Del_M_Hl = zeros(size(M_Holdup));
% Calculate step size
del_t = t - t_prv;
% d(M_Holdup)/dt approximated by forward differentiation
Del_M_Holdup = (M_Holdup - M_Holdup_prv) / del_t;
% d(H_l_Enthalpy)/dt approximated by forward differentiation
Del_H_l_Enthalpy = (H_l_Enthalpy - H_l_Enthalpy_prv) / del_t;
% d(M_Holdup*H_l_Enthalpy)/dt approximated by forward differentiation
Del_M_Hl = (M_Holdup.*H_l_Enthalpy - M_Holdup_prv.*H_l_Enthalpy_prv)/del_t;
%end P_Calc_Apprx_Deriv

%=====================================================
% P_Calc_LV_for_Total_Reflux
% Return
%   Calculates liquid and vapor flow rates for total reflux condition
%   or for D=0
% given
%   Time, t;
%=====================================================
function [D_DistillRate, L_flow, V_flow] = P_Calc_LV_for_Total_Reflux(t, ...
H_l_Enthalpy, H_v_Enthalpy, M_Holdup, Del_M_Holdup, Del_H_l_Enthalpy, ...
Del_M_Hl, Q_Boiler, R_Ratio_inv, rate_ro_mw, epsilon_t)
Glob_Decs;
% set sizes
D_DistillRate = zeros(size(1,1));
L_flow = zeros(size(M_Holdup));
V_flow = zeros(size(M_Holdup));
% Calculate distillate flow rate
```

```
D_DistillRate = 0.0;
% Calculate liquid flow rate from reflux drum
sum_Del_M_Holdup = sum( Del_M_Holdup(2:NT+1) );
sum_Del_M_Hl = sum( Del_M_Hl(2:NT+1) );
L_flow(NT+2)=(Q_Boiler+H_l_Enthalpy(1)*sum_Del_M_Holdup-M_Holdup(1)* ...
Del_H_l_Enthalpy(1)-sum_Del_M_Hl+epsilon_t*rate_ro_mw(NT+2)* ...
M_Holdup(NT+2)* H_v_Enthalpy(NT+1))/(H_v_Enthalpy(NT+1)-H_l_Enthalpy(NT+2));
% fprintf(['Qb=' num2str(Q_Boiler) '\n']);
% fprintf(['h(1)=' num2str(H_l_Enthalpy(1)) '   SumDel_M=' ...
num2str(sum_Del_M_Holdup) '\n']);
% fprintf(['M(1)=' num2str(M_Holdup(1)) '   Del_Hl(1)=' ...
num2str(Del_H_l_Enthalpy(1)) '\n']);
% fprintf(['SumDel_Mh=' num2str(sum_Del_M_Hl) '\n']);
% fprintf(['H(NT+1)=' num2str(H_v_Enthalpy(NT+1)) '   h(NT+2)=' ...
num2str(H_l_Enthalpy(NT+2)) '\n']);
% Calculate vapor flow rate from top tray
V_flow(NT+1) = L_flow(NT+2) - epsilon_t*rate_ro_mw(NT+2)*M_Holdup(NT+2);
% % % % % % fprintf(['V' num2str(NT+1) '=' num2str(V_flow(NT+1)) '   L'...
num2str(NT+2) '=' num2str(L_flow(NT+2)) '\n']);
% Calculate other vapor and liquid flow rates
for i=NT+1:-1:2;
V_flow(i-1)=(V_flow(i)*(H_v_Enthalpy(i)- H_l_Enthalpy(i)) + L_flow(i+1)* ...
(H_l_Enthalpy(i) - H_l_Enthalpy(i+1))+ M_Holdup(i)*Del_H_l_Enthalpy(i) + ...
H_l_Enthalpy(i)*epsilon_t*rate_ro_mw(i)*M_Holdup(i)) / ...
(H_v_Enthalpy(i-1) - H_l_Enthalpy(i));
L_flow(i) = V_flow(i-1) + L_flow(i+1) - V_flow(i) - Del_M_Holdup(i)+ ...
epsilon_t*rate_ro_mw(i)*M_Holdup(i);
% % % fprintf(['V' num2str(i-1) '=' num2str(V_flow(i-1)) '   L' ...
num2str(i) '=' num2str(L_flow(i)) '\n']);
end;
% pause
%end P_Calc_LV_for_Total_Reflux

%=================================================
% P_Calc_LV_for_Finite_Reflux
% Return
%   Calculates liquid and vapor flow rates for distillate withdrawal
%   or for D!=0 or D different than zero.
% given
%   Time, t;
%=================================================
function [D_DistillRate,L_flow, V_flow] = P_Calc_LV_for_Finite_Reflux(t, ...
H_l_Enthalpy, H_v_Enthalpy, M_Holdup, Del_M_Holdup, Del_H_l_Enthalpy, ...
Del_M_Hl, Q_Boiler, R_Ratio_inv, rate_ro_mw, epsilon_t)
Glob_Decs;
% set sizes
D_DistillRate = zeros(size(1,1));
L_flow = zeros(size(M_Holdup));
V_flow = zeros(size(M_Holdup));
% Calculate distillate flow rate
R = 1 / R_Ratio_inv;
Dummy1 =  sum(Del_M_Hl(1:NT+1));
D_DistillRate = ( Q_Boiler - Dummy1 + epsilon_t*rate_ro_mw(NT+2)* ...
M_Holdup(NT+2)*(H_v_Enthalpy(NT+1)-H_l_Enthalpy(NT+2))) / (...
(R+1)*H_v_Enthalpy(NT+1) - R*H_l_Enthalpy(NT+2) );
% Calculate vapor flow rate from top tray
V_flow(NT+1)= D_DistillRate*(R+1)-epsilon_t*rate_ro_mw(NT+2)*M_Holdup(NT+2);
% Calculate liquid flow rate from reflux drum
L_flow(NT+2) = D_DistillRate * R;
% Calculate other vapor and liquid flow rates
for i=NT+1:-1:2;
V_flow(i-1) = ( V_flow(i)*(H_v_Enthalpy(i) - H_l_Enthalpy(i)) + ...
L_flow(i+1)*(H_l_Enthalpy(i) - H_l_Enthalpy(i+1))+ M_Holdup(i)* ...
Del_H_l_Enthalpy(i)+ H_l_Enthalpy(i)*epsilon_t*rate_ro_mw(i)* ...
M_Holdup(i)) / (H_v_Enthalpy(i-1) - H_l_Enthalpy(i));
L_flow(i) = V_flow(i-1) + L_flow(i+1) - V_flow(i) - Del_M_Holdup(i) + ...
epsilon_t*rate_ro_mw(i)*M_Holdup(i);
```

```
end;
%end P_Calc_LV_for_Finite_Reflux

%================================================
% P_Calc_Distilled_Amount
% Return
%   Calculate current amount of product distilled
% given
%   Time, t;
%================================================
%function [M_Distilled] = P_Calc_Distilled_Amount(t,t_prv,D_DistillRate, ...
M_Distilled_pr)
function [M_Distilled] = P_Calc_Distilled_Amount(t,delta_t,D_DistillRate,...
M_Distilled_pr)
Glob_Decs;
% set size
M_Distilled = 0.0;
% Calculate amount of product distilled
M_Distilled = M_Distilled_pr + D_DistillRate*delta_t;
%end P_Calc_Distilled_Amount

%================================================
% P_Calc_Mol_Still_Holdup
% Return
%   Calculate Still Pot Holdup
% given
%   Time, t;
%================================================
function [M_Holdup] = P_Calc_Mol_Still_Holdup(t, M_Holdup_pr, M_Distilled);
Glob_Decs;
% Set sizes
M_Holdup = M_Holdup_pr;
% Instantaneous still pot total holdup amount (moles/hour)
M_Holdup(1) = M_Feed-sum(M_Holdup_pr(2:NT+1))-M_Holdup_pr(NT+2)-M_Distilled;
%end P_Calc_Mol_Still_Holdup

%================================================
% P_f
% Return the derivatives for the continuous states.
%================================================
function [DX_frac] = P_f(t, X_frac, Y_frac, Temp, M_Holdup, L_flow, ...
V_flow, Q_Boiler, D_DistillRate, R_Ratio_inv, rate_ro_mw, epsilon_t)
Glob_Decs;
DX_frac = zeros(size(X_frac));
% Instantaneous still pot,tray(s),reflux drum liquid compositions
%derivatives (moles/moles/hour)
for j=1:NC;
% still pot
DX_frac(1,j) = (( L_flow(2)*(X_frac(2,j)-X_frac(1,j)) - ...
V_flow(1)*(Y_frac(1,j)-X_frac(1,j)) ) / M_Holdup(1)) + ...
(epsilon(j)*rate_ro_mw(1)-X_frac(1,j)*epsilon_t*rate_ro_mw(1));
% tray(s)
for i=2:NT+1;
DX_frac(i,j) = (( V_flow(i-1)*(Y_frac(i-1,j)-X_frac(i,j)) + ...
L_flow(i+1)*(X_frac(i+1,j)-X_frac(i,j)) - V_flow(i)*(Y_frac(i,j)- ...
X_frac(i,j)) ) / M_Holdup(i)) + (epsilon(j)*rate_ro_mw(i)- ...
X_frac(i,j)*epsilon_t*rate_ro_mw(i));
end;
% reflux drum
DX_frac(NT+2,j) =  (( V_flow(NT+1)*(Y_frac(NT+1,j)-X_frac(NT+2,j)) ) / ...
M_Holdup(NT+2)) + (epsilon(j)*rate_ro_mw(NT+2)- ...
X_frac(NT+2,j)*epsilon_t*rate_ro_mw(NT+2));
end;
% end mdlDerivatives

%================================================
% P_Keep_Current_Vars
```

```
% Return
%   Keeps current parameters for future use
% given
%   Time: t; Variables at t: ...;
%=====================================================
function [t_prv, M_Holdup_prv, H_l_Enthalpy_prv]= P_Keep_Current_Vars(t, ...
M_Holdup, H_l_Enthalpy);
% Previous step time
t_prv = t;
% Previous step still pot, tray(s), reflux drum total holdup amount (moles)
M_Holdup_prv = M_Holdup;
% Previous step still pot,tray(s),reflux drum liq. phase enthalpy (J/moles)
H_l_Enthalpy_prv = H_l_Enthalpy;
%end P_Keep_Current_Vars

%=====================================================
% P_Int_Euler
% Return
%   Take integral
% given
%   Time: t; Integration Step: delta_t; Previous States: X_frac;
%   Derivatives: DX_frac;
%=====================================================
function [t_new, X_frac_new] = P_Int_Euler(t, delta_t, X_frac, DX_frac)
t_new = t + delta_t;
X_frac_new = X_frac + DX_frac*delta_t;
%end Int_Euler

%=====================================================
% P_Normalize_States
% Return
%   Normalizes Plant States
% given
%   Time: t;States: X_frac_Pr;
%=====================================================
function [X_frac] = P_Normalize_States(t, X_frac_Pr)
Glob_Decs;
% Set size
X_frac = X_frac_Pr;
% %%%%%%%%%%%%%%   Make the low compositions zero %%%%%%%%
% Check for still pot,tray(s),reflux drum liquid compositions (moles/moles)
for j=1:NC;
if isnan(X_frac_Pr(1,j));
error(['Stage no ', num2str(1) ,' component ', num2str(j), ' liquid fraction
is Nan'] );
end;
if X_frac_Pr(1,j) < zero_tolerance;
display(['Stage no ',num2str(1),' composition of comp. ', num2str(j), ' (...
',num2str(X_frac_Pr(1,j)),' )  made zero']);
X_frac(1,j) = 0.0;
end;
end;
% %%%%%%% Normalize the liquid compositions (moles/moles) %%%%%%%%%%
% Normalize still pot,tray(s),reflux drum liquid compositions (moles/moles)
dummy1 = sum(X_frac_Pr(1,:));
if ~(dummy1 > 0.0);
error(['Sum of comp. fraction in the Stage no  ',num2str(1),'  is zero. ',
num2str(X_frac_Pr(1,:)) ]);
else
X_frac(1,:) = X_frac_Pr(1,:) / dummy1;
end;
% %%%%%%%%%% Check compositions are in the limit of [0,1] %%%%%%%%%
% Check still pot,tray(s),reflux drum liquid compositions (moles/moles)
for j=1:NC;
if (X_frac(1,j)<0 | X_frac(1,j)>1);
error(['Composition out of limit ! - [Normalize_States]
X_frac(',num2str(1),',:) = ', num2str(X_frac(1,:))] );
```

```
end;
end;
%end P_Normalize_States

%=================================================
% P_Calc_Tanks
% Return
%   Storage Tank Holdups and Compositions
%=================================================
function [X, M] = P_Calc_Tanks(t, DeltaT, X, M, X_Drum, D_Rate, Active);
if Active~=0;
% Calculate increase in holdup
deltaM = D_Rate*DeltaT;
% Calculate tank's composition
if ((deltaM == 0) & M(Active, 1)==0)
X(Active, :) = zeros(size(X(Active, :)));
else
X(Active, :) = ( X(Active, :) * M(Active, 1) + X_Drum(1, :) * deltaM )/( ...
M(Active, 1) + deltaM );
end;
% Calculate tank's current holdup
M(Active, 1) = M(Active, 1) + D_Rate*DeltaT;
end;
%end P_Calc_Tanks
% --------------------End Real Plant Simulation Functions   -------------- %

% -------------------------  Estimator Functions  ------------------------%
% ================================================
% INIT_ESTIMATE (don't modify global variables not owned by this function)
% perform
%   Initialize esimator
% given
%   All Global variables
% output
%   any output required
%=================================================
function [outputs] = INIT_ESTIMATE
Glob_Decs;
outputs = [];
% end INIT_ESTIMATE
% ================================================
%ESTIMATE (don't modify global variables not owned by this function)
% perform
%   Estimates the states of the system
% given
%   temperatures on selected trays
% output
%   distillate compositions
% ================================================
function [outputs] = ESTIMATE(t)
Glob_Decs;
outputs = [];
% end ESTIMATE
%=================================================
% EST_Norm_States
% Return
%   Normalizes Plant States
% given
%   Time: t;States: X_frac_Pr;
%=================================================
function [X_frac] = EST_Norm_States(t, X_frac_Pr)
Glob_Decs;
% Set size
X_frac = X_frac_Pr;
% %%%%%%%%%%%%%%   Make the low compositions zero %%%%%%%%
% Check for still pot,tray(s),reflux drum liquid compositions (moles/moles)
for j=1:NC;
```

```
if isnan(X_frac_Pr(1,j));
error(['Stage no ', num2str(1) ,' component ', num2str(j), ' ...
liquid fraction is Nan'] );
end;
if X_frac_Pr(1,j) < zero_tolerance;
 % display(['Stage no ',num2str(1),' composition of comp. ', num2str(j), ...
' ( ',num2str(X_frac_Pr(1,j)),' )  made zero']);
X_frac_Pr(1,j) = 0.0;
end;
end;
% %%%%%%%% Normalize the liquid compositions (moles/moles) %%%%%%%%
% Normalize still pot,tray(s),reflux drum liquid compositions (moles/moles)
dummy1 = sum(X_frac_Pr(1,:));
if ~(dummy1 > 0.0);
error(['Sum of comp. fraction in the Stage no  ',num2str(1),' ...
is zero. ', num2str(X_frac_Pr(1,:)) ]);
else
X_frac(1,:) = X_frac_Pr(1,:) / dummy1;
end;
% %%%%%%%%%%%% Check compositions are in the limit of [0,1] %%%%%%%%
% Check still pot, tray(s), reflux drum liquid compositions (moles/moles)
for j=1:NC;
if (X_frac(1,j)<0 | X_frac(1,j)>1);
error(['Composition out of limit ! - [Normalize_States]
X_frac(',num2str(1),',:) = ', num2str(X_frac(1,:))] );
end;
end;
%end EST_Norm_States
%==================================================
% EST_Tanks
% Return
% Storage Tank Holdups and Compositions
%==================================================
function [X, M] = EST_Tanks(t, DeltaT, X, M, X_Drum, D_Rate, Active);
if Active~=0;
% Calculate increase in holdup
deltaM = D_Rate*DeltaT;
% Calculate tank's composition
if ((deltaM == 0) & M(Active, 1)==0)
X(Active, :) = zeros(size(X(Active, :)));
else
X(Active, :) = ( X(Active, :) * M(Active, 1) + X_Drum(1, :) * deltaM ) /...
( M(Active, 1) + deltaM );
end;
% Calculate tank's current holdup
M(Active, 1) = M(Active, 1) + D_Rate*DeltaT;
end;
%end EST_Tanks
% ------------------  End Estimator Functions  ------------------------ %

% -------------------  Controller Functions  --------------------------- %
%==================================================
% INIT_CONTROL   (don't MODIFY GLOBAL VARIABLES not owned by this function)
% perform
%   Initilialize controller
% given
%   All Global variables
% output
%   any output required
% Number of operating stages is increased for a four component system
%==================================================
function [SetPoints, Num_Oper_Stage, DistillProfile] = INIT_CONTROL
Glob_Decs;
% Controller Set Points of Product Specifications
SetPoints   =  zeros(NC,1);
%SetPoints   = [0.52; 0.5; 0.65; 0.999];
```

```
% Estimator predicts only distillate comps,so use AcAc dist.setpoint that
%corresponds to AcAc reboiler setpoint (0.999)
SetPoints   = [ 0.52; 0.5; 0.645; 0.272865];
% Number of different operation stage
Num_Oper_Stage  =  7;
% Distillate Flow rate values of different operation stages (Products and/or
%Slopcuts)
DistillProfile  =  zeros(Num_Oper_Stage,1);
%end INIT_CONTROL

%=================================================
% CONTROL_real   (don't MODIFY GLOBAL VARIABLES not owned by this function)
% perform
%   Controls the system
% given
%   Current Time, t; All Global variables at time t;
% output
%   any output required
%=================================================
function [Q_Boiler, R_Ratio_inv, Tank_Active, CONT_Curr_Stage_new] = ...
CONTROL_real(t, X_frac, Q_Boiler_pr, R_Ratio_inv_pr, CONT_SetPoints, ...
CONT_Num_Oper_Stage, CONT_DistillProfile, Tank_Active_prv, X_tank, ...
CONT_Curr_Stage)
Glob_Decs;
% Find new Reflux ratio (L0/D)
% here X_frac (distillate compositions) is a 1x4 matrix!
if (CONT_Curr_Stage==0) %----------- Total Reflux operation
if (X_frac(1,1)<CONT_SetPoints(1,1))
R_Ratio = -1;
Tank_Active = 0;
CONT_Curr_Stage = 0;
elseif (X_frac(1,1)>=CONT_SetPoints(1,1))
'0->1'
CONT_Curr_Stage = 1;
end;
end;
if (CONT_Curr_Stage==1)%1st product-cut distillation to 1st product-cut tank
if ((X_tank(1,1)==0) | (X_tank(1,1)>=CONT_SetPoints(1,1)))
R_Ratio = 0.891382/(1.0-0.891382);
Tank_Active = 1;
CONT_Curr_Stage = 1;
elseif (X_tank(1,1)<CONT_SetPoints(1,1))
'1->2'
CONT_Curr_Stage = 2;
end;
end;
if (CONT_Curr_Stage==2) % 1st slop-cut distillation to 1st slop-cut tank
if (X_frac(1,2)<CONT_SetPoints(2,1))
  R_Ratio = 0.675/(1.0-0.675);
  Tank_Active = 2;
  CONT_Curr_Stage = 2;
elseif (X_frac(1,2)>=CONT_SetPoints(2,1))
   '2->3'
CONT_Curr_Stage = 3;
end;
end;
if (CONT_Curr_Stage==3)%2nd product-cut distillation to 2nd product-cut tank
    if ((X_tank(3,2)==0) | (X_tank(3,2)>=CONT_SetPoints(2,1)))
        R_Ratio = 0.764999/(1.0-0.764999);
        Tank_Active = 3;
        CONT_Curr_Stage = 3;
    elseif (X_tank(3,2)<CONT_SetPoints(2,1))
        '3->4'
        CONT_Curr_Stage = 4;
    end;
end;
if (CONT_Curr_Stage==4)   % 2nd slop-cut distillation to 2nd slop-cut tank
```

```
    if (X_frac(1,3)<CONT_SetPoints(3,1))
        R_Ratio = 0.659999/(1.0-0.659999);
        Tank_Active = 4;
        CONT_Curr_Stage = 4;
    elseif (X_frac(1,3)>=CONT_SetPoints(3,1))
        '4->5'
        CONT_Curr_Stage = 5;
    end;
end;
if (CONT_Curr_Stage==5)%3rd product-cut distillation to 3rd product-cut tank
    if ((X_tank(5,3)==0) | (X_tank(5,3)>=CONT_SetPoints(3,1)))
        R_Ratio = 0.799/(1.0-0.799);
        Tank_Active = 5;
        CONT_Curr_Stage = 5;
    elseif (X_tank(5,3)<CONT_SetPoints(3,1))
        '5->6'
        CONT_Curr_Stage = 6;
    end;
end;
if (CONT_Curr_Stage==6) % 3rd slop-cut distillation to 3rd slop-cut tank
% In reality, the fraction of the total amount of the component in the whole
%column is to be checked
% Estimator predicts only distillate comps,so use AcAc dist.setpoint that
%corresponds to AcAc reboiler setpoint (0.999)
    if (X_frac(1,4)<CONT_SetPoints(4,1))
        R_Ratio = 0.9561/(1.0-0.9561);
        Tank_Active = 6;
        CONT_Curr_Stage = 6;
    elseif (X_frac(1,4)>=CONT_SetPoints(4,1))
        '6->7'
        CONT_Curr_Stage = 7;
    end;
end;
if (CONT_Curr_Stage==7)         %----------- Distillation stops
    R_Ratio = 1/R_Ratio_inv_pr;
    Tank_Active = -1;
end;
% Keep Current Stage #
CONT_Curr_Stage_new = CONT_Curr_Stage;
% Convert Reflux ratio (L0/D) to One Over Reflux ratio (D/L0)
if (R_Ratio==-1)
    R_Ratio_inv = 0;
else
    R_Ratio_inv = 1.0 / R_Ratio;
end;
% Find Reboiler load (J/hour)
    % Constant Reboiler Load
    Q_Boiler = Q_Boiler_pr;
%end CONTROL_real

%=====================================================
% CONTROL           (don't MODIFY GLOBAL VARIABLES not owned by this
function)
% perform
%           Controls the system
% given
%           Current Time, t; All Global variables at time t;
% output
%           any output required
%=====================================================
function [Q_Boiler, R_Ratio_inv, Tank_Active] = CONTROL(t, X_frac, ...
Q_Boiler_pr, R_Ratio_inv_pr, CONT_SetPoints, CONT_Num_Oper_Stage, ...
CONT_DistillProfile)
Glob_Decs;
if constant_R_Ratio ==1.0
 if (t < 9.15)
   R_Ratio = -1;
```

```
   Tank_Active = 0;
 elseif (t >= 9.15)
    R_Ratio_p = 0.83;
    R_Ratio = R_Ratio_p/(1-R_Ratio_p);
    Tank_Active = 1;
 end;
end;
if constant_R_Ratio == 0.0
%Find new Reflux ratio (L0/D)
    if (t<9.15)
        R_Ratio = -1;
        Tank_Active = 0;
    elseif (t>=9.15 & t<27.7512)
        R_Ratio = 0.891382/(1.0-0.891382);
        R_Ratio = R_Ratio + R_Ratio*Rpercent/100.0;
        Tank_Active = 1;
    elseif (t>=27.7512 & t<29.4318)
        R_Ratio = 0.675/(1.0-0.675);
        R_Ratio = R_Ratio + R_Ratio*Rpercent/100.0;
        Tank_Active = 1;
    elseif (t>=29.4318 & t<32.3259)
        R_Ratio = 0.764999/(1.0-0.764999);
        R_Ratio = R_Ratio + R_Ratio*Rpercent/100.0;
        Tank_Active = 1;
    elseif (t>=32.3259 & t<35.1237)
        R_Ratio = 0.659999/(1.0-0.659999);
        R_Ratio = R_Ratio + R_Ratio*Rpercent/100.0;
        Tank_Active = 1;
    elseif (t>=35.1237 & t<36.2127)
        R_Ratio = 0.799/(1.0-0.799);
        R_Ratio = R_Ratio + R_Ratio*Rpercent/100.0;
        Tank_Active = 1;
    elseif (t >= 36.2127)
        R_Ratio = 0.9561/(1.0-0.9561);
        R_Ratio = R_Ratio + R_Ratio*Rpercent/100.0;
        Tank_Active = 1;
    end;
end;
% Convert Reflux ratio (L0/D) to One Over Reflux ratio (D/L0)
if (R_Ratio==-1)
    R_Ratio_inv = 0;
else
    R_Ratio_inv = 1.0 / R_Ratio;
end;
% Find Reboiler load (J/hour)
    % Constant Reboiler Load
    Q_Boiler = Q_Boiler_pr;
%end CONTROL
% ------------------  End Controller Functions  ---------------------- %

% -------       Simulation loop control user interface functions    ------ %
%=====================================================
% write_plant_to_scr
%=====================================================
function write_plant_to_scr(t, X_frac)
Glob_Decs;
    % Reflux Drum Liquid composition
    data = X_frac(NT+2,:);
    l = prod(size(data));
    formats = ''; for i=1:l; formats = [formats ' %f']; end;
    fprintf(['%9.4f' '      Real: ' formats ' '], t, reshape(data, 1, ...
prod(size(data))) );
%end write_plant_to_scr

%=====================================================
% write_estcont_to_scr
%=====================================================
```

```
function write_estcont_to_scr(t, EST_X)
Glob_Decs;
% Estimated Reflux Drum Liquid composition
dummy = EST_X(1,:) ;
l = prod(size(dummy));
formats = ''; for i=1:l; formats = [formats ' %f']; end;
fprintf(['    Estd: ' formats '\n'], reshape(dummy, 1, prod(size(dummy))) );
%end write_estcont_to_scr

%=================================================
% write_plant_to_file
%=================================================
function  write_plant_to_file(t,X_frac,Y_frac,Temp,M_Holdup, L_flow, V_flow)
Glob_Decs;
% Liquid Profile file
l = prod(size(X_frac));
formats = ''; for i=1:l; formats = [formats '; %f']; end;
fprintf(FID_lprofile, ['%9.4f' formats '\n'], t, reshape(X_frac, 1, l));
% Vapor Profile file
l = prod(size(Y_frac));
formats = ''; for i=1:l; formats = [formats '; %f']; end;
fprintf(FID_vprofile, ['%9.4f' formats '\n'], t, reshape(Y_frac, 1, l));
% Temperature Profile file
l = prod(size(Temp));
formats = ''; for i=1:l; formats = [formats '; %f']; end;
fprintf(FID_tprofile, ['%9.4f' formats '\n'], t, reshape(Temp, 1, l));
% Holdup Profile file
l = prod(size(M_Holdup));
formats = ''; for i=1:l; formats = [formats '; %f']; end;
fprintf(FID_holdup, ['%9.4f' formats '\n'], t, reshape(M_Holdup, 1, l));
% Liquid and Vapor Flowrate Profile file
l = prod(size(L_flow)) +  prod(size(V_flow));
formats = ''; for i=1:l; formats = [formats '; %f']; end;
fprintf(FID_lvflow, ['%9.4f' formats '\n'],t,reshape([L_flow;V_flow],1,l));
%end write_plant_to_file

%=================================================
% write_estcont_to_file : Write estimator and controller data to file
%=================================================
function write_estcont_to_file(t, CONT_QBoiler, CONT_RRatioinv, EST_X);
Glob_Decs;
% Controller Outputs file
l = prod(size(CONT_QBoiler)) + prod(size(CONT_RRatioinv));
formats = ''; for i=1:l; formats = [formats '; %f']; end;
fprintf(FID_control, ['%9.4f ' formats '\n'], t, CONT_QBoiler,
CONT_RRatioinv);
% Estimator Outputs file
%   l = prod(size(EST_X)) + prod(size(IAE));
l = prod(size(EST_X));
formats = ''; for i=1:l; formats = [formats '; %f']; end;
%   fprintf(FID_estimator, ['%9.4f ' formats '\n'], t, reshape(EST_X, 1, ...
prod(size(EST_X))), reshape(IAE, 1, prod(size(IAE))) );
fprintf(FID_estimator, ['%9.4f ' formats '\n'], t, reshape(EST_X, 1, ...
prod(size(EST_X))) );
%end write_estcont_to_file

%=================================================
% write_tank_to_file : Write tank data to file
%=================================================
function write_tank_to_file(t, X_actual, M_actual, X_est, M_est);
Glob_Decs;
l = prod(size(X_actual)) + prod(size(M_actual))+prod(size(X_est)) + ...
prod(size(M_est)) ;
formats = ''; for i=1:l; formats = [formats '; %f']; end;
fprintf(FID_tank, ['%9.4f ' formats '\n'], t, reshape(X_actual, 1, ...
prod(size(X_actual))),   reshape(M_actual, 1, prod(size(M_actual))),...
```

```
reshape(X_est, 1, prod(size(X_est))),...
reshape(M_est, 1, prod(size(M_est))));
%end write_tank_to_file

%================================================
% write_cap_to_file : Write capacity factor to file
%================================================
function write_cap_to_file(t, Tank1_Holdup,Tank2_Holdup,Tank3_Holdup, ...
Tank4_Holdup,Tank5_Holdup,Tank6_Holdup,M_Holdup,CAP);
Glob_Decs;
l=prod(size(Tank1_Holdup))+prod(size(Tank2_Holdup))+ ...
prod(size(Tank3_Holdup))+prod(size(Tank4_Holdup))+ ...
prod(size(Tank5_Holdup))+prod(size(Tank6_Holdup))+ ...
prod(size(M_Holdup)),prod(size(CAP));
formats = ''; for i=1:l; formats = [formats '; %f']; end;
fprintf(FID_cap, ['%9.4f ' formats '\n'], t,reshape(Tank1_Holdup, 1, ...
prod(size(Tank1_Holdup))),reshape(Tank2_Holdup, 1, ...
prod(size(Tank2_Holdup))),...
reshape(Tank3_Holdup, 1, prod(size(Tank3_Holdup))),...
reshape(Tank4_Holdup, 1, prod(size(Tank4_Holdup))),...
reshape(Tank5_Holdup, 1, prod(size(Tank5_Holdup))),...
reshape(Tank6_Holdup, 1, prod(size(Tank6_Holdup))),reshape(M_Holdup, 1, ...
prod(size(M_Holdup))),reshape(CAP, 1, prod(size(CAP))));
%end write_opt_to_file
% ------   End Simulation loop control user interface functions    ------- %
```

Pressure_Profile.m

```
%================================================
% PressureProfile
% Return
%           pressure profile through the column (Pa)
% for a given
%           Still pot and reflux drum pressures (Pa)
%================================================
function [fP_Tray]=PressureProfile(fP_Pot,fP_Drum)
Glob_Decs;
if (size(fP_Pot)~=1 | size(fP_Drum)~=1)
    error('fP_Pot and/or fP_Drum are not scalar(s). [PressureProfile]');
end;
fP_Tray = zeros(NT,1);
fdelP_Tray = (fP_Pot-fP_Drum)/NT;
for i=1:NT; fP_Tray(i) = fP_Pot - i*fdelP_Tray; end;
% end PressureProfile
```

C.2 Thermodynamic Library MATLAB Interface Code

thermo_Init.m

```
function thermo_Init(check_input_parameters)
% function thermo_Init(check_input_parameters)
% Thermophysical and physical property calculation MEX File Interface
% ------------- Initialization routine -------------
%  if check_input_parameters = 1 then initialization routine writes the
%  input parameters read from 'plant_data.dat' to 'plant_data_check.dat'
thermo_LIBRARY('init',check_input_parameters);
```

thermo_Equilibrium.m

```
function [Tequi, y] = thermo_Equilibrium(T,P,x)
% function [Tequi, y] = thermo_Equilibrium(T,P,x)
% Thermophysical and physical property calculation MEX File Interface
% ------------- Equilibrium routine -------------
%[Tequi,y]: Equilibrium temperature(K), Equil. Vap. phase fractions(mol/mol)
```

```
%(T,P,x): Initial Equilibrium temperature guess(K), Pressure(Pa), Liquid
%phase fractions(mol/mol)
 [Tequi, y] = thermo_LIBRARY('equilibrium',T,P,x);
```

thermo_Enthaly.m

```
function [hl,hv] = thermo_Enthalpy(T,P,x,y)
% function [hl,hv] = thermo_Enthalpy(T,P,x,y)
% Thermophysical and physical property calculation MEX File Interface
% ------------- Enthalpy routine -------------
%  [hl,hv]      : Liquid and Vapor phase specific enthalpy (J/mol)
%  (T,P,x,y)    : Initial Equilibrium temperature guess(K), Pressure(Pa),
%         Liquid phase fractions(mol/mol), vapor phase fractions(mol/mol)
 [hl_d,hv_d] = thermo_LIBRARY('enthalpy',T,P,x,y);
hl = hl_d - 20000.0;
hv = hv_d - 20000.0;
```

thermo_Density.m

```
function [mwa, densa] = thermo_Density(T,P,x)
% function [mwa, densa] = thermo_Density(T,P,x)
% Thermophysical and physical property calculation MEX File Interface
% ------------- Density routine -------------
%[mwa, densa]: Avg. molecular weight(kg/mol),avg. liq. phase density (kg/m3)
%(T,P,x) : Initial Equilibrium temperature guess(K), Pressure(Pa), Liquid
%phase fractions(mol/mol)
 [mwa, densa] = thermo_LIBRARY('density',T,P,x);
```

C.3 Thermodynamic Library FORTRAN dll Code

thermo_LIBRARY.f

```
C -------------------------------------------------------------------------
C      MEX File Gateway implementation for Plant_Subroutines
C      Date : 08-05-2001 by Uğur YILDIZ
C -------------------------------------------------------------------------
C      Modified for gama-fi approach by ALMILA BAHAR Date:01/05/2006
C -------------------------------------------------------------------------
subroutine mexFunction(nlhs, plhs, nrhs, prhs)
include 'thermo_LIBRARY.h'
include 'parameter.h'
integer plhs(*), prhs(*)! pointer to left-hand and right-hand side variables
integer nlhs, nrhs         ! # of variables in plhs, prhs
integer mxCreateFull, mxGetString              ! mx Functions declarations
integer mxGetM, mxGetN, mxIsNumeric, mxIsString! mx Functions declarations
integer m, n, size, status, alloc_err          ! Dummy variables
integer Func_name_ptr                    ! Function name fortran pointers
character*100 Func_name                  ! Function name for fortran use
c ------------------------------------------------Input fortran pointers
integer Input1_pr, Input2_pr, Input3_pr, Input4_pr, Input5_pr, Input6_pr,
 & x_pr, y_pr, z_pr
c ------------------------------------------------ Output fortran pointers
integer Output1_pr, Output2_pr, Output3_pr, Output4_pr, Output5_pr,
 & Output6_pr
c --------------------------------------- Input arguments for fortran use
integer,allocatable, dimension (:) :: int_Input1, int_Input2, int_Input3,
 & int_Input4, int_Input5, int_Input6
real*8,allocatable, dimension (:) ::  real_Input1, real_Input2, real_Input3,
real_Input4, real_Input5, real_Input6
integer Input1_sz,Input2_sz,Input3_sz,Input4_sz,Input5_sz,Input6_sz
c --------------------------------------- Output arguments for fortran use
integer,allocatable, dimension (:) :: int_Output1, int_Output2, int_Output3,
 & int_Output4, int_Output5, int_Output6
```

```
real*8,allocatable, dimension (:) ::  real_Output1, real_Output2,
real_Output3, real_Output4, real_Output5, real_Output6
integer Output1_sz,Output2_sz,Output3_sz,Output4_sz,Output5_sz,Output6_sz
real*8  x, y(3,3), z(3,3)
C     ------------------------ Check for at least one function is requested.
if (nrhs .lt. 1) then
call mexErrMsgTxt('Not a proper function selected. - [thermo_LIBRARY.dll]')
endif
if (mxIsString(prhs(1)) .ne. 1) then
call mexErrMsgTxt('Function name parameter must be a valid string. -
[thermo_LIBRARY.dll]')
endif
m = mxGetM(prhs(1))
n = mxGetN(prhs(1))
if (m .ne. 1) then
call mexErrMsgTxt('Function name parameter must be a row vector. -
&[thermo_LIBRARY.dll]')
endif
C     ---------------------------------------- Call the requested function.
C      Get the string contents (dereference the input integer).
status = mxGetString(prhs(1),Func_name,100)
C      Check if mxGetString is successful.
if (status .ne. 0) then
call mexErrMsgTxt('String length must be less than 100. -
[thermo_LIBRARY.dll]')
endif
c ---------------------------------------- ! Call initialization function
if (Func_name.eq.'init') then
status = 1
if (nrhs .ne. 2) then
call mexErrMsgTxt('One input (number of components) is required for the
initialization. - (init) [thermo_LIBRARY.dll]')
endif
if (mxIsNumeric(prhs(2)) .ne. 1) call mexErrMsgTxt('Input #1 is not a
&numeric. - (init) [thermo_LIBRARY.dll]')
m = mxGetM(prhs(2))
n = mxGetN(prhs(2))
if (n .ne. 1 .or. m .ne. 1) call mexErrMsgTxt('Input #1 is not a scalar. -
&(init) [thermo_LIBRARY.dll]')
       Input1_pr = mxGetPr(prhs(2))
status = 0
allocate (int_Input1(1),STAT = alloc_err)
status = status + alloc_err
if (status .ne. 0) then
if (allocated(int_Input1)) deallocate(int_Input1)
call mexErrMsgTxt('Memory allocation error. - (init) [thermo_LIBRARY.dll]')
endif
call mxCopyPtrToInteger4(Input1_pr,int_Input1,1)
status = 1
call init(int_Input1,status)
if (status.eq.0) then
call mexPrintf('thermo_LIBRARY is initialized. - (init)
[thermo_LIBRARY.dll]')
else
call mexErrMsgTxt('thermo_LIBRARY can not be initialized. - (init)
[thermo_LIBRARY.dll]')
endif
c ---------------------------------------------- ! Call enthalpy function
elseif ((Func_name.eq.'enthalpy') .and. (lib_Inited.eq.1)) then
if (nrhs .ne. 5) then
call mexErrMsgTxt('Four inputs (T,P,x,y) is required. - (enthalpy)
&[thermo_LIBRARY.dll]')
elseif (nlhs .ne. 2) then
call mexErrMsgTxt('Two outputs (liquid and vapor entalphies) are required. -
&(enthalpy) [thermo_LIBRARY.dll]')
endif
```

```
if (mxIsNumeric(prhs(2)) .ne. 1) call mexErrMsgTxt('Input #1 is not a
&numeric. - (enthalpy) [thermo_LIBRARY.dll]')
if (mxIsNumeric(prhs(3)) .ne. 1) call mexErrMsgTxt('Input #2 is not a
&numeric. - (enthalpy) [thermo_LIBRARY.dll]')
if (mxIsNumeric(prhs(4)) .ne. 1) call mexErrMsgTxt('Input #3 is not a
&numeric. - (enthalpy) [thermo_LIBRARY.dll]')
if (mxIsNumeric(prhs(5)) .ne. 1) call mexErrMsgTxt('Input #4 is not a
&numeric. - (enthalpy) [thermo_LIBRARY.dll]')
m = mxGetM(prhs(2))
n = mxGetN(prhs(2))
Input1_sz = m*n
if (n .ne. 1 .or. m .ne. 1) call mexErrMsgTxt('Input #1 is not a scalar. -
&(enthalpy) [thermo_LIBRARY.dll]')
        m = mxGetM(prhs(3))
        n = mxGetN(prhs(3))
        Input2_sz = m*n
if (n .ne. 1 .or. m .ne. 1) call mexErrMsgTxt('Input #2 is not a scalar. -
&(enthalpy) [thermo_LIBRARY.dll]')
        m = mxGetM(prhs(4))
        n = mxGetN(prhs(4))
        Input3_sz = m*n
if (n .ne. nj .or. m .ne. 1) call mexErrMsgTxt('Input #3 is not a NC-element
&row vector. - (enthalpy) [thermo_LIBRARY.dll]')
       m = mxGetM(prhs(5))
        n = mxGetN(prhs(5))
        Input4_sz = m*n
if (n .ne. nj .or. m .ne. 1) call mexErrMsgTxt('Input #4 is not a NC-element
&row vector. - (enthalpy) [thermo_LIBRARY.dll]')
Input1_pr = mxGetPr(prhs(2))
Input2_pr = mxGetPr(prhs(3))
Input3_pr = mxGetPr(prhs(4))
Input4_pr = mxGetPr(prhs(5))
plhs(1) = mxCreateFull(1,1,0)
plhs(2) = mxCreateFull(1,1,0)
Output1_pr = mxGetPr(plhs(1))
Output2_pr = mxGetPr(plhs(2))
status = 0
allocate (real_Input1(1),STAT = alloc_err)
status = status + alloc_err
allocate (real_Input2(1),STAT = alloc_err)
status = status + alloc_err
allocate (real_Input3(nj),STAT = alloc_err)
status = status + alloc_err
allocate (real_Input4(nj),STAT = alloc_err)
status = status + alloc_err
allocate (real_Output1(1),STAT = alloc_err)
status = status + alloc_err
allocate (real_Output2(1),STAT = alloc_err)
status = status + alloc_err
if (status .ne. 0) then
if (allocated(real_Input1)) deallocate(real_Input1)
if (allocated(real_Input2)) deallocate(real_Input2)
if (allocated(real_Input3)) deallocate(real_Input3)
if (allocated(real_Input4)) deallocate(real_Input4)
if (allocated(real_Output1)) deallocate(real_Output1)
if (allocated(real_Output2)) deallocate(real_Output2)
call mexErrMsgTxt('Memory allocation error. - (enthalpy)
&[thermo_LIBRARY.dll]')
endif
call mxCopyPtrToReal8(Input1_pr,real_Input1,Input1_sz)
call mxCopyPtrToReal8(Input2_pr,real_Input2,Input2_sz)
call mxCopyPtrToReal8(Input3_pr,real_Input3,Input3_sz)
call mxCopyPtrToReal8(Input4_pr,real_Input4,Input4_sz)
call enth(real_Input1, real_Input2, real_Input3, real_Input4, real_Output1,
&real_Output2)
call mxCopyReal8ToPtr(real_Output1,Output1_pr,1)
call mxCopyReal8ToPtr(real_Output2,Output2_pr,1)
```

```
c ---------------------------------------------- ! Call density function
elseif ((Func_name.eq.'density') .and. (lib_Inited.eq.1)) then
if (nrhs .ne. 4) then
call mexErrMsgTxt('Three inputs (T,P,x) is required. - (density)
&[thermo_LIBRARY.dll]')
elseif (nlhs .ne. 2) then
call mexErrMsgTxt('Two outputs (Avg. mol. weight and density) are required.
&- (density) [thermo_LIBRARY.dll]')
endif
if (mxIsNumeric(prhs(2)) .ne. 1) call mexErrMsgTxt('Input #1 is not a
&numeric. - (density) [thermo_LIBRARY.dll]')
if (mxIsNumeric(prhs(3)) .ne. 1) call mexErrMsgTxt('Input #2 is not a
&numeric. - (density) [thermo_LIBRARY.dll]')
if (mxIsNumeric(prhs(4)) .ne. 1) call mexErrMsgTxt('Input #3 is not a
&numeric. - (density) [thermo_LIBRARY.dll]')
m = mxGetM(prhs(2))
n = mxGetN(prhs(2))
Input1_sz = m*n
if (n .ne. 1 .or. m .ne. 1) call mexErrMsgTxt('Input #1 is not a scalar. -
&(density) [thermo_LIBRARY.dll]')
m = mxGetM(prhs(3))
n = mxGetN(prhs(3))
Input2_sz = m*n
if (n .ne. 1 .or. m .ne. 1) call mexErrMsgTxt('Input #2 is not a scalar. -
&(density) [thermo_LIBRARY.dll]')
m = mxGetM(prhs(4))
n = mxGetN(prhs(4))
Input3_sz = m*n
if (n .ne. nj .or. m .ne. 1) call mexErrMsgTxt('Input #3 is not a NC-element
&row vector. - (density) [thermo_LIBRARY.dll]')
Input1_pr = mxGetPr(prhs(2))
Input2_pr = mxGetPr(prhs(3))
Input3_pr = mxGetPr(prhs(4))
plhs(1) = mxCreateFull(1,1,0)
plhs(2) = mxCreateFull(1,1,0)
Output1_pr = mxGetPr(plhs(1))
Output2_pr = mxGetPr(plhs(2))
status = 0
allocate (real_Input1(1),STAT = alloc_err)
status = status + alloc_err
allocate (real_Input2(1),STAT = alloc_err)
status = status + alloc_err
allocate (real_Input3(nj),STAT = alloc_err)
status = status + alloc_err
allocate (real_Output1(1),STAT = alloc_err)
status = status + alloc_err
allocate (real_Output2(1),STAT = alloc_err)
status = status + alloc_err
if (status .ne. 0) then
if (allocated(real_Input1)) deallocate(real_Input1)
if (allocated(real_Input2)) deallocate(real_Input2)
if (allocated(real_Input3)) deallocate(real_Input3)
if (allocated(real_Output1)) deallocate(real_Output1)
if (allocated(real_Output2)) deallocate(real_Output2)
call mexErrMsgTxt('Memory allocation error. - (density)
&[thermo_LIBRARY.dll]')
endif
call mxCopyPtrToReal8(Input1_pr,real_Input1,Input1_sz)
call mxCopyPtrToReal8(Input2_pr,real_Input2,Input2_sz)
call mxCopyPtrToReal8(Input3_pr,real_Input3,Input3_sz)
c      subroutine pr_dens(t,p,x,mwa,densa)
call pr_dens(real_Input1, real_Input2, real_Input3, real_Output1,
real_Output2)
call mxCopyReal8ToPtr(real_Output1,Output1_pr,1)
call mxCopyReal8ToPtr(real_Output2,Output2_pr,1)
c ------------------------------------------- ! Call equilibrium function
elseif ((Func_name.eq.'equilibrium') .and. (lib_Inited.eq.1)) then
```

```
if (nrhs .ne. 4) then
call mexErrMsgTxt('Three inputs (T, P, x) is required. - (equilibrium)
&[thermo_LIBRARY.dll]')
elseif (nlhs .ne. 2) then
call mexErrMsgTxt('Two outputs (T and vapor comp.) are required. -
&(equilibrium) [thermo_LIBRARY.dll]')
endif
if (mxIsNumeric(prhs(2)) .ne. 1) call mexErrMsgTxt('Input #1 is not a
&numeric. - (equilibrium) [thermo_LIBRARY.dll]')
if (mxIsNumeric(prhs(3)) .ne. 1) call mexErrMsgTxt('Input #2 is not a
&numeric. - (equilibrium) [thermo_LIBRARY.dll]')
if (mxIsNumeric(prhs(4)) .ne. 1) call mexErrMsgTxt('Input #3 is not a
&numeric. - (equilibrium) [thermo_LIBRARY.dll]')
m = mxGetM(prhs(2))
n = mxGetN(prhs(2))
Input1_sz = m*n
if (n .ne. 1 .or. m .ne. 1) call mexErrMsgTxt('Input #1 is not a scalar. -
&(equilibrium) [thermo_LIBRARY.dll]')
m = mxGetM(prhs(3))
n = mxGetN(prhs(3))
Input2_sz = m*n
if (n .ne. 1 .or. m .ne. 1) call mexErrMsgTxt('Input #2 is not a scalar. -
&(equilibrium) [thermo_LIBRARY.dll]')
m = mxGetM(prhs(4))
n = mxGetN(prhs(4))
Input3_sz = m*n
if (n .ne. nj .or. m .ne. 1)
call mexErrMsgTxt('Input #3 is not a NC-element row vector. - (equilibrium)
&[thermo_LIBRARY.dll]')
Input1_pr = mxGetPr(prhs(2))
Input2_pr = mxGetPr(prhs(3))
Input3_pr = mxGetPr(prhs(4))
plhs(1) = mxCreateFull(1,1,0)
plhs(2) = mxCreateFull(1,nj,0)
Output1_pr = mxGetPr(plhs(1))
Output2_pr = mxGetPr(plhs(2))
status = 0
allocate (real_Input1(1),STAT = alloc_err)
status = status + alloc_err
allocate (real_Input2(1),STAT = alloc_err)
status = status + alloc_err
allocate (real_Input3(nj),STAT = alloc_err)
status = status + alloc_err
allocate (real_Output1(1),STAT = alloc_err)
status = status + alloc_err
allocate (real_Output2(nj),STAT = alloc_err)
status = status + alloc_err
if (status .ne. 0) then
if (allocated(real_Input1)) deallocate(real_Input1)
if (allocated(real_Input2)) deallocate(real_Input2)
if (allocated(real_Input3)) deallocate(real_Input3)
if (allocated(real_Output1)) deallocate(real_Output1)
if (allocated(real_Output2)) deallocate(real_Output2)
call mexErrMsgTxt('Memory allocation error. - (equilibrium)
&[thermo_LIBRARY.dll]')
endif
call mxCopyPtrToReal8(Input1_pr,real_Input1,Input1_sz)
call mxCopyPtrToReal8(Input2_pr,real_Input2,Input2_sz)
call mxCopyPtrToReal8(Input3_pr,real_Input3,Input3_sz)
c      subroutine pr_equil(t,p,x,yy)     ' t is also an output
call pr_equil(real_Input1, real_Input2, real_Input3, real_Output2)
real_Output1 = real_Input1
call mxCopyReal8ToPtr(real_Output1,Output1_pr,1)
call mxCopyReal8ToPtr(real_Output2,Output2_pr,nj)
c ------------------------------------------- ! No relevant function
else
```

```
call mexErrMsgTxt('Library is not initialized or No relevant function is
&requested. - [thermo_LIBRARY.dll]')
endif
c ------------------------------------------- ! Memory deallocation
if (allocated(real_Input1)) deallocate(real_Input1)
if (allocated(real_Input2)) deallocate(real_Input2)
if (allocated(real_Input3)) deallocate(real_Input3)
if (allocated(real_Input4)) deallocate(real_Input4)
if (allocated(real_Input5)) deallocate(real_Input5)
if (allocated(real_Input6)) deallocate(real_Input6)
if (allocated(int_Input1)) deallocate(int_Input1)
if (allocated(int_Input2)) deallocate(int_Input2)
if (allocated(int_Input3)) deallocate(int_Input3)
if (allocated(int_Input4)) deallocate(int_Input4)
if (allocated(int_Input5)) deallocate(int_Input5)
if (allocated(int_Input6)) deallocate(int_Input6)
if (allocated(real_Output1)) deallocate(real_Output1)
if (allocated(real_Output2)) deallocate(real_Output2)
if (allocated(real_Output3)) deallocate(real_Output3)
if (allocated(real_Output4)) deallocate(real_Output4)
if (allocated(real_Output5)) deallocate(real_Output5)
if (allocated(real_Output6)) deallocate(real_Output6)
if (allocated(int_Output1)) deallocate(int_Output1)
if (allocated(int_Output2)) deallocate(int_Output2)
if (allocated(int_Output3)) deallocate(int_Output3)
if (allocated(int_Output4)) deallocate(int_Output4)
if (allocated(int_Output5)) deallocate(int_Output5)
if (allocated(int_Output6)) deallocate(int_Output6)
return
end
C ------------------------------------------------ thermo_LIBRARY
Initialization routine
subroutine init(check_input,st)
integer check_input, st
include 'thermo_LIBRARY.h'
include 'parameter.h'
include 'common_plant.h'
C-- Initialization of the 'plant' common statement in 'common_plant.h' -----
C      ------ written by MTD (Revised by Uğur Yıldız)
integer :: i,j,I_O_err
integer :: thermo_LIBRARY_dummy_pr, thermo_LIBRARY_dummy_pi
C      tolerance = 1.d-7
open(5,file='thermo_data.dat',IOSTAT=I_O_err, ERR = 100)
read(5,*)
read(5,*) tolerance
read(5,*)
read(5,*)
do i=1,nj
read(5,*) mw(i),tc(i),tboil(i),pc(i),wc(i)
enddo
read(5,*)
read(5,*)
do i=1,nj
read(5,*) (del(i,j),j=1,nj)
enddo
read(5,*)
read(5,*)
do i=1,nj
read(5,*) cenh1(i),cenh2(i),cenh3(i),cenh4(i)
enddo
close(5)
C      ---------------------------------------------------------------------
if (check_input .eq. 1) then
open(6,file='thermo_data_check.dat')
write(6,*) 'tolerance'
write(6,1) tolerance
write(6,*)
```

```
write(6,*) 'Mw(kg/mol)      Tc(K)      Tboil(K)    Pc(Pa)            w'
do i=1,nj
write(6,2) mw(i),tc(i),tboil(i),pc(i),wc(i)
enddo
write(6,*)
write(6,*) 'del(binary interaction parameters)'
do i=1,nj
write(6,3) (del(i,j),j=1,nj)
enddo
write(6,*)
write(6,*) '   cenh1         cenh2          cenh3       cenh4(J/molK)'
do i=1,nj
write(6,4) cenh1(i),cenh2(i),cenh3(i),cenh4(i)
enddo
close(6)
endif
lib_Inited = 1
st = 0
return
100    if (I_O_err.ne.0) then
call mexErrMsgTxt('"thermo_data.dat" couldn"t be opened. - (init)
&[thermo_LIBRARY.dll]')
lib_Inited = 0
st = 1
return
endif
1      format(d11.3)
2      format(5d15.3)
3
format(d13.1,:,d13.1,:,d13.1,:,d13.1,:,d13.1,:,d13.1,:,d13.1,:,d13.1,:,d13.1
&,:,d13.1,:,d13.1,:,d13.1,:,d13.1,:,d13.1)
4      format(4d15.3)
end subroutine
C  Write statements in these routines are exchanged with mexErrMsgTxt and
CmexPrintf and also 'parameter.h' and 'plant_data.dat' are modified.
C -------------------------------------------------------------------------
C Peng-Rabinson EOS Subroutines Written by Mustafa T. DOKUCU Date:16-05-2001
c -------------------------------------------------------------------------
subroutine enth(t,p,x,y,hl,hv)
!Usage:
!      to calculate the ideal gas mixture enthalpy
!Record of revisions:
!   date      programmer      description of change
!   ====      ==========      =====================
! 18/03/2001    MTD           original code
implicit none
include 'parameter.h'
include 'common_plant.h'
!                                      Inputs
!                                      ========
real*8 :: t                          !temperature
real*8 :: p                          !pressure
real*8 :: x(nj)                      !liquid phase fractions
real*8 :: y(nj)                      !vapour phase fractions
!                                        Locals
!                                       ========
real*8 :: hl1                        !ideal liquid mixture enthalpy
real*8 :: hv1                        !ideal vapour mixture enthalpy
real*8 :: dhl                        !liquid enthalpy departure
real*8 :: dhv                        !vapour enthalpy departure
real*8 :: enigl                      !ideal gas enthalpy
real*8 :: enigv                      !ideal gas enthalpy
real*8 :: cl1,cv1
real*8 :: cl2,cv2
real*8 :: cl3,cv3
real*8 :: cl4,cv4
integer:: i,j
```

```
integer:: ifase
real*8 :: enthv(nj),enthl(nj),lheat(nj),lheatb(nj),trb(nj),tr(nj)
real*8 :: lheata,lheatk,lheatc
!                                      Outputs
!                                     =========
real*8 ::  hl                        !liquid enthalpy
real*8 ::  hv                        !vapour enthalpy
cl1  = 0.d0
cv1  = 0.d0
cl2  = 0.d0
cv2  = 0.d0
cl3  = 0.d0
cv3  = 0.d0
cl4  = 0.d0
cv4  = 0.d0
do i = 1,nj
cl1 = cl1 + cenh1(i) * x(i)
cl2 = cl2 + cenh2(i) * x(i)
cl3 = cl3 + cenh3(i) * x(i)
cl4 = cl4 + cenh4(i) * x(i)
cv1 = cv1 + cenh1(i) * y(i)
cv2 = cv2 + cenh2(i) * y(i)
cv3 = cv3 + cenh3(i) * y(i)
cv4 = cv4 + cenh4(i) * y(i)
enddo
enigl=cl1*(t-trf)+(1.d0/2.d0)*cl2*(t**2-trf**2)+(1.d0/3.d0)*cl3*(t**3-
trf**3)+(1.d0/4.d0)*cl4*(t**4 - trf**4)
enigv = cv1 * (t-trf) + (1.d0/2.d0) * cv2 * (t**2 - trf**2) + (1.d0/3.d0) *
cv3 * (t**3 - trf**3) + (1.d0/4.d0) * cv4 * (t**4 - trf**4)
ifase = 0
call pr_enth(t,p,x,ifase,dhl)
ifase = 1
call pr_enth(t,p,y,ifase,dhv)
hl = enigl + dhl + 20000.d0
hv = enigv + dhv + 20000.d0
return
end subroutine
c -------------------------------------------------------------------------------
subroutine pr_compr(a_mixture,b_mixture,z_liq,z_vap)
!Usage:
!      to solve the cubic eqution for the liquid and vapor
!compressibility factors used for the estimation of species
!fugacities
!Record of revisions:
!   date       programmer   description of change
!   ====       =========    ====================
! 14/02/2001     MTD           original code
implicit none
include 'parameter.h'
include 'common_plant.h'
!                            Inputs
!                           ========
real*8 :: a_mixture
real*8 :: b_mixture
!                            Locals
!                           ========
complex*8:: z_vap_cplx
complex*8:: z_liq_cplx
complex*8:: s1
complex*8:: a
complex*8:: b
!                            Outputs
!                           =========
real*8:: z_vap
real*8:: z_liq

!convert the type declaration of the input variables to complex
```

```
a = cmplx(a_mixture,0.d0)
b = cmplx(b_mixture,0.d0)
!calculate the liquid phase compressibility
s1=-(-36.0d0*a+144.0d0*a*b-48.0d0*b**2-
 & 224.0d0*b**3+48.0d0*b+8.0d0+12.0d0*sqrt(24.0d0*a*b-24.0d0*b**2-
 & 192.0d0*b**3+264.0d0*a*b**2-3.0d0*a**2+24.0d0*a**2*b**2-120.0d0*a**2*b-
 & 48.0d0*a*b**4+336.0d0*a*b**3-480.0d0*b**4+12.0d0*a**3-96.0d0*b**6-
 & 384.0d0*b**5))**(1.d0/3.d0)/12.0d0+(a-10.d0/3.d0*b**2-4.d0/3.d0*B-
 & 1.d0/3.d0)/(-36.0d0*a+144.0d0*a*b-48.0d0*b**2-
 & 224.0d0*b**3.0d0+48.0d0*b+8.0d0+12.0d0*sqrt(24.0d0*a*b-24.0d0*b**2-
 & 192.0d0*b**3+264.0d0*a*b**2-3.0d0*a**2+24.0d0*a**2*b**2-120.0d0*a**2*b-
 & 48.0d0*a*b**4+336.0d0*a*b**3-480.0d0*b**4+12.0d0*a**3-96.0d0*b**6-384.0d0
 & *B**5))**(1.d0/3.d0)

z_liq_cplx = s1+1.d0/3.d0-b/3.d0+cmplx(0.d0,1.d0)*sqrt(3.d0)*((-
& 36.d0*a+144.d0*a*b-&48.d0*b**2-224.d0*b**3+48.d0*b+8.d0+12*sqrt(24.d0*a*b-
& 24.d0*b**2-192.d0*b**3+264.d0*a*b**2-3.d0*a**2+24.d0*a**2*b**2-
&120.d0*a**2*b-48.d0*a*b**4+336.d0*a*b**3-480.d0*b**4+12.d0*a**3-96.d0*b**6-
& 384.d0*b**5))**(1.d0/3.d0)/6.d0+(2.d0*a-20.d0/3.d0*b**2-8.d0/3.d0*b-
& 2.d0/3.d0)/(-36.d0*a+144.d0*a*b-48.d0*b**2-
& 224.d0*b**3+48.d0*b+8+12.d0*sqrt(24.d0*a*b-24.d0*b**2-
& 192.d0*b**3+264.d0*a*b**2-3.d0*a**2+24.d0*a**2*b**2-120.d0*a**2*b-
& 48.d0*a*b**4+336.d0*a*b**3-480.d0*b**4+12.d0*a**3-96.d0*b**6-
& 84.d0*b**5))**(1.d0/3.d0))/2.d0

!calculate the vapor phase compressibility
z_vap_cplx =(-36.d0*a+144.d0*a*b-48.d0*b**2-224.d0*b**3+48.d0*b+8.d0+12.d0*
& sqrt(24.d0*a*b-24.d0*b**2-192.d0*b**3+264.d0*a*b**2-
& 3.d0*a**2+24.d0*a**2*b**2-120.d0*a**2*b-48.d0*a*b**4+336.d0*a*b**3-
& 480.d0*b**4+12.d0*a**3-96.d0*b**6-384.d0*b**5))**(1.d0/3.d0)/6.d0-(2.d0*a-
& 20.d0/3.d0*b**2-8.d0/3.d0*b-2.d0/3.d0)/(-36.d0*a+144.d0*a*b-48.d0*b**2-
& 224.d0*b**3+48.d0*b+8.d0+12.d0*sqrt(24.d0*a*b-24.d0*b**2-
& 192.d0*b**3+264.d0*a*b**2-3.d0*a**2+24.d0*a**2*b**2-120.d0*a**2*b-
& 48.d0*a*b**4+336.d0*a*b**3-480.d0*b**4+12.d0*a**3-96.d0*b**6-384.d0
&*b**5))**(1.d0/3.d0)+1.d0/3.d0-b/3.d0

!there is no liquid phase if the liquid compressibility root is a complex #
!in this case the compressibility root returned as equal to vapor phase
!compressibility
if (aimag(z_liq_cplx) > tolerance) then
z_liq = real(z_vap_cplx)
else
z_liq = real(z_liq_cplx)
endif
!the root found for the vapor compressibility is erronaeous if it is
!a complex # in this case the compressibility root returned as zero to the
!mainprogram
if (aimag(z_vap_cplx) > tolerance) then
call mexPrintf('vapor phase compressibility can not be calculated. -
&(pr_compr) [thermo_LIBRARY.dll]\n')
c        write(*,*) 'vapor phase compressibility can not be calculated'
z_vap = 0.d0
else
z_vap = real(z_vap_cplx)
endif
return
end subroutine
c -------------------------------------------------------------------------
subroutine pr_cons(t,a,aij,b)
!Usage:
!to calculate the constants A and B of the Peng-Robinson EOS which is
!explained in p239 (Sandler)
!Record of revisions:
!    date    programmer    description of change
!   ====     ==========   ====================
! 12/03/2001    MTD           original code
implicit none
```

```
include 'parameter.h'
include 'common_plant.h'
!                          Inputs
!                         ========
real*8:: t              !temperature [K]
!                          Locals
!                         ========
real*8 :: ac(nj)         !constant
real*8 :: xk             !constant
real*8 :: alsqr         !constant
real*8 :: alpha         !constant
real*8 :: tr            !reduced temperature
integer:: i
integer:: j
!                          Outputs
!                         =========
real*8 :: a(nj)         !a of the species
real*8 :: b(nj)         !b of the species
real*8 :: aij(nj,nj)    !interacting a's of the species
do i=1,nj
ac(i) = 0.457235529d0 * ((rg * tc(i))**2) / pc(i)     !eqn 4.7-1(first part)
b(i)  = 7.779607400000001d-2 * rg * tc(i) / pc(i)           !eqn 4.7-2
xk    = 0.37464d0 + (1.54226d0 - 0.26992d0 * wc(i)) * wc(i) !eqn 4.7-4
tr    = t / tc(i)
alsqr = 1.d0 + xk * (1.d0 - dsqrt(tr))
alpha = alsqr * alsqr                                        !eqn 4.7-3
a(i)  = alpha * ac(i)                                      !eqn 4.7-1(whole)
enddo
do i=1,(nj-1)
do j=(i+1),nj
  aij(i,j) = (1.d0 - del(i,j)) * dsqrt(a(i) * a(j))         !eqn 7.4-9
  aij(j,i) = aij(i,j)
  enddo
enddo
return
end subroutine
c -------------------------------------------------------------------------------
subroutine pr_dens(t,p,x,mwa,densa)
!Usage:
!     to calculate the average molecular weight and the
!density of the liquid phase using Peng-Robinson EOS
!Record of revisions:
!   date     programmer          description of change
!   ====     ==========          =====================
! 25/03/2001    MTD               original code
implicit none
include 'parameter.h'
include 'common_plant.h'
!                          Inputs
!                         ========
real*8 :: t
real*8 :: p
real*8 :: x(nj)
!                          Locals
!                         ========
real*8 :: aa
real*8 :: bb
real*8 :: ca
real*8 :: cb
real*8 :: a(nj)
real*8 :: b(nj)
real*8 :: zx(nj)
real*8 :: aij(nj,nj)
real*8 :: z_liq
real*8 :: z_vap
real*8 :: zz
real*8 :: vv
```

```
real*8 :: sumx
integer:: i
integer:: j
real*8 :: adens(nj),bdens(nj),dens(nj),tr(nj), conca,dens_cons
!                          Outputs
!                         =========
real*8 :: mwa
real*8 :: densa
mwa  = 0.d0
sumx = 0.d0
do i = 1,nj
sumx = sumx + x(i)
enddo
do i = 1,nj
  zx(i) = x(i) / sumx
enddo
do i = 1,nj
        mwa = mwa + mw(i) * zx(i)
enddo

call pr_cons(t,a,aij,b)
aa = 0.d0
bb = 0.d0
do i = 1,nj
bb = bb + zx(i) * b(i)
do j = 1,nj
if (i == j) then
aa = aa + zx(i) * zx(i) * a(i)
else
aa = aa + zx(i) * zx(j) * aij(i,j)
endif
enddo
enddo
ca = aa * p / ((rg * t)**2)
cb = bb * p /  (rg * t)
call pr_compr(ca,cb,z_liq,z_vap)
zz    = z_liq
vv    = zz * rg * t / p
densa = mwa / vv
return
end subroutine
c ------------------------------------------------------------------------------
subroutine pr_enth(t,p,zx,ifase,dh)
!Usage:
!to calculate the enthalpy departure of a mixture
!as explained in Sandler p425
!Peng-Robinson EOS is explained in p239
!Record of revisions:
!   date         programmer          description of change
!   ====         ==========         =======================
!12/03/2001        MTD                  original code
implicit none
include 'parameter.h'
include 'common_plant.h'
!                          Inputs
!                         ========
real*8 :: t
real*8 :: p
real*8 :: zx(nj)
!                          Locals
!                         ========
real*8 :: zz
real*8 :: a(nj)
real*8 :: b(nj)
real*8 :: xk(nj)
real*8 :: aij(nj,nj)
real*8 :: c1,c2,c3,c4,c5,c6,c7,c8,c9
```

```
real*8 :: a1,a2,a3,a4
real*8 :: anum,aden
real*8 :: cnum,cden
real*8 :: damdt
real*8 :: aa
real*8 :: ca
real*8 :: bb
real*8 :: cb
real*8 :: z_liq
real*8 :: z_vap
real*8 :: tr(nj)
real*8 :: dh0,dh1,dh2
integer:: i
integer:: j
integer:: ifase
!                          Outputs
!                         =========
real*8:: dh
call pr_cons(t,a,aij,b)
aa = 0.d0
bb = 0.d0
do i = 1,nj
bb = bb + zx(i) * b(i)
do j = 1,nj
if (i == j) then
aa = aa + zx(i) * zx(i) * a(i)
else
aa = aa + zx(i) * zx(j) * aij(i,j)
endif
enddo
enddo
ca = aa * p / ((rg * t)**2)
cb = bb * p / (rg * t)
call pr_compr(ca,cb,z_liq,z_vap)
if (ifase == 0) then
zz = z_liq
else
zz = z_vap
endif
do i = 1,nj
 xk(i) = 0.37464d0 + (1.54226d0 - 0.26992d0 * wc(i)) * wc(i) !eqn 4.7-4
enddo
damdt = 0.d0
do i= 1,nj
do j = 1,nj
tr(i) = t / tc(i)
tr(j) = t / tc(j)
c1 = (-0.457235529d0/2.d0) * (-1 + del(i,j)) * rg**4
c2 =   tc(i) * (-1.d0 - xk(i) + xk(i) * dsqrt(tr(i)))
c3 =   tc(j) * (-1.d0 - xk(j) + xk(j) * dsqrt(tr(j)))
c4 =  -tc(j) * xk(i) * dsqrt(tr(j)) -tc(j) * xk(i) * dsqrt(tr(j)) * xk(j)
c5 =   2.d0 * xk(i) * t * xk(j)-tc(i) * xk(j) * dsqrt(tr(i))
c6 =  -tc(i) * xk(j) * dsqrt(tr(i)) * xk(i)
c7 = (-1.d0 - xk(i) + xk(i) * dsqrt(tr(i)))**2
c8 = (-1.d0 - xk(j) + xk(j) * dsqrt(tr(j)))**2 / pc(i) / pc(j)
c9 =   pc(i) * pc(j) * dsqrt(tr(i)) * dsqrt(tr(j))
cnum = c4 + c5 + c6
cden = dsqrt(rg**4 * tc(i)**2 * c7 * tc(j)**2 * c8) * pc(i) * pc(j) *
dsqrt(tr(i)) * dsqrt(tr(j))
damdt = damdt + zx(i) * zx(j) * (c1 * c2 * c3 * cnum / cden)
enddo
enddo
a1 = dsqrt(2.d0)
a2 = a1 + 1.d0
a3 = a1 - 1.d0
a4 = a1 * 2.d0
anum = zz + (a2 * cb)
```

```
aden = zz - (a3 * cb)
dh0 = (rg * t)*(zz -1.d0)
dh1 = (t*damdt - aa)/a4/bb
dh2 = dlog(anum/aden)
dh  = (dh0 + dh1*dh2)
return
end subroutine
c -----------------------------------------------------------------------
subroutine pr_equil(t,p,x,yy)
!Usage:
!      to calculate the bubble point temperature using
!Peng-Robinson EOS similar to VLMU.BAS of Sandler
!Record of revisions:
!    date    programmer  description of change
!   ===      ========    ====================
! 12/03/2001   MTD        original code
! 12/06/2001   UGUR       to be able to find equilibrium staff
! when a zero-fraction component exist. some checks were performed before
!calculation.
! 01/05/2006   ALMILA    modified for gama-fi method with NRTL activity
!coefficient model
implicit none
include 'parameter.h'
include 'common_plant.h'
!                          Inputs
!                         ========
real*8:: t                 ! t is also an output
real*8:: p
real*8:: x(nj)
!                          Locals
!                         ========
real*8 :: s(2),sum,sumy
real*8 :: dt1,dt2
real*8 :: dlnp
real*8 :: ps(nj)
real*8 :: xk1(nj)
real*8 :: a(nj)
real*8 :: aij(nj,nj)
real*8 :: b(nj)
real*8 :: zx(nj)
real*8 :: fugacity(nj)
real*8 :: f1(nj),f2(nj)
real*8 :: zz,zz1,zz2
real*8 :: y1(nj),y2(nj)
real*8 :: yk
real*8 :: test,ttest
real*8 :: dsdt,dlt,dd
real*8 :: neg_dd,neg_dlt
real*8 :: tbg,tcg
integer:: i
integer:: j
integer:: k
integer:: nc
integer:: kkk
integer:: nloop
integer:: iconv
integer:: ifase
integer:: itest
integer:: kvalue
logical:: reguess
real*8 :: ant_cons_a(nj),ant_cons_b(nj),ant_cons_c(nj)
real*8 :: gama(nj),sumy1
integer:: loop
real*8 :: alact(nj),tagx1,tagx2,total,xgkj,xgij1
real*8 :: bij(nj,nj),alp(nj,nj),ta(nj,nj),gij(nj,nj)
!                          Outputs
!                         =========
```

```
real*8 yy(nj)
integer:: comp_index(nj)
common /nc/ nc
! zero component check
j=0
do i=1,nj
if (x(i).gt.0.0d0) then
j=j+1
comp_index(j) = i
else
yy(i) = 0.0d0
endif
end do
nc = j
ant_cons_a(1) = 4.13361d0
ant_cons_a(2) = 5.33675d0
ant_cons_a(3) = 5.11564d0
ant_cons_a(4) = 4.54456d0
ant_cons_b(1) = 1195.130d0
ant_cons_b(2) = 1648.220d0
ant_cons_b(3) = 1687.537d0
ant_cons_b(4) = 1555.120d0
ant_cons_c(1) = 212.470d0
ant_cons_c(2) = 230.918d0
ant_cons_c(3) = 230.170d0
ant_cons_c(4) = 224.650d0
sum = 0.0d0
do i = 1,nc
  sum = sum + x(comp_index(i))
enddo
do i = 1,nc
  x(comp_index(i)) = x(comp_index(i))/sum
enddo
loop = 0
10loop = loop+1
if (loop .gt. 1800) go to 4630
sumy = 0.0d0
c Activity coeffients from NRTL model
aij(1,2) = -4.41293d0
aij(2,1) =  1.817306d0
aij(1,3) = -2.34561d0
aij(3,1) =  3.853826d0
aij(1,4) =  0.0d0
aij(4,1) =  0.0d0
aij(2,3) =  0.806535d0
aij(3,2) =  0.514285d0
aij(2,4) =  0.0d0
aij(4,2) =  0.0d0
aij(3,4) =  3.3293d0
aij(4,3) = -1.9763d0
bij(1,2) =  1614.287d0
bij(2,1) = -421.289d0
bij(1,3) =  1290.464d0
bij(3,1) = -4.42868d0
bij(1,4) =  515.8212d0
bij(4,1) = -235.279d0
bij(2,3) = -266.533d0
bij(3,2) =  444.8857d0
bij(2,4) =  225.4756d0
bij(4,2) = -252.482d0
bij(3,4) = -723.888d0
bij(4,3) =  609.8886d0
alp(1,2) = 0.1d0
alp(1,3) = 0.364313d0
alp(1,4) = 0.3d0
alp(2,3) = 0.4d0
alp(2,4) = 0.3d0
```

```
alp(3,4) = 0.3d0
do i=1,nj-1
do j=i+1,nj
alp(j,i) = alp(i,j)
enddo
enddo
do i=1,nj
ta(i,i) = 0.0d0
enddo
do i=1,nj
do j=1,nj
if (i .eq. j) then
ta(i,j) = 0.0d0
else
ta(i,j) = aij(i,j) + bij(i,j)/t
endif
gij(i,j) = dexp(-alp(i,j)*ta(i,j))
enddo
enddo
do i=1,nc
tagx1 = 0.0d0
xgij1 = 0.0d0
do j = 1,nc
tagx1=tagx1+a(comp_index(j),comp_index(i))*gij(comp_index(j),comp_index(i))*
& x(comp_index(j))
xgij1 = xgij1 + x(comp_index(j))*gij(comp_index(j),comp_index(i))
enddo
total = 0.0d0
do j=1,nc
xgkj = 0.0d0
do k=1,nc
xgkj = xgkj + x(comp_index(k))*gij(comp_index(k),comp_index(j))
enddo
tagx2 = 0.0d0
do k=1,nc
tagx2=tagx2+x(comp_index(k))*ta(comp_index(k),comp_index(j))*gij(comp_index(
& k),comp_index(j))
enddo
total = total + x(comp_index(j))*gij(comp_index(i),comp_index(j))/xgkj*
     &(ta(comp_index(i),comp_index(j)) - tagx2/xgkj)
enddo
alact(comp_index(i)) = tagx1 / xgij1 + total
gama(comp_index(i)) = dexp(alact(comp_index(i)))
enddo
c end of NRTL lngama
!Vapor Pressure calculation
do i=1,nc
ps(comp_index(i)) = ant_cons_a(comp_index(i))-(ant_cons_b(comp_index(i))
     &/(t-273.15d0+ant_cons_c(comp_index(i))))
ps(comp_index(i)) = (10**ps(comp_index(i))) * 1.0d5
enddo
!End of vapor pressure calculation
do i=1,nc
!y*p*fiv=x*gama*pvap
yy(comp_index(i)) = x(comp_index(i))*gama(comp_index(i))*ps(comp_index(i))/p
sumy = sumy + yy(comp_index(i))
enddo
do j=1,nc
y2(comp_index(j)) = yy(comp_index(j))*1.0d0/sumy
yy(comp_index(j)) = y2(comp_index(j))
enddo
!call pr_cons
call pr_cons(t,a,aij,b)
11 do i = 1,nc
  zx(comp_index(i)) = yy(comp_index(i))
enddo
!phase vapor 0
```

```
ifase = 0
!call pr_fuga
call pr_fuga(t,p,ifase,zx,zz,a,aij,b,fugacity)
do i = 1,nc
  f2(comp_index(i)) = fugacity(comp_index(i))
enddo
sumy1 = 0.0d0
do i=1,nc
yy(comp_index(i))=x(comp_index(i))*gama(comp_index(i))*ps(comp_index(i))*yy(
comp_index(i))/f2(comp_index(i))
if (yy(comp_index(i)) .lt. 1.0d-16) yy(comp_index(i))=0.0d0
if (yy(comp_index(i)) .gt. 1.0d0)   yy(comp_index(i))=1.0d0
enddo
do i=1,nc
sumy1 = sumy1+yy(comp_index(i))
enddo
dsdt = (sumy-sumy1)/0.005d0
if ((sumy-sumy1) .lt. tolerance) go to 12
sumy=sumy1
do i=1,nc
yy(comp_index(i)) = yy(comp_index(i))*1.0d0/sumy1
enddo
go to 11
12 if (dabs((sumy1-1.0d0)/dsdt) < 0.0026d0) return
dlt = (sumy1-1.0d0)/dsdt
cif (loop < 11) then
cdd = 20.0d0
cendif
cif (loop >= 11) then
dd = 5.0d0
cendif
if (dlt > dd) then
t = t+dd
endif
if (dlt > dd) go to 10
neg_dd = -1.d0 * dd
if (dlt < neg_dd) then
t = t-dd
endif
neg_dlt = -1.0d0 * dlt
if (neg_dlt > dd) go to 10
t = t + dlt + 0.0025d0
go to 10
4630   call mexErrMsgTxt('not converging: one-phase region or poor initial
guess. - (pr_equil) [thermo_LIBRARY.dll]')
c4630  write(*,*) 'not converging: one-phase region or poor initial guess'
return
end subroutine
c -------------------------------------------------------------------------
subroutine pr_fuga(t,p,ifase,zx,zz,a,aij,b,fugacity)
   !Usage:
   !to calculate the species fugacity f(T,P,xi) as explained in Sandler p409
   !Peng-Robinson EOS is explained in p239
!Record of revisions:
!  date     programmer   description of change
!  ====     ========  ======================
!12/03/2001  MTD           original code
!12/06/2001  UGUR     to be able to find equilibrium staff when a zero-
!fraction component exist.
!some checks were performed before calculation.
implicit none
include 'parameter.h'
include 'common_plant.h'
!                          Inputs
!                         ========
real*8 :: t
real*8 :: p
```

```
real*8 :: zx(nj)
real*8 :: a(nj)
real*8 :: b(nj)
real*8 :: aij(nj,nj)
integer:: ifase
!                          Locals
!                         ========
real*8 :: c1
real*8 :: c2
real*8 :: c3
real*8 :: sa(nj)
real*8 :: aa
real*8 :: bb
real*8 :: cb
real*8 :: ca
real*8 :: zz
real*8 :: z_liq
real*8 :: z_vap
real*8 :: fox(nj)
real*8 :: ag1
real*8 :: ag2
real*8 :: ag3
integer:: nc
integer:: i
integer:: j
real*8 :: ant_cons_a(nj),ant_cons_b(nj),ant_cons_c(nj),ps(nj)
!                          Outputs
!                         =========
real*8 ::fugacity(nj)
integer:: comp_index(nj)
common /nc/ nc
! zero component check
j=0
do i=1,nj
if (zx(i) .gt. 0.d0) then
j=j+1
comp_index(j) = i
endif
end do
nc = j
c1 = dsqrt(2.d0)
c2 = 1.d0 + c1
c3 = c1 - 1.d0
do i = 1,nc
sa(comp_index(i)) = 0.d0
enddo
aa = 0.d0
bb = 0.d0
do i = 1,nc
bb = bb + zx(comp_index(i)) * b(comp_index(i))
do j = 1,nc
if (i == j) then
aa    = aa + zx(comp_index(i)) * zx(comp_index(i)) * a(comp_index(i))
sa(comp_index(j)) = sa(comp_index(j)) + zx(comp_index(j)) * a(comp_index(j))
else
aa=aa+zx(comp_index(i))*zx(comp_index(j))*aij(comp_index(i),comp_index(j))
sa(comp_index(j))=sa(comp_index(j))+zx(comp_index(i))*
& aij(comp_index(i),comp_index(j))
endif
enddo
enddo
ca = aa * p / ((rg*t)**2)
cb = bb * p / (rg*t)
call pr_compr(ca,cb,z_liq,z_vap)
if (ifase == 0) then
zz  = z_vap
else
```

```
zz  = z_liq
endif
ag1 = (zz + c2 * cb) / (zz - c3 * cb)
ag1 = dlog(ag1)
ag2 = ca / (2.d0 * cb * c1)
do i = 1,nc
ag3    = (2.d0 * sa(comp_index(i)) / aa) - (b(comp_index(i)) / bb)
fox(comp_index(i))=(b(comp_index(i))*(zz-1.d0)/bb)-dlog(zz-cb)-ag1*ag2*ag3
fox(comp_index(i)) = dexp(fox(comp_index(i)))
fugacity(comp_index(i)) = zx(comp_index(i)) * p * fox(comp_index(i))
enddo
return
end subroutine
```

thermo_LIBRARY.h

```
common /thermo_LIBRARY/    lib_Inited
integer :: lib_Inited      ! Toggle for checking whether thermo_LIBRARY.dll
!is initialized.
```

common_plant.h

```
common /plant/ whs,whr,ds,dr,wls,wlr,mvb,mvd,tolerance,
& mw(nj),tc(nj),tboil(nj),pc(nj),wc(nj),del(nj,nj),cenh1(nj),
& cenh2(nj),cenh3(nj),cenh4(nj)
real*8 :: whs,whr,ds,dr,wls,wlr,mvb,mvd,tolerance
real*8 :: mw,tc,tboil,pc,wc,del
real*8 :: cenh1,cenh2,cenh3,cenh4
```

parameter.h

```
C This parameters were modified as the common statement labeled as
C'parameter'
integer ,parameter :: nj = 4            ! number of components
real*8 ,parameter :: rg = 8.313999999999999d0 ! ideal gas constant
real*8 ,parameter :: trf= 273.15d0            ! reference temperature
```

thermo_data.dat

```
tolerance (Component order: ethyl acetate, ethanol, water, acetic acid)
  1.000d-7

Mw(kg/mol)      Tc(K)       Tboil(K)    Pc(Pa)            w
 88.106d-3     523.25d0    350.3d0    3.830d6         0.361d0
 46.069d-3     513.92d0    351.5d0    6.148d6         0.644d0
 18.015d-3     647.30d0    373.2d0    2.209d7         0.344d0
 60.052d-3     592.71d0    391.1d0    5.786d6         0.459d0

del(binary interaction parameters)
  0.000d0   0.022d0    -0.280d0    -0.226d0
  0.022d0   0.000d0    -0.935d0    -0.0436d0
 -0.280d0  -0.935d0     0.000d0    -0.144d0
 -0.226d0  -0.0436d0   -0.144d0     0.000d0

   cenh1         cenh2          cenh3       cenh4(J/molK)
  7.235d0      4.072d-1      -2.092d-4     2.555d-8
  9.014d0      2.095d-1      -1.037d-4     2.004d-8
 32.218d0      0.192d-2       1.055d-5    -3.593d-9
  4.840d0      2.549d-1      -1.753d-4     4.949d-8
```

C.4 ANN Estimator Code

training.m

```
net = newelm(minmax(input_norm),[20,34,4],{'tansig','tansig','purelin'},...
'trainbfg');
net.trainParam.show=1;
net.trainParam.lr=0.0001;
net.trainParam.epochs=500;
net.trainParam.goal=1e-7;
[net,tr]=train(net,input_norm,output_norm);
```

find_R_interval.m

```
% ================================================
% This function finds the reflux ratio interval and finds the max. and min.
% values of the input and output variables by interpolation.
% ------------------------------------------------------------------------
% inputs
% R_Ratio_p: L/V ratio of the column = R/(1+R)
% constant_R_Ratio:
% it is 1.0,when the column is operating at constant R after total reflux;
% it is 0.0,when the column is operating at variable R after total reflux
% Rpercent: indicates the percent change from the optimal R profile when the
%column operates at variable reflux ratio after total reflux
% ------------------------------------------------------------------------
% outputs
% i_max: Maximum values of the input variables
% i_min: Minimum values of the input variables
% o_max: Maximum values of the output variables
% o_min: Minimum values of the output variables
% ================================================
function [i_max, i_min, o_max, o_min] = find_R_interval(R_Ratio_p, ...
constant_R_Ratio, Rpercent)
i_max = zeros(1,4);
i_min = zeros(1,4);
o_max = zeros(1,4);
o_min = zeros(1,4);
i_max_1 = zeros(1,4); % Maximum value of the first input
i_min_1 = zeros(1,4); % Minimum value of the first input
o_max_1 = zeros(1,4); % Maximum value of the first output
o_min_1 = zeros(1,4); % Minimum value of the first output
i_max_2 = zeros(1,4); % Maximum value of the second input
i_min_2 = zeros(1,4); % Minimum value of the second input
o_max_2 = zeros(1,4); % Maximum value of the second output
o_min_2 = zeros(1,4); % Minimum value of the second output
if constant_R_Ratio == 1.0
    if (R_Ratio_p == 1.0)
        i_max = [364.7221 364.2783 364.2783 364.2783];
        i_min = [354.6084 343.9560 343.2418 343.1541];
        o_max = [0.5760 0.8532 0.2261 0.5];
        o_min = [0 0.1974 0 1.17e-4];
    elseif (R_Ratio_p>=0.5) & (R_Ratio_p<0.6)
        i_max_1 = [389.8986 389.8963 389.8866 388.2249];
        i_min_1 = [361.1863 346.7504 344.1106 343.8205];
        o_max_1 = [0.5203 0.8532 0.3197 0.8773];
        o_min_1 = [0 0.0325 0 1.17e-4];
        i_max_2 = [389.9045 389.9041 389.8937 388.4182];
        i_min_2 = [361.1863 346.7502 344.1106 343.8205];
        o_max_2 = [0.5204 0.8532 0.4059 0.9077];
        o_min_2 = [0 0.0187 0 1.17e-4];
        d1 = abs(R_Ratio_p - 0.5);
        d2 = abs(0.6 - R_Ratio_p);
        i_max = d1/(d1+d2)*i_max_2 + d2/(d1+d2)*i_max_1;
        i_min = d1/(d1+d2)*i_min_2 + d2/(d1+d2)*i_min_1;
        o_max = d1/(d1+d2)*o_max_2 + d2/(d1+d2)*o_max_1;
```

```
    o_min = d1/(d1+d2)*o_min_2 + d2/(d1+d2)*o_min_1;
elseif (R_Ratio_p>=0.6) & (R_Ratio_p<0.7)
    i_max_1 = [389.9045 389.9041 389.8937 388.4182];
    i_min_1 = [361.1863 346.7502 344.1106 343.8205];
    o_max_1 = [0.5204 0.8532 0.4059 0.9077];
    o_min_1 = [0 0.0187 0 1.17e-4];
    i_max_2 = [389.8650 389.8617 389.7480 384.0034];
    i_min_2 = [361.1863 346.7502 344.1106 343.8205];
    o_max_2 = [0.5206 0.8532 0.5075 0.6979];
    o_min_2 = [0 0.0645 0 1.17e-4];
    d1 = abs(R_Ratio_p - 0.6);
    d2 = abs(0.7 - R_Ratio_p);
    i_max = d1/(d1+d2)*i_max_2 + d2/(d1+d2)*i_max_1;
    i_min = d1/(d1+d2)*i_min_2 + d2/(d1+d2)*i_min_1;
    o_max = d1/(d1+d2)*o_max_2 + d2/(d1+d2)*o_max_1;
    o_min = d1/(d1+d2)*o_min_2 + d2/(d1+d2)*o_min_1;
elseif (R_Ratio_p>=0.7) & (R_Ratio_p<0.8)
    i_max_1 = [389.8650 389.8617 389.7480 384.0034];
    i_min_1 = [361.1863 346.7502 344.1106 343.8205];
    o_max_1 = [0.5206 0.8532 0.5075 0.6979];
    o_min_1 = [0 0.0645 0 1.17e-4];
    i_max_2 = [389.9063 389.9074 389.8468 387.6059];
    i_min_2 = [361.1863 346.7502 344.1106 343.8205];
    o_max_2 = [0.5210 0.8532 0.6145 0.8907];
    o_min_2 = [0 0.0095 0 1.17e-4];
    d1 = abs(R_Ratio_p - 0.7);
    d2 = abs(0.8 - R_Ratio_p);
    i_max = d1/(d1+d2)*i_max_2 + d2/(d1+d2)*i_max_1;
    i_min = d1/(d1+d2)*i_min_2 + d2/(d1+d2)*i_min_1;
    o_max = d1/(d1+d2)*o_max_2 + d2/(d1+d2)*o_max_1;
    o_min = d1/(d1+d2)*o_min_2 + d2/(d1+d2)*o_min_1;
elseif (R_Ratio_p>=0.8) & (R_Ratio_p<0.85)
    i_max_1 = [389.9063 389.9074 389.8468 387.6059];
    i_min_1 = [361.1863 346.7502 344.1106 343.8205];
    o_max_1 = [0.5210 0.8532 0.6145 0.8907];
    o_min_1 = [0 0.0095 0 1.17e-4];
    i_max_2 = [384.6589 377.6219 372.2896 364.7180];
    i_min_2 = [361.1861 346.7502 344.1106 343.8205];
    o_max_2 = [0.5216 0.8532 0.6196 0.5];
    o_min_2 = [0 0.1905 0 1.17e-4];
    d1 = abs(R_Ratio_p - 0.8);
    d2 = abs(0.85 - R_Ratio_p);
    i_max = d1/(d1+d2)*i_max_2 + d2/(d1+d2)*i_max_1;
    i_min = d1/(d1+d2)*i_min_2 + d2/(d1+d2)*i_min_1;
    o_max = d1/(d1+d2)*o_max_2 + d2/(d1+d2)*o_max_1;
    o_min = d1/(d1+d2)*o_min_2 + d2/(d1+d2)*o_min_1;
elseif (R_Ratio_p>=0.85) & (R_Ratio_p<0.89)
    i_max_1 = [384.6589 377.6219 372.2896 364.7180];
    i_min_1 = [361.1861 346.7502 344.1106 343.8205];
    o_max_1 = [0.5216 0.8532 0.6196 0.5];
    o_min_1 = [0 0.1905 0 1.17e-4];
    i_max_2 = [376.4271 372.5482 371.2858 364.2783];
    i_min_2 = [361.1795 346.7502 344.1106 343.5710];
    o_max_2 = [0.5301 0.8532 0.3516 0.5];
    o_min_2 = [0 0.3157 0 1.17e-4];
    d1 = abs(R_Ratio_p - 0.85);
    d2 = abs(0.89 - R_Ratio_p);
    i_max = d1/(d1+d2)*i_max_2 + d2/(d1+d2)*i_max_1;
    i_min = d1/(d1+d2)*i_min_2 + d2/(d1+d2)*i_min_1;
    o_max = d1/(d1+d2)*o_max_2 + d2/(d1+d2)*o_max_1;
    o_min = d1/(d1+d2)*o_min_2 + d2/(d1+d2)*o_min_1;
elseif (R_Ratio_p>=0.89) & (R_Ratio_p<0.9)
    i_max_1 = [376.4271 372.5482 371.2858 364.2783];
    i_min_1 = [361.1795 346.7502 344.1106 343.5710];
    o_max_1 = [0.5301 0.8532 0.3516 0.5];
    o_min_1 = [0 0.3157 0 1.17e-4];
    i_max_2 = [370.4641 364.2783 364.2783 364.2783];
```

```
        i_min_2 = [361.1397 346.7502 343.9951 343.4998];
        o_max_2 = [0.5360 0.8532 0.1653 0.5];
        o_min_2 = [0 0.2999 0 1.17e-4];
        d1 = abs(R_Ratio_p - 0.89);
        d2 = abs(0.9 - R_Ratio_p);
        i_max = d1/(d1+d2)*i_max_2 + d2/(d1+d2)*i_max_1;
        i_min = d1/(d1+d2)*i_min_2 + d2/(d1+d2)*i_min_1;
        o_max = d1/(d1+d2)*o_max_2 + d2/(d1+d2)*o_max_1;
        o_min = d1/(d1+d2)*o_min_2 + d2/(d1+d2)*o_min_1;
    elseif (R_Ratio_p>=0.9) & (R_Ratio_p<0.95)
        i_max_1 = [370.4641 364.2783 364.2783 364.2783];
        i_min_1 = [361.1397 346.7502 343.9951 343.4998];
        o_max_1 = [0.5360 0.8532 0.1653 0.5];
        o_min_1 = [0 0.2999 0 1.17e-4];
        i_max_2 = [364.7221 364.2783 364.2783 364.2783];
        i_min_2 = [359.5993 345.1851 343.3504 343.2073];
        o_max_2 = [0.56724 0.8532 0.21387 0.5];
        o_min_2 = [0 0.21851 0 1.17e-4];
        d1 = abs(R_Ratio_p - 0.9);
        d2 = abs(0.95 - R_Ratio_p);
        i_max = d1/(d1+d2)*i_max_2 + d2/(d1+d2)*i_max_1;
        i_min = d1/(d1+d2)*i_min_2 + d2/(d1+d2)*i_min_1;
        o_max = d1/(d1+d2)*o_max_2 + d2/(d1+d2)*o_max_1;
        o_min = d1/(d1+d2)*o_min_2 + d2/(d1+d2)*o_min_1;
    elseif (R_Ratio_p>=0.95) & (R_Ratio_p<1.0)
        i_max_1 = [364.7221 364.2783 364.2783 364.2783];
        i_min_1 = [359.5993 345.1851 343.3504 343.2073];
        o_max_1 = [0.56724 0.8532 0.21387 0.5];
        o_min_1 = [0 0.21851 0 1.17e-4];
        i_max_2 = [364.7221 364.2783 364.2783 364.2783];
        i_min_2 = [354.6084 343.9560 343.2418 343.1541];
        o_max_2 = [0.5760 0.8532 0.2261 0.5];
        o_min_2 = [0 0.1974 0 1.17e-4];
        d1 = abs(R_Ratio_p - 0.95);
        d2 = abs(1.0 - R_Ratio_p);
        i_max = d1/(d1+d2)*i_max_2 + d2/(d1+d2)*i_max_1;
        i_min = d1/(d1+d2)*i_min_2 + d2/(d1+d2)*i_min_1;
        o_max = d1/(d1+d2)*o_max_2 + d2/(d1+d2)*o_max_1;
        o_min = d1/(d1+d2)*o_min_2 + d2/(d1+d2)*o_min_1;
    end;
end;
if constant_R_Ratio == 0.0
    if (Rpercent == 0.0)
        i_max = [389.8419 389.6200 385.4282 370.2828];
        i_min = [361.1776 346.7498 344.1105 343.5611];
        o_max = [0.5309 0.8532 0.6532 0.5];
        o_min = [0 0.0882 0 1.17e-4];
    elseif (Rpercent>0.0) & (Rpercent<10.0)
        i_max_1 = [389.8419 389.6200 385.4282 370.2828];
        i_min_1 = [361.1776 346.7498 344.1105 343.5611];
        o_max_1 = [0.5309 0.8532 0.6532 0.5];
        o_min_1 = [0 0.0882 0 1.17e-4];
        i_max_2 = [386.9120 378.8623 372.5038 366.3068];
        i_min_2 = [361.1377 346.7502 343.9911 343.4980];
        o_max_2 = [0.5361 0.8532 0.6673 0.5];
        o_min_2 = [0 0.1466 0 1.17e-4];
        d1 = abs(Rpercent - 0.0);
        d2 = abs(10.0 - Rpercent);
        i_max = d1/(d1+d2)*i_max_2 + d2/(d1+d2)*i_max_1;
        i_min = d1/(d1+d2)*i_min_2 + d2/(d1+d2)*i_min_1;
        o_max = d1/(d1+d2)*o_max_2 + d2/(d1+d2)*o_max_1;
        o_min = d1/(d1+d2)*o_min_2 + d2/(d1+d2)*o_min_1;
    elseif (Rpercent>=10.0) & (Rpercent<20.0)
        i_max_1 = [386.9120 378.8623 372.5038 366.3068];
        i_min_1 = [361.1377 346.7502 343.9911 343.4980];
        o_max_1 = [0.5361 0.8532 0.6673 0.5];
        o_min_1 = [0 0.1466 0 1.17e-4];
```

```
        i_max_2 = [382.0114 374.2861 371.9075 364.2783];
        i_min_2 = [361.0607 346.7502 343.8830 343.4504];
        o_max_2 = [0.5403 0.8532 0.6031 0.5];
        o_min_2 = [0 0.2186 0 1.17e-4];
        d1 = abs(Rpercent - 10.0);
        d2 = abs(20.0 - Rpercent);
        i_max = d1/(d1+d2)*i_max_2 + d2/(d1+d2)*i_max_1;
        i_min = d1/(d1+d2)*i_min_2 + d2/(d1+d2)*i_min_1;
        o_max = d1/(d1+d2)*o_max_2 + d2/(d1+d2)*o_max_1;
        o_min = d1/(d1+d2)*o_min_2 + d2/(d1+d2)*o_min_1;
    elseif (Rpercent>=-10.0) & (Rpercent<0.0)
        i_max_1 = [389.9035 389.8734 389.4124 379.6643];
        i_min_1 = [361.1834 346.7502 344.1106 343.6387];
        o_max_1 = [0.5247 0.8532 0.6412 0.5928];
        o_min_1 = [0 0.0418 0 1.17e-4];
        i_max_2 = [389.8419 389.6200 385.4282 370.2828];
        i_min_2 = [361.1776 346.7498 344.1105 343.5611];
        o_max_2 = [0.5309 0.8532 0.6532 0.5];
        o_min_2 = [0 0.0882 0 1.17e-4];
        d1 = abs(Rpercent - (-10.0));
        d2 = abs(0.0 - Rpercent);
        i_max = d1/(d1+d2)*i_max_2 + d2/(d1+d2)*i_max_1;
        i_min = d1/(d1+d2)*i_min_2 + d2/(d1+d2)*i_min_1;
        o_max = d1/(d1+d2)*o_max_2 + d2/(d1+d2)*o_max_1;
        o_min = d1/(d1+d2)*o_min_2 + d2/(d1+d2)*o_min_1;
    elseif (Rpercent>=-20.0) & (Rpercent<-10.0)
        i_max_1 = [389.9044 389.9010 389.7948 385.5087];
        i_min_1 = [361.1853 346.7502 344.1106 343.7321];
        o_max_1 = [0.5221 0.8532 0.6392 0.8141];
        o_min_1 = [0 0.0151 0 1.17e-4];
        i_max_2 = [389.9035 389.8734 389.4124 379.6643];
        i_min_2 = [361.1834 346.7502 344.1106 343.6387];
        o_max_2 = [0.5247 0.8532 0.6412 0.5928];
        o_min_2 = [0 0.0418 0 1.17e-4];
        d1 = abs(Rpercent - (-20.0));
        d2 = abs(-10.0 - Rpercent);
        i_max = d1/(d1+d2)*i_max_2 + d2/(d1+d2)*i_max_1;
        i_min = d1/(d1+d2)*i_min_2 + d2/(d1+d2)*i_min_1;
        o_max = d1/(d1+d2)*o_max_2 + d2/(d1+d2)*o_max_1;
        o_min = d1/(d1+d2)*o_min_2 + d2/(d1+d2)*o_min_1;
    end;
end;
```

normalize_sim_input.m

```
% ================================================
% This function normalizes the network inputs to a value btw -1 and 1.
% inputs
% sim_i: inputs to the network
% i_max: maximum values for the network inputs
% i_min: minimum values for the network inputs
% outputs
% sim_i_norm: normalized network inputs
%================================================
function [sim_i_norm] = normalize_sim_input(sim_i, i_max, i_min)
in1 = 2*(sim_i(:,1)-i_min(1,1))/(i_max(1,1)-i_min(1,1)) -1;
in2 = 2*(sim_i(:,2)-i_min(1,2))/(i_max(1,2)-i_min(1,2)) -1;
in3 = 2*(sim_i(:,3)-i_min(1,3))/(i_max(1,3)-i_min(1,3)) -1;
in4 = 2*(sim_i(:,4)-i_min(1,4))/(i_max(1,4)-i_min(1,4)) -1;
% fprintf('INPUTS Normalized \n');
sim_i_norm = [in1 in2 in3 in4];
sim_i_norm = sim_i_norm';
% fprintf('INPUTS Combined \n');
```

simulate.m

```
% ================================================
% Simulates the network and finds output to given input by interpolation
% ------------------------------------------------------------------------
% inputs
% R_Ratio_p: L/V ratio of the column = R/(1+R)
% constant_R_Ratio:
% it is 1.0,when the column is operating at constant R after total reflux;
% it is 0.0,when the column is operating at variable R after total reflux
%Rpercent:indicates percent change from the optimal R profile when the
%column operates at variable reflux ratio after total reflux
% sim_i_norm: normalized network inputs
% ------------------------------------------------------------------------
% output
% sim_output: network output
% ================================================
function [sim_output] = simulate(R_Ratio_p, constant_R_Ratio, Rpercent,
sim_i_norm)
sim_output=zeros(4,1);
if constant_R_Ratio == 1.0
    if (R_Ratio_p == 1.0)
        load E1;
        sim_output(:,1)=sim(E1,sim_i_norm(:,1));
    elseif (R_Ratio_p>=0.5) & (R_Ratio_p<0.6)
        d1 = abs(R_Ratio_p - 0.5);
        d2 = abs(0.6 - R_Ratio_p);
        load E05;
        net1=E05;
        load E06;
        net2=E06;
        sim_output_1(:,1)=sim(net1,sim_i_norm(:,1));
        sim_output_2(:,1)=sim(net2,sim_i_norm(:,1));
  sim_output(:,1)=d1/(d1+d2)*sim_output_2(:,1)+d2/(d1+d2)*sim_output_1(:,1);
    elseif (R_Ratio_p>=0.6) & (R_Ratio_p<0.7)
        d1 = abs(R_Ratio_p - 0.6);
        d2 = abs(0.7 - R_Ratio_p);
        load E06;
        net1=E06;
        load E07;
        net2=E07;
        sim_output_1(:,1)=sim(net1,sim_i_norm(:,1));
        sim_output_2(:,1)=sim(net2,sim_i_norm(:,1));
        sim_output(:,1)=d1/(d1+d2)*sim_output_2(:,1) +
d2/(d1+d2)*sim_output_1(:,1);
    elseif (R_Ratio_p>=0.7) & (R_Ratio_p<0.8)
        d1 = abs(R_Ratio_p - 0.7);
        d2 = abs(0.8 - R_Ratio_p);
        load E07;
        net1=E07;
        load E08;
        net2=E08;
        sim_output_1(:,1)=sim(net1,sim_i_norm(:,1));
        sim_output_2(:,1)=sim(net2,sim_i_norm(:,1));
  sim_output(:,1)=d1/(d1+d2)*sim_output_2(:,1)+d2/(d1+d2)*sim_output_1(:,1);
    elseif (R_Ratio_p>=0.8) & (R_Ratio_p<0.85)
        d1 = abs(R_Ratio_p - 0.8);
        d2 = abs(0.85 - R_Ratio_p);
        load E08;
        net1=E08;
        load E085;
        net2=E085;
        sim_output_1(:,1)=sim(net1,sim_i_norm(:,1));
        sim_output_2(:,1)=sim(net2,sim_i_norm(:,1));
sim_output(:,1)=d1/(d1+d2)*sim_output_2(:,1)+d2/(d1+d2)*sim_output_1(:,1);
    elseif (R_Ratio_p>=0.85) & (R_Ratio_p<0.89)
        d1 = abs(R_Ratio_p - 0.85);
```

```
        d2 = abs(0.89 - R_Ratio_p);
        load E085;
        net1=E085;
        load E089;
        net2=E089;
        sim_output_1(:,1)=sim(net1,sim_i_norm(:,1));
        sim_output_2(:,1)=sim(net2,sim_i_norm(:,1));
sim_output(:,1)=d1/(d1+d2)*sim_output_2(:,1)+d2/(d1+d2)*sim_output_1(:,1);
    elseif (R_Ratio_p>=0.89) & (R_Ratio_p<0.9)
        d1 = abs(R_Ratio_p - 0.89);
        d2 = abs(0.9 - R_Ratio_p);
        load E089;
        net1=E089;
        load E09;
        net2=E09;
        sim_output_1(:,1)=sim(net1,sim_i_norm(:,1));
        sim_output_2(:,1)=sim(net2,sim_i_norm(:,1));
 sim_output(:,1)=d1/(d1+d2)*sim_output_2(:,1)+d2/(d1+d2)*sim_output_1(:,1);
    elseif (R_Ratio_p>=0.9) & (R_Ratio_p<0.95)
        d1 = abs(R_Ratio_p - 0.9);
        d2 = abs(0.95 - R_Ratio_p);
        load E09;
        net1=E09;
        load E095;
        net2=E095;
        sim_output_1(:,1)=sim(net1,sim_i_norm(:,1));
        sim_output_2(:,1)=sim(net2,sim_i_norm(:,1));
  sim_output(:,1)=d1/(d1+d2)*sim_output_2(:,1)+d2/(d1+d2)*sim_output_1(:,1);
    elseif (R_Ratio_p>=0.95) & (R_Ratio_p<1.0)
        d1 = abs(R_Ratio_p - 0.95);
        d2 = abs(1.0 - R_Ratio_p);
        load E095;
        net1=E095;
        load E1;
        net2=E1;
        sim_output_1(:,1)=sim(net1,sim_i_norm(:,1));
        sim_output_2(:,1)=sim(net2,sim_i_norm(:,1));
  sim_output(:,1)=d1/(d1+d2)*sim_output_2(:,1)+d2/(d1+d2)*sim_output_1(:,1);
    end;
end;
if constant_R_Ratio == 0.0
    if (Rpercent == 0.0)
        load Eopt;
        sim_output(:,1)=sim(Eopt,sim_i_norm(:,1));
    elseif (Rpercent>0.0) & (Rpercent<10.0)
        d1 = abs(Rpercent - 0.0);
        d2 = abs(10.0 - Rpercent);
        load Eopt;
        net1=Eopt;
        load Epl10opt;
        net2=Epl10opt;
        sim_output_1(:,1)=sim(net1,sim_i_norm(:,1));
        sim_output_2(:,1)=sim(net2,sim_i_norm(:,1));
  sim_output(:,1)=d1/(d1+d2)*sim_output_2(:,1)+d2/(d1+d2)*sim_output_1(:,1);
    elseif (Rpercent>=10.0) & (Rpercent<20.0)
        d1 = abs(Rpercent - 10.0);
        d2 = abs(20.0 - Rpercent);
        load Epl10opt;
        net1=Epl10opt;
        load Epl20opt;
        net2=Epl20opt;
        sim_output_1(:,1)=sim(net1,sim_i_norm(:,1));
        sim_output_2(:,1)=sim(net2,sim_i_norm(:,1));
  sim_output(:,1)=d1/(d1+d2)*sim_output_2(:,1)+d2/(d1+d2)*sim_output_1(:,1);
    elseif (Rpercent>=-10.0) & (Rpercent<0.0)
        d1 = abs(Rpercent - (-10.0));
        d2 = abs(0.0 - Rpercent);
```

```
        load Eopt;
        net1=Eopt;
        load Emin10opt;
        net2=Emin10opt;
        sim_output_1(:,1)=sim(net1,sim_i_norm(:,1));
        sim_output_2(:,1)=sim(net2,sim_i_norm(:,1));
  sim_output(:,1)=d1/(d1+d2)*sim_output_2(:,1)+d2/(d1+d2)*sim_output_1(:,1);
    elseif (Rpercent>=-20.0) & (Rpercent<-10.0)
        d1 = abs(Rpercent - (-20.0));
        d2 = abs(-10.0 - Rpercent);
        load Emin10opt;
        net1=Emin10opt;
        load Emin20opt;
        net2=Emin20opt;
        sim_output_1(:,1)=sim(net1,sim_i_norm(:,1));
        sim_output_2(:,1)=sim(net2,sim_i_norm(:,1));
  sim_output(:,1)=d1/(d1+d2)*sim_output_2(:,1)+d2/(d1+d2)*sim_output_1(:,1);
    end;
end;
```

unnormalize_sim_output.m

```
% ================================================
% This function unnormalizes the network output
% ------------------------------------------------------------------------
% inputs
% sim_output: network output
% o_max: maximum values of the network outputs
% o_min: minimum values of the network outputs
% ------------------------------------------------------------------------
% output
% sim_output: unnormalized network output
% ================================================
function [sim_output] = unnormalize_sim_output(sim_output, o_max, o_min) ...
sim_output=sim_output';
s1 = sim_output(1,1);
s2 = sim_output(1,2);
s3 = sim_output(1,3);
s4 = sim_output(1,4);
o1=(s1+1)/2*(o_max(1,1)-o_min(1,1))+o_min(1,1);
o2=(s2+1)/2*(o_max(1,2)-o_min(1,2))+o_min(1,2);
o3=(s3+1)/2*(o_max(1,3)-o_min(1,3))+o_min(1,3);
o4=(s4+1)/2*(o_max(1,4)-o_min(1,4))+o_min(1,4);
% fprintf('sim_output UNnormalized \n');
sim_output = [o1 o2 o3 o4];
%sim.output is summed up&divided by this sum to have the sum of conc.=1.0
    sum=sim_output(1,1)+sim_output(1,2)+sim_output(1,3)+sim_output(1,4);
    sim_output(1,1)=sim_output(1,1)/(sum);
    sim_output(1,2)=sim_output(1,2)/(sum);
    sim_output(1,3)=sim_output(1,3)/(sum);
    sim_output(1,4)=sim_output(1,4)/(sum);
%the concentrations are checked, if less then zero, it is equated to zero.
for j=1:4;
    if sim_output(1,j) < 0
            sim_output(1,j)=0;
    end;
end;
```

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name : Bahar, Almıla
Nationality: Turkish (TC)
Date and Place of Birth: 5 September 1978, Ankara
Marital Status: Single
Phone: +90 312 2102636
Fax: +90 312 2102600
email : abahar@metu.edu.tr

EDUCATION

Degree	Institution	Year of Graduation
MS	METU Chemical Engineering Department	2003
BS	METU Chemical Engineering Department	2000
High School	Atatürk High School, Ankara	1995

WORK EXPERIENCE

Year	Place	Enrollment
2000-2007	METU Chemical Engineering Department	Research Assistant

FOREIGN LANGUAGES

Fluency in English, Intermediate German

PUBLICATIONS

1. Bahar A., Güner E., Özgen C., Halıcı U., "Design of State Estimators for the Inferential Control of an Industrial Distillation Column", 2006 IEEE World Congress on Computational Intelligence – International Joint Conference on Neural Networks, Vancouver, BC, Canada, July 16-21, 2006.

2. Bahar A., Özgen C., Leblebicioğlu K., Halıcı U., "An Artificial Neural Network Estimator Design for the Inferential Model Predictive Control of an Industrial

Distillation Column", Industrial and Engineering Chemistry Researchl, Vol. 43, No.19, 2004.

3. Bahar A., Güner E., Özgen C., "Endüstriyel Bir Damıtma Kolonunda Yapay Sinir Ağı ve Adaptif Sinirsel Bulanık Tahmin Metotlarının Kullanımı", 6. Ulusal Kimya Mühendisliği Kongresi, Ege Üniversitesi, İzmir, Eylül 2004.

4. Bahar A., Özgen C., Halıcı U., "Endüstriyel Çok Bileşenli Bir Damıtma Kolonunun Yapay Sinir Ağı Kullanan Model Öngörümlü Denetleçle Denetimi", Türk Otomatik Kontrol Ulusal Toplantısı, ODTÜ, Ankara, Türkiye, Eylül 2002.

HOBBIES

Tennis, Reading Book, Movies