

Fall 2009

Design and implementation of embedded adaptive controller using ARM processor.

Hoan The Nguyen
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_theses

Recommended Citation

Nguyen, Hoan The, "Design and implementation of embedded adaptive controller using ARM processor." (2009). *Master's Theses*. 3991.

DOI: <https://doi.org/10.31979/etd.uff7-zkwd>

https://scholarworks.sjsu.edu/etd_theses/3991

This Thesis is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Theses by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

DESIGN AND IMPLEMENTATION OF EMBEDDED ADAPTIVE CONTROLLER
USING ARM PROCESSOR

A Thesis

Presented to

The Faculty of the Department of Computer Engineering
San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Hoan The Nguyen

December 2009

UMI Number: 1484326

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

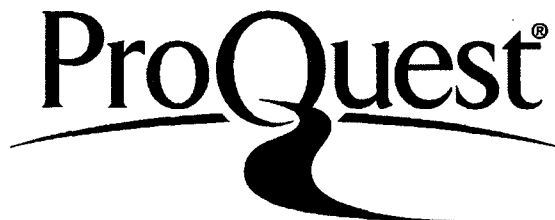
In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 1484326

Copyright 2010 by ProQuest LLC.

All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

© 2009

Hoan The Nguyen

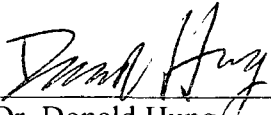
ALL RIGHTS RESERVED

SAN JOSE STATE UNIVERSITY

The Undersigned Thesis Committee Approves the Thesis Titled
DESIGN AND IMPLEMENTATION OF EMBEDDED ADAPTIVE
CONTROLLER USING ARM PROCESSOR

by
Hoan The Nguyen


APPROVED FOR THE DEPARTMENT OF COMPUTER ENGINEERING



Dr. Donald Hung, Department of Computer Engineering 11-4-09
Date

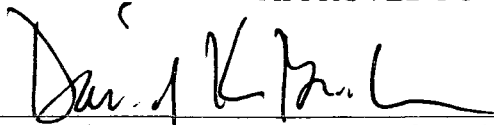


Dr. Lee Chang, Department of Computer Engineering 11/4/09
Date



Dr. Xiao Su, Department of Computer Engineering 11/4/09
Date

APPROVED FOR THE UNIVERSITY



Associate Dean Office of Graduate Studies and Research 12/3/09
Date

ABSTRACT

DESIGN AND IMPLEMENTATION OF EMBEDDED ADAPTIVE CONTROLLER USING ARM PROCESSOR

by Hoan The Nguyen

This thesis is concerned with development of embedded adaptive controllers for industrial applications. Many industrial processes present challenging control problems such as high nonlinearity, time-varying dynamic behaviors, and unpredictable external disturbances. Conventional controllers are too limited to successfully resolve these problems. Therefore, the adaptive control strategy, an advanced control theory, is applied to overcome deficiencies of the conventional controllers.

Through the thesis, an embedded adaptive controller is designed and implemented for the specific case study, a gasoline-refining plant. The adaptive controller design is initially achieved in continuous-time space and then converted to discrete-time space by using z -transform. It is finally implemented on an advanced reduced instruction set computer machine (ARM) processor. A plant simulator written in C++ executes functions of the gasoline-refining plant. Therefore, an integrated testing environment is developed in order that the embedded adaptive controller can interact in real-time fashion with the plant simulator located in a remote computer. In all system tests, the embedded adaptive controller successfully controlled the remote plant simulator and fully satisfied all control objectives.

ACKNOWLEDGMENTS

I wish to express my gratitude to my advisor Dr. Donald Hung. I am most grateful for his invaluable advice and for the freedom he allowed me in conducting my research. I admire the illuminating inspiration of his ideas and his ability to motivate people.

I would like to thank Dr. Lee Chang and Dr. Xiao Su for serving as the members of my thesis committee.

TABLE OF CONTENTS

	Page
List of Tables	xiv
List of Figures	xvi
List of Abbreviations	xxi
List of Symbols	xxii
Chapter 1 Introduction.....	1
1.1. Background.....	1
1.2. Statement of the Problem.....	2
1.3. Objectives of the Study.....	3
1.4. Scope of the Study	4
1.5. Methodology	5
Chapter 2 Literature Review.....	7
2.1. Gasoline Refinery	7
2.2. Distillation Equipment.....	7
2.3. Distillation Operation Principles.....	9
2.4. Methods of Distillation Column Control	10
2.5. Principles of Adaptive Control	11
2.6. Adaptive Schemes.....	13
2.6.1. Gain Scheduling.....	13
2.6.2. Model Reference Adaptive Control Systems.....	14
2.7. Lyapunov Stability Theory	14

2.7.1. Definition of Lyapunov Stability	15
2.7.2. Lyapunov Stability Theorem	16
2.7.3. Lyapunov Function	16
2.8. ARM Processor	16
2.8.1. System Architecture of ARM Development Board	17
2.8.2. Memory Organization	19
Chapter 3 Plant Modeling and Simulation	20
3.1. Introduction	20
3.2. Process Description	20
3.3. Process Calculation	21
3.4. Process Description and Control Scheme	22
3.5. Plant Modeling	22
3.6. Plant Simulation	24
Chapter 4 Analog Controller Design	26
4.1. Introduction	26
4.2. System Architecture	26
4.3. Construction of the Reference Model	27
4.3.1. Stability Test	30
4.3.2. Controllability and Observability Test	30
4.4. Analog Controller Synthesis	31
4.4.1. Plant	31
4.4.2. Reference Model	32

4.4.3. Feedback Control Loop.....	32
4.4.4. Compatibility Condition	33
4.4.5. Error Equation.....	35
4.4.6. Adaptation Law.....	36
4.4.7. Stability of the Analog Controller.....	37
4.5. Analog Controller Simulation.....	38
4.5.1. Simulation Program	38
4.5.2. Simulation Result.....	39
4.5.3. Error Reduction.....	42
Chapter 5 Digital Controller Design.....	43
5.1. Introduction.....	43
5.2. Digital Controller Synthesis.....	44
5.2.1. Plant Model.....	44
5.2.2. Reference Model	45
5.2.3. Linear Feedback Controller	45
5.2.4. Compatibility Condition	46
5.2.5. Adaptive Mechanism	47
Chapter 6 Simulation of the Adaptive Digital Controller.....	50
6.1. Introduction.....	50
6.2. Dynamic Simulation Using MATLAB.....	50
6.2.1. Simulation Program	50
6.2.2. Error Calculation.....	51

6.2.3. MATLAB Simulation Result.....	52
6.3. C++ Simulation Project	55
6.3.1. Plant Model Class	55
6.3.2. Reference Model Class	57
6.3.3. Linear Control Class	58
6.3.4. Comparator Class.....	59
6.3.5. Adaptive Mechanism Class.....	60
6.3.6. Create Makefile and Build Project.....	62
6.3.7. C++ Simulation Result.....	62
Chapter 7 Implementation of the Embedded Adaptive Controller.....	66
7.1. Introduction.....	66
7.1.1. Embedded Adaptive Controller Using ARM Processor	66
7.1.2. In-Hardware Validation Scheme.....	67
7.1.3. System Architecture and Operations	68
7.1.4. Kernel Image of the ARM Board.....	72
7.2. Common Database	73
7.3. Plant Simulator.....	74
7.4. Development of the Main Testing Form in HTML	75
7.5. Adaptive Mechanism Program	77
7.5.1. CGI Program.....	78
7.5.2. NFS Server Setup.....	82
7.6. Building the Kernel Image.....	84

7.6.1. NFS Client Setup	84
7.6.2. Enabling the CGI Protocol.....	87
7.6.3. Enabling the Boa Web Server.....	89
7.7. Testing.....	93
7.7.1. Test Procedure	93
7.7.2. Test Results.....	98
Chapter 8 Conclusion and Future Work.....	104
8.1. Conclusion	104
8.2. Future Work.....	105
Appendix A: Distillation Control Techniques	109
A.1. Column Pressure Control.....	109
A.1.1. Coolant Manipulation	109
A.1.2. Vent-bleed.....	110
A.2. Column Level Control	111
A.3. Methods of Distillation Column Control	113
A.3.1. Degrees of Freedom of Distillation Process	113
A.3.2. Control Structures.....	114
A.3.3. Energy Balance Structure	116
A.3.4. Material Balance Structure	117
Appendix B: Process Calculation.....	119
B.1. Basic Engineering Data.....	119
B.2. Distillation Process Calculation.....	122

B.2.1. Equilibrium Flash Vaporization Curves	122
B.2.2. Yield of Fractions	123
B.2.3. Operating Pressure	124
B.3. Calculation for the Feed Section.....	124
B.3.1. Description.....	124
B.3.2. Calculation	125
B.4. Calculation for the Stripping Section.....	126
B.4.1. Description.....	126
B.4.2. Calculation	127
B.5. Calculation for the Rectifying Section.....	128
B.5.1. Description.....	128
B.5.2. Calculation	129
B.6. Calculation Results	131
B.6.1. Raw Gasoline Property	131
B.6.2. Main Stream Property	131
B.7. Process Description.....	132
B.7.1. Simplified Process Flow Diagram	132
B.7.2. Distillation Column.....	133
B.7.3. Blending System and Product Distribution	134
B.7.4. Feed Control.....	135
B.7.5. Top Column Section	136
B.7.6. Bottom Section.....	136

Appendix C: Mathematical Model.....	137
C.1. Introduction.....	137
C.2. Dynamic Study of Distillation Process.....	138
C.2.1. Generic Trays.....	138
C.2.2. Feed Tray	140
C.2.3. Top Section.....	141
C.2.4. Bottom Section.....	143
C.3. Mathematical Model of Distillation Process.....	145
C.4. Simplified Model	145
C.5. Mathematical Model of the Gasoline Refinery.....	147
C.5.1. Relative Volatility.....	147
C.5.2. Latent Heat and Boilup.....	147
C.5.3. Liquid Holdups on Tray and Column Base	148
C.5.4. Liquid Holdup in Reflux Drum.....	150
C.6. Basic Mathematical Model of the Plant.....	150
Appendix D: Dynamic Simulation.....	154
D.1. Modular Decomposition of the Column	154
D.2. Simulation with MATLAB Simulink	155
Appendix E: Construction of Reference Model.....	161
E.1. Model Construction	161
E.2. Stability Test	167

Appendix F: Source Code	169
F.1. Adaptive Mechanism	169
F.2. Plant Model.....	170
F.3. Reference Model.....	171
F.4. Linear Controller.....	172
F.5. Comparator	173
F.6. Controlled Output	174
F.7. Reference Outputs.....	175
F.8. CGI Program.....	175
Appendix G: MATLAB/C++ Simulation Output Files	198
G.1. State Variables Files	198
G.2. Reference and Controlled Outputs.....	202
G.3. Plant Error.....	206
G.4. Adaptive Gains.....	210
Appendix H: Hardware Validation Output Files.....	214
H.1. Validation Output Files of State Variables	214
H.2. Reference and Controlled Outputs.....	218
H.3. Plant Error	222
H.4. Adaptive Gains.....	225
References.....	230

LIST OF TABLES

Table 2.1.	Main parameters of a distillation column.	9
Table 2.2.	Main features of the ARM-7 development board.	18
Table 3.1.	Main streams of the plant.	21
Table 3.2.	Steady-state compositions of the plant.	25
Table 4.1.	Plant output errors for different adaptation rates.	42
Table 7.1.	Structured data files.	73
Table 7.2.	Some key features of an HTML form.	76
Table 7.3.	Structured output data files.	98
Table 7.4.	Output data of state variables.	99
Table 7.5.	Comparison result between the embedded and software models.	102
Table A.1.	Manipulated variables and controlled variables of a distillation column. .	114
Table A.2.	Typical column control structures.	115
Table B.1.	Condensate composition analyzed by gas chromatography.	119
Table B.2.	Distillation data.	120
Table B.3.	Gasoline quality requirement.	121
Table B.4.	Relationship between ASTM, TBP, and EFV.	122
Table B.5.	Material balances for the feed section.	126
Table B.6.	Material and energy balances of the stripping section.	128
Table B.7.	Material and energy balances around the boundary (A).	130
Table B.8.	ASTM distillation curve of the raw gasoline.	131
Table B.9.	Main streams of the plant.	132

Table C.1.	Parameters of a generic tray.....	138
Table G.1.	Simulation result of state variables.	198
Table G.2.	Simulation result of reference and controlled outputs.	202
Table G.3.	Simulation result of plant errors.	206
Table G.4.	Simulation result of adaptive gains.....	210
Table H.1.	Validation result of state variables.....	214
Table H.2.	Simulation result of reference and controlled outputs.	218
Table H.3.	Simulation result of plant errors.	222
Table H.4.	Simulation result of adaptive gains.....	226

LIST OF FIGURES

Figure 2.1.	Distillation column.....	8
Figure 2.2.	Contacting between vapor and liquid phases at a tray.....	9
Figure 2.3.	Energy balance control structure.	10
Figure 2.4.	Block diagram of an adaptive system.	12
Figure 2.5.	Procedure for selection of an appropriate adaptive scheme.....	12
Figure 2.6.	Block diagram of a system with gain scheduling.	13
Figure 2.7.	Block diagram of a model reference adaptive control scheme.	14
Figure 2.8.	Illustration of Lyapunov stability.....	15
Figure 2.9.	System block diagram of the ARM development board.....	17
Figure 2.10.	Memory map.....	19
Figure 3.1.	Simplified diagram of the condensate distillation.	20
Figure 3.2.	Process flow diagram of the gasoline refinery.....	22
Figure 3.3.	Vapor liquid equilibrium relationship.....	24
Figure 3.4.	Simulation result of the concentration on each stage.....	25
Figure 4.1.	System architecture of the adaptive control system.....	26
Figure 4.2.	Two-input two-output representation for the plant.....	28
Figure 4.3.	Bode responses of two models.....	29
Figure 4.4.	Simulation program for the analog adaptive controller.	38
Figure 4.5.	External disturbances.	40
Figure 4.6.	Reference states and plant states.....	40
Figure 4.7.	Controlled outputs and reference outputs.	41

Figure 4.8. Plot of plant output errors during simulation.	41
Figure 5.1. Input and output signals of ZOH.	43
Figure 5.2. Block diagram of a sampled-data system.	43
Figure 5.3. Logic diagram of the adaptation law.	48
Figure 6.1. Simulation program for the digital adaptive system.	51
Figure 6.2. External disturbances.	52
Figure 6.3. Reference states and plant states.	53
Figure 6.4. Controlled outputs and reference outputs.	53
Figure 6.5. Error plots.	54
Figure 6.6. Plant class's header file.	56
Figure 6.7. Reference model class's header file.	57
Figure 6.8. Linear control class's header file.	59
Figure 6.9. Comparator class's header file.	60
Figure 6.10. Adaptive mechanism class's header file.	61
Figure 6.11. Running the "adaptive_control" executive file.	63
Figure 6.12. State variables.	63
Figure 6.13. Controlled outputs and reference outputs.	64
Figure 6.14. Plant output errors.	64
Figure 7.1. Elementary block diagram of the system.	66
Figure 7.2. In-hardware validation scheme.	67
Figure 7.3. Integrated testing environment.	68
Figure 7.4. NFS client/server interaction for mounting a network file system.	69

Figure 7.5. Sequence diagram.	71
Figure 7.6. Compile and build the plant simulator in a console terminal.	75
Figure 7.7. Main testing form in HTML.	76
Figure 7.8. Pseudo code of the adaptive mechanism program.	78
Figure 7.9. Flow chart of the CGI program.	80
Figure 7.10. Pseudo code of the CGI program.	82
Figure 7.11. Enable NFS server for Ubuntu machine.	83
Figure 7.12. Check rpc daemon status.	83
Figure 7.13. Select network file systems.	84
Figure 7.14. Select NFS supports.	85
Figure 7.15. Select the “mount” option.	85
Figure 7.16. Select the “umount” option.	86
Figure 7.17. Modifications for Busybox Makefile.	87
Figure 7.18. Start “xinetd” for Linux Ubuntu.	91
Figure 7.19. Manually load the kernel image to the ARM board.	92
Figure 7.20. Manually run the kernel image.	92
Figure 7.21. Process status.	93
Figure 7.22. Mount network files for the NFS client.	94
Figure 7.23. Ping ARM board from the Linux machine.	95
Figure 7.24. Ping the Linux machine from the ARM board.	95
Figure 7.25. Open the main page of Boa web server.	96
Figure 7.26. Start the plant simulator in a Linux machine.	97

Figure 7.27. Testing result displayed on the web browser.	97
Figure 7.28. Testing result displayed on the plant simulator's monitor.	98
Figure 7.29. Variable states during simulation of the embedded adaptive controller.	101
Figure 9.1. Block diagram of the experimental pilot plant.	106
Figure 9.2. Adaptive mechanism programmed in a PLC.	107
Figure 9.3. Integrated testing environment for SoCs and other digital systems.	108
Figure A.1. Column pressure control using coolant manipulation.	110
Figure A.2. Column pressure control using vent bleed.	111
Figure A.3. Some typical types of reboiler.	112
Figure A.4. Energy balance structure.	116
Figure A.5. D–V control structure.	117
Figure A.6. L–B control structure.	118
Figure B.1. Yield curve.	124
Figure B.2. Equilibrium phase flows at the feed section.	125
Figure B.3. Equilibrium phase flows at the stripping section.	127
Figure B.4. Equilibrium phase flows at the rectifying section.	129
Figure B.5. Simplified process flow diagram of the gasoline plant.	133
Figure B.6. A local control devices for feed pumps.	136
Figure C.1. A generic tray.	138
Figure C.2. Variation of liquid depth across a generic tray.	139
Figure C.3. Feed section.	140
Figure C.4. Top section.	141

Figure C.5. Bottom section.	143
Figure D.1. Modular decomposition scheme for the distillation column.....	154
Figure D.2. Hierarchical structure of the simulation program.	155
Figure D.3. Main program in MATLAB Simulink.....	155
Figure D.4. Module of the rectifying section.	156
Figure D.5. Module of the stripping section.	157
Figure D.6. Module of the column base and reboiler.....	158
Figure D.7. Module of a generic tray.	158
Figure D.8. Module of the feed tray.....	159
Figure D.9. Module of the eighth tray.....	159
Figure D.10. Module of the condenser and reflux drum.....	160
Figure E.1. Model of a generic tray.....	161

LIST OF ABBREVIATIONS

API	American Petroleum Institute
ARM	Advanced RISC machine
ASTM	American Society for Testing and Materials
CGI	Common gateway interface
DCS	Distributed control system
EFV	Equilibrium flash vaporization
LPG	Liquefied petroleum gas
MIMO	Multi-input multi-output
MON	Motor octane number
MRAC	Model-reference adaptive control
MTBE	Methyl tert-butyl ether
NFS	Network file system
PC	Personal computer
PID	Proportional integral differential
PLC	Programmable logic controller
RISC	Reduced instruction set computer
RON	Research octane number
RVP	Reid vapor pressure
SISO	Single-input single-output
TBP	True boiling point
ZOH	Zero-order hold

LIST OF SYMBOLS

A	State matrix
B	Input matrix
C	Output matrix
e	State error
L	Feedback matrix
M	Feedforward matrix
u	Control signal
x	State variable
y	Controlled output

Greek Symbols

α	Relative volatility
γ	Adaptation rate
θ	Adaptive gain

Subscript

B	Bottom product
D	Distillate or overhead product
F	Feed
m	Reference model
r	Reduced-order model

CHAPTER 1

INTRODUCTION

1.1. Background

We have seen that the price of microprocessors has dropped significantly since their creation in the early 1970s. As a consequence, microprocessors have increasingly become the main part of many control systems. In the 1980s, microcontrollers were developed due to the integration of microprocessors and other peripheral devices in the same chip. Microcontrollers have widely increased applications of embedded systems due to their low cost and high performance. Therefore, embedded controllers using microprocessors such as ARM processors are aspired to develop for robotics and industrial applications.

ARM processor has a 32-bit RISC architecture invented by ARM Company. There are various ARM processors; however, they are based on a common architecture and provide high performance, low power consumption, and reduced cost. They are licensed by most of leading semiconductor manufacturers, who have shipped more than ten billion ARM processors since the company was established in 1990 [1].

In this study, an embedded adaptive controller is designed and implemented for a gasoline refinery. In the petroleum refining industry, distillation is the most popular and important process. Crude oil, a mixture of thousands of organic substances, is refined by the distillation process to produce a host of liquid fuels, pure chemical substances, and petrochemicals [2]. The size of the refining industry is truly immense with the total processing capacity of approximately 4 billion tons per year [3]. In addition, the

percentage of energy consumption by distillation is very high, about 40% for a typical chemical plant [4]. Humphrey [5] estimates that, with advanced control strategy, there is potential for an average 15% reduction in the energy consumption by distillation in the United States' refining industry. Therefore, enhancement of distillation control would result in major economic improvement for petroleum refineries.

1.2. Statement of the Problem

Distillation processes are highly multivariable and nonlinear. The dynamic analysis and simulation of a distillation process are thus very complicated. Its theoretical model can contain several hundreds of state variables; however, in practice, the amount of information on the process is usually insufficient. Therefore, reduced state space models should be employed [6].

Distillation processes are multiple-input multiple-output (MIMO) systems. Conventional PID controllers normally employ a single-input single-output (SISO) approach with appropriate pairings. For distillation processes, a controlled variable is affected by many manipulated variables so that conventional controllers will have multiple control loops. Consequently, there exists coupling, which causes serious problems, particularly in the cases of high-purity distillation systems. De-couplers are usually deployed to reduce the interaction between the control loops. However, the SISO approach with pairing and de-coupling is only partially successful. Shen and Lee [7] believe that the use of multivariable controllers such as adaptive controllers can greatly improve the situation since they treat the MIMO process as a single system instead of many individual subsystems.

Time-varying dynamic behaviors of distillation processes and the existence of unpredictable external disturbances also present challenging control problems. If conventional controllers such as PID controllers are used, they must be re-tuned for different operating conditions. This task is costly, time-consuming, and even unfeasible. Unlike conventional controllers, adaptive control systems use special adaptation mechanisms that can self-adjust their control settings to compensate unanticipated changes in the process or in the environment [8]. Therefore, adaptive control strategy has been progressively applied for solving the difficult control problems [9].

There has been some research on applying adaptive control strategy to distillation processes. For example, Nguyen and Afzulpurkar [10] propose an adaptive control system for a natural gasoline plant; and Narendra and Annaswamy [11] suggest an adaptive scheme for distillation column. However, most of these works focus on continuous-time space. To implement adaptive algorithms on digital computers, they must be discretized using z-transform.

After the embedded adaptive controller is successfully implemented, it must interact with either a real pilot plant or a software-based plant simulator running on a remote computer to close the control loop and allow real-time data transfer between the plant and the controller. Therefore, an integrated testing environment must be developed to be able to verify the controller performances.

1.3. Objectives of the Study

The structure of distillation as well as other chemical processes has become increasingly complex due to better management of energy and raw materials. Hence, the

design of the control system for a complete plant has become intimately related to the design of the process itself [12].

In this study, we design and implement the embedded adaptive controller and plant simulator through various phases: 1) development of mathematical model; 2) process calculation and modeling; 3) controller design in both continuous-time and discrete spaces; and 4) hardware implementation and testing. The objectives of the study are described as follows:

- Create adaptive algorithms for the plant using Lyapunov stability theory;
- Design adaptive controllers for the plant in both continuous-time and discrete-time spaces;
- Create a software-based plant simulator, which executes functions of the gasoline plant and its local auxiliaries;
- Implement the embedded adaptive controller using an ARM processor, which fully satisfies all control objectives under different operating conditions of the plant;
- Develop an integrated testing environment for verification of the embedded adaptive controller.

1.4. Scope of the Study

Building a pilot plant is not feasible for this study since it is very expensive. Hence, identification method, which requires operational data of the plant, is not applicable. Instead, the reference model of the gasoline refinery will be developed using mathematical method.

The embedded adaptive controller will be implemented on an ARM-7 development board. It will interact with the plant simulator written in C++ language on a remote computer through an Ethernet network.

1.5. Methodology

The plant simulator, a software simulator, is created to play the roles of the distillation column and other local auxiliaries. The embedded adaptive controller is an adaptive mechanism that governed by an adaptation law and programmed in the ARM kernel image. The embedded adaptive controller is thus physically an ARM processor, residing on an ARM development board. An integrated environment is developed via an Ethernet for testing the adaptive system. The major steps in this study are as follows:

- Study the plant dynamics and adaptive control strategy;
- Develop the mathematical model of the plant;
- Perform the plant simulation using MATLAB, a numerical computing environment and fourth generation programming language developed by Mathworks [13], to obtain the steady-state data;
- Construct the reference model of the plant using mathematical approach to prepare for the next phase of adaptive controller design;
- Design the analog adaptive controller for the plant in continuous-time space based on Lyapunov stability theory;
- Discretize the adaptive system using z-transform and design the digital adaptive controller in discrete-time space;

- Simulate both analog and digital designs using either MATLAB or C++ and verify their control performances;
- Implement the embedded adaptive controller on an ARM development board;
- Connect the embedded adaptive controller and the plant simulator via a local network so that they form closed loop;
- Use network file system (NFS) technology to allow the embedded adaptive controller and the plant simulator to have read or write access to the common database in an NFS server.

CHAPTER 2

LITERATURE REVIEW

2.1. Gasoline Refinery

Petroleum refining emphasizes distillation of crude oil into various fractions. The crude oil is heated to 370 degrees to 470 degrees Celsius and pumped into a distillation column. The crude oil is then separated into the following major fractions: 1) naphtha; 2) light naphtha; and 3) heavy naphtha. The naphtha or light naphtha fraction can be used as a component of finished gasoline. The heavy naphtha needs further processing such as catalytically reforming to become high-octane blending stock.

In this thesis, we study a gasoline refinery whose feed stream is condensate produced from associated gas or natural gas fields. It contains fewer high-boiling components than crude oil does. Moreover, its antiknock quality is quite low since its composition has a large amount of straight paraffinic hydrocarbons [2]. In the gasoline refinery, the distillation process is responsible for cutting off light components as propane and butane to ensure the saturated vapor pressure and volatility of the gasoline product. It will be finally blended with high octane number boosters and additives to ensure the octane number and other quality criteria such as anti-oxidization.

2.2. Distillation Equipment

Distillation process is done in a special chemical apparatus, called distillation column. It is made up of several parts, each of which is used to transfer heat energy or enhance mass transfer. A typical distillation column consists of the following

components: 1) a vertical shell; 2) column internals such as trays; 3) a reboiler; 4) a condenser; and 5) a reflux drum.

The reboiler provides heat for vaporizing the feed stream. The condenser is to chill and condense overhead vapor. The reflux drum stores the condensed vapor and recycles liquid reflux back to the column. The vertical shell covers the column internals. A schematic of a typical distillation column with a single feed and two product streams is shown in Figure 2.1. Main parameters of a distillation column are shown in Table 2.1.

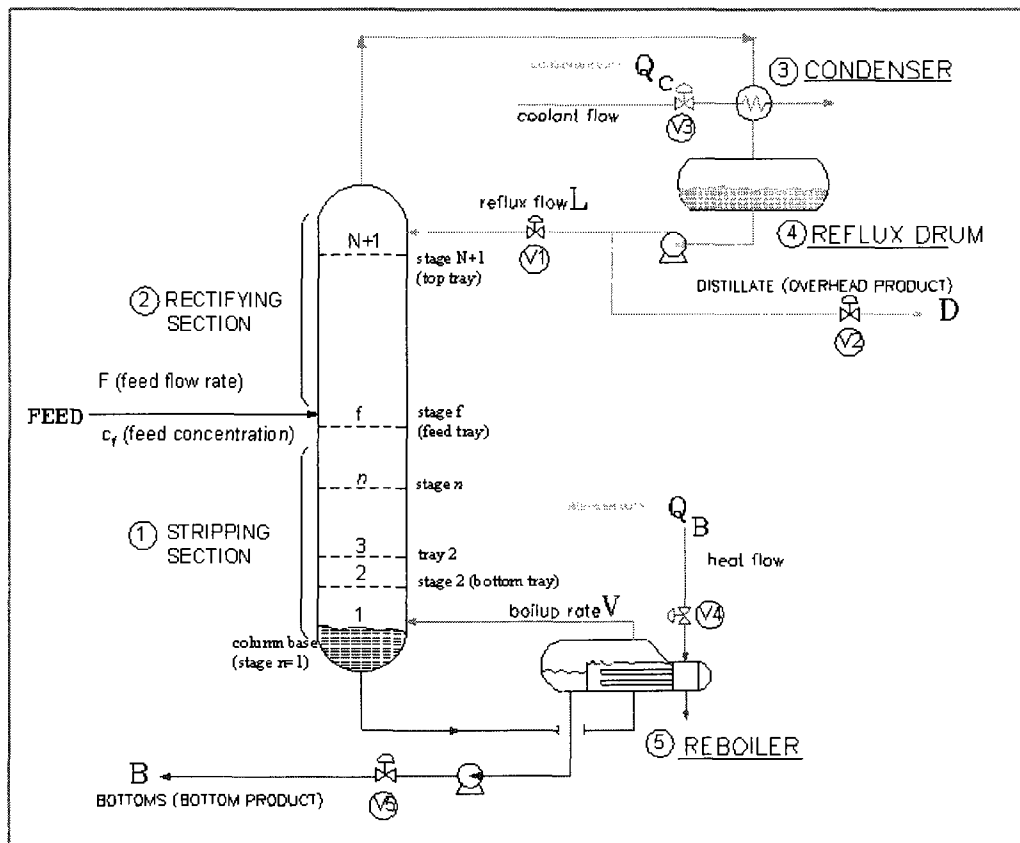


Figure 2.1. Distillation column.

Table-2.1. Main parameters of a distillation column.

Material stream		Position	Flow rate	Concentration
Feed stream		Stage f	F	c_F
Overhead	Reflux stream	Stage N	L	x_D
	Distillate stream	Reflux drum	D	x_D
Bottoms	Return stream	Stage 1	V	x_B
	Bottoms stream	Reboiler	B	x_B

2.3. Distillation Operation Principles

The liquid mixture is fed onto the feed tray. In normal operation, there is a certain amount of liquid on each plate. The reboiler is used to supply energy for generating vapor, which moves upward and passes through the liquid on each tray. The intimate contact between liquid and vapor is usually accomplished by using bubble caps as shown in Figure 2.2.

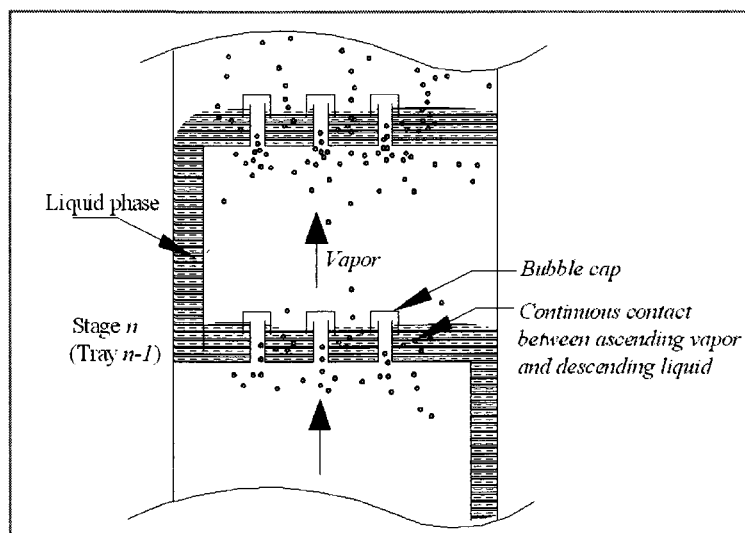


Figure 2.2. Contacting between vapor and liquid phases at a tray.

The overhead vapor, upon leaving the top plate, enters the condenser where it is condensed. The liquid is then collected in the reflux drum from which the reflux stream and the top product stream are withdrawn. The top product is also called distillate.

The liquid that leaves the bottom tray of the column enters the reboiler, where it is vaporized. The produced vapor is forced to flow back up through the column; and the liquid withdrawn from the reboiler is called bottoms or bottom product.

2.4. Methods of Distillation Column Control

Distillation process has two degrees of freedom; therefore, there are various control structures. One of the most common control structures is energy balance structure as shown in Figure 2.3. Refer to Appendix A for more details on control structures.

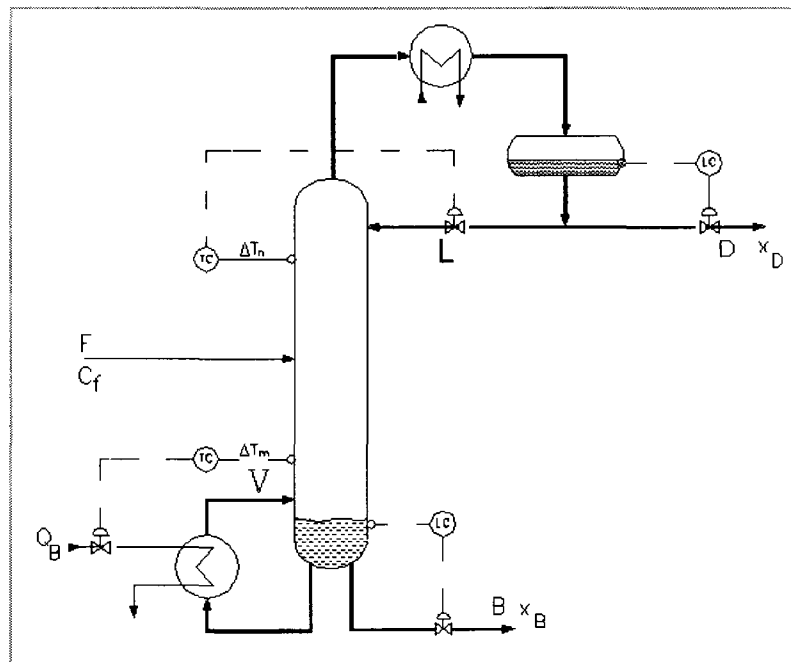


Figure 2.3. Energy balance control structure.

The energy balance structure or $L-V$ structure can be considered the standard control structure for dual composition control. In Figure 2.3, the reflux flow rate L and the boilup manipulator V are used to control the primary outputs associated with the product specifications. The secondary outputs, which are the liquid holdups in the reflux drum and in the column base, are usually controlled by distillate flow rate D and the bottoms flow rate B .

2.5. Principles of Adaptive Control

An adaptive control system can synthesize adaptive gains in such a way as to compensate for variations in the characteristics of the process it controls. There are many types of adaptive control systems, which differ only in the way the controller parameters are adjusted [12].

Adaptive controllers are needed for chemical and petroleum processes since most of them are nonlinear and nonstationary. The linearized models that are used to design linear controllers depend on particular steady states. When their desired steady-state operation has variation, the best values of the controller parameters will change. In addition, their time-varying dynamic characteristics usually cause deterioration in the performance of the linear controller. Therefore, adaptation of the controller parameters is required.

Stephanopoulos [12] defines the objective of an adaptation law. It must guide the adaptive mechanism to the best adjustment of the controller parameters rather than keep the controlled variables at the specified set points, which can be accomplished by conventional control loops.

An adaptive control system can be considered as consisting of two loops as shown in Figure 2.4. One loop is a conventional feedback loop. The other loop is the parameter adjustment loop [14].

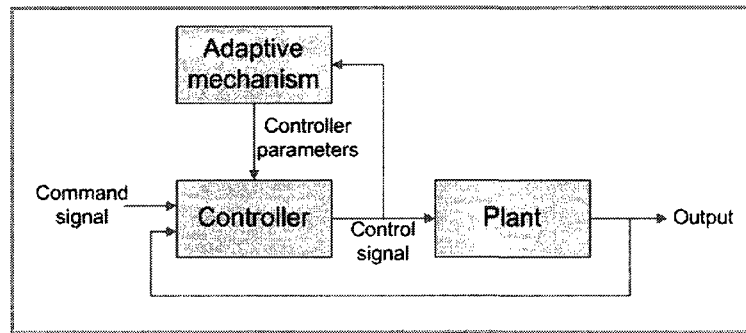


Figure 2.4. Block diagram of an adaptive system.

There are two different methods for adjustment of the controller parameters: 1) direct method; and 2) indirect method. In the direct method, the adaptation law directs the controller parameters adjustment such as gain scheduling and model-reference adaptive systems. In the indirect method, at any adjustment step, new controller parameters are obtained by solving the design problem such as self-tuning regulators.

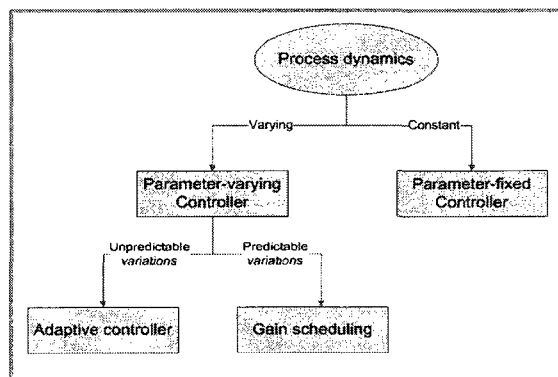


Figure 2.5. Procedure for selection of an appropriate adaptive scheme.

Adaptive controllers, being inherently nonlinear, are more sophisticated than conventional feedback controllers. The decision whether to use adaptive control is based on the dynamic behaviors of the process as depicted in Figure 2.5.

2.6. Adaptive Schemes

2.6.1. Gain Scheduling

In many control systems, it is possible to determine measurable variables that have well-defined connections with changes in process dynamics. These variables can be used to adjust the controller parameters. This approach is called gain scheduling because the scheme was originally used to measure the gain and then change the controller parameters, as shown in Figure 2.6.

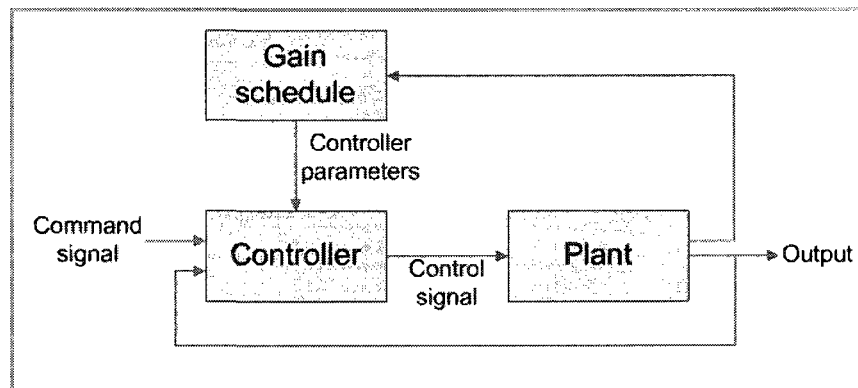


Figure 2.6. Block diagram of a system with gain scheduling.

The system can be viewed as consisting of two loops: 1) an inner loop composed of the plant and the controller; and 2) an outer loop that adjusts the controller parameters on the basis of operating conditions. Gain scheduling can be considered as a mapping

from plant parameters to controller parameters [14]; hence, it can be implemented as a function or a lookup table.

2.6.2. Model Reference Adaptive Control Systems

The model reference adaptive control (MRAC) scheme was originally proposed to solve the problem in which the performance specifications are given in terms of a reference model. This model generates the desired output corresponding to the command signal.

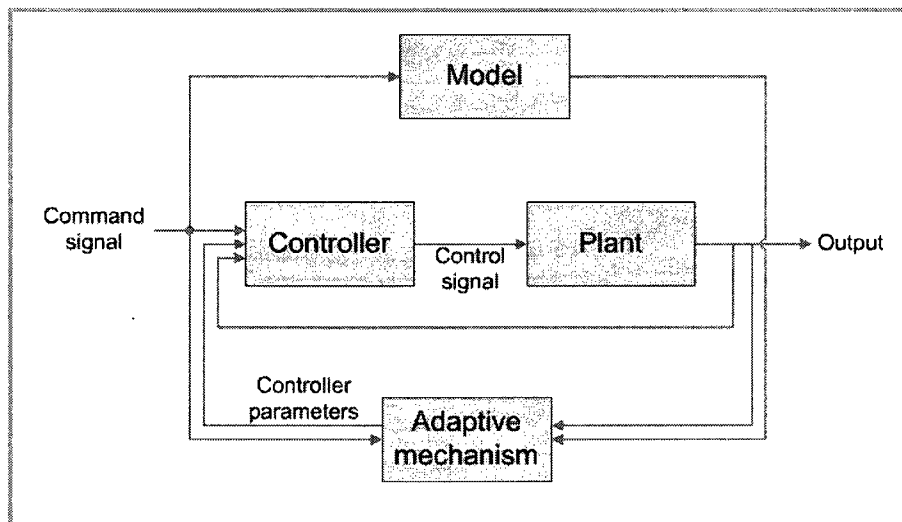


Figure 2.7. Block diagram of the model reference adaptive control scheme.

The controller consists of two loops, as shown in Figure 2.7. The inner loop is an ordinary feedback loop composed of the plant and the controller. The outer loop adjusts the controller parameters in such a way that the error, the difference between the plant output and the model output, is small. The key problem is to design the adaptive mechanism so that the adaptive control system is stable, and the error goes zero [14].

2.7. Lyapunov Stability Theory

Lyapunov stability theory primarily addresses the stability problem of any system regardless of being linear or nonlinear. It is very significant for nonlinear control system design since it is the only tool available when other methods are failed [15].

Lyapunov considers the nonlinear differential equation with zero initial condition:

$$\frac{dx}{dt} = f(x). \quad (2.1)$$

Lyapunov investigates whether the solution of (2.1) is stable with respect to disturbances or not.

2.7.1. Definition of Lyapunov Stability

The solution $x(t) = 0$ is stable if for a given $\varepsilon > 0$ there exists a number $\delta(\varepsilon) > 0$ such that all solutions with initial conditions $\|x(0)\| < \delta$ have the following property:

$$\|x(t)\| < \varepsilon, \text{ for } 0 \leq t < \infty.$$

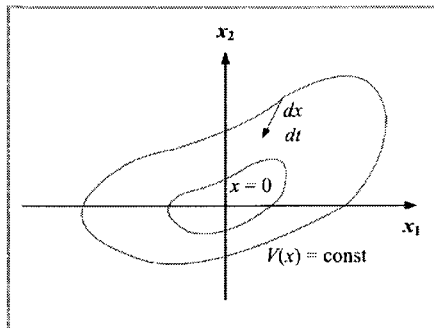


Figure 2.8. Illustration of Lyapunov stability.

The solution is asymptotically stable if a positive number δ can be found such that all solutions with $\|x(0)\| < \delta$ have the following property:

$$\|x(t)\| \rightarrow 0 \text{ as } t \rightarrow \infty.$$

2.7.2. Lyapunov Stability Theorem

The solution $x(t) = 0$ is stable if there exists a function $V: R^n \rightarrow R$ that is positive definite such that its derivative along the solution of (2.1) is negative definite as

$$\frac{dV}{dt} = \frac{\partial V^T}{\partial x} \frac{dx}{dt} = \frac{\partial V^T}{\partial x} f(x) = -W(x) \quad (2.2)$$

If dV/dt is negative semi-definite, the solution is asymptotically stable. The function V is called a Lyapunov function for the system in (2.1).

2.7.3. Lyapunov Function

Assume that the following linear system is asymptotically stable:

$$\frac{dx}{dt} = Ax \quad (2.3)$$

For each symmetric positive definite matrix Q , there exists a unique symmetric positive definite matrix P such that

$$A^T P + PA = -Q \quad (2.4)$$

Equation (2.4) has always a unique solution with P positive definite; and the following function:

$$V(x) = x^T P x \quad (2.5)$$

is a Lyapunov function [14].

2.8. ARM Processor

We use an ARM-7 development board manufactured by Samsung to implement the embedded adaptive controller for the gasoline refinery. The following sections describe its architecture and key specifications.

2.8.1. System Architecture of ARM Development Board

The S3C44B0X ARM-7 development board has various features including 8KB cache, internal SRAM, LCD controller, 2-channel UART with handshake, 4-channel DMA, system manager with chip select logic and FP/EDO/SDRAM controller, 5-channel timers, I/O ports, RTC, 8-channel 10-bit ADC, IIC-BUS interface, and IIS-BUS interface [16].

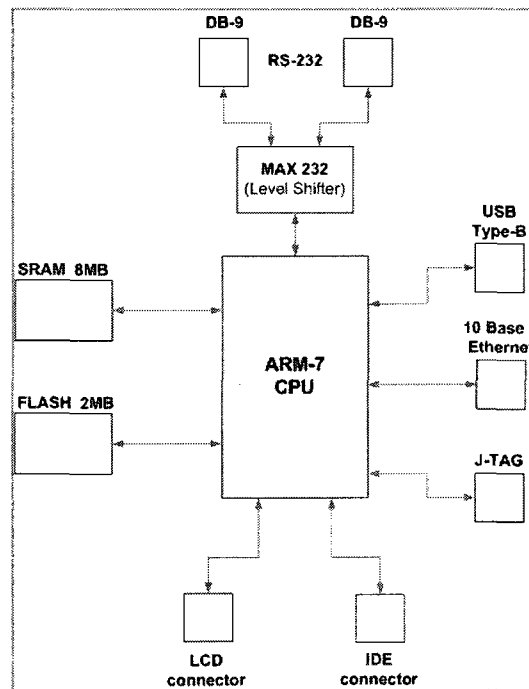


Figure 2.9. System block diagram of the ARM development board.

The S3C44B0X is developed using an ARM7TDMI core and new bus architecture, SAMBA II or Samsung ARM CPU embedded microcontroller bus architecture [16]. The main features are shown in Table 2.2.

Table 2.2. Main features of the ARM-7 development board.

Features	Descriptions
CPU	“ARM7TDMI” CPU core for 16/32 bit operations.
	Cache 8KB for memory management.
	On-chip ICE breaker debug support with JTAG based debugging solution.
	32x8 bit hardware multiplier.
	SAMBA II bus architecture up to 66MHz.
	Little/big endian support.
Memory subsystem	Address space: total 256 MB.
	8 memory banks.
	Supports programmable 8/16/32-bit data bus width for each bank.
	Fixed bank start address and programmable bank size for 7 banks.
	Programmable bank start address and bank size for one bank.
	Fully programmable access cycles for all memory banks.
	Supports self-refresh mode in DRAM/SDRAM for power-down.
	Supports asymmetric/symmetric address of DRAM.
GPIO ports	8 external interrupt ports.
	71 multiplexed input/output ports.
UART	2-channel UART with DMA-based or interrupt based operation.
	Supports 5-bit, 6-bit, 7-bit, or 8-bit serial data.
	Supports hardware handshaking during transmission/receiving.
	Programmable baud rate.
	Supports IrDA 1.0 (115.2kbps).
	Loop back mode for testing.
	Each channel have two internal 32-byte FIFO for Rx and Tx.
Ethernet	1 port 10 Base T (10/100Mbps)

2.8.2. Memory Organization

The memory space is 256MB in total. There are 8 memory banks: 1) the first 6 memory banks are used only for ROM; and 2) the last 2 memory banks can be realized by RAM.

The 8-MB SDRAM is in Bank 6. The beginning address of Bank 6 is 0x0c000000. Therefore, the kernel image will be load to the ARM board at address 0x0c008000. The memory map is shown in Figure 2.10.

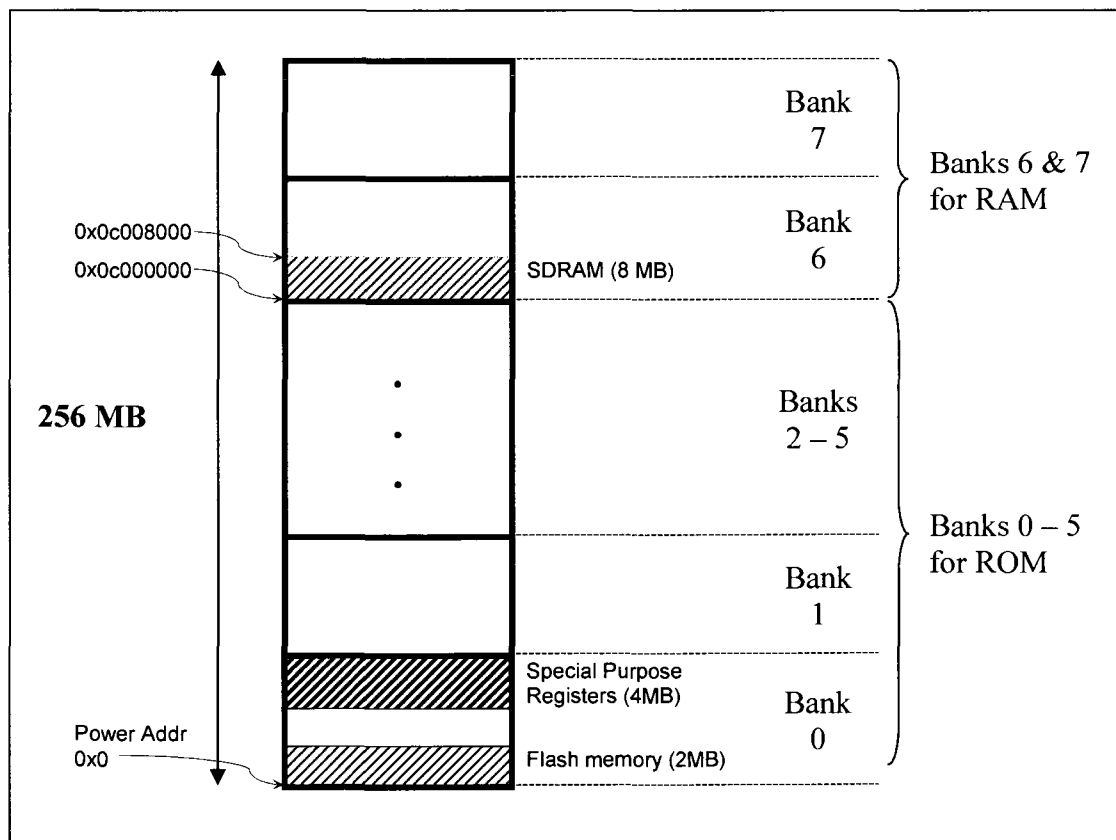


Figure 2.10. Memory map.

CHAPTER 3

PLANT MODELING AND SIMULATION

3.1. Introduction

Process modeling is the first phase in the whole system design procedure. This phase is significant since it provides steady-state data of the plant. We will base on these data to design the adaptive control system. In this section, we study the distillation process of the gasoline refinery and perform the process modeling and simulation.

3.2. Process Description

Condensate, which is condensed from associated gas or natural gas in gas processing plants, will be stabilized by cutting off light components such as propane and butane in a distillation column.

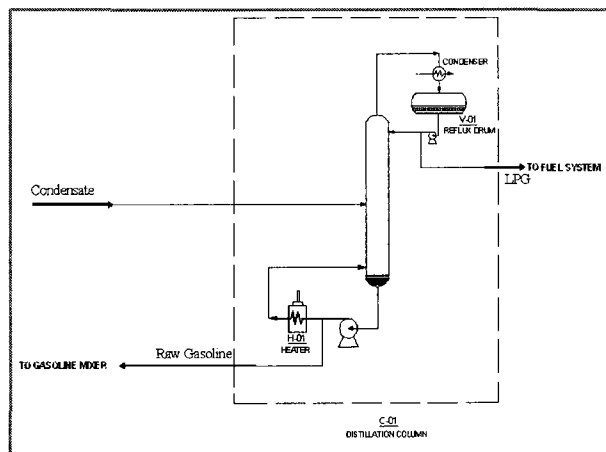


Figure 3.1. Simplified diagram of the condensate distillation.

The distillation column has 24 actual trays, which is equivalent to 14 theoretical trays. Condensate is fed to the seventh tray. The top product is Liquefied Petroleum Gas

(LPG). The bottom product is naphtha, which will be blended with high octane number boosters and additives to produce the finished gasoline.

The control objective is to keep the product qualities within the following limits under different operating conditions:

$$x_D \geq 98\% \quad (3.1)$$

and

$$x_B \leq 2.0\% \quad (3.2)$$

where x_D and x_B are the product compositions or the product qualities.

3.3. Process Calculation

Based on the design basis, we calculate steady-state values of the gasoline-refining plant. The process calculation is given in Appendix B. The key values of the plant design are listed in Table 3.1.

Table 3.1. Main streams of the plant.

	Condensate	LPG	Raw gasoline
Temperature, C	118	46	144
Pressure, atm	8.6	4.0	4.6
Density, kg/m ³	670	585	727
Volume flow rate, m ³ /h	227.6	8.78	21.88
Mass flow rate, kg/h	15480	5061	10405
Mass flow rate, ton/year	130000	43000	87000

3.4. Process Description and Control Scheme

The process flow diagram of the plant is shown in Figure 3.2. Refer to Appendix B for more details on process description.

The control structure is selected as $L-V$ structure that directly controls separation quality. Based on this structure, we construct the control scheme for the distillation system.

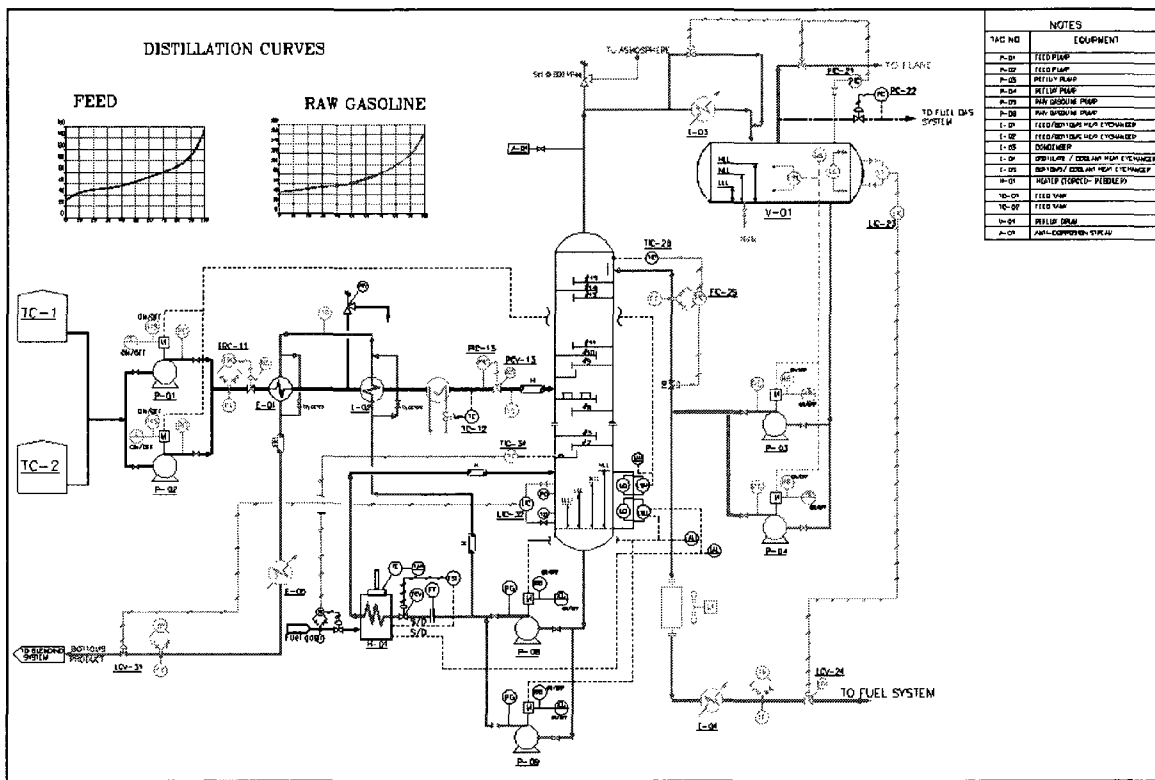


Figure 3.2. Process flow diagram of the gasoline refinery.

3.5. Plant Modeling

For the distillation column with 14 trays and 20 components, the number of differential and algebraic equations is equal to $14 \cdot (2 \cdot 20 + 3) = 602$ equations [17]. The

plant order should be reduced. We can lump a group of components together to make pseudo-component; and the column dynamics are modeled on pseudo-component only [18]. The feed can be approximated by a pseudo binary mixture of LPG (iso-butane, n-butane and propane) and naphtha (iso-pentane, n-pentane, and heavier components).

As a result, the mathematical model of the gasoline refinery is represented by a set of 31 nonlinear differential and algebraic equations:

$$14.03 \dot{x}_{16} = 164.6291y_{15} - 75.6380x_{16} - 92.7597x_{16}$$

$$5.8\dot{x}_{15} = 164.6291(y_{14} - y_{15}) + 75.6380(x_{16} - x_{15})$$

$$5.8\dot{x}_{14} = 164.6291(y_{13} - y_{14}) + 75.6380(x_{15} - x_{14})$$

$$5.8\dot{x}_{13} = 164.6291(y_{12} - y_{13}) + 75.6380(x_{14} - x_{13})$$

$$5.8\dot{x}_{12} = 164.6291(y_{11} - y_{12}) + 75.6380(x_{13} - x_{12})$$

$$5.8\dot{x}_{11} = 164.6291(y_{10} - y_{11}) + 75.6380(x_{12} - x_{11})$$

$$5.8\dot{x}_{10} = 164.6291(y_9 - y_{10}) + 75.6380(x_{11} - x_{10})$$

$$5.8\dot{x}_9 = 66.1139y_8 - 156.38y_9 + 75.6380(x_{10} - x_9) + 59.95$$

$$5.8\dot{x}_8 = 66.1139(y_7 - y_8) + 75.6380 x_9 - 188.59x_8 + 33.99$$

$$5.8\dot{x}_7 = 66.1139(y_6 - y_7) + 179.8871(x_8 - x_7)$$

$$5.8\dot{x}_6 = 66.1139(y_5 - y_6) + 179.8871(x_7 - x_6)$$

$$5.8\dot{x}_5 = 66.1139(y_4 - y_5) + 179.8871(x_6 - x_5)$$

$$5.8\dot{x}_4 = 66.1139(y_3 - y_4) + 179.8871(x_5 - x_4)$$

$$5.8\dot{x}_3 = 66.1139(y_2 - y_3) + 179.8871(x_4 - x_3)$$

$$5.8\dot{x}_2 = 66.1139(y_1 - y_2) + 179.8871(x_3 - x_2)$$

$$24.88\dot{x}_1 = 179.8871x_2 - 110.9235x_1 - 66.1139y_1. \quad (3.3)$$

Vapor liquid equilibrium (VLE) relationship on each tray:

$$y_1 = \frac{5.68x_1}{1 + 4.68x_1}$$

$$y_2 = \frac{5.68x_2}{1 + 4.68x_2}$$

$$\vdots$$

$$y_{15} = \frac{5.68x_{15}}{1 + 4.68x_{15}}. \quad (3.4)$$

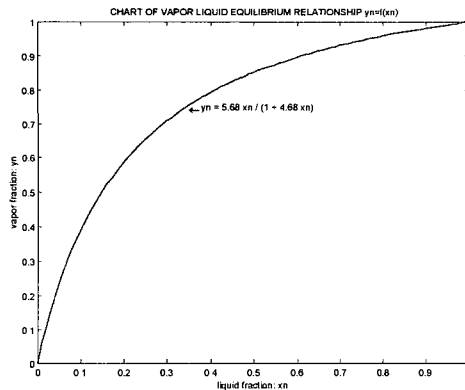


Figure 3.3. Vapor liquid equilibrium relationship.

Refer to Appendix C for more detailed on establishment of mathematical model.

3.6. Plant Simulation

We perform simulation of the plant using MATLAB Simulink. In the simulation program, each stage of the plant is represented by a specific subsystem. Refer to

Appendix D for more detailed on the simulation program. The steady-state solution is summarized in Table 3.2.

Table 3.2. Steady-state compositions of the plant.

n	1	2	3	4	5	6	7	8
x_n^*	0.0111	0.0303	0.0666	0.1196	0.1765	0.2203	0.2461	0.2591
y_n^*	0.0599	0.1507	0.2884	0.4355	0.5490	0.6161	0.6496	0.6651
n	9	10	11	12	13	14	15	16
x_n^*	0.2715	0.2993	0.3637	0.4889	0.6665	0.8319	0.9354	0.9851
y_n^*	0.6792	0.7081	0.7645	0.8446	0.9190	0.9656	0.9880	0.9974

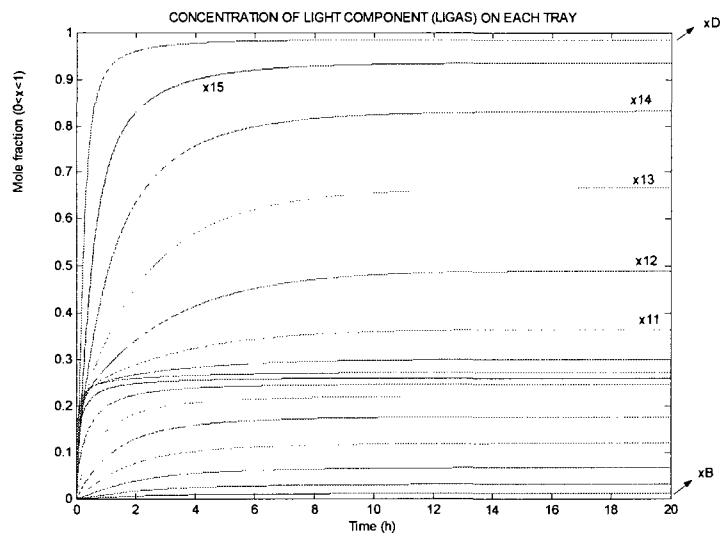


Figure 3.4. Simulation result of the concentration on each stage.

In summary, the plan can obtain the operational objectives in which the purity of the bottom product is greater than 98%, and the impurity of the overhead product is less than 2%.

CHAPTER 4

ANALOG CONTROLLER DESIGN

4.1. Introduction

We apply adaptive control strategy to design the control system for the gasoline plant. The adaptive system is more complicated than other conventional controller since it was synthesized with adaptation law that enables it to operate properly for wide range of operation as demonstrated in [19]–[21]. In the following sections, we will study generic architecture of an analog adaptive controller. We then establish the adaptation law and design the adaptive controller for the plant in continuous-time space.

4.2. System Architecture

As mentioned earlier in [14], Astrom and Wittenmark define adaptive control strategy in which the system can self-adjust its settings to compensate for unpredictable changes in the process or the environment. The system architecture of the adaptive controller is shown in Figure 4.1.

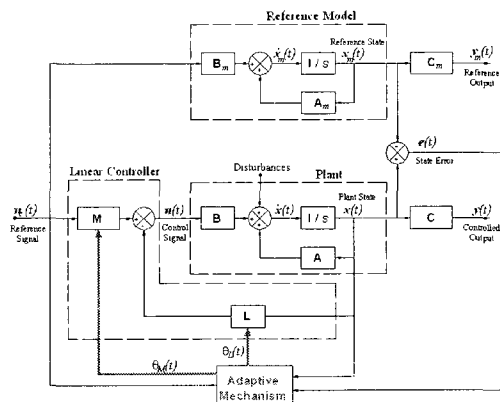


Figure 4.1. System architecture of the adaptive control system.

The adaptive control system will consist of two loops. The inner loop is an ordinary feedback loop composed of the plant and the conventional controller. The outer loop is an adaptive loop that adjusts the conventional controller parameters in such a way that the plant error is small. The adaptive loop will be synthesized based on the Lyapunov stability theory introduced in Sec. 2.7.

There are two kinds of gains including adaptive gain and feedback gain. The purpose of the adaptive mechanism is to enable synthesis of the adaptive gains, which finally change the feedback gains as soon as state errors are detected. The adaptive mechanism is thus the most important component of the adaptive system.

4.3. Construction of the Reference Model

Basically, there are two different methods of reference model construction [22]. They include: 1) mathematical approach; and 2) experimental approach. The mathematical approach is based on physical laws and prior knowledge about the process. The advantages of this approach are: 1) insightful understanding of the process; and 2) physical interpretations of process parameters and variables. However, it is quite difficult and time-consuming to build the model from fundamental knowledge. On the contrary, the experimental approach is based on experimentation. It is also known as system identification. This approach includes the following tasks: 1) experimental planning; 2) selection of model structure; 3) parameter estimation; and 4) validation. For distillation control, system identification is sometimes impractical since the experimentation needs to build a real distillation column or a pilot plant, which is very expensive. Therefore, we select the mathematical approach to construct the reference model.

We observe that the plant has many internal variables; however, its input–output relationship is quite simple, as shown in Figure 5. We adjust the reflux flow rate L and the vapor rate V so that the product quality is met the control objectives defined in (3.1) and (3.2).

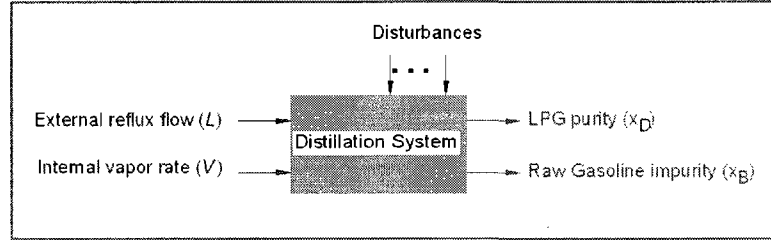


Figure 4.2. Two-input two-output representation for the plant.

We can make linearization at the nominal operating point as follows:

$$\dot{x}(t) = A_\ell x(t) + B_\ell u(t) \quad (4.1)$$

and

$$y(t) = C_\ell x(t) \quad (4.2)$$

where A_ℓ is a 16x16 matrix; B_ℓ is a 16x2 matrix; and C_ℓ is a 2x16 matrix.

The values of A_ℓ , B_ℓ , and C_ℓ matrices are calculated with an algebraic method described in Appendix E.

Many researchers state that the dynamic response of most distillation column is dominated by one large time constant, which is nearly the same, regardless of where an input or disturbance is introduced or where composition is measured. This is well known both from plant measurements [23] and from theoretical studies [24]–[25]. This means that most distillation columns can approximate by first order systems.

We can use Gramian-based input/output balancing of state-space realizations as

```
[sysb,g] = balreal(sys);
```

The last 14 elements of the balanced Gramian matrix are small or zero; so we can eliminate the last 14 states with the MATLAB command of model order reduction:

```
sysmred = modred(sysb,3:16,'del');
```

As the result, the reduced-order model of the plant is a second-order two-input two-output system:

$$\dot{x}_m(t) = A_m x_m(t) + B_m u_c(t) \quad (4.3)$$

and

$$y_m(t) = C_m x_m(t) \quad (4.4)$$

where $A_m = \begin{bmatrix} -6.7941 & -0.9095 \\ 1.4686 & -0.2497 \end{bmatrix}$, $B_m = \begin{bmatrix} -0.1461 & 0.2073 \\ -0.0021 & -0.0281 \end{bmatrix}$, and $C_m = \begin{bmatrix} -0.0624 & -0.0281 \\ 0.2458 & 0.0009 \end{bmatrix}$.

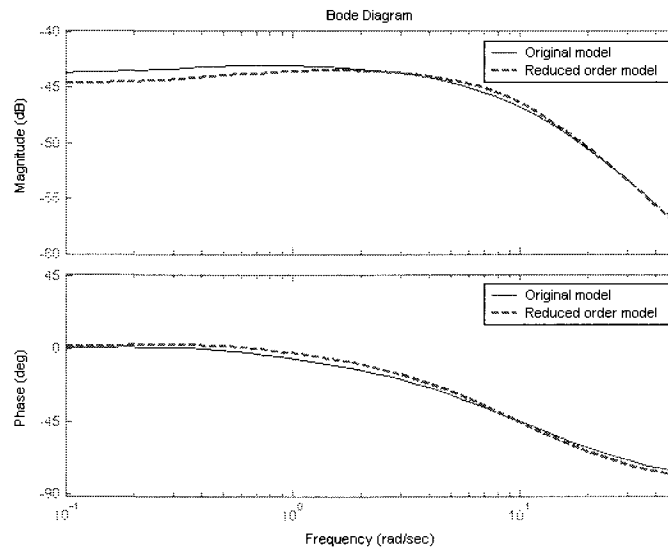


Figure 4.3. Bode responses of two models.

In Figure 4.3, Bode diagram of the reduced-order model is nearly in agreement with the one of the original model. The reduced-order model will be checked with stability and other tests before deciding whether it is a reference model or not.

4.3.1. Stability Test

The system is stable because all eigenvalues of the state matrix $\{-6.5832, -0.4606\}$ are in left-hand side of the complex plane.

4.3.2. Controllability and Observability Test

A system is said to be controllable if and only if it is possible, by means of the input, to transfer the system from any initial state $x(t) = x_t$ to any other state $x_T = x(T)$ in a finite time $T-t \geq 0$.

For any system described in the following forms

$$\dot{x} = Ax(t) + Bu(t) \quad (4.5)$$

and

$$y(t) = Cx(t) + Du(y) \quad (4.6)$$

where A , B , C , and D are matrices.

The matrix $M = [B \ AB \ \dots A^{n-l}B]$, where l is the rank of B and n is the system dimension, is called controllability matrix. The system is completely controllable if M is full rank of n .

A system is said to be observable if and only if it is possible to determine any arbitrary initial state $x(t) = x_t$ by using only a finite record, $y(\tau)$ for $t \leq \tau \leq T$, of the output.

For the system in (4.5) and (4.6), the observability matrix S is defined as

$$S = [C \ CA \ CA^2 \ \dots \ CA^{n-1}]^T$$

where l is the rank of C and n is the system dimension.

The system is completely observable if S is full rank (n).

We can use the following MATLAB commands to test controllability and observability of the reduced-order model:

```
M = ctrb(Ar,Br);    % controllability matrix
S = obsv(Ar,Cr);    % observability matrix
```

As the result, the M and S matrices have full rank; therefore, the model is completely controllable and observable.

In conclusion, the reduced-order linear model fully satisfies the steady-state property. Therefore, it is selected as the reference model for the MRAC system design in the next section.

4.4. Analog Controller Synthesis

4.4.1. Plant

The model of the plant can be expressed in the state space as

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{4.7}$$

and

$$y(t) = Cx(t) \tag{4.8}$$

where $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, $B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$, and $C = \begin{bmatrix} -0.0624 & -0.0281 \\ 0.2458 & 0.0009 \end{bmatrix}$.

The plant parameters including a_{11} , a_{12} , a_{21} , a_{22} , b_{11} , b_{12} , b_{21} , and b_{22} are unknown and dependent on the plant dynamics. The plant model has the following vectors of variables:

- 1) A vector of state variables $x = [x_1 \ x_2]^T$;
- 2) A vector of control signals $u = [u_1 \ u_2]^T$;
- 3) A vector of controlled outputs $y = [y_1 \ y_2]^T$.

4.4.2. Reference Model

The reference model for the plant is a linear time-invariant system as developed in Sec. 4.3:

$$\dot{x}_m(t) = A_m x_m(t) + B_m u_c(t) \quad (4.9)$$

and

$$y_m(t) = C_m x_m(t) \quad (4.10)$$

where $A_m = \begin{bmatrix} -6.7941 & -0.9095 \\ 1.4686 & -0.2497 \end{bmatrix}$, $B_m = \begin{bmatrix} -0.1461 & 0.2073 \\ -0.0021 & -0.0281 \end{bmatrix}$, and $C_m = \begin{bmatrix} -0.0624 & -0.0281 \\ 0.2458 & 0.0009 \end{bmatrix}$.

4.4.3. Feedback Control Loop

A general linear control law is given by:

$$u(t) = M u_c(t) - L x(t) \quad (4.11)$$

The matrices L and M may be chosen as follows:

$$L = \begin{bmatrix} \theta_1 & \theta_2 \\ \theta_3 & \theta_4 \end{bmatrix}$$

$$M = \begin{bmatrix} \theta_5 & \theta_6 \\ \theta_7 & \theta_8 \end{bmatrix}.$$

The closed-loop system is obtained by substituting (4.11) into (4.9):

$$\dot{x} = (A - BL)x + BMu_c.$$

The closed-loop system depends on a parameter $\theta = [\theta_1 \ \theta_2 \ \theta_3 \ \theta_4 \ \theta_5 \ \theta_6 \ \theta_7 \ \theta_8]^T$;

hence, we define:

$$A_c(\theta) = (A - BL)$$

$$B_c(\theta) = BM.$$

As a result, the closed-loop system can be expressed as:

$$\dot{x}(t) = A_c(\theta)x(t) + B_c(\theta)u_c(t)$$

$$A_c(\theta) = A - BL = \begin{bmatrix} a_{11} - b_{11}\theta_1 - b_{12}\theta_3 & a_{12} - b_{11}\theta_2 - b_{12}\theta_4 \\ a_{21} - b_{21}\theta_1 - b_{22}\theta_3 & a_{22} - b_{21}\theta_2 - b_{22}\theta_4 \end{bmatrix}$$

$$B_c(\theta) = BM = \begin{bmatrix} b_{11}\theta_5 + b_{12}\theta_7 & b_{11}\theta_6 + b_{12}\theta_8 \\ b_{21}\theta_5 + b_{22}\theta_7 & b_{21}\theta_6 + b_{22}\theta_8 \end{bmatrix}. \quad (4.12)$$

4.4.4. Compatibility Condition

It is necessary to find at least one value of θ such that the closed-loop equation is equivalent to the reference equation (4.9). A sufficient condition is existence of at least one special value θ^0 such that

$$A_c(\theta^0) = A_m$$

$$B_c(\theta^0) = B_m.$$

This condition ensures a perfect model-following: $x \rightarrow x_m$ when t increases.

Substitute $\theta^0 = [\theta_1^0 \ \theta_2^0 \ \theta_3^0 \ \theta_4^0 \ \theta_5^0 \ \theta_6^0 \ \theta_7^0 \ \theta_8^0]^T$ into the compatibility

condition, we get:

$$\begin{aligned}
\alpha_{11} - b_{11}\theta_1^0 - b_{12}\theta_3^0 &= -6.7941 \\
\alpha_{12} - b_{11}\theta_2^0 - b_{12}\theta_4^0 &= -0.9095 \\
\alpha_{21} - b_{21}\theta_1^0 - b_{22}\theta_3^0 &= 1.4686 \\
\alpha_{22} - b_{21}\theta_2^0 - b_{22}\theta_4^0 &= -0.2497 \\
b_{11}\theta_5^0 + b_{12}\theta_7^0 &= -0.1461 \\
b_{11}\theta_6^0 + b_{12}\theta_8^0 &= 0.2073 \\
b_{21}\theta_5^0 + b_{22}\theta_7^0 &= -0.0021 \\
b_{21}\theta_6^0 + b_{22}\theta_8^0 &= -0.0281.
\end{aligned} \tag{4.13}$$

Clearly, there always exists a parameter value θ^0 satisfying the compatibility condition:

$$\begin{aligned}
\theta_1^0 &= \frac{b_{22}(a_{11} + 6.7941) - b_{12}(a_{21} - 1.4686)}{b_{11}b_{22} - b_{12}b_{21}} \\
\theta_2^0 &= \frac{b_{22}(a_{12} + 0.9095) - b_{12}(a_{22} + 0.2497)}{b_{11}b_{22} - b_{12}b_{21}} \\
\theta_3^0 &= \frac{-b_{21}(a_{11} + 6.7941) + b_{11}(a_{21} - 1.4686)}{b_{11}b_{22} - b_{12}b_{21}} \\
\theta_4^0 &= \frac{-b_{21}(a_{12} + 0.9095) + b_{11}(a_{22} + 0.2497)}{b_{11}b_{22} - b_{12}b_{21}} \\
\theta_5^0 &= \frac{0.0021b_{12} - 0.1461b_{22}}{-b_{21}b_{12} + b_{11}b_{22}} \\
\theta_6^0 &= \frac{0.0281b_{12} + 0.2073b_{22}}{-b_{21}b_{12} + b_{11}b_{22}}
\end{aligned}$$

$$\theta_7^0 = \frac{-0.0021b_{11} + 0.1461b_{21}}{-b_{21}b_{12} + b_{11}b_{22}}$$

$$\theta_8^0 = \frac{0.0281b_{11} + 0.2073b_{21}}{-b_{21}b_{12} + b_{11}b_{22}}. \quad (4.14)$$

4.4.5. Error Equation

The model following error is defined as

$$e(t) = x(t) - x_m(t)$$

$$\dot{e}(t) = Ax(t) + Bu(t) - A_m x_m(t) - B_m u_c(t). \quad (4.15)$$

Substitute the term of $u(t)$ into (4.15) and rearrange the equation as follows:

$$\dot{e}(t) = A_m e(t) + (A - A_m - BL)x(t) + (BM - B_m)u_c(t)$$

or,

$$\dot{e}(t) = A_m e(t) + (A_c(\theta_L) - A_m)x(t) + (B_c(\theta_M) - B_m)u_c(t)$$

or,

$$\dot{e}(t) = A_m e(t) + (A_c(\theta_L) - A_c(\theta_L^0))x(t) + (B_c(\theta_M) - B_c(\theta_M^0))u_c(t). \quad (4.16)$$

Finally, the equation above is equivalent to the following:

$$\dot{e}(t) = A_m e(t) + \psi(t)(\theta - \theta^0) \quad (4.17)$$

where,

$$\psi = \begin{bmatrix} -b_{11}x_1 & -b_{11}x_2 & -b_{12}x_1 & -b_{12}x_2 & b_{11}u_{c1} & b_{11}u_{c2} & b_{12}u_{c1} & b_{12}u_{c2} \\ -b_{21}x_1 & -b_{22}x_2 & -b_{22}x_1 & -b_{22}x_2 & b_{21}u_{c1} & b_{21}u_{c2} & b_{22}u_{c1} & b_{22}u_{c2} \end{bmatrix}$$

$$\theta - \theta^0 = [\theta_1 - \theta_1^0 \quad \theta_2 - \theta_2^0 \quad \dots \quad \theta_7 - \theta_7^0 \quad \theta_8 - \theta_8^0]^T.$$

4.4.6. Adaptation Law

To derive a parameter adjustment law, we introduce a Lyapunov candidate function

$$V(e, \theta) = \frac{1}{2} (\gamma e^T P e + (\theta - \theta^0)^T (\theta - \theta^0)) \quad (4.18)$$

where $e = [e_1 \ e_2]^T$ is a vector of state errors; γ is an adaptation rate; and $P = [1 \ 0; 0 \ 1]$ is chosen as a positive matrix.

The function V is positive definite. To find out whether it is a Lyapunov function, we calculate its total time derivative

$$\frac{dV}{dt} = -\frac{\gamma}{2} e^T Q e + \gamma (\theta - \theta^0)^T \psi^T P e + (\theta - \theta^0)^T \frac{d\theta}{dt} \quad (4.19)$$

or,

$$\frac{dV}{dt} = -\frac{\gamma}{2} e^T Q e + (\theta - \theta^0)^T \left(\frac{d\theta}{dt} + \gamma \psi^T P e \right) \quad (4.20)$$

where Q is positive definite and such that

$$A_m^T P + P A_m = -Q. \quad (4.21)$$

The matrix $Q = [13.5882 \ -0.5591; -0.5591 \ 0.4994]$ is a positive-definite matrix. Based on Lyapunov stability theory as introduced in (2.5), the function $V(e, \theta)$ above is a Lyapunov function. In another way, Nguyen and Afzulpurkar [26] proved the Lyapunov function candidate in (4.18) as a Lyapunov function by using another approach, MATLAB Symbolic Algebra.

As the result, the adaptation law is chosen as

$$\frac{d\theta(t)}{dt} = -\gamma\psi^T(t)Pe(t) \quad (4.22)$$

where,

$$\psi = \begin{bmatrix} -b_{11}x_1 & -b_{11}x_2 & -b_{12}x_1 & -b_{12}x_2 & b_{11}u_{c1} & b_{11}u_{c2} & b_{12}u_{c1} & b_{12}u_{c2} \\ -b_{21}x_1 & -b_{22}x_2 & -b_{22}x_1 & -b_{22}x_2 & b_{21}u_{c1} & b_{21}u_{c2} & b_{22}u_{c1} & b_{22}u_{c2} \end{bmatrix}$$

Therefore, the adaptation law can be expanded as follows:

$$\dot{\theta} = \begin{bmatrix} \gamma(b_{11}e_1 + b_{21}e_2)x_1 \\ \gamma(b_{11}e_1 + b_{21}e_2)x_2 \\ \gamma(b_{12}e_1 + b_{22}e_2)x_1 \\ \gamma(b_{12}e_1 + b_{22}e_2)x_2 \\ -\gamma(b_{11}e_1 + b_{21}e_2)u_{c1} \\ -\gamma(b_{11}e_1 + b_{21}e_2)u_{c2} \\ -\gamma(b_{12}e_1 + b_{22}e_2)u_{c1} \\ -\gamma(b_{12}e_1 + b_{22}e_2)u_{c2} \end{bmatrix} \quad (4.23)$$

4.4.7. Stability of the Analog Controller

We compute differentiation of the Lyapunov function in (4.18). Substitution of the $d\theta/dt$ term determined in (4.22), we get:

$$\frac{dV}{dt} = -\frac{\gamma}{2}e^T Qe$$

or,

$$\frac{dV}{dt} = -\frac{\gamma}{2}[(0.5591e_1 - 0.7066e_2)^2 + 13.2756e_1^2]. \quad (4.24)$$

The system is stable because the time derivative of the Lyapunov function is negative definite. As the result, the error goes to zero according to Lyapunov stability theory, which will be demonstrated with dynamic simulation in the next section.

4.5. Analog Controller Simulation

4.5.1. Simulation Program

As shown in Figure 4.4, the program consists of all components of the adaptive controller as follows:

- 1) Reference signals $u_c(t)$;
- 2) Disturbances $f(t)$;
- 3) Linear controller governed by the general linear control law;
- 4) Reference model;
- 5) Plant representing for the gasoline refinery with time-varying parameters;
- 6) Adaptive mechanism governed by the adaptation law in (4.23).

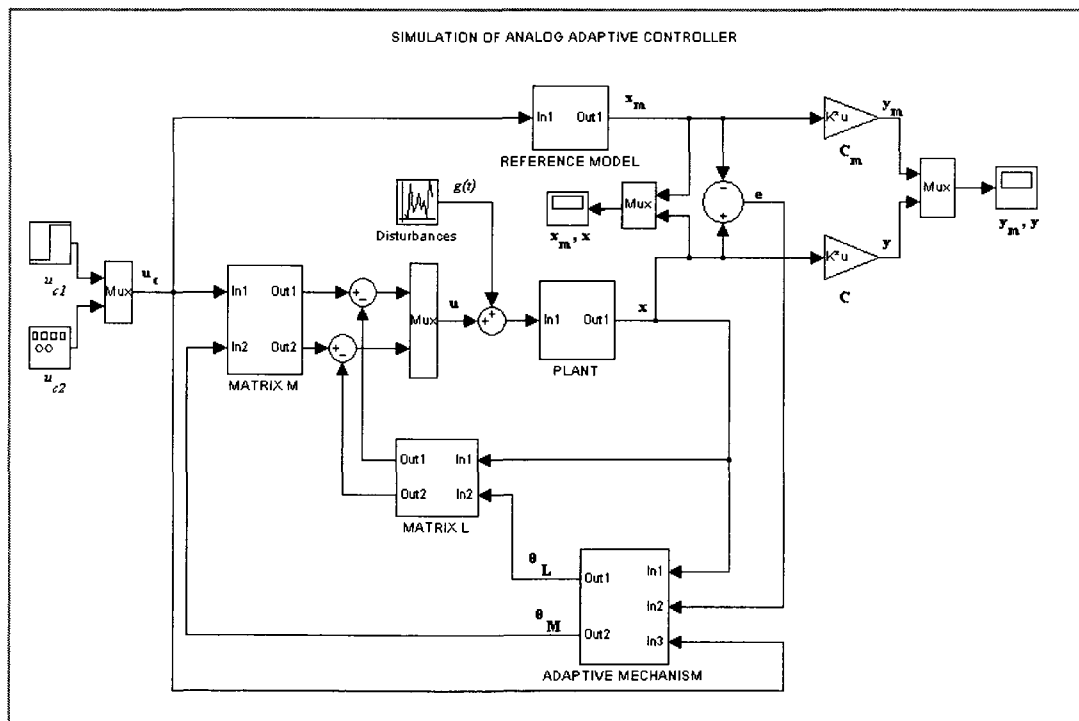


Figure 4.4. Simulation program for the analog adaptive controller.

To evaluate the controller performance, we compute mean error of the plant output. As earlier mentioned, the state error is defined as

$$e = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} x_1 - x_{m1} \\ x_2 - x_{m2} \end{bmatrix} = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix}.$$

We now define mean error \bar{e}_i based on root-mean-square of the vector e_i as follows:

$$\bar{e}_i = RMS(e_i) = \frac{\|e_i\|}{\sqrt{n_i}}, \text{ for } i = 1 \text{ or } 2$$

where n_i is the dimension of the vector e_i ; and $\|e_i\|$ is the norm of the vector e_i .

The norm of the vector e_i can be found by using the following MATLAB commands:

```
norm_e1 = norm(e1);
norm_e2 = norm(e2);
```

When the simulation is finished, all element values of the vector e_1 and e_2 are sent to MATLAB workspace so that we can easily calculate the mean errors.

4.5.2. Simulation Result

Adaptation rate is set at value 10. The plant parameters are simulated by random function. The reference inputs are step functions. External disturbances are simulated with random noise, as shown in Figure 4.5. The reference state and plant states are shown in Figure 4.6.

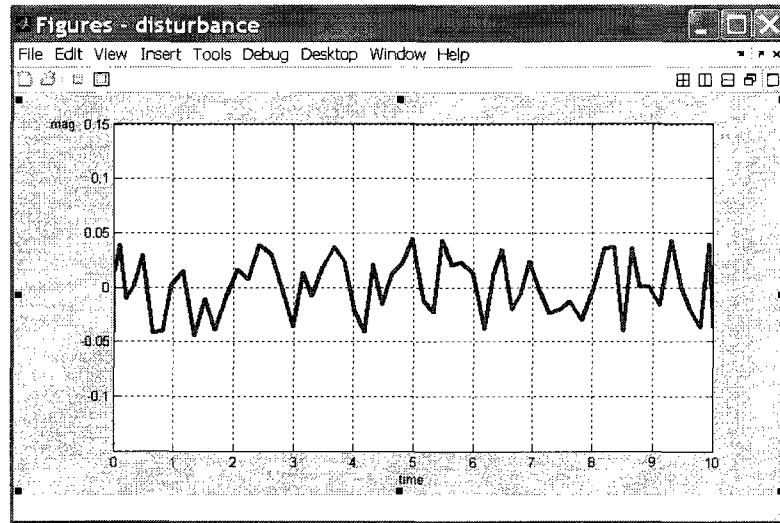


Figure 4.5. External disturbances.

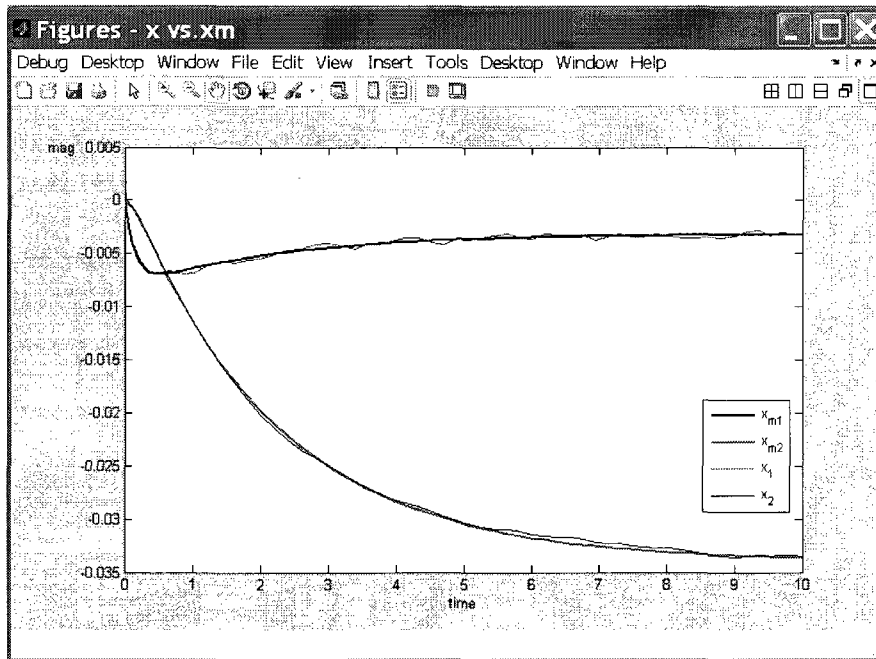


Figure 4.6. Reference states and plant states.

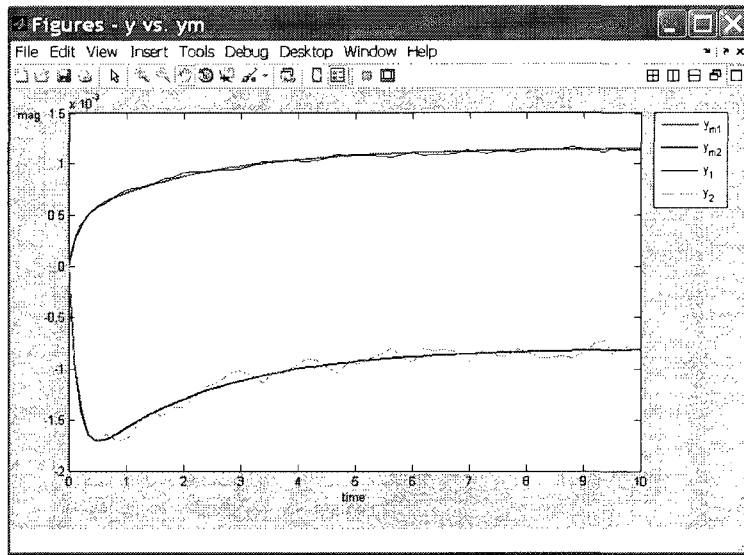


Figure 4.7. Controlled outputs and reference outputs.

The controlled outputs rapidly approach the desired values as shown in Figure 4.7. This is a clear illustration for the stability of the MRAC system as theoretically proved in Sec. 4.4.7. The state errors reduce when time increases as shown in Figure 4.8.

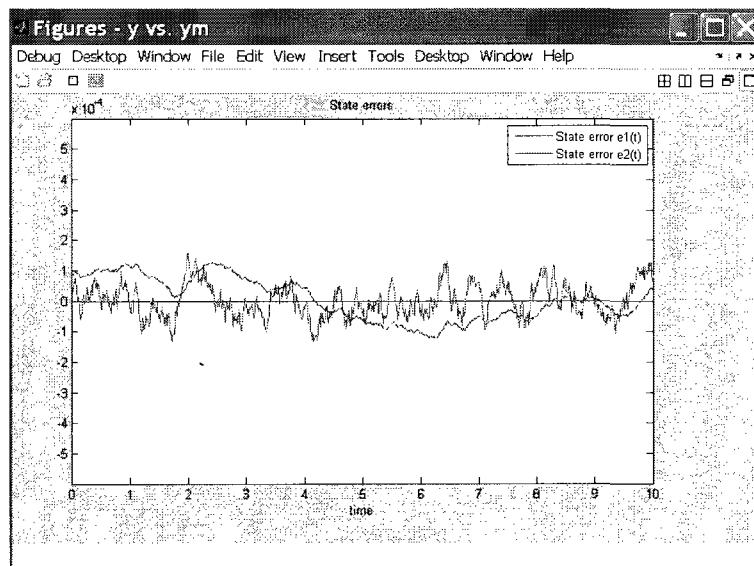


Figure 4.8. Plot of plant output errors during simulation.

The mean errors in the simulation duration can be determined as follows:

$$\bar{e}_1 = \frac{\|e_1\|}{\sqrt{n_1}} = \frac{0.0017}{\sqrt{1001}} = 5.4284 * 10^{-5}$$

$$\bar{e}_2 = \frac{\|e_2\|}{\sqrt{n_2}} = \frac{0.0022}{\sqrt{1001}} = 7.1062 * 10^{-5}.$$

Error reduction will be introduced in the next section.

4.5.3. Error Reduction

Adaptation rate γ has a strong effect on plant errors. In general, increasing adaptation rate will weaken plant output errors. In Table 4.1, the plant errors rapidly reduce to zero when the adaptation rate increases.

Table 4.1. Plant output errors for different adaptation rates.

Plant errors	Adaptation rate, γ		
	1	10	100
\bar{e}_1	$5.4304 * 10^{-5}$	$5.4284 * 10^{-5}$	$5.4076 * 10^{-5}$
\bar{e}_2	$7.1399 * 10^{-5}$	$7.1062 * 10^{-5}$	$7.1003 * 10^{-5}$

CHAPTER 5

DIGITAL CONTROLLER DESIGN

5.1. Introduction

In the previous chapter, the control system is designed and analyzed in continuous-time space. However the plant will be controlled by a digital computer. So we will discretize the plant. Zero-order hold (ZOH) is a mathematical model of the practical signal reconstruction accomplished by a conventional digital-to-analog converter (DAC). It describes the effect of converting a continuous-time signal into discrete-time signal by holding each sample value for one sample interval. The input and output signals of a zero-order hold is shown in Figure 5.1.

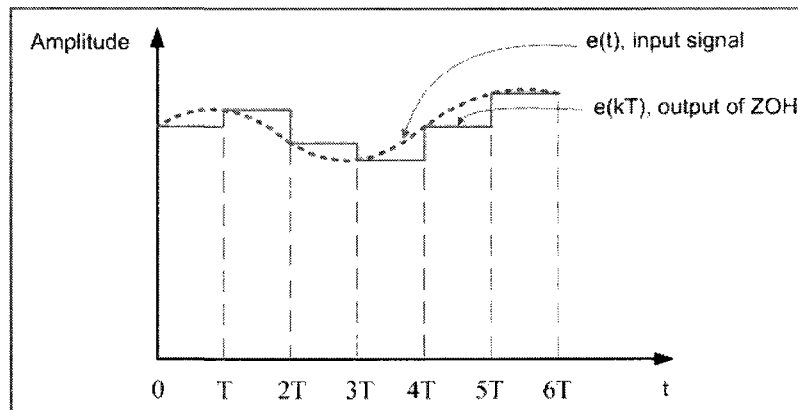


Figure 5.1. Input and output signals of ZOH.

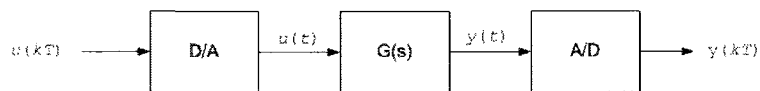


Figure 5.2. Block diagram of a sampled-data system.

In Figure 5.2, the zero-order-hold equivalent to $G(s)$ is given by

$$G(z) = \frac{z-1}{z} \mathcal{Z} \left\{ \frac{G(s)}{s} \right\}$$

where \mathcal{Z} is the z-transform of the bracketed term in time continuous space.

5.2. Digital Controller Synthesis

5.2.1. Plant Model

The plant in continuous-time space is given by

$$\dot{x}(t) = Ax(t) + Bu(t)$$

and

$$y(t) = Cx(t)$$

where $A = [a_{11} \ a_{12}; a_{21} \ a_{22}]$, $B = [b_{11} \ b_{12}; b_{21} \ b_{22}]$, and $C = [-0.0624 \ -0.0281; 0.2458 \ 0.0009]$. The elements of the matrices A and B above are unknown and dependent on the dynamics of the system.

The plant has the Laplace transform as

$$x(s) = \frac{1}{s} [Ax(s) + Bu(s)]$$

We assume that $x(t)$ and $u(t)$ are constant during the sampling interval T ; hence, the ZOH equivalence of the plant is given by

$$x(z) = \frac{T}{z-1} [Ax(z) + Bu(z)]$$

Now we can easily convert it to the new form of difference equations, which is conveniently implemented in a digital computer:

$$x(kT + T) = (I_n + TA)x(kT) + TBU(kT). \quad (5.1)$$

The controlled output $y(t)$ has ZOH equivalence as follow:

$$y(kT) = Cx(kT). \quad (5.2)$$

5.2.2. Reference Model

The reference model for the plant is a linear time-invariant system, as developed in Chapter 6:

$$\dot{x}_m(t) = A_m x_m(t) + B_m u_c(t)$$

and

$$y_m(t) = C_m x_m(t)$$

where $A_m = [-6.7941 \ -0.9095; \ 1.4686 \ -0.2497]$, $B_m = [-0.1461 \ 0.2073; \ -0.0021 \ -0.0281]$, and $C_m = [-0.0624 \ -0.0281; \ 0.2458 \ 0.0009]$.

Similarly to previous section, the reference model has the ZOH equivalence as follows:

$$x_m(kT + T) = (I_n + TA_m)x_m(kT) + TBu_c(kT). \quad (5.3)$$

The reference output $y_m(t)$ has ZOH equivalence as follow:

$$y_m(kT) = Cx_m(kT). \quad (5.4)$$

5.2.3. Linear Feedback Controller

A general linear control law is given by:

$$u(t) = Mu_c(t) - Lx(t).$$

The ZOH equivalence is determined as

$$u(kT) = Mu_c(kT) - Lx(kT). \quad (5.5)$$

All elements of the matrices L and M are adjusted by the adaptive mechanism:

$$L = \begin{bmatrix} \theta_1(kT) & \theta_2(kT) \\ \theta_3(kT) & \theta_4(kT) \end{bmatrix}$$

$$M = \begin{bmatrix} \theta_5(kT) & \theta_6(kT) \\ \theta_7(kT) & \theta_8(kT) \end{bmatrix}. \quad (5.6)$$

5.2.4. Compatibility Condition

Similarly to the design in continuous-time space, we need to determine compatibility value $\theta^0 = [\theta_1^0 \ \theta_2^0 \ \theta_3^0 \ \theta_4^0 \ \theta_5^0 \ \theta_6^0 \ \theta_7^0 \ \theta_8^0]^T$ as follows:

$$\theta_1^0 = \frac{b_{22}^0(a_{11}^0 + 6.7941) - b_{12}^0(a_{21}^0 - 1.4686)}{b_{11}^0 b_{22}^0 - b_{12}^0 b_{21}^0}$$

$$\theta_2^0 = \frac{b_{22}^0(a_{12}^0 + 0.9095) - b_{12}^0(a_{22}^0 + 0.2497)}{b_{11}^0 b_{22}^0 - b_{12}^0 b_{21}^0}$$

$$\theta_3^0 = \frac{-b_{21}^0(a_{11}^0 + 6.7941) + b_{11}^0(a_{21}^0 - 1.4686)}{b_{11}^0 b_{22}^0 - b_{12}^0 b_{21}^0}$$

$$\theta_4^0 = \frac{-b_{21}^0(a_{12}^0 + 0.9095) + b_{11}^0(a_{22}^0 + 0.2497)}{b_{11}^0 b_{22}^0 - b_{12}^0 b_{21}^0}$$

$$\theta_5^0 = \frac{0.0021b_{12}^0 - 0.1461b_{22}^0}{-b_{21}^0 b_{12}^0 + b_{11}^0 b_{22}^0}$$

$$\theta_6^0 = \frac{0.0281b_{12}^0 + 0.2073b_{22}^0}{-b_{21}^0 b_{12}^0 + b_{11}^0 b_{22}^0}$$

$$\theta_7^0 = \frac{-0.0021b_{11}^0 + 0.1461b_{21}^0}{-b_{21}^0 b_{12}^0 + b_{11}^0 b_{22}^0}$$

$$\theta_8^0 = \frac{0.0281b_{11}^0 + 0.2073b_{21}^0}{-b_{21}^0 b_{12}^0 + b_{11}^0 b_{22}^0}. \quad (5.7)$$

5.2.5. Adaptive Mechanism

The adaptive law obtained in continuous-time space is given by:

$$\dot{\theta}(t) = -\gamma \psi^T(t) P e(t)$$

where $\psi = \begin{bmatrix} -b_{11}x_1 & -b_{11}x_2 & -b_{12}x_1 & -b_{12}x_2 & b_{11}u_{c1} & b_{11}u_{c2} & b_{12}u_{c1} & b_{12}u_{c2} \\ -b_{21}x_1 & -b_{22}x_2 & -b_{22}x_1 & -b_{22}x_2 & b_{21}u_{c1} & b_{21}u_{c2} & b_{22}u_{c1} & b_{22}u_{c2} \end{bmatrix}$.

The Laplace transform is as follows:

$$\theta(s) = -\frac{\gamma \psi^T(s) P e(s)}{s}$$

Then we determine the ZOH equivalence in z-space:

$$\theta(z) = -\frac{T}{z-1} \gamma \psi^T(z) P e(z).$$

The discrete-time adaptation law is therefore obtained as

$$\theta(kT) = \theta(kT - T) + T[-\gamma \psi^T(kT - T) P e(kT - T)]. \quad (5.8)$$

The general adaptation law in (5.8) can be expanded as follows:

$$\begin{aligned} \theta_1(kT) &= \theta_1(kT - T) + T\gamma[b_{11}e_1(kT - T) + b_{21}e_2(kT - T)]x_1(kT - T) \\ \theta_2(kT) &= \theta_2(kT - T) + T\gamma[b_{11}e_1(kT - T) + b_{21}e_2(kT - T)]x_2(kT - T) \\ \theta_3(kT) &= \theta_3(kT - T) + T\gamma[b_{12}e_1(kT - T) + b_{22}e_2(kT - T)]x_1(kT - T) \\ \theta_4(kT) &= \theta_4(kT - T) + T\gamma[b_{12}e_1(kT - T) + b_{22}e_2(kT - T)]x_2(kT - T) \\ \theta_5(kT) &= \theta_5(kT - T) - T\gamma[b_{11}e_1(kT - T) + b_{21}e_2(kT - T)]u_{c1}(kT - T) \\ \theta_6(kT) &= \theta_6(kT - T) - T\gamma[b_{12}e_1(kT - T) + b_{22}e_2(kT - T)]u_{c2}(kT - T) \\ \theta_7(kT) &= \theta_7(kT - T) - T\gamma[b_{12}e_1(kT - T) + b_{22}e_2(kT - T)]u_{c1}(kT - T) \\ \theta_8(kT) &= \theta_8(kT - T) - T\gamma[b_{12}e_1(kT - T) + b_{22}e_2(kT - T)]u_{c2}(kT - T). \end{aligned} \quad (5.9)$$

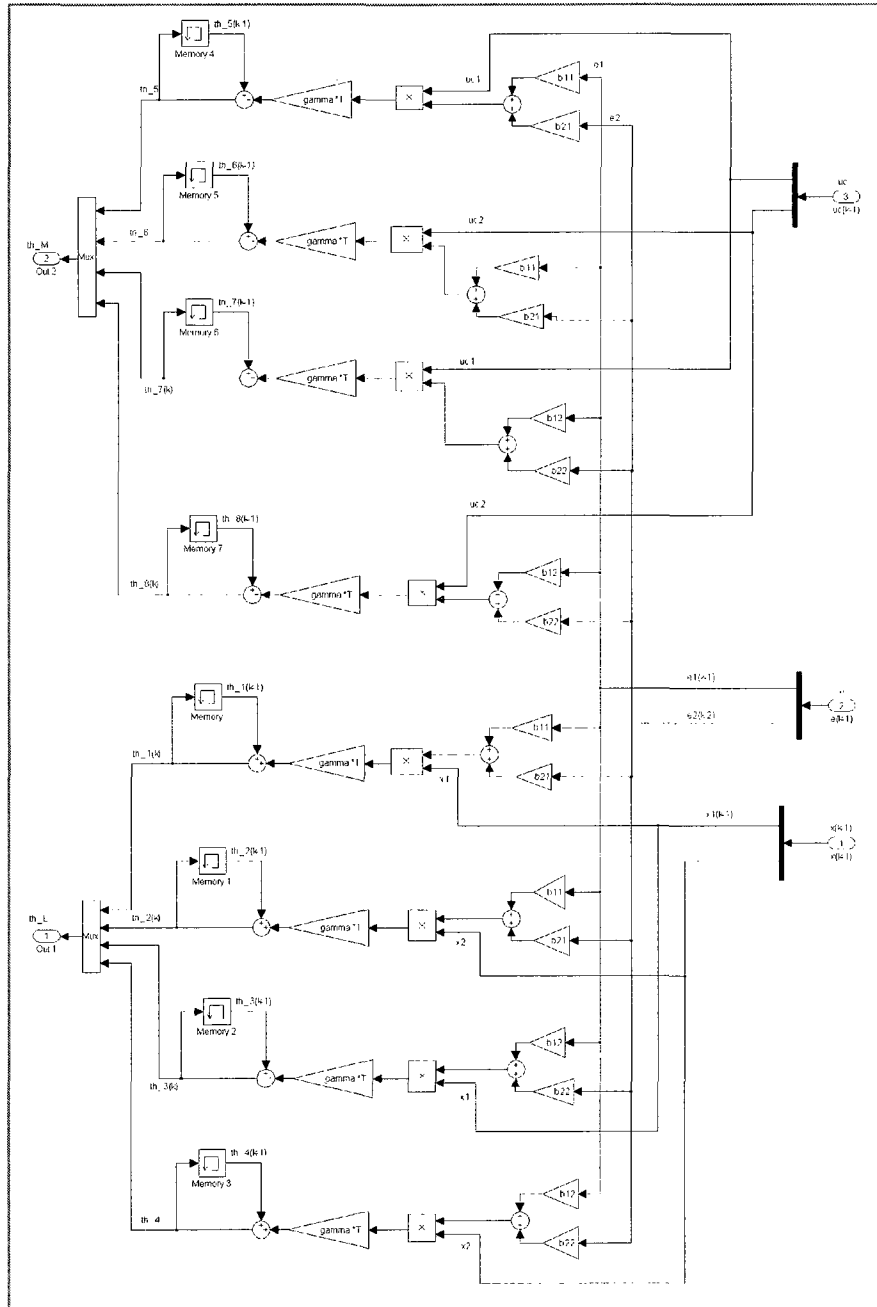


Figure 5.3. Logic diagram of the adaptation law.

The logic diagram of the adaptation law is shown in Figure 5.3. Base on the adaptation law, we are able to implement the embedded adaptive controller in hardware

using ARM processor, FPGA, or whatever type of digital computer. For the one using ARM development board, the adaption law will be translated into C language; and the adaptive mechanism will be an executable running in Linux environment.

CHAPTER 6

SIMULATION OF THE ADAPTIVE DIGITAL CONTROLLER

6.1. Introduction

The adaptive controller for the gasoline refinery is designed in Chapter 5. We now perform simulation of the adaptive system to verify its control performance. In the realm of control system design, MATLAB is normally used to perform simulation of control systems; hence, we also follow this approach. Moreover, we develop an alternative effective method for simulation of digital systems using C++ in which each subsystem of the system will be represented by a C++ class.

The simulation program written in C++ has many advantages. First, it is an executable so it can directly run in popular operating systems such as DOS/Windows or Linux. Second, its I/O file handling and data exchange with other software systems are quite simple since C++ has plentiful I/O library. Third, we can re-use the code to develop the plant simulator in the next chapter, which will interact with the embedded adaptive controller via an Ethernet.

6.2. Dynamic Simulation Using MATLAB

6.2.1. Simulation Program

The simulation program consists of major components of the adaptive system in discrete-time space, as shown in Figure 6.1:

- 1) Reference signals $u_c(kT)$;
- 2) Disturbances $f(kT)$;
- 3) Subsystem of Linear Controller governed by the general linear control law as

$$u(kT) = Mu_c(kT) - Lx(kT);$$

- 4) Subsystem of Reference Model representing the desired responses to reference signals

$$x_m(kT) = A_m x_m(kT) + B_m u_c(kT);$$

- 5) Subsystem of Plant representing the unknown process model as

$$x(kT) = Ax(kT) + Bu(kT);$$

- 6) Subsystem of Adaptive Mechanism governed by the adaptive law as

$$\theta(kT) = \theta(kT - T) + T[-\gamma\psi^T(kT - T)Pe(kT - T)].$$

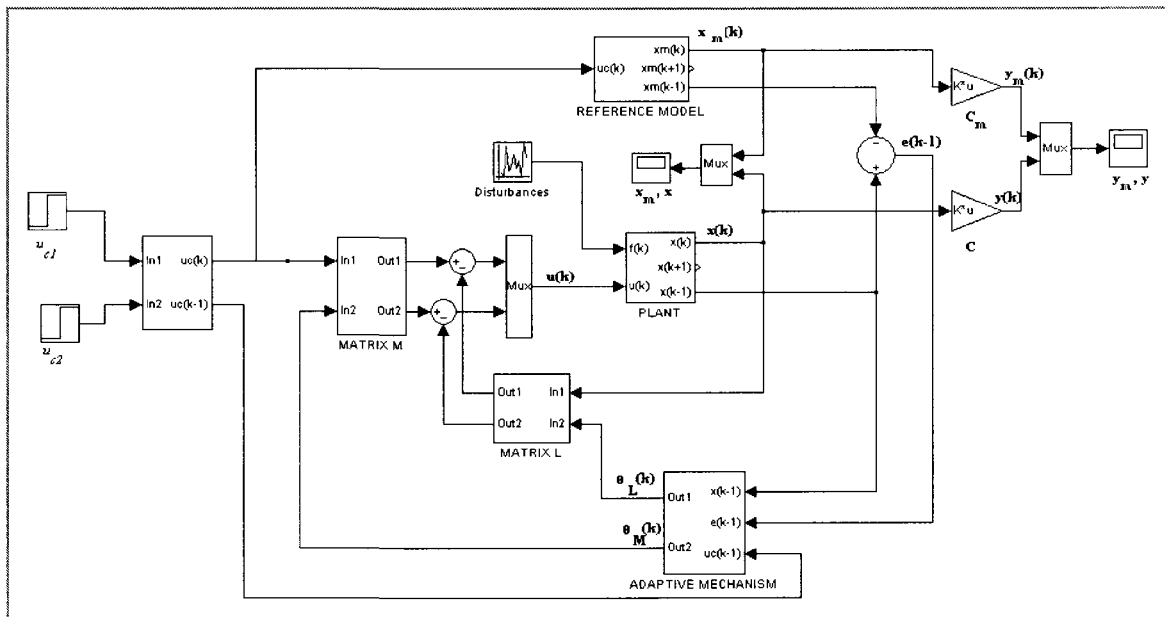


Figure 6.1. Simulation program for the digital adaptive system.

6.2.2. Error Calculation

As earlier mentioned, the plant output errors is defined as

$$e(kT) = \begin{bmatrix} e_1(kT) \\ e_2(kT) \end{bmatrix} = \begin{bmatrix} x_1(kT) - x_{m1}(kT) \\ x_2(kT) - x_{m2}(kT) \end{bmatrix} = \begin{bmatrix} \Delta x_1(kT) \\ \Delta x_2(kT) \end{bmatrix}.$$

We now define mean error \bar{e}_i based on root-mean-square of the vector e_i as follows:

$$\bar{e}_i = RMS(e_i) = \frac{\|e_i\|}{\sqrt{n_i}}, \quad \text{for } i = 1 \text{ or } 2 \quad (6.1)$$

where n_i is the dimension of vector e_i ; and $\|e_i\|$ is the norm of vector e_i .

6.2.3. MATLAB Simulation Result

The plant parameters are simulated by random functions. The reference inputs $u_c(kT)$ are step functions. External disturbances are simulated with random noise, as shown in Figure 6.2.

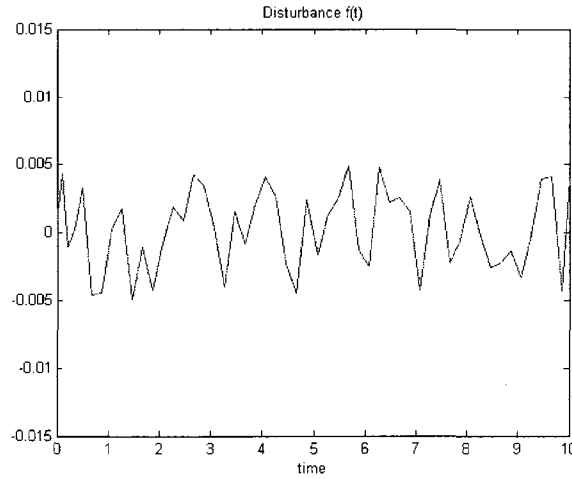


Figure 6.2. External disturbances.

As a result, the reference states (x_{m1}, x_{m2}) are well tracked by plant states (x_1, x_2) , as depicted in Figure 6.3.

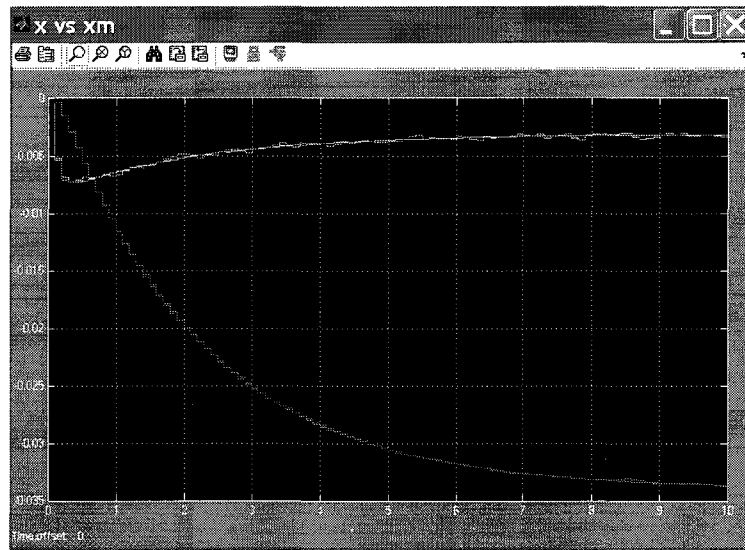


Figure 6.3. Reference states and plant states.

The controlled outputs rapidly reach the desired values, as shown in Figure 6.4. This is a clear illustration for the stability of the digital adaptive controller.

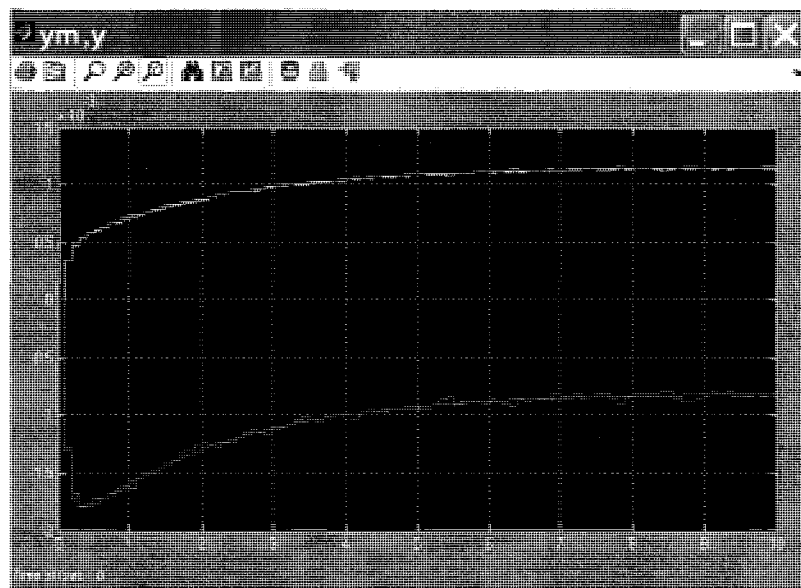
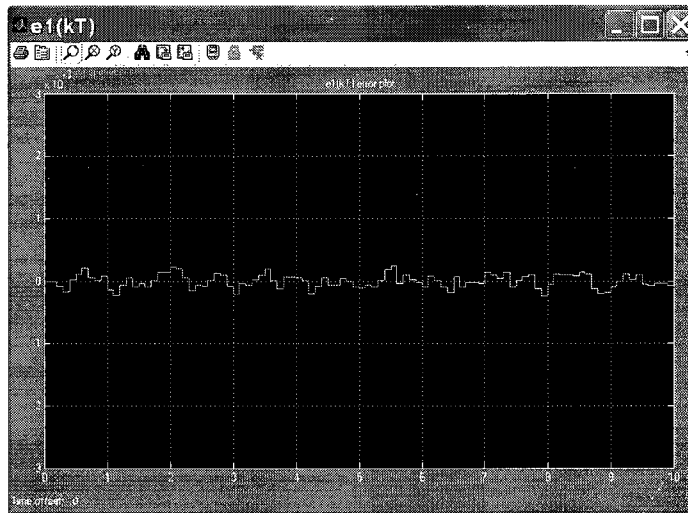
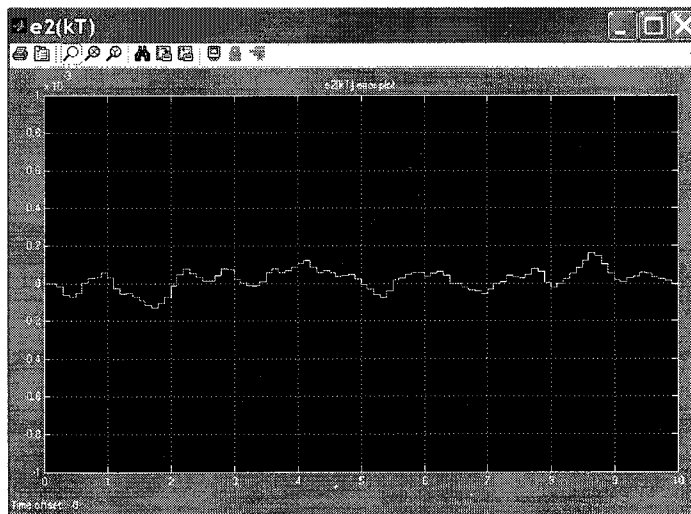


Figure 6.4. Controlled outputs and reference outputs.

The plant output errors approach zero; and the system has good control performance, as depicted in Figure 6.5.



a) $e_1(kT)$



a) $e_2(kT)$

Figure 6.5. Error plots.

The mean errors over the simulation duration are calculated by using the error equation (6.1) as follows:

$$\bar{e}_1 = \frac{\|e_1\|}{\sqrt{n_1}} = \frac{9.4543e-004}{\sqrt{101}} = 9.4074 * 10^{-5}$$

$$\bar{e}_2 = \frac{\|e_2\|}{\sqrt{n_2}} = \frac{9.6557e-004}{\sqrt{101}} = 9.6078 * 10^{-5}.$$

6.3. C++ Simulation Project

The functional verification project written in C++ is to simulate all subsystems including plant, reference model, linear controller, comparator, and adaptive mechanism. The algorithms and data structures of each C++ class are summarized in the following sections.

6.3.1. Plant Model Class

Plant model class is to simulate the plant. It receives signals from the linear controller and then generates plant state $x(kT)$. The header file of the plant class is shown in Figure 6.6.

```

1  #ifndef __PLANT_MODEL_H__
2  #define __PLANT_MODEL_H__
3  #include <stdlib.h>
4  #include "tbdefs.h"
5  class Plant{
6  private:
7  Pkt u_k; //control signal
8  Pkt x_k; //plant state
9  double a11, a12, a21, a22;
10 double b11, b12, b21, b22;
11 double c11, c12, c21, c22;

```

```

12 public:
13 Plant();
14 ~Plant();
15 void getPlantPkt(int k, Pkt u_k, Pkt x_k);
16 void genPlantPkt(int k, Pkt &x_kp1);
17 };
18 #endif

```

Figure 6.6. Plant class's header file.

The plant class has several important member functions:

```

Plant();

void getPlantPkt(int k, Pkt u_k, Pkt x_k);

void genPlantPkt(int k, Pkt &x_kp1);

```

The algorithms of these functions are described below.

- 1) The class constructor `Plant()` is to create a new object of plant whose parameters are randomized due to invariant dynamics of the plant:

```

a11 = -6.7941 + 0.67*(rand()%2-0.5);
a12 = -0.9095 + 0.09*(rand()%2-0.5);
a21 = 1.4686 + 0.15*(rand()%2-0.5);
a22 = -0.2497 + 0.02*(rand()%2-0.5);
b11 = -0.1461 + 0.014*(rand()%2-0.5);
b12 = 0.2073 + 0.02*(rand()%2-0.5);
b21 = -0.0021 + 0.0002*(rand()%2-0.5);
b22 = -0.0281 + 0.003*(rand()%2-0.5);
c11 = -0.0624 + 0.006*(rand()%2-0.5);
c12 = -0.0281 + 0.003*(rand()%2-0.5);

```



```

c21 = 0.2458 + 0.025*(rand()%2-0.5);
c22 = 0.0009 + 0.00009*(rand()%2-0.5);

```

- 2) The member function `getPlantPkt` is to receive input signals of the plant such as control signals from the linear controller.
- 3) The member function `genPlantPkt` is to generate plant states, which will be stored in the common database.

6.3.2. Reference Model Class

Reference model class is to simulate the reference model. It receives signals from the reference signal generator and then produces reference state $x_m(kT)$. The header file of the reference model class is shown in Figure 6.7.

```

1  #ifndef __REFERENCE_MODEL_H__
2  #define __REFERENCE_MODEL_H__
3  #include <stdlib.h>
4  #include "tbdefs.h"
5  class Refmdl{
6  private:
7   Pkt uc_k;
8   Pkt xm_k;
9  public:
10  Refmdl();
11  ~Refmdl();
12  void refmdlGetPkt(int k, Pkt uc_k, Pkt xm_k);
13  void genRefPkt(int k, Pkt &xm_kpl);
14  };
15 #endif

```

Figure 6.7. Reference model class's header file.

The plant class has two important member functions:

```
void refmdlGetPkt(int k, Pkt uc_k, Pkt xm_k);  
void genRefPkt(int k, Pkt &xm_kp1);
```

The algorithms of these functions are quite simple:

- 1) The function `refmdlGetPkt` is to receive reference signals from the reference signal generator.
- 2) The function `genRefPkt` is to generate reference states, which will be stored in the common database.

6.3.3. Linear Control Class

Linear control class is to simulate the linear controller. It receives signals from the reference signal generator and adaptive mechanism. It then produces control signals $u(kT)$ to manipulate the plant. The header file of the linear control class is shown in Figure 6.8.

```
1  #ifndef __LINEAR_CONTROL_H__  
2  #define __LINEAR_CONTROL_H__  
3  #include <stdlib.h>  
4  #include "tbdefs.h"  
5  class Linctrl{  
6  private:  
7  thetaPkt th_k;  
8  Pkt x_k;  
9  Pkt uc_k;  
10 public:  
11 Linctrl();  
12 ~Linctrl();
```

```

13 void getLinctrlPkt(int k, thetaPkt th_k, Pkt uc_k, Pkt x_k);
14 void genLinctrlPkt(int k, Pkt &u_k);
15 };
16 #endif

```

Figure 6.8. Linear control class's header file.

The linear control class has two important member functions:

```

void getLinctrlPkt(int k, thetaPkt th_k, Pkt uc_k, Pkt
x_k);
void genLinctrlPkt(int k, Pkt &u_k);

```

The algorithms of these functions are quite simple:

- 1) The function `getLinctrlPkt` is to receive signals from the adaptive mechanism, the reference signal generator, and the plant.
- 2) The function `genLinctrlPkt` is to generate control signals, which directly manipulate the plant. The control signals are also stored in the common database.

6.3.4. Comparator Class

Comparator class is to simulate the comparator. It receives signals from the reference model and the plant. It then compares them and computes the error $e(kT)$, which is an important element to synthesize adaptive gains. The header file of the comparator class is shown in Figure 6.9.

```

1 #ifndef __COMPARATOR_H__
2 #define __COMPARATOR_H__
3 #include <stdlib.h>
4 #include "tbdefs.h"

```

```

5 class Comparator{
6 private:
7   Pkt x_k;
8   Pkt xm_k;
9   Pkt e_k;
10 public:
11   Comparator();
12   ~Comparator();
13 void getCmprPkt(int k, Pkt x_k, Pkt xm_k);
14 void genCmprPkt(int k, Pkt &e_k);
15 };
16 #endif

```

Figure 6.9. Comparator class's header file.

The linear control class has two important member functions:

```

void getCmprPkt(int k, Pkt x_k, Pkt xm_k);
void genCmprPkt(int k, Pkt &e_k);

```

The algorithms of these functions are quite simple:

- 1) The function `getCmprPkt` is to get state variables of the reference model and the plant.
- 2) The function `genCmprPkt` is to compare the state variables above and generate error signals.

6.3.5. Adaptive Mechanism Class

Adaptive mechanism class is to simulate the adaptive mechanism. It receives signals from the signal generator, plant, and comparator. It then generates adaptive gains $\theta(kT)$. The header file of the adaptive mechanism is shown in Figure 6.10.

```

1  #ifndef __ADAPTIVE_MECH_H__
2  #define __ADAPTIVE_MECH_H__
3  #include <stdlib.h>
4  #include "tbdefs.h"
5  class AdaptiveMech{
6  private:
7   thetaPkt th_kml;
8   Pkt x_kml;
9   Pkt e_kml;
10  Pkt uc_kml;
11 public:
12  AdaptiveMech();
13  ~AdaptiveMech();
14   void getAdapMechPkt(int k, Pkt x_kml, Pkt e_kml, Pkt
        uc_kml, thetaPkt th_kml);
15  void genAdapMechPkt(int k, thetaPkt &th_k);
16 };
17 #endif

```

Figure 6.10. Adaptive mechanism class's header file.

The adaptive mechanism class has two important member functions:

```

void getAdapMechPkt(int k, Pkt x_kml, Pkt e_kml, Pkt
uc_kml, thetaPkt th_kml);

```

```

void genAdapMechPkt(int k, thetaPkt &th_k);

```

The algorithms of these functions are quite simple:

- 1) The function `getAdapMechPkt` is to get state variables of the reference model and the plant.
- 2) The function `genAdapMechPkt` is to compute all adaptive gains and store them into the common database.

6.3.6. Create Makefile and Build Project

We create the Makefile of the project with the following content:

```
CC = g++
all:
    $(CC) -g -c -o AdaptiveMech.o AdaptiveMech.cpp
    $(CC) -g -c -o Comparator.o Comparator.cpp
    $(CC) -g -c -o LinearControl.o LinearControl.cpp
    $(CC) -g -c -o PlantModel.o PlantModel.cpp
    $(CC) -g -c -o ReferenceModel.o ReferenceModel.cpp
    $(CC) -g -o adaptive_control AdaptiveMech.o
    Comparator.o LinearControl.o PlantModel.o
    ReferenceModel.o
```

In a console, we run the Makefile to compile and build the “adaptive_control” executive file.

6.3.7. C++ Simulation Result

We run the executable to perform simulation of the adaptive system. We find that it gives the same results as MATLAB simulation in Sec. 6.2. Thus we can deploy either the simulation program written in C++ or the simulation written in MATLAB for testing the embedded adaptive controller in the next chapter.

The most significant results are shown in the following figures. Refer to Appendix G for more simulation data.

```

hoan@hoan:~/thesis/adapcon$
File Edit View Terminal Tabs Help
th2[9] = -0.510735
th3[9] = 0.280902
th4[9] = -0.309865
th5[9] = 1.028989
th6[9] = -0.045742
th7[9] = 0.017589
th8[9] = 1.010035
Plant model has been created with random parameters:
A = [-0.459100 -0.064500 1.543600 -0.239700]
B = [-0.139100 0.217300 -0.002200 -0.029600]
C = [-0.065400 -0.029600 0.233300 0.000855]
Adapmech receives signals...
Adapmech generates adaptive gains:
th1[10] = 0.371379
th2[10] = -0.510663
th3[10] = 0.280843
th4[10] = -0.309971
th5[10] = 1.032299
th6[10] = -0.045088
th7[10] = 0.012710
th8[10] = 1.009059
Simulation is done.
hoan@hoan:~/thesis/adapcon$

```

Figure 6.11. Running the “adaptive_control” executive file.

The reference state x_m and plant states x are shown in Figure 6.12.

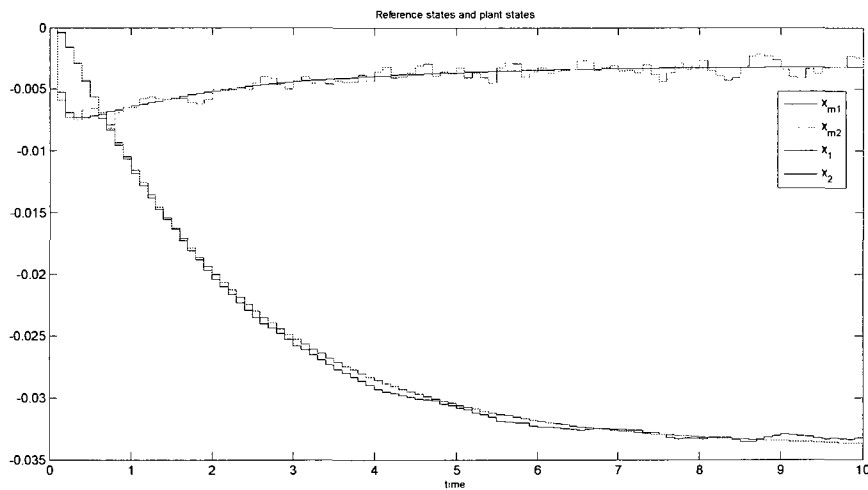


Figure 6.12. State variables.

The controlled outputs rapidly reach the desired values as shown in Figure 6.13.

This is a definite illustration for the stability of the digital control system.

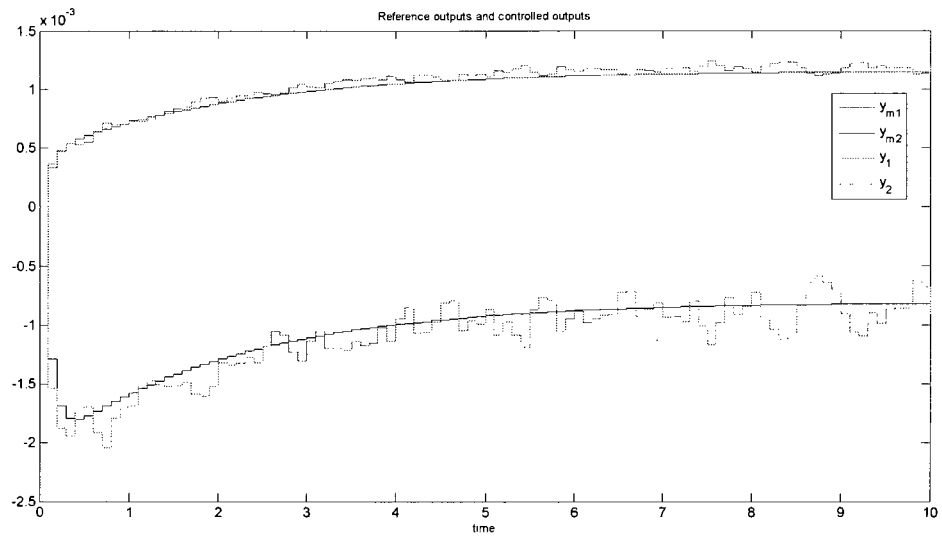


Figure 6.13. Controlled outputs and reference outputs.

The plant output errors rapidly approach zero as shown in Figure 6.14.

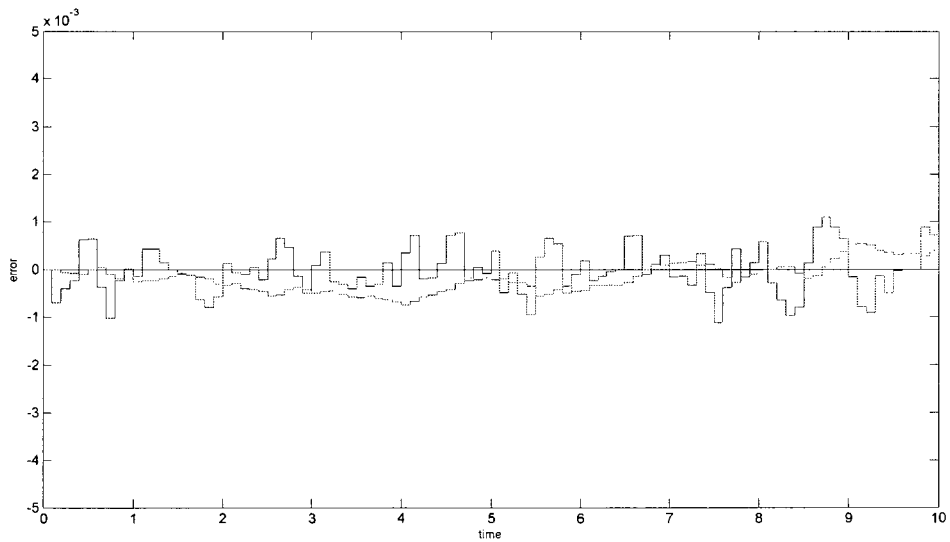


Figure 6.14. Plant output errors.

The mean errors \bar{e} in the simulation duration can be determined as follows:

$$\bar{e}_1 = \frac{\|e_1\|}{\sqrt{n_1}} = \frac{9.4543e-004}{\sqrt{101}} = 9.4074*10^{-5}$$

$$\bar{e}_2 = \frac{\|e_2\|}{\sqrt{n_2}} = \frac{9.6557e-004}{\sqrt{101}} = 9.6078*10^{-5}.$$

The mean errors determined by the C++ simulation program is the same as the ones calculated by the MATLAB Simulink program in Sec. 6.2.

CHAPTER 7

IMPLEMENTATION OF THE EMBEDDED ADAPTIVE CONTROLLER

7.1. Introduction

7.1.1. Embedded Adaptive Controller Using ARM Processor

In previous chapters, we successfully design and simulate the adaptive controller for the gasoline refinery. We now implement it using the ARM-7 processor. We firstly develop a plant simulator, which runs on a Linux machine to simulate the plant and other local instruments such as the conventional feedback control loop. The embedded adaptive controller will be implemented in an ARM-7 development board, which governs the adaptation law and remotely controls the plant via an Ethernet network. The elementary block diagram of the plant simulator and embedded adaptive controller is shown in Figure 7.1.

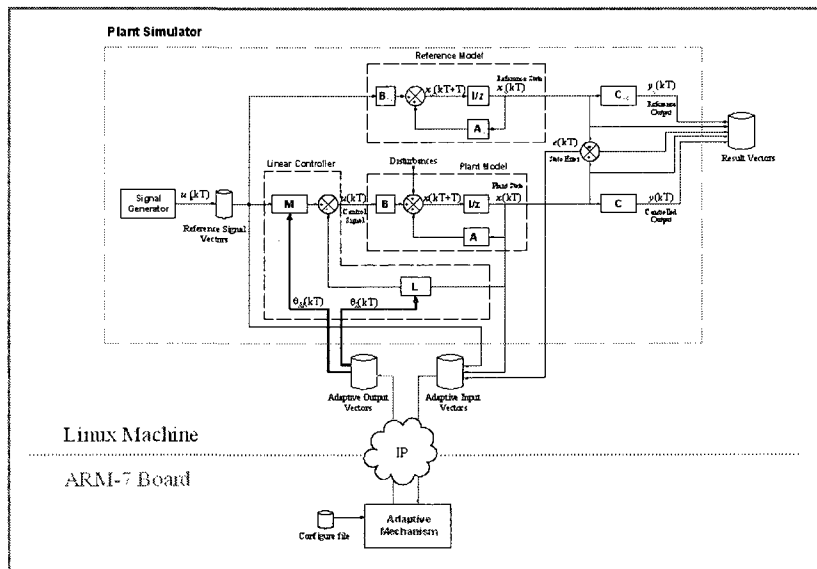


Figure 7.1. Elementary block diagram of the system.

7.1.2. In-Hardware Validation Scheme

In Chapter 6, we carried out simulation of the adaptive system in both MATLAB and C++ in which the adaptive mechanism was represented by either a MATLAB Simulink module or a C++ class. The simulation results showed that the design fully obtained the control objectives under variant environment such as time-varying process dynamics of the plant and unpredictable disturbances. Therefore, we can consider these software models as golden models, which execute the functions of the adaptive mechanism and generate accurate output data to compare against the actual results from the hardware implementation [28].

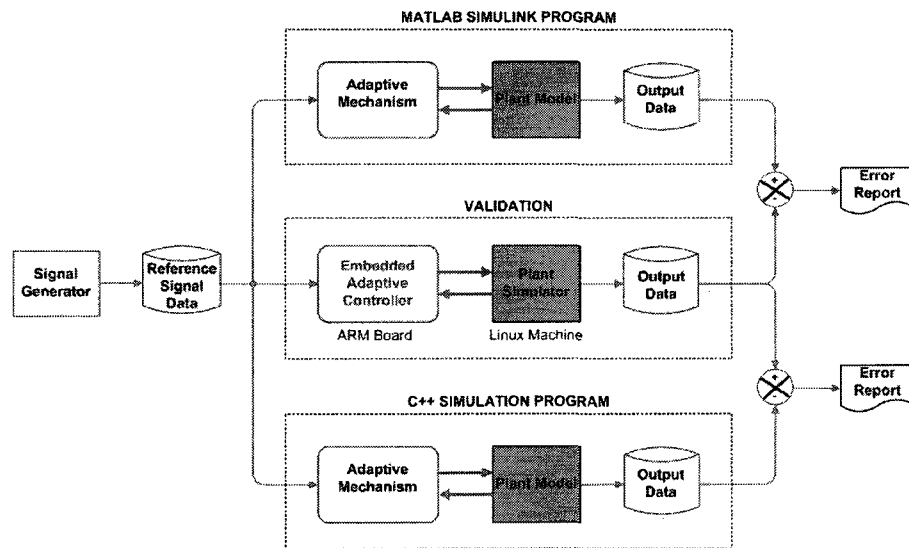


Figure 7.2. In-hardware validation scheme.

In Figure 7.2, we use the same data of reference signal to feed the MATLAB Simulink program, C++ simulation program, and the validation blocks. Their output data are collected and sent to the result comparator to prepare error reports.

7.1.3. System Architecture and Operations

The integrated testing environment consists of the plant simulator, the embedded adaptive controller, a Boa web server with a built-in CGI program, an NFS server/ client, web clients, and engineering stations. These machines are connected by a LAN network as shown in Figure 7.3.

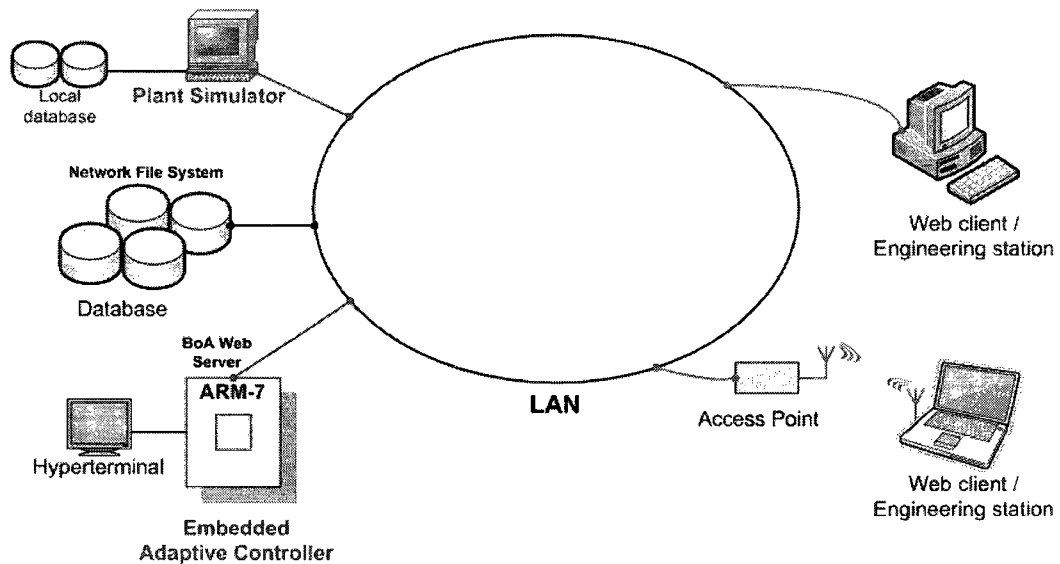


Figure 7.3. Integrated testing environment.

The ARM board is an NFS client; and a Linux machine plays the role of the NFS server. The ARM board will mount and open the shared database on the NFS server. The embedded adaptive controller resides on the ARM board whereas the plant simulator runs on another Linux machine. Moreover, there is a type of computer called web client/ engineering station, which is either a personal computer (PC) or Linux computer. A user sitting on this station can use a web browser to open the main graphical user interface

(GUI) page of the Boa web server to begin testing the adaptive system. He or she can also access the shared database on the NFS server.

By default, the ARM board has an IP address 192.168.0.128 and subnet mask 255.255.255.0. We set the plant simulator machine an IP address 192.168.0.10/255.255.255.0 and the NFS server an IP address 192.168.0.8.

After the kernel image is loaded into the ARM board, we perform the command of NFS mount from the HyperTerminal. In Figure 7.4, the interaction between the NFS client and the NFS server for mounting a network file system is described as follows:

- The NFS client initiates mounting the network file system;
- The NFS client makes RPC “get_port” request;
- The NFS server performs RPC “get_port” reply;
- The NFS client requests the server to mount the file system;
- The NFS server performs local mount for the requested files [29].

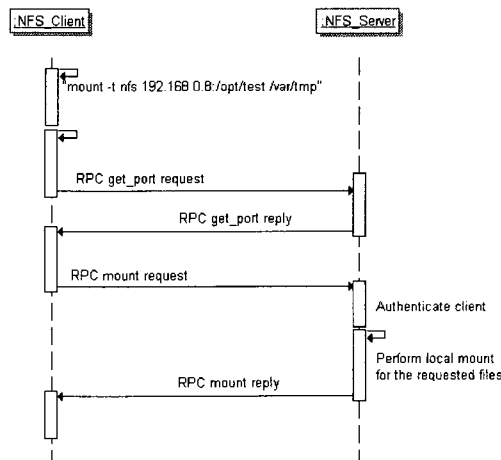


Figure 7.4. NFS client/server interaction for mounting a network file system.

After the network file system is successful mounted, the ARM board (i.e., the NFS client) can access the shared database on the remote NFS server.

The interactions among the web browser, the Boa web server, the CGI program, the embedded adaptive controller, and the plant simulator are described as follows:

- 1) A user opens the web browser to request the form by entering the URL <http://192.168.0.128/embedded_adaptive_controller.html>.
- 2) The browser makes a connection to the Boa server by the following steps:
 - Break the URL into 3 parts including the protocol (http), the IP address (192.168.0.128), and the file name (embedded_adaptive_controller.html);
 - Form a connection to the IP address on the port 80;
 - Send a “get” request to the Boa server.
- 3) The web server responds the request by sending HTML text for the web page to the browser.
- 4) The browser reads HTML tags and displays the input form onto the screen.
- 5) The user enters testing parameters and submits them by clicking on “Run” button.
- 6) The browser allows the user to enter testing parameters and submits information to the Boa web server.
- 7) The Boa web server forwards the information and activates the CGI program.
- 8) The CGI program makes a function call for the adaptive mechanism.
- 9) The plant simulator running on a Linux machine interacts with the embedded adaptive controller and forms the closed control loop.

- 10) At every step, the CGI program sends the status to the Boa server.
- 11) The Boa server then sends the status in HTML text to the browser.
- 12) The browser reads HTML tags and formats the page onto the screen.

As shown in Figure 7.5, the sequence diagram describes interactions among the web browser, the Boa web server, the CGI program, the embedded adaptive controller, and the plant simulator.

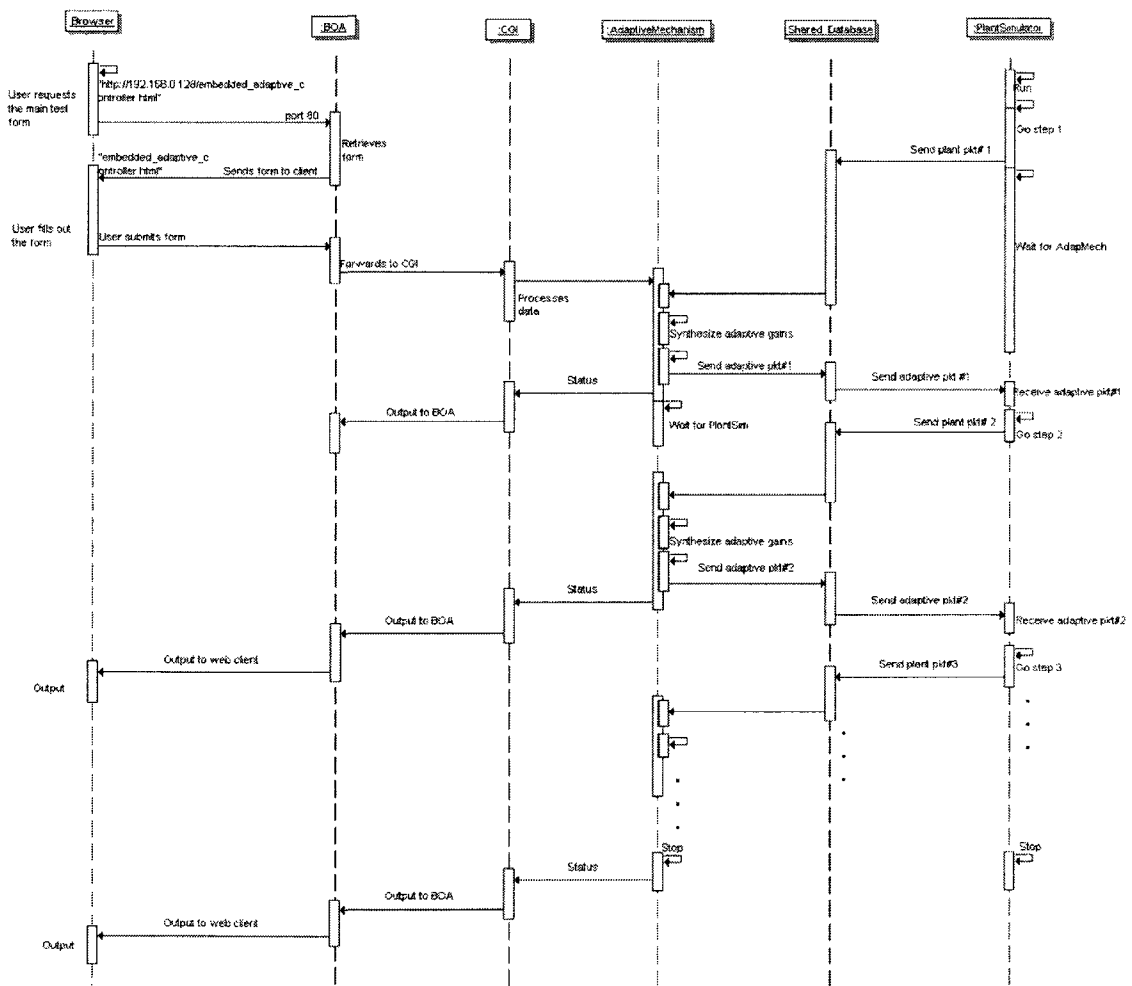


Figure 7.5. Sequence diagram.

7.1.4. Kernel Image of the ARM Board

The uClinux kernel is developed for the ARM board using Cygwin and Armtools software [30]. The distribution CD provides installation files for Cygwin and Armtools as well as various helpful documents such as datasheet and reference manual [31]. We write a C program that plays the role of the adaptive mechanism and combine this program with other predefined source files to build the uClinux kernel image. The kernel image will consist of the following elements: 1) the adaptive mechanism program; 2) the CGI program; 3) the Boa web server; and 4) the NFS client.

The CGI protocol includes a standard for interfacing applications with information servers such as web servers. The executable in a CGI can be any type of executable that handles standard input and output [32]. CGI programs are the most common server-side method for performing an executable that go beyond HTML. In this implementation, the adaptive mechanism and the CGI program are written in C language.

The Boa web servers is enabled from uClinux distribution through 3 steps: 1) customize kernel settings, customize vendor/user settings, and update default vendor settings; 2) select networking options and select network device support; and 3) establish Ethernet connection between the ARM board and other computers such as the plant simulator and engineering stations throughout a LAN network.

The Boa web server is enabled for the ARM board to allow a user to use a web browser in a client machine or an engineering station to point to the Boa web server and start the adaptive controller testing form. The user then set testing mode to send the information in the form to the CGI program. It will make a function call for the adaptive

mechanism. At every simulation step, execution status is posted back to the Boa server and finally received by the user's browser.

7.2. Common Database

The shared database is located in the NFS server. The database can be accessed by any computer in the LAN. The database consists of a number of structured text files as listed in Table 7.1.

Table 7.1. Structured data files.

No.	File name	Data description
1	kx.dat	Time stamp of plant state
2	x1.dat	Plant state x_1
3	x2.dat	Plant state x_2
4	ke.dat	Time stamp of error variables
5	e1.dat	Error variable e_1
6	e2.dat	Error variable e_2
7	kuc.dat	Time stamp of reference signal
8	uc1.dat	Reference signal u_{c1}
9	uc2.dat	Reference signal u_{c2}
10	kth.dat	Time stamp of adaptive gains
11	th1.dat	Adaptive gain θ_1
12	th2.dat	Adaptive gain θ_2
13	th3.dat	Adaptive gain θ_3
14	th4.dat	Adaptive gain θ_4
15	th5.dat	Adaptive gain θ_5
16	th6.dat	Adaptive gain θ_6
17	th7.dat	Adaptive gain θ_7

18	th8.dat	Adaptive gain θ_8
19	adapout.dat	Output data of embedded adaptive controller
20	kmax.dat	Maximal step size

7.3. Plant Simulator

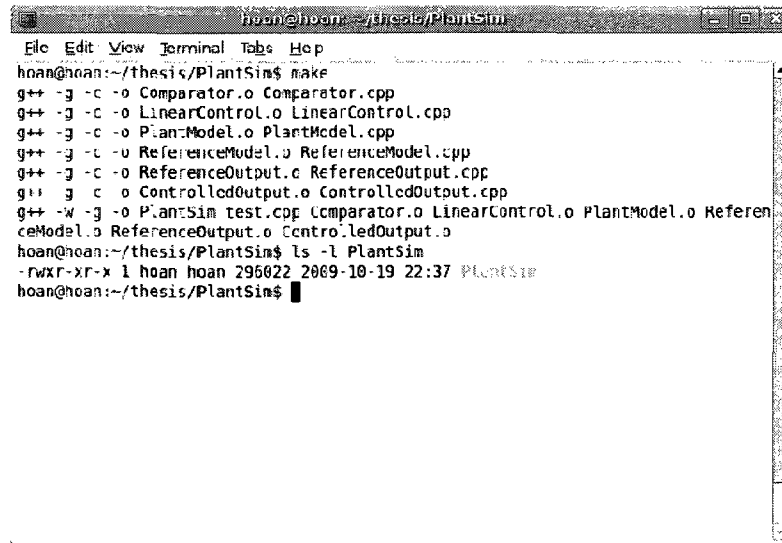
We build a C++ project, so called “plant simulator,” to simulate the plant and other auxiliaries including feedback control loop, reference model, and comparator. Each subsystem is represented by a C++ class, which has been well developed in Chapter 6.

Hence, we just create the “Makefile” of the new project as follows:

```
CC = g++
all:
    $(CC) -g -c -o Comparator.o Comparator.cpp
    $(CC) -g -c -o LinearControl.o LinearControl.cpp
    $(CC) -g -c -o PlantModel.o PlantModel.cpp
    $(CC) -g -c -o ReferenceModel.o ReferenceModel.cpp
    $(CC) -g -o PlantSim AdaptiveMech.o Comparator.o
        LinearControl.o PlantModel.o ReferenceModel.o
```

In a console terminal, we run the “Makefile” to compile and build the PlantSim executive file, as shown in Figure 7.6:

```
$ make
$ ls -l PlantSim
```



```
hoan@hoan:~/thesis/PlantSim
File Edit View Terminal Tabs Help
hoan@hoan:~/thesis/PlantSim$ make
g++ -g -c -o Comparator.o Comparator.cpp
g++ -g -c -o LinearControl.o LinearControl.cpp
g++ -g -c -o PlantModel.o PlantModel.cpp
g++ -g -c -o ReferenceModel.o ReferenceModel.cpp
g++ -g -c -o ReferenceOutput.o ReferenceOutput.cpp
g++ -g -c -o ControlledOutput.o ControlledOutput.cpp
g++ -w -g -o PlantSim test.cpp Comparator.o LinearControl.o PlantModel.o ReferenceModel.o ReferenceOutput.o ControlledOutput.o
hoan@hoan:~/thesis/PlantSim$ ls -l PlantSim
-rwxr-xr-x 1 hoan hoan 295022 2009-10-19 22:37 PlantSim
hoan@hoan:~/thesis/PlantSim$
```

Figure 7.6. Compile and build the plant simulator in a console terminal.

7.4. Development of the Main Testing Form in HTML

We write the testing form in HTML to allow user to enter testing parameters. The input form is quite simple and described as follows:

```
1 <html>
2 <head>
3 <title> UDP streaming </title>
4 </head>
5 <body>
6 <div align="center">
7 <br>
8 <br>
9 </div>
10 <h1>Embedded Adaptive Controller</h1>
11 <form action="/cgi-bin/mycgi" method="get" target="dest">
12 <h3>Please enter simulation parameters.</h3>
13 <p>Sampling time (T) : <input type="text"
    name="sptime" value="" size=40>
```

```

14 <p>Adaptation rate (gamma): <input type="text" name="gamma"
    value="" size=40>
15 <p><input type="submit" value = "Run"><input type="reset">
16 <input type=hidden name=cmd value="run">
17 </form>
18 </body>
19 </html>

```

Figure 7.7. Main testing form in HTML.

The beginning section is normal HTML similarly to the start of any HTML web page. The next section starting with the line `<form action="/cgi-bin/mycgi" method="get" target="dest">` is the actual form [30]. Some key features of HTML form needed for CGI interface are listed in Table 7.2.

Table 7.2. Some key features of an HTML form.

Feature	Code example	Description
action="..."	<code><form action="/cgi-bin/mycgi" method="get" target="dest"></code>	Define the URL of the CGI program
method="..."	<code><form action="/cgi-bin/mycgi" method="get" target="dest"></code>	Define how the information is passed to the server
target="..."	<code><form action="/cgi-bin/mycgi" method="get" target="dest"></code>	Define the target frame to load the destination page
type=...	<code><input type=text name=sptime></code>	Input tag to enter sampling time and text type
	<code><input type = submit value = "Run" ></code>	Collect and post data

7.5. Adaptive Mechanism Program

The adaptive mechanism program, the core of the embedded adaptive controller, will be run in the ARM-7 development board. Similarly to the C++ simulation project in Chapter 6, the adaptive mechanism program written in C has two basic functions as follows:

```
static void getAdapMechPkt(int k, struct Pkt *xk-1, struct
Pkt *ek-1, struct Pkt *uck-1, struct thetaPkt *thk-1);
void genAdapMechPkt(int k, struct Pkt* xk-1, struct Pkt* ek-1,
struct Pkt* uck-1, struct thetaPkt* thk-1, struct thetaPkt
*thk);
```

The first function, `getAdapMechPkt`, is to receive signals from the common database and save them to local variables. The second one, `genAdapMechPkt`, is to synthesize adaptive gains and sent them to the plant simulator via the common database. The pseudo code of the adaptive mechanism program is shown in Figure 7.8.

```
1 static void getAdapMechPkt(int k, struct Pkt *x[k-1],
  struct Pkt *e[k-1], struct Pkt *uc[k-1], struct thetaPkt
  *th[k-1]){
2 //Receive signals from database
3 getsig("/var/tmp/fkx.dat", &kx);
4 getsig("/var/tmp/fx1.dat", &x1);
5 getsig("/var/tmp/fx2.dat", &x2);
6 getsig("/var/tmp/fke.dat", &ke);
7 getsig("/var/tmp/fel.dat", &e1);
8 getsig("/var/tmp/fe2.dat", &e2);
9 //...
10 getsig("/var/tmp/fth8.dat", &th8);
```

```

11 return;
12 }

13 void genAdapMechPkt(int k, struct Pkt* x[k-1], struct Pkt*
    e[k-1], struct Pkt* uc[k-1], struct thetaPkt* th[k-1],
    struct thetaPkt *th[k]){
14 //compute adaptive gains
15 th->k = k;
16 th1[k]=th1[k-1]+Tgamma*(b11*e1[k-1]+b21*e2[k-1])*x1[k-
    1]/10000000;
17 th2[k]=th2[k-1]+Tgamma*(b11*e1[k-1]+b21*e2[k-1])*x2[k-
    1]/10000000;
18 //...
19 th8[k]=th8[k-1]-Tgamma*(b12*e1[k-1]+b22*e2[k-1])*uc2[k-
    1]/10000000;
20 //write to database
21 fwritethpkt("/var/tmp/adapout.dat", th_k);
22 settime("/var/tmp/fka.dat", k);
23 return;
24 }

```

Figure 7.8. Pseudo code of the adaptive mechanism program.

7.5.1. CGI Program

As described in the introduction section, CGI programs are the most common server-side method for interfacing applications with web servers. Parameter passing will be done in both ways of communication: 1) from client (browser) to CGI; and 2) from CGI to the browser [30]. There are three steps of parameter passing as described below.

First, the CGI program communicates with the browser via web server:

```
printf("Content-type: text/html\n\n");
```

```
printf("<html> <head> <title> Embedded Adaptive Mech  
</title> </head>\n");
```

Second, the CGI program captures the value of the command from the browser:

```
unsigned char *str1,*str2, *cmd;  
cmd = getval((unsigned char *)"cmd");
```

We note that the HTML file on the web server uses “hidden” attribute to pass variables:

```
<input type=hidden name=cmd value="run">
```

The CGI program captures parameters passed by HTML page through “getenv(...)” as follows:

```
vstr = (unsigned char*) getenv("REQUEST_METHOD");  
if(vstr==NULL) return 0;  
if(strcmp((const char*)vstr,"POST") == 0)
```

Third, the CGI program captures parameters. It finds the length of the parameters:

```
vstr = (unsigned char*) getenv("CONTENT_LENGTH");  
if (vstr==NULL || strlen((const char*)vstr)==0) return 0;
```

It then captures the parameters from the browser using fgets(...) function in the C stdio library as shown below:

```
if(vstr==NULL) return 0;  
fgets((char*)vstr,cl+1,stdin);
```

The flow chart of CGI program is shown in Figure 7.9.

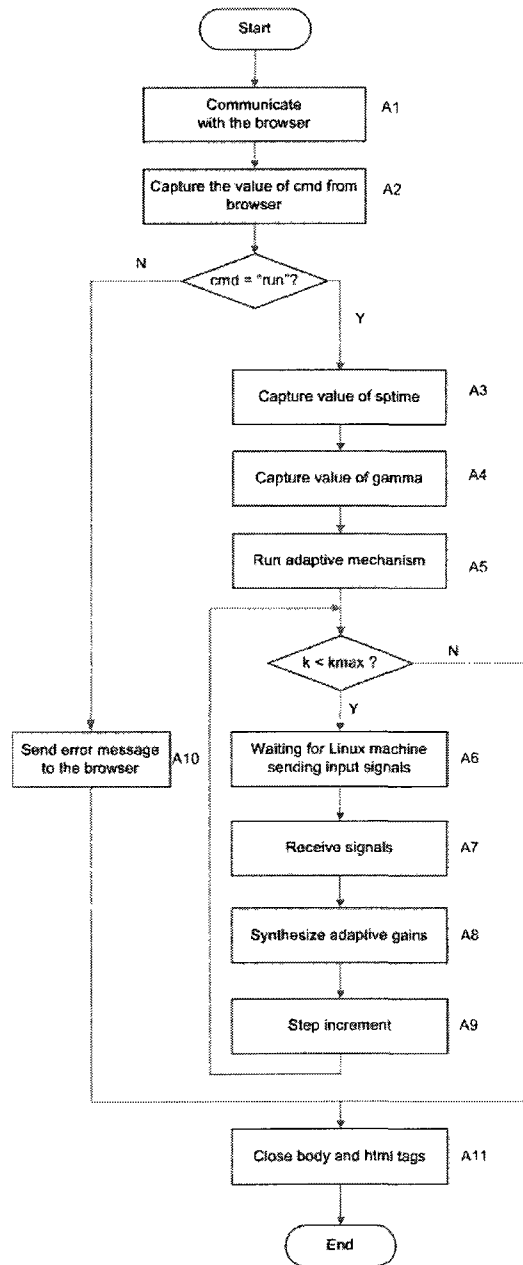


Figure 7.9. Flow chart of the CGI program.

The C pseudo code of the CGI program is shown in Figure 7.10.

```

1 main ()
2 {
3 int valid;

```



```

4 unsigned char *username,*password, *cmd;
5 //A1: Communicate with the browser
6 printf("Content-type: text/html\n\n");
7 printf("<html> <head> <title> Embedded Adaptive Mech
  </title> </head>\n");
8 //A2: Capture the value of cmd from browser
9 cmd = getval((unsigned char *)"cmd");
10 if(strcmp((const char *)cmd,"run")==0)
11 {
12     //A3: Capture value of sampling time
13     sptime=getval((unsigned char*)"sptime")/60;
14     //A4: Capture value of gamma
15     gamma=getval((unsigned char*)"gamma");
16     //A5: Run adaptive mech
17     while(k< kmax){
18         //A6: Waiting for Linux machine sending input signals
19         while(!getkt){
20             getkt = getcsig("/var/tmp/fkt.dat", &kT);
21             wait(5000);
22         }
23         //Check time stamp
24         while(k!=kT){
25             getsig("/var/tmp/fkt.dat", &kT);
26             wait(10000);
27         }
28         //A7: Receive signals
29         getAdapMechPkt(k, &x_kml, &e_kml, &uc_kml, &th_kml);
30         //A8: Generate adaptive gains
31         genAdapMechPkt(k, &x_kml, &e_kml, &uc_kml, &th_kml,
  &th_k);
32         //A9: Step increment
33         k++;
34     }

```

```

35 }
36 else
37     //A10: Error message
38     printf ("<p>Sorry, the request is invalid.\n");
39 //A11: Close body and html tags
40 printf ("</body></html>\n");
41 exit (0);
42 }

```

Figure 7.10. Pseudo code of the CGI program.

7.5.2. NFS Server Setup

The NFS server resides in a Linux machine. There are three utilities needed to run an NFS server: 1) portmap daemon; 2) mount daemon; and 3) NFS daemon. The following procedure describes the NFS server setup for Linux Ubuntu machine:

1) Install the package

```
# sudo apt-get install nfs-kernel-server nfs-common
```

2) Modify /etc/exports to make the file system

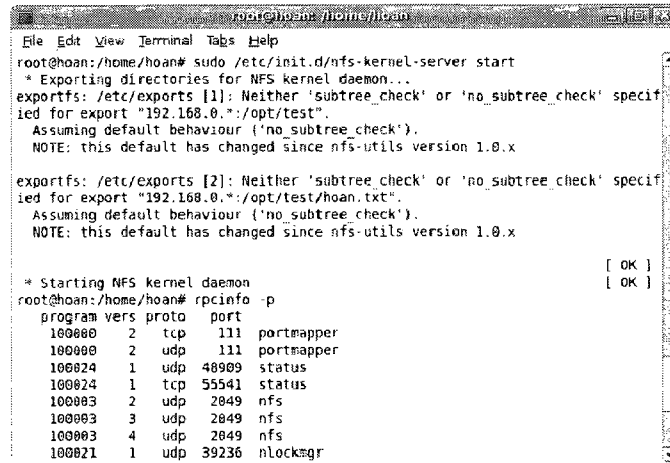
```
/opt/test 192.168.0.128(rw,fsid=0, no_root_squash)
```

where:

- /opt/test is the directory to be exported;
- 192.168.0.128 is the IP address of ARM board (NFS client);
- rw is to allow client machine to read and write access to the directory;
- no_root_squash is to allow root on the client machine to have the same level of access to the files on the system as root on the server.

- 3) Enable the NFS server, which will start multiple services and export the file system as shown in Figure 7.11.

```
#sudo /etc/init.d/nfs-kernel-server start
```



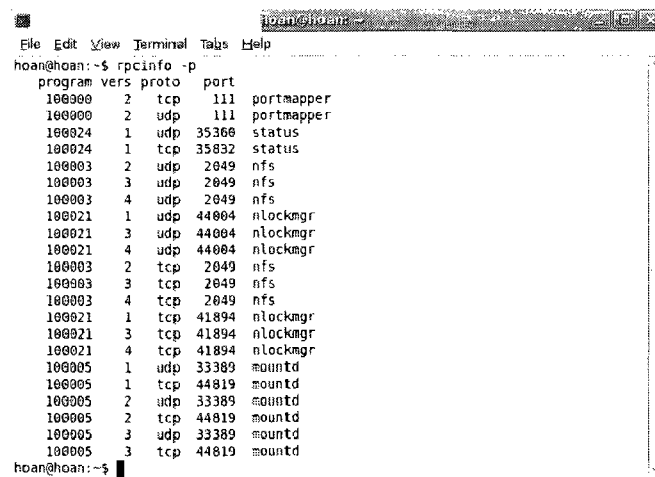
```
root@hoan:/home/hoan# sudo /etc/init.d/nfs-kernel-server start
* Exporting directories for NFS kernel daemon...
exportfs: /etc/exports [1]: Neither 'subtree_check' or 'no_subtree_check' specified for export "192.168.0.*:/opt/test".
Assuming default behaviour ('no_subtree_check').
NOTE: this default has changed since nfs-utils version 1.0.x
exportfs: /etc/exports [2]: Neither 'subtree_check' or 'no_subtree_check' specified for export "192.168.0.*:/opt/test/hoan.txt".
Assuming default behaviour ('no_subtree_check').
NOTE: this default has changed since nfs-utils version 1.0.x

* Starting NFS kernel daemon
root@hoan:/home/hoan# rpcinfo -p
program vers proto port
100000 2 tcp 111 portmapper
100000 2 udp 111 portmapper
100024 1 udp 48909 status
100024 1 tcp 55541 status
100003 2 udp 2049 nfs
100003 3 udp 2049 nfs
100003 4 udp 2049 nfs
100021 1 udp 39236 nlockmgr
```

Figure 7.11. Enable NFS server for Ubuntu machine.

Finally, we can check status of RPC daemon using the following command:

```
#sudo rpcinfo -p
```



```
hoan@hoan:~$ rpcinfo -p
program vers proto port
100000 2 tcp 111 portmapper
100000 2 udp 111 portmapper
100024 1 udp 35360 status
100024 1 tcp 35832 status
100003 2 udp 2049 nfs
100003 3 udp 2049 nfs
100003 4 udp 2049 nfs
100021 1 udp 44004 nlockmgr
100021 3 udp 44004 nlockmgr
100021 4 udp 44004 nlockmgr
100003 2 tcp 2049 nfs
100003 3 tcp 2049 nfs
100003 4 tcp 2049 nfs
100021 1 tcp 41894 nlockmgr
100021 3 tcp 41894 nlockmgr
100021 4 tcp 41894 nlockmgr
100005 1 udp 33389 mountd
100005 1 tcp 44819 mountd
100005 2 udp 33389 mountd
100005 2 tcp 44819 mountd
100005 3 udp 33389 mountd
100005 3 tcp 44819 mountd
```

Figure 7.12. Check rpc daemon status.

7.6. Building the Kernel Image

7.6.1. NFS Client Setup

Firstly, we do “make menuconfig” for configuring the uClinux kernel. Under “FileSystems” configuration, we choose “Network File Systems.”

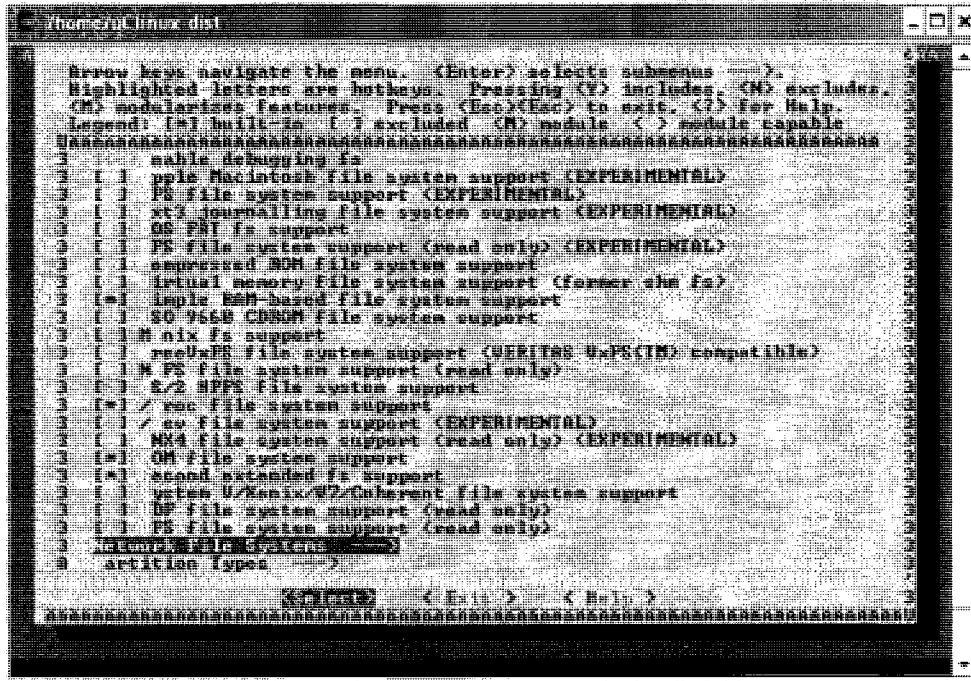


Figure 7.13. Select network file systems.

Now, under “Network File Systems”, the following options should be chosen: 1) “NFS File System Support”; and 2) “Provide NFSv3 client support” as shown in Figure 7.14. This completes the required configuration for the NFS client set up.

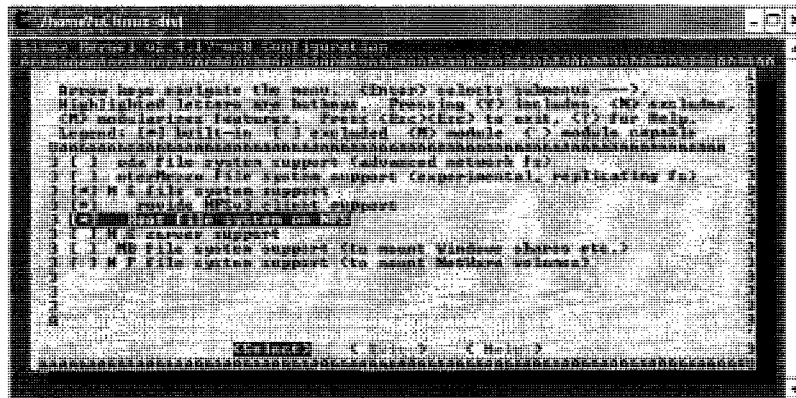


Figure 7.14. Select NFS supports.

Secondly, we add user level utility for remote mounting of the server exported directory. Since the “mount” under uClinux-dist/user does not work well with Linux 2.4 kernel, we use the busybox utility. The busybox user level utility needs to be configured to include mount and umount. We choose “BusyBox” configuration under “Vendor Settings” and select “mount” and “umount” under “BusyBox configuration.”

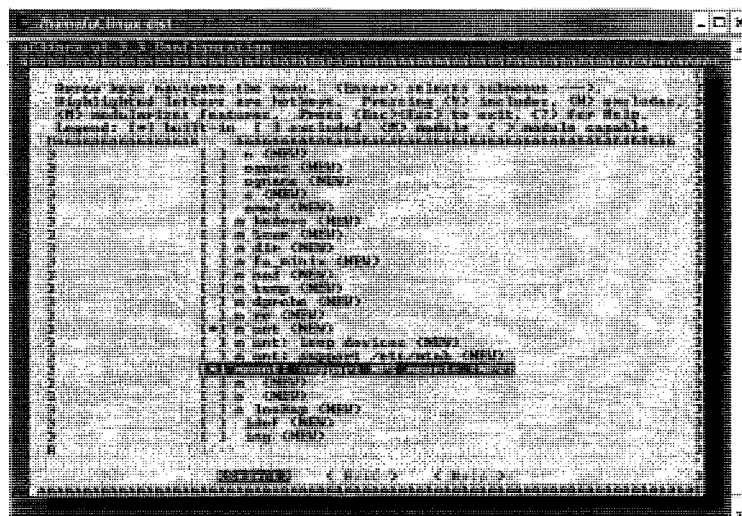


Figure 7.15. Select the “mount” option.

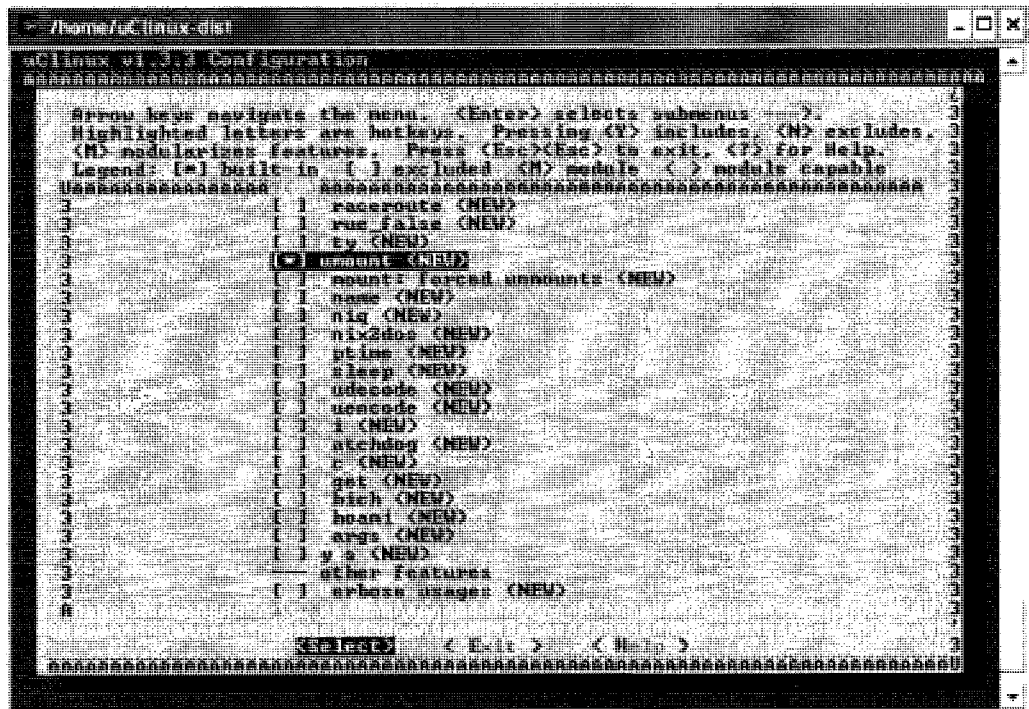


Figure 7.16. Select the “umount” option.

Lastly, we modify “Makefile” under /user/busybox directory. We make some modifications for ROMFS as follows:

```
romfs: #install-romfs.sh busybox.links
syslog-install
    cp $(PROG) $(ROMFSDIR)/bin/$(PROG)
    $(ROMFSINST) /bin/$(PROG)
    $(SHELL) $< $(ROMFSDIR)/bin/
```

We fix for env, xargs, and rm: 1) change env -i to /bin/env; 2) change xarg to /bin/xargs ; and 3) change rm -f to /bin/rm -f. The modifications are shown in Figure 7.17.

```

/home/uClinux-dist/user/busybox
tar -czf busybox-$(VERSION).tar.gz busybox-$(VERSION)/;

.PHONY: tags
tags:
    etags -R .

.PHONY: syslog-install
syslog-install:
    $(ROMFSINST) -e CONFIG_USER_BUSYBOX_SYSLOGD -s /var/tmp/log /dev/log

ifdef CONFIG_UCLINUX
romfs: #install-romfs.sh busybox.links syslog-install
    cp $(PROG) $(ROMFSDIR)/bin/$(PROG)
    $(ROMFSINST) /bin/$(PROG)
    $(SHELL) $(ROMFSDIR)/bin/
else
romfs: install.sh busybox.links syslog-install
    $(SHELL) $(ROMFSDIR)
endif

# Heat rule to build a new config.h and overwrite if it is different
# This avoids bulk busybox builds.
# We also trash any existing busybox setup in the romfs bin
build-config: build-config.awk
    if ! -f "$${ROMFSDIR}/bin/busybox" ; then \
        inode=$(ls -li "$${ROMFSDIR}/bin/busybox" | awk '{print $$1}'); \
        ls -li "$${ROMFSDIR}/bin" | grep "^ *$$inode" | awk '{print $$2}' | \
        sed "s:^:$$ROMFSDIR/bin/:" | /bin/env -i /bin/xargs /bin/
    /rm -f; \
    fi
    awk -f build-config.awk Config.h >Config.temp
    if cmp -s Config.temp Config.h ; then

```

Figure 7.17. Modifications for Busybox Makefile.

7.6.2. Enabling the CGI Protocol

The CGI protocol is built into the Boa web server by default for the uClinux distribution. We do the following two steps to enable the CGI protocol: 1) modify “boa.conf” to make sure “cgi-bin” is mapped to Boa document root directory; and 2) customize Boa “Makefile” to make sure cgi program is properly compiled, built, and placed at the proper directory of the kernel image.

In the first step, we modify the “boa.conf” configuration file. We note that the configuration file for Boa is located in /uClinux-dist/user/boa/src directory. The first thing is to map the virtual location of /cgi-bin/ to the actual location /usr/lib/cgi-bin/. We usually set /etc/ as the document root of Boa server; hence, we can use it as the physical

location for the CGI program. We edit “boa.conf” by replacing the line of the Boa server root location as

```
ScriptAlias /cgi-bin/ /etc/
```

In the final step, we modify Boa “Makefile” to build the CGI program to the kernel image as follows:

- 1) Add one line to define “mycgi” as the CGI executable to be built:

```
TESTEXEC = mycgi
```

- 2) Add one line to define “mycgi.o” as the temporary object file of “mycgi” during compilation:

```
TESTOBS = mycgi.o
```

- 3) Add multiple lines to define the way to build “mycgi” including depending object (“mycgi.o”), the compiler (“\$(CC)”), and the referenced libraries as:

```
$(TESTEXEC) : $(TESTOBS)
```

```
$(CC) $(LDFLAGS) -o $@
```

```
$(TESTOBS) $(SSL_LIBS) $(EXTRALIBS) $(LDLIBS)
```

- 4) Add multiple lines under “romfs” entry to define the built program and other files to the file systems:

```
romfs:
```

```
$(ROMFSINST) /bin/$(EXEC)
```

```
$(ROMFSINST) /etc/$(CONFIG)
```

```
$(ROMFSINST) /etc/$(MIME)
```

```
$(ROMFSINST) /etc/$(INDEX)
```

```
$(ROMFSINST) /etc/$(TESTEXEC)
```



```
$(ROMFSINST) /etc/embedded_adaptive_controller.html
```

```
$(ROMFSINST) /etc/logo.gif
```

```
$(ROMFSINST) /etc/image.jpg
```

```
$(ROMFSINST) /etc/clip.mpg
```

7.6.3. Enabling the Boa Web Server

Enabling the Boa web server from uClinux distribution takes 3 steps: 1) customize kernel settings, customize vendor/user settings, and update default vendor settings; 2) select networking options and select network device support; and 3) establish a LAN connection of the ARM board and a laptop through a router.

In the first step, when performing “make manconfig”, make sure to check the following three boxes:

- 1) Customize kernel settings;
- 2) Customize vendor/user settings,
- 3) Update default vendor settings;

In the second step, we select networking options and select network device support:

- 1) TCP/IP networking;
- 2) IP kernel level autoconfig;
- 3) IP DHCP support;
- 4) IP BOOTP
- 5) IP RARP

Then exit from the networking options, we enter network device support and check:

- 1) Ethernet (10 or 100 Mbit)

- 2) RTL8019AS

At the application settings customization, we go to “Network applications” screen and check:

- 1) Boa

- 2) dhcpd-new

- 3) dhclient

- 4) ifconfig

- 5) inetd

- 6) ping

- 7) portmap (to be used for NFS port mapping)

- 8) route

- 9) routed

- 10) telnetd

- 11) tftpd

Finally, we build the kernel image using the command “make” from the console terminal.

Once the kernel image is ready, it can be loaded into the ARM board by using use Trivial File Transfer Protocol (TFPT) from either Windows or Linux computer. If a Windows machine is used, we run the TFTP utility “tftp32.exe” under the /Cygwin/tftpboot directory. Otherwise, the following command is to enable TFTP server for Linux machines:

```
root# /etc/init.d/xinetd start
```

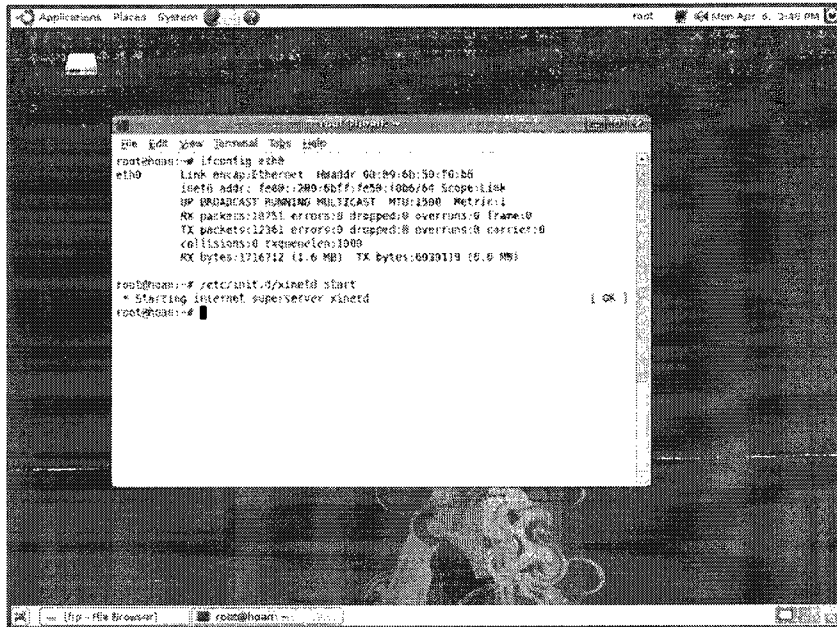


Figure 7.18. Start “xinetd” for Linux Ubuntu.

We then copy the kernel image to the /tftpboot directory. When power up the ARM board, it automatically loads and run the kernel image.

Alternately, we can manually load and run the kernel image using the following commands in the console terminal (Linux machine) or the hyperterminal (Windows machine):

```
tftp 0x0c008000 linux_bootram.bin  
go 0x0c008000
```

```

test - HyperTerminal
File Edit View Call Transfer Help
ARMboot 1.0.2 (Aug 12 2004 - 10:43:21)
ARMboot code: 0c700000 -> 0c719860
CFG_ENV_SIZE=00001000,CFG_ENV_ADDR=00040000.
DRAM Configuration:
Bank #0: 0c000000 8 MB
Flash Configuration:
Flash: 2 MB
*** Using default environment
Hit any key to stop autoboot: 0
S3C44B0 # ifip 0x0c008000 linux_bootram.bin
RTL8019AS ethernet driver v1.0 2003/09/18
ARP broadcast 1
eth addr: 00:15:c5:7b:80:86
IFIP from server 192.168.0.7; our IP address is 192.168.0.128
Filename 'linux_bootram.bin'.
Load address: 0xc008000
Loading: .....
done
Bytes transferred = 1431920 (15d970 hex)
S3C44B0 # go 0x0c008000_

```

Figure 7.19. Manually load the kernel image to the ARM board.

```

test - HyperTerminal
File Edit View Call Transfer Help
Command: mkdir /var/run
Command: mkdir /var/lock
Command: ifconfig lo 127.0.0.1
Command: route add -net 127.0.0.0 netmask 255.255.255.0 lo
Command: dhcpd -p -a eth0 &
[11]
Command: ifconfig eth0 192.168.0.128
Command: /bin/boa -c /etc &
[13]
Command: ps
  PID PORT STAT SIZE SHARED %CPU COMMAND
    1   S   37K   0K  48.5  init
    2   S    0K   0K   0.0  keventd
    3   S    0K   0K   0.0  ksoftirqd_CPU0
    4   S    0K   0K   0.0  kswapd
    5   S    0K   0K   0.0  bdflush
    6   S    0K   0K   0.0  kupdated
    7   R   72K   0K  99.9  /bin/sh /etc/rc
   11   Z    0K   0K   0.0  dhcpd
   13   S  136K   0K   0.0  /bin/boa -c /etc
Execution Finished, Exiting
Sash command shell (version 1.1.1)
/> _

```

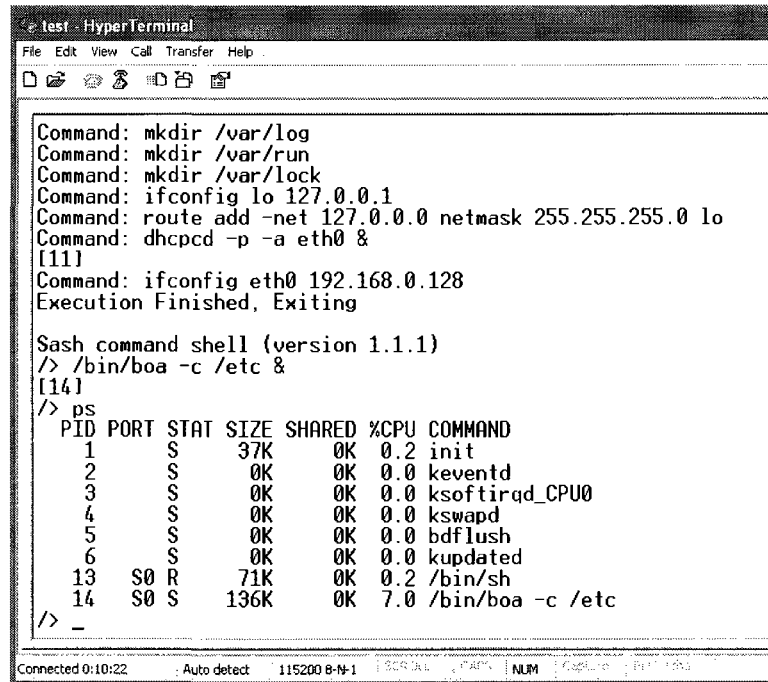
Figure 7.20. Manually run the kernel image.

We are now able to ping ARM board from the host and to ping the host from ARM board. With ping working, we can now start the Boa server as follows:

```
/>/bin/boa -c /etc &
```

where /etc is the internet document root directory of the Boa web server; and “&” makes the Boa process run in the background.

We can use “ps” to check out its status as shown in Figure 7.21.



```
test - HyperTerminal
File Edit View Call Transfer Help
Command: mkdir /var/log
Command: mkdir /var/run
Command: mkdir /var/lock
Command: ifconfig lo 127.0.0.1
Command: route add -net 127.0.0.0 netmask 255.255.255.0 lo
Command: dhcpcd -p -a eth0 &
[11]
Command: ifconfig eth0 192.168.0.128
Execution Finished, Exiting

Sash command shell (version 1.1.1)
/> /bin/boa -c /etc &
[14]
/> ps
  PID  PORT  STAT  SIZE  SHARED  %CPU  COMMAND
   1    -    S      37K   0K     0.2  init
   2    -    S      0K    0K     0.0  keventd
   3    -    S      0K    0K     0.0  ksoftirqd_CPU0
   4    -    S      0K    0K     0.0  kswapd
   5    -    S      0K    0K     0.0  bdflood
   6    -    S      0K    0K     0.0  kupdated
  13    -    S0 R    71K   0K     0.2  /bin/sh
  14    -    S0 S   136K   0K     7.0  /bin/boa -c /etc
/> -
```

Figure 7.21. Process status.

7.7. Testing

7.7.1. Test Procedure

The test procedure for the whole system is as follows:

- 1) Establish the serial communication between a computer and the ARM board using serial communication port 0.
- 2) On the PC side, run hyper terminal software by start > all program > accessories > communications > hyper terminal.

- 3) Set up a LAN network consisting of the ARM board, the NFS server (Linux machine), the plant simulator (PC/ Linux machine), and engineering stations (PC/ Linux machine).
- 4) Power up the ARM board, the hyper terminal console will display the following information:
 - TFTP service loading the kernel image.
 - Uncompressing Linux and booting the kernel.
 - Shell invoked to run the boot script file: /etc/rc.
- 5) Enable Boa and run commands for portmap and NFS mount as depicted in Figure 7.22.

```

test - HyperTerminal
File Edit View Call Transfer Help
[11]
Command: ifconfig eth0 192.168.0.128
Command: boa -c /etc &
[13]
Command: portmap &
[14]
Command: ps
  PID PORT STAT SIZE SHARED %CPU COMMAND
   1   S    37K   0K  33.3  init
   2   S    0K    0K   0.0  keventd
   3   S    0K    0K   0.0  ksoftirqd_CPU0
   4   S    0K    0K   0.0  kswapd
   5   S    0K    0K   0.0  bdflush
   6   S    0K    0K   0.0  kupdated
   7   R   72K   0K  78.5  /bin/sh /etc/rc
  11   Z    0K    0K  45.0  dhcpcd
  13   S   136K   0K   0.0  boa -c /etc
  14   S    91K   0K   0.0  portmap
Command: busybox mount -t nfs -o rsize=1024,wsiz=1024 192.168.0.20:/opt/test /var/tmp
Execution Finished, Exiting
Sash command shell (version 1.1.1)
/> _
Connected 0:02:38 Auto detect 115200 8-N-1 C/POLL C/MS NUM | Capture | Print | ...

```

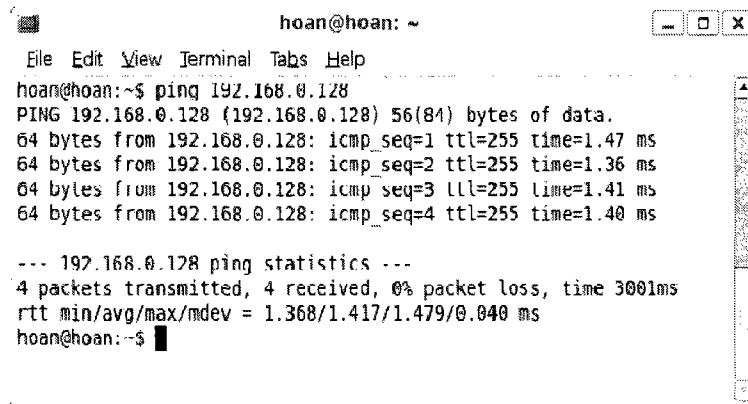
Figure 7.22. Mount network files for the NFS client.

- 6) Mount the network file system for NFS client (i.e., ARM board) using the following command:

```
busybox mount -t nfs -o rsize=1024,wsiz=1024
192.168.0.8:/opt/test /var/tmp
```

where 192.168.0.8 is the IP address of NFS server; /opt/test is the exported directory; and /var/tmp is the local directory on NFS client.

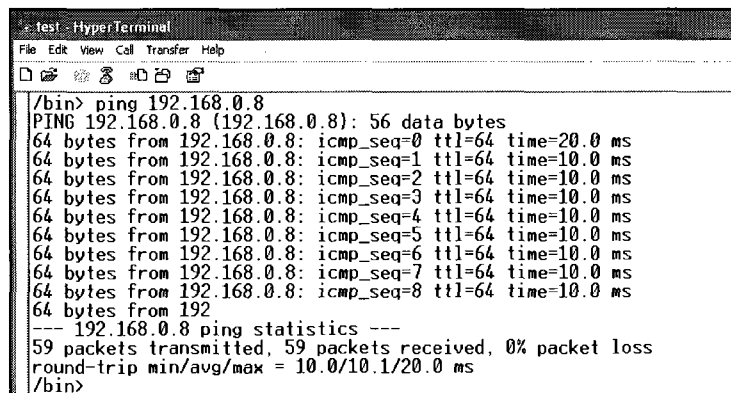
- 7) Ping the ARM board from the Linux machine and vice versa.



```
hoan@hoan: ~
File Edit View Terminal Tabs Help
hoan@hoan:~$ ping 192.168.0.128
PING 192.168.0.128 (192.168.0.128) 56(84) bytes of data.
64 bytes from 192.168.0.128: icmp_seq=1 ttl=255 time=1.47 ms
64 bytes from 192.168.0.128: icmp_seq=2 ttl=255 time=1.36 ms
64 bytes from 192.168.0.128: icmp_seq=3 ttl=255 time=1.41 ms
64 bytes from 192.168.0.128: icmp_seq=4 ttl=255 time=1.40 ms

--- 192.168.0.128 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 1.368/1.417/1.479/0.040 ms
hoan@hoan:~$
```

Figure 7.23. Ping ARM board from the Linux machine.



```
test - HyperTerminal
File Edit View Call Transfer Help
/bin> ping 192.168.0.8
PING 192.168.0.8 (192.168.0.8): 56 data bytes
64 bytes from 192.168.0.8: icmp_seq=0 ttl=64 time=20.0 ms
64 bytes from 192.168.0.8: icmp_seq=1 ttl=64 time=10.0 ms
64 bytes from 192.168.0.8: icmp_seq=2 ttl=64 time=10.0 ms
64 bytes from 192.168.0.8: icmp_seq=3 ttl=64 time=10.0 ms
64 bytes from 192.168.0.8: icmp_seq=4 ttl=64 time=10.0 ms
64 bytes from 192.168.0.8: icmp_seq=5 ttl=64 time=10.0 ms
64 bytes from 192.168.0.8: icmp_seq=6 ttl=64 time=10.0 ms
64 bytes from 192.168.0.8: icmp_seq=7 ttl=64 time=10.0 ms
64 bytes from 192.168.0.8: icmp_seq=8 ttl=64 time=10.0 ms
64 bytes from 192
--- 192.168.0.8 ping statistics ---
59 packets transmitted, 59 packets received, 0% packet loss
round-trip min/avg/max = 10.0/10.1/20.0 ms
/bin>
```

Figure 7.24. Ping the Linux machine from the ARM board.

- 8) A user using a web browser to open the main GUI page of the Boa web server at the URL location $\langle \text{http://192.168.0.128/embedded_adaptive_controller.html} \rangle$.



Figure 7.25. Open the main page of Boa web server.

- 9) Set sampling time and adaptation rate γ and click “Run” to submit information to the CGI program.
- 10) At the same time, run the plant simulator on a Windows/ Linux computer and start testing the embedded adaptive system as shown in Figure 7.26.


```

hcan@hcan: ~/thesis/PlantSim
File Edit View Terminal Tabs Help
hcan@hcan:~/thesis/PlantSim$ ./adaptivecontrol
Time = 0T:
Plant parameters:
A = -7.129100  -6.954500  1.393600  -0.229700
B =  0.139100  0.197300  0.002000  0.026600
B = -9.059400  -6.025600  0.258300  0.006855
u1[0] = 0.517830
u2[0] = 0.100000
x1[0] = 0.000000
x2[0] = 0.000000
xw1[0] = 0.000000
xw2[0] = 0.000000
Time = 1T:
Plant parameters:
A = -7.129100  -6.954500  1.393600  -0.229700
B = -9.139100  0.197300  -0.002200  -0.029600
B = -9.065400  -6.029600  0.258300  0.006945
Input pkt sent to Adaptive Mechanism.
Waiting for adaptive gain pkt.
Press any key to continue...
█

```

Figure 7.26. Start the plant simulator in a Linux machine.

- 11) The testing results can be observed at either the web browser (Figure 7.27) or the plant simulator's consol (Figure 7.28).

```

Web Server - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.0.128/cgi-bin/mycgi?vu
Google Search
e(k-1).p1 = -119
e(k-1).p2 = -18
uc(k-1).k = 6
uc(k-1).p1 = 50000
uc(k-1).p2 = 10000
th(k-1).k = 65995
th(k-1).th1 = 66000
th(k-1).th2 = 5
th(k-1).th3 = 1
th(k-1).th4 = -309998
th(k-1).th5 = 10442254
th(k-1).th6 = -42712
th(k-1).th7 = 4240
Done

```

Figure 7.27. Testing result displayed on the web browser.

```

huang@huang ~/thesis/PlantSim
File Edit View Terminal Tabs Help
Get time::Press any key to continue...
<p> Opening the input file /opt/test/fka.dat
<p> Opening the input file /opt/test/adapout.dat
Adaptive gains synthesized:
th.k[5] = 6001
th1[5] = 0.371599
th2[5] = -0.510600
th3[5] = 0.280602
th4[5] = -0.309999
th5[5] = 1.044203
th6[5] = -0.042708
th7[5] = 0.000449
th8[5] = 1.000624
u1[5] = 0.517732
u2[5] = 0.101250
x1[5] = -0.007699
x2[5] = -0.005796
xm1[5] = -0.007186
xm2[5] = -0.005626
Time = 6T:
Plant parameters:
A = -7.129100   -0.954500   1.543600   -0.239700
B = -0.153100   0.217300   -0.002000  -0.029600
B = -0.059400  -0.029600   0.233300   0.000945
Input pkt sent to Adaptive Mechanism.
Waiting for adaptive gain pkt.
Press any key to continue...

```

Figure 7.28. Testing result displayed on the plant simulator's monitor.

7.7.2. Test Results

As aforementioned, all testing outputs are stored in the shared database, which can be accessed from any computer of the LAN network. The output data files are listed in Table 7.3.

Table 7.3. Structured output data files.

No.	File name	Data description
1	t_x.dat	Plant state x_1 and x_2 .
2	t_xm.dat	Reference state x_{m1} and x_{m2} .
3	t_e.dat	Error variables e_1 and e_2 .
4	t_uc.dat	Reference signals u_{c1} and u_{c2} .
5	t_u.dat	Control signals u_1 and u_2 .
6	t_theta.dat	Adaptive gains $\theta_1 - \theta_8$.
7	t_y.dat	Controlled outputs y_1 and y_2 .
8	t_ym.dat	Reference outputs y_{m1} and y_{m2} .

The plant simulator has varying dynamics by randomized parameters and interfered by unpredictable disturbances. The embedded adaptive controller synthesizes adaptive gains at every step time. We find that the plant states always keep track of the reference states all the testing time as shown in Table 7.4.

Table 7.4. Output data of state variables.

k	Time	x_1	x_2	x_{m1}	x_{m2}
0	0	0.00000	0.00000	0.00000	0.00000
1	0.1	-0.00521	-0.00037	-0.00523	-0.00039
2	0.2	-0.00667	-0.00151	-0.00687	-0.00153
3	0.3	-0.00817	-0.00291	-0.00730	-0.00289
4	0.4	-0.00764	-0.00447	-0.00731	-0.00427
5	0.5	-0.00770	-0.00580	-0.00719	-0.00563
6	0.6	-0.00815	-0.00713	-0.00702	-0.00693
7	0.7	-0.00738	-0.00862	-0.00685	-0.00817
8	0.8	-0.00650	-0.00986	-0.00669	-0.00936
9	0.9	-0.00665	-0.01091	-0.00652	-0.01049
10	1	-0.00641	-0.01202	-0.00637	-0.01158
11	1.1	-0.00623	-0.01313	-0.00622	-0.01261
12	1.2	-0.00645	-0.01403	-0.00608	-0.01359
13	1.3	-0.00594	-0.01503	-0.00595	-0.01453
14	1.4	-0.00587	-0.01586	-0.00582	-0.01543
15	1.5	-0.00551	-0.01667	-0.00569	-0.01628
16	1.6	-0.00514	-0.01748	-0.00558	-0.01710
17	1.7	-0.00587	-0.01817	-0.00546	-0.01788
18	1.8	-0.00641	-0.01903	-0.00536	-0.01862

19	1.9	-0.00522	-0.01982	-0.00526	-0.01933
20	2	-0.00480	-0.02051	-0.00516	-0.02000
21	2.1	-0.00564	-0.02115	-0.00507	-0.02065
22	2.2	-0.00550	-0.02190	-0.00498	-0.02126
23	2.3	-0.00539	-0.02251	-0.00489	-0.02185
24	2.4	-0.00547	-0.02311	-0.00481	-0.02241
25	2.5	-0.00528	-0.02367	-0.00474	-0.02294
26	2.6	-0.00496	-0.02427	-0.00466	-0.02345
27	2.7	-0.00503	-0.02476	-0.00460	-0.02394
28	2.8	-0.00430	-0.02525	-0.00453	-0.02440
29	2.9	-0.00472	-0.02558	-0.00447	-0.02484
30	3	-0.00441	-0.02603	-0.00440	-0.02526
31	3.1	-0.00491	-0.02641	-0.00435	-0.02566
32	3.2	-0.00511	-0.02687	-0.00429	-0.02605
33	3.3	-0.00390	-0.02732	-0.00424	-0.02641
34	3.4	-0.00446	-0.02756	-0.00419	-0.02676
35	3.5	-0.00383	-0.02781	-0.00414	-0.02710
36	3.6	-0.00369	-0.02813	-0.00410	-0.02741
37	3.7	-0.00381	-0.02832	-0.00405	-0.02772
38	3.8	-0.00435	-0.02862	-0.00401	-0.02800
39	3.9	-0.00441	-0.02894	-0.00397	-0.02828
40	4	-0.00439	-0.02916	-0.00393	-0.02854
41	4.1	-0.00447	-0.02952	-0.00390	-0.02879
42	4.2	-0.00373	-0.02980	-0.00386	-0.02903
43	4.3	-0.00418	-0.02995	-0.00383	-0.02926
44	4.4	-0.00404	-0.03014	-0.00380	-0.02948
45	4.5	-0.00373	-0.03035	-0.00377	-0.02969
46	4.6	-0.00440	-0.03052	-0.00374	-0.02989

47	4.7	-0.00453	-0.03078	-0.00371	-0.03007
48	4.8	-0.00436	-0.03109	-0.00369	-0.03025
49	4.9	-0.00385	-0.03139	-0.00366	-0.03043
50	5.0	-0.00428	-0.03161	-0.00364	-0.03059

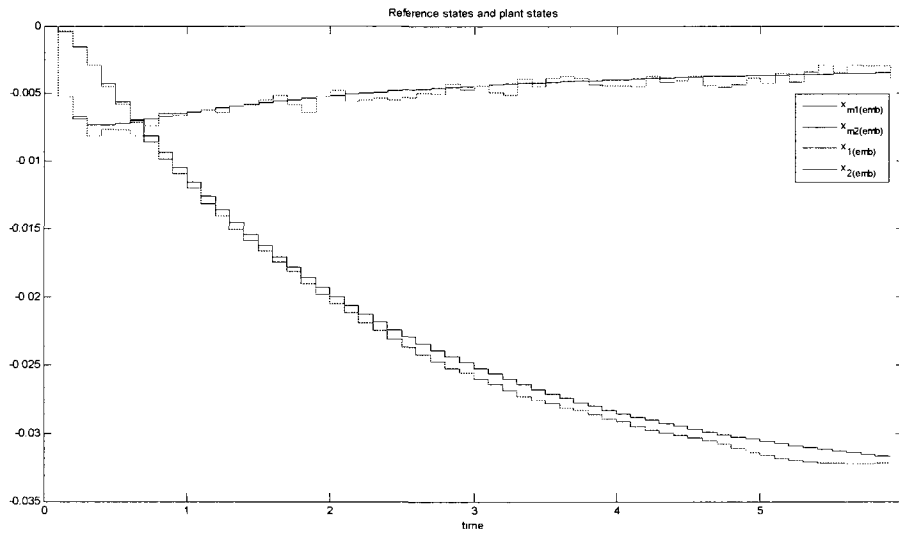


Figure 7.29. Variable states during simulation of the embedded adaptive controller.

As introduced in Sec.7.1.2, the output data of the embedded adaptive controller will be compared with the result of software models, the MATLAB Simulink program and the C++ executable. We note that MATLAB and C++-based simulation programs have the same results. The comparison result is shown in Table 7.5. We find that all data are almost identical.

Table 7.5. Comparison result between the embedded and software models.

k	Time	Δx_1	Δx_2	Δx_{m1}	Δx_{m2}
0	0	0.0000	0.0000	0.0000	0.0000
1	0.1	0.0007	0.0000	0.0000	0.0000
2	0.2	0.0006	0.0001	0.0000	0.0000
3	0.3	-0.0006	0.0001	0.0000	0.0000
4	0.4	-0.0009	-0.0001	0.0000	0.0000
5	0.5	-0.0011	-0.0002	0.0000	0.0000
6	0.6	-0.0008	-0.0002	0.0000	0.0000
7	0.7	0.0005	-0.0003	0.0000	0.0000
8	0.8	0.0004	-0.0003	0.0000	0.0000
9	0.9	-0.0001	-0.0002	0.0000	0.0000
10	1	0.0001	-0.0002	0.0000	0.0000
11	1.1	-0.0004	-0.0003	0.0000	0.0000
12	1.2	-0.0008	-0.0002	0.0000	0.0000
13	1.3	-0.0001	-0.0003	0.0000	0.0000
14	1.4	0.0000	-0.0003	0.0000	0.0000
15	1.5	0.0003	-0.0003	0.0000	0.0000
16	1.6	0.0006	-0.0002	0.0000	0.0000
17	1.7	0.0002	-0.0001	0.0000	0.0000
18	1.8	-0.0002	-0.0002	0.0000	0.0000
19	1.9	0.0006	-0.0002	0.0000	0.0000
20	2	0.0002	-0.0002	0.0000	0.0000
21	2.1	-0.0005	-0.0002	0.0000	0.0000
22	2.2	-0.0004	-0.0002	0.0000	0.0000
23	2.3	-0.0005	-0.0002	0.0000	0.0000
24	2.4	-0.0004	-0.0002	0.0000	0.0000
25	2.5	-0.0008	-0.0002	0.0000	0.0000

26	2.6	-0.0010	-0.0003	0.0000	0.0000
27	2.7	-0.0009	-0.0004	0.0000	0.0000
28	2.8	0.0004	-0.0005	0.0000	0.0000
29	2.9	0.0002	-0.0003	0.0000	0.0000
30	3	-0.0001	-0.0003	0.0000	0.0000
31	3.1	-0.0009	-0.0003	0.0000	0.0000
32	3.2	-0.0006	-0.0004	0.0000	0.0000
33	3.3	0.0006	-0.0004	0.0000	0.0000
34	3.4	0.0001	-0.0002	0.0000	0.0000
35	3.5	0.0005	-0.0001	0.0000	0.0000
36	3.6	0.0008	-0.0002	0.0000	0.0000
37	3.7	0.0006	0.0000	0.0000	0.0000
38	3.8	-0.0005	0.0000	0.0000	0.0000
39	3.9	-0.0001	0.0000	0.0000	0.0000
40	4	-0.0008	0.0001	0.0000	0.0000
41	4.1	-0.0013	-0.0001	0.0000	0.0000
42	4.2	0.0003	-0.0002	0.0000	0.0000
43	4.3	-0.0002	-0.0002	0.0000	0.0000
44	4.4	-0.0004	-0.0002	0.0000	0.0000
45	4.5	-0.0007	-0.0002	0.0000	0.0000
46	4.6	-0.0014	-0.0003	0.0000	0.0000
47	4.7	-0.0006	-0.0005	0.0000	0.0000
48	4.8	-0.0007	-0.0006	0.0000	0.0000
49	4.9	-0.0001	-0.0008	0.0000	0.0000
50	5.0	-0.0010	-0.0008	0.0000	0.0000

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1. Conclusion

Adaptive control is applied for solving the control problem of the gasoline refinery with high nonlinearity and unpredictable disturbances. For this class of control problems, conventional controllers are very limited and have a lot of deficiencies.

The adaptive mechanism governed by an adaptation law is the heart of any adaptive controller. We establish the adaptation law for the plant control system using Lyapunov stability theory. This adaptation law is accurate for a generic second order plant; hence, it is obviously applicable for adaptive control of other second order systems in different realms such as chemical industry, military industry, and robotics.

Adaptive control system design is much complicated than conventional controller design due to the complexity of adaptive structure and the issue of unknown or time-varying parameters. We use an effective methodology for system modeling and design using state space. All computational works are based on matrix manipulation, which is fully supported by MATLAB. We can apply this methodology to extend the result to higher order systems.

We have shown the design – verification flow with various phases: 1) mathematical model; 2) process calculation and modeling; 3) controller design in both continuous-time and discrete spaces; 4) controller simulation; and 5) in-hardware implementation and testing. The mathematical model, which is given in term of a set of mathematical equations of energy and material balances, provides insightful

understanding of the dynamic behaviors of the plant. The process calculation is necessary to obtain steady-state data for control system design in later phases. The reference model is constructed with mathematical approach and proven to be stable and ensure the plant's steady-state properties.

We design the adaptive controller in continuous-time space and then convert it into discrete-time space using z - transform. The discrete-time version must be accomplished prior to implementation of the adaptive design on digital computers. We successfully design and implement the embedded adaptive controller using ARM processor. We apply the state-of-the-art testing technique in which we employ NFS and concept of distributed system over an Ethernet network. We have overcome the new challenges, which require a reliable closed control loop and real-time data transfer between the controller and the plant simulator. As a result, the embedded adaptive controller remotely controls the plant simulator on a network node.

8.2. Future Work

First, we can design embedded adaptive controllers using combination of Lyapunov theory and hyperstability concept, which give better rejection of disturbances and time-varying parameter problems [34]. Many authors concern this perspective, for example, Nguyen and Nitin [10] design an adaptive control system with hyperstability in continuous-time space. We can extend the existing works to discrete-time space and use the same methodology proposed in the thesis to carry out in-hardware implementation.

Second, we can build a real pilot plant. It is costly; however, we can carry out experiments and measure actual control performance. The experimental scheme is

proposed as displayed in Figure 9.1. Ideally, the composition is continuously measured by an online analyzer or gas chromatography. However, we can select tray temperatures as secondary measurements, which indicate product concentration in an indirect way. The reason is to reduce equipment cost. Temperatures T_σ and T_ρ are the most sensitive temperatures, which are best related to product concentrations x_B and x_D .

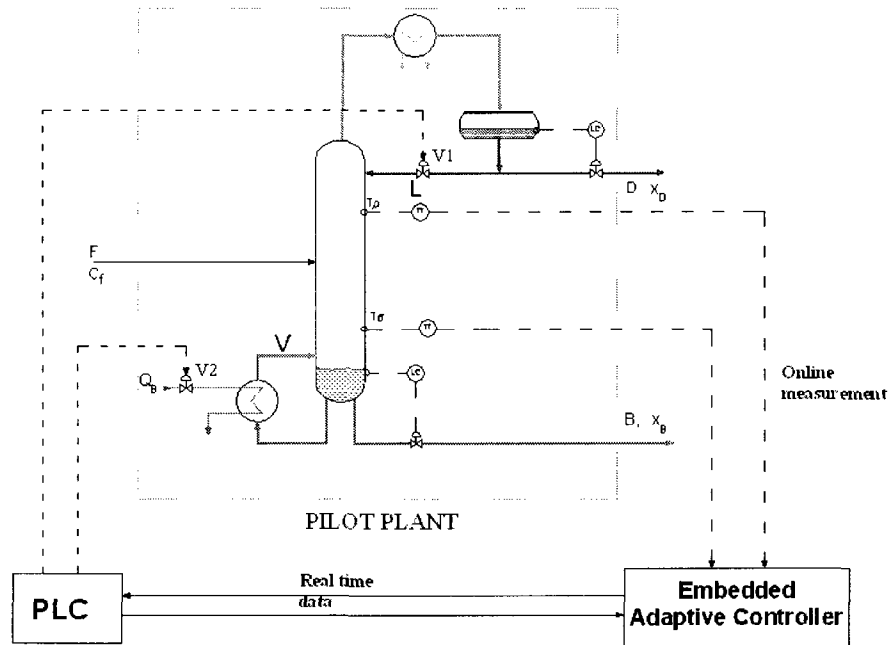


Figure 9.1. Block diagram of the experimental pilot plant.

The proposed scheme includes three main parts. The first part is a distillation column in laboratory. The second part is an embedded adaptive controller implemented in a PC or microcontroller. The third part is a programmable logic controller (PLC) system for handling control valves V_1 and V_2 .

Third, adaptive mechanism can be programmed in a PLC. By using this concept, we can upgrade existing conventional controllers using PLCs just by modification of their programs. This is important to save equipment and engineering cost.

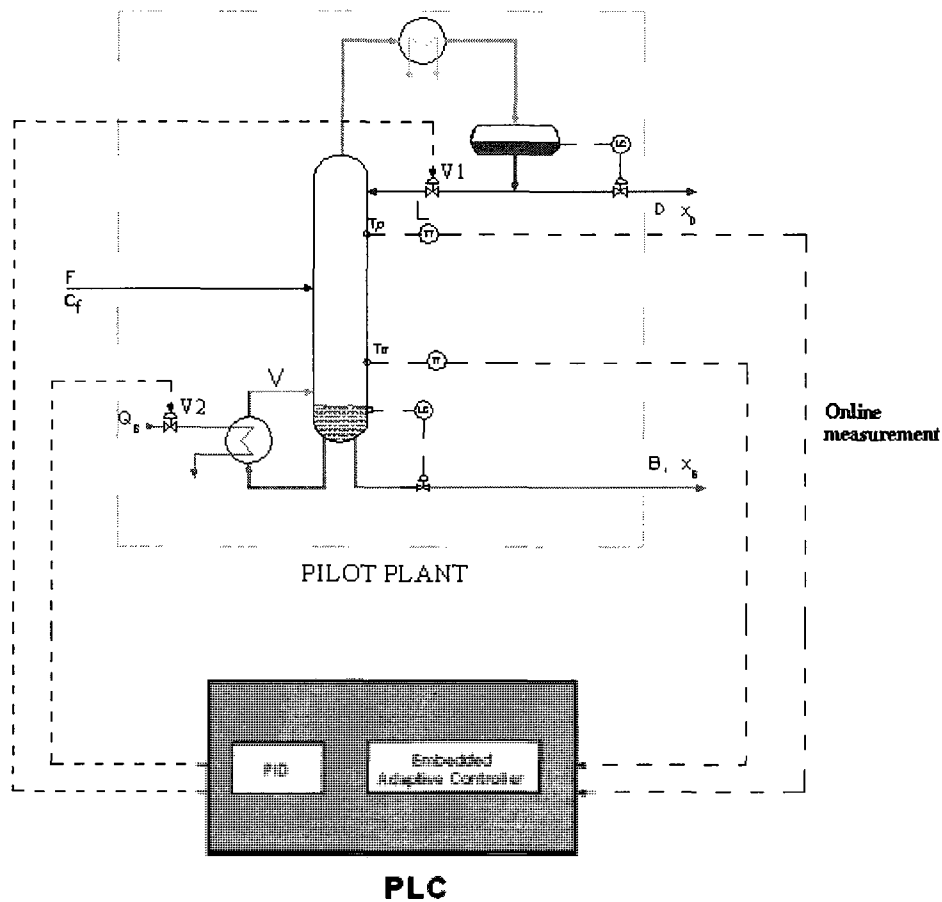


Figure 9.2. Adaptive mechanism programmed in a PLC.

Fourth, we can design adaptive controller as System on a Chip (SoC) which contains the complete system. SoCs are developed with Verilog or any HDL language and can be implemented using a Field-Programmable Gate Array (FPGA) such as an Ethernet-supported Xilinx FPGA board.

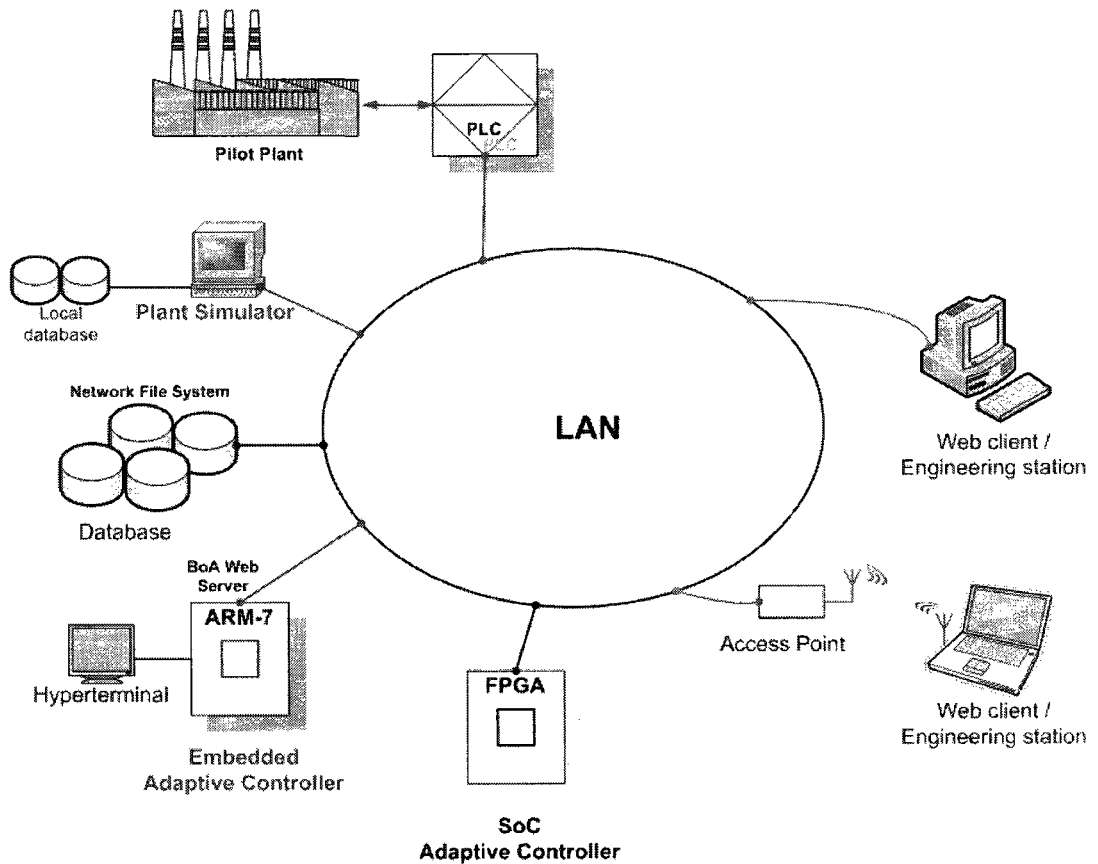


Figure 9.3. Integrated testing environment for SoCs and other digital systems.

Finally, we can extend testing environment introduced in Sec. 7.1.3 by integrating SoC adaptive controller and pilot plant as shown in Figure 9.3. The embedded adaptive controller remotely controls the pilot plant over the LAN network. Obviously, we can deploy this testing methodology for validation of SoCs and other digital systems, particularly distributed systems.

APPENDIX A: DISTILLATION CONTROL TECHNIQUES

A.1. Column Pressure Control

Most distillation control systems, either conventional or advanced, assume that distillation columns operate at a constant pressure. Maintaining constant operation gives stable operation and increases overall plant profit.

It is important to prevent pressure of a distillation column from changing rapidly, either up or down. Sudden decreases in pressure can cause flashing of the liquid on the trays; and the excessive vapor rate can flood the column. Sudden increase in pressure can cause condensation of vapor; and the low vapor rates can cause weeping and dumping of trays.

We consider again the schematic of distillation column, as shown in Figure 2.1. The overhead vapor after being heat removal in the condenser will consist of two phases, liquid and vapor. The objective is to condense maximum quantity of distillate (i.e., maximize the profit) at its true boiling point to minimize the energy cost for condenser duty. There is a pressure balance established between the column top and the reflux drum for the purpose of stabilizing the column pressure. Several common types of column pressure control are described in the following sections.

A.1.1. Coolant Manipulation

As shown in Figure A.1, the condenser is normally a heat exchanger using coolant (e.g., cooling water or refrigerant). The control valve adjusts the flow rate of coolant to obtain the desired condenser duty.

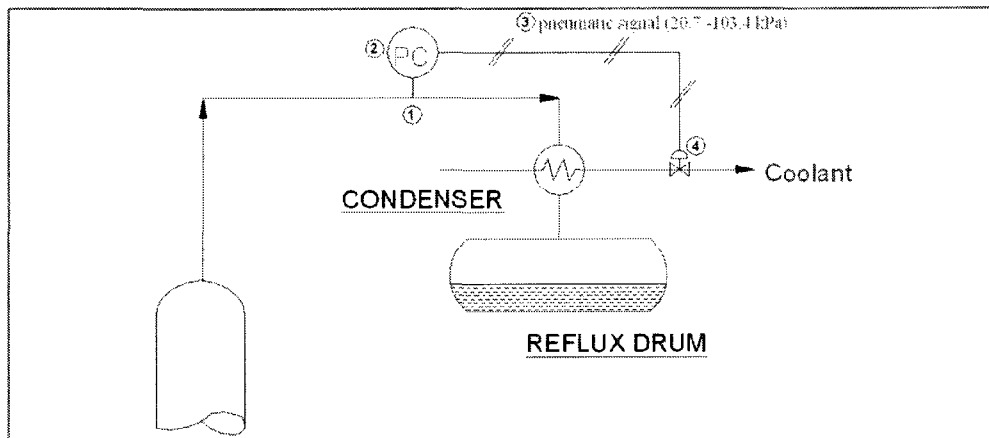


Figure A.1. Column pressure control using coolant manipulation.

Pressure signal from the pressure sensor installed in the overhead vapor flow is transmitted to the pressure controller. The control output, which is a function of the error signal based on the deviation of the measured pressure from the set point, supplies controlled pressurized air (20.7 – 103.4 kPa) to the pneumatic valve actuator. As the result, the coolant flow rate is adjusted to the desired value. If the overhead flow rate increases (or decreases), the coolant flow rate will increase (or decrease) correspondingly. This maintains the column pressure stable.

A.1.2. Vent-bleed

As shown in Figure A.2, inert gas is added or bled from the system using a dual split-ranged valve system so that under normal conditions, both control valves are closed. The reflux flow must be considerably sub-cooled in order to keep the product concentration in the vent gas stream low. The reflux temperature is directly affected by the coolant temperature.

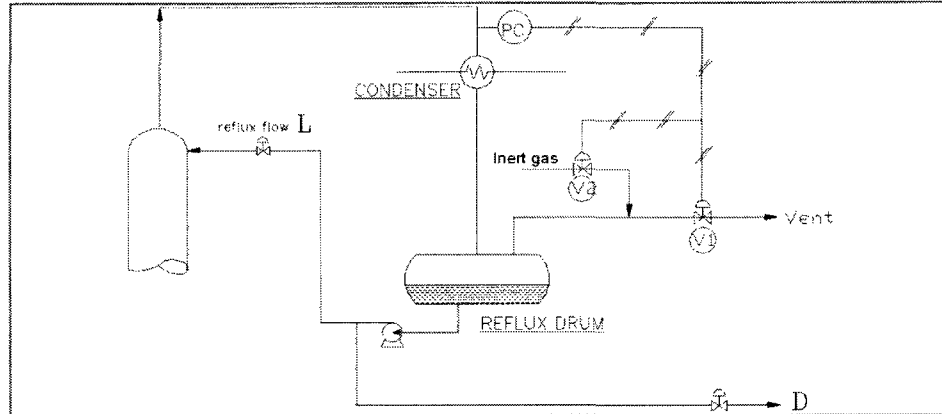


Figure A.2. Column pressure control using vent bleed.

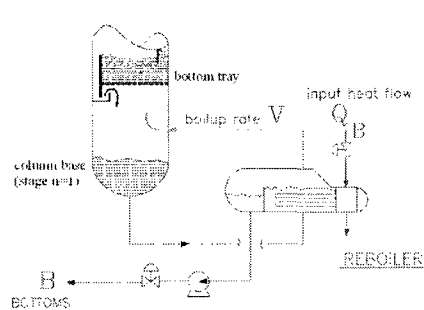
The cooling capacity of condenser is preset. When the vapor rate increases significantly, it is not totally condensed. As a consequence, the pressure in the reflux drum will increase. To stabilize it, the pressure controller will command the control valve V1 to vent the exceed vapor.

In contrast, the vapor rate is condensed too much, which causes a pressure drop at the top column section. Consequently, the control valve V1 is gradually closed; and the control valve V2 is slightly open.

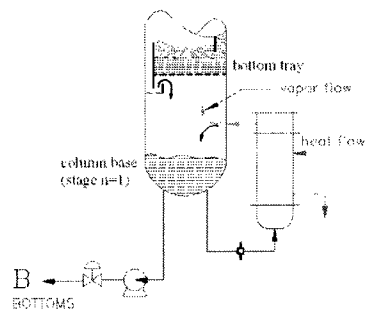
A.2. Column Level Control

The two liquid levels that must be controlled are in the reflux drum and column base. The levels are controlled in different ways, depending on a number of factors [35].

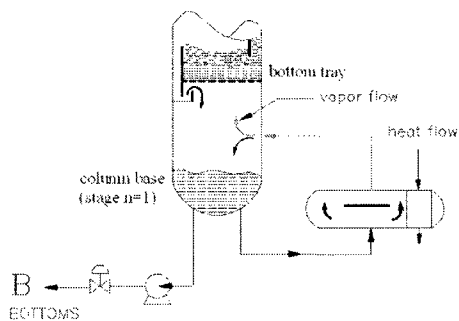
If the column is part of a series of units in a plant, it is usually important from a plant-wide control viewpoint to use the liquid levels as surge capacities to reduce effect of disturbances. In such an environment, it is usually preferable to control the base level with the bottoms flow and the reflux drum level with the distillate flow.



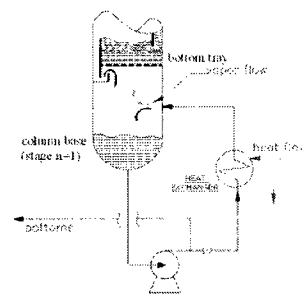
a) Kettle type.



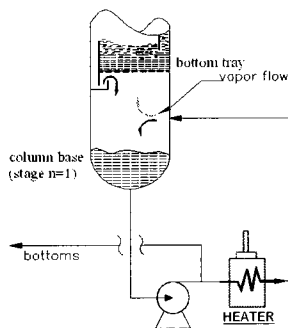
b) Vertical thermo-siphon type.



c) Horizontal thermo-siphon type.



d) Forced-circulation type with heat exchanger.



e) Forced-circulation type with heater.

Figure A.3. Some typical types of reboiler.

In high reflux ratio columns (i.e., $L/D > 5$), using distillate stream to control level would require large changes in D for fairly small changes in L or V . Thus, the

disturbances would be amplified in the distillate flow rate. The reflux drum level should be controlled by reflux in accordance with the Richardson's rule—we always control level with the level with the largest stream [36].

In practice, we should care of potential problems with “inverse response” that may happen and cause the plant unstable. An increase in reboiler heat input can quickly increase the fraction of vapor. In a thermo-siphon reboiler, this can push liquid back into the base of the column, resulting in a momentary increase in the liquid level in the column base. In a kettle reboiler, the increase in vapor fraction causes the material in the reboiler to swell and more liquid flows over the outlet weir into the surge volume in the end of the reboiler. Therefore, the liquid level in this section momentarily increases. Various reboiler types are shown in Figure A.3.

A.3. Methods of Distillation Column Control

A.3.1. Degrees of Freedom of Distillation Process

The degrees of freedom of a processing system are the independent variables that must be specified in order to define the process completely. Consequently, the desired control of a process will be achieved when and only when all the degrees of freedom have been specified.

The mathematical approach to finding the degrees of freedom of any process is to total all the variables and subtract the number of independent equations [12]. However, there is a simple approach developed by Luyben [35]. In Figure 2.1, there are five control valves, one on each of the following streams: distillate, reflux, coolant, bottoms, and heating medium. The feed stream is considered being set by the upstream process.

So this column has five degrees of freedom. But inventories in any process always must be controlled. Inventory loops involve liquid levels and pressures. This means that the liquid level in the reflux drum, the liquid level in the column base; and the column pressure must be controlled.

If we subtract the three variables that must be controlled from five, we end up with two degrees of freedom. Thus, there are two and only two additional variables that can be controlled in the distillation column.

Notice that we have made no assumptions about the number or type of chemical components being distilled. Therefore, a simple, ideal, binary system has two degrees of the freedom; and a complex, multi-component, non-ideal distillation system also has two degrees of freedom.

A.3.2. Control Structures

The manipulated variables and controlled variables of a distillation column are displayed in Table A.1.

Table A.1. Manipulated variables and controlled variables of a distillation column.

	Controlled variables	Manipulated variables	Control valve location
1	Concentration (temperature) of distillate	Reflux flow rate	Reflux flow (V1)
2	Concentration (temperature) of bottoms	Reboiler duty	Heat flow (V4)
3	Column pressure	Condenser duty	Coolant flow (V3)
4	Liquid level in the column base	Bottoms flow rate	Bottoms flow (V5)
5	Liquid level in the reflux drum	Distillate flow rate	Distillate flow (V2)

The column has 2 degrees of freedom; hence, a control structure is a selective combination of two manipulated variables. As shown in Table A.2, there are many common control structures are used in practical distillation [37].

Table A.2. Typical column control structures.

	Control Structure	Role of valve				Manipulated variable	
		D (V2)	L (V1)	QB (V4)	B (V5)	Inventory Control (IC)	Separation Control (SC)
1	D-V(or D-QB)	SC	IC	SC	IC	L, B	D, V
2	L-V(or L-QB)	IC	SC	SC	IC	D, B	L, V
3	L-B	IC	SC	IC	SC	D, V	L, B
4	L/D-V	IC	SC	SC	IC	D, B	L/D, V
5	L/D-B	IC	SC	IC	SC	D, V	L/D, B
6	D-V/B	SC	IC	SC	IC	L, B	D, V/B
7	L-V/B	IC	SC	SC	IC	D, B	L, V/B
8	L/D-V/B	IC	SC	SC	IC	D, B	L/D, V/B

For a binary distillation, the most common structures are the energy balance structure, $L-V$, and the material balance structure, $D-V$ and $L-B$. Most industrial distillation columns on two-point control are probably operated by one of these control structures.

Selecting a control structure is a complicated problem with many facets. It requires looking at the column control problem from several perspectives:

- 1) A local perspective considering the steady state characteristics of the column;
- 2) A local perspective considering the dynamic characteristics of the column;

- 3) A global perspective considering the interaction of the column with other units in the plant.

A.3.3. Energy Balance Structure

As shown in Figure A.4, the energy balance structure, which is usually called $L-V$ structure, can be considered to be the standard control structure for dual composition control of distillation. In this control structure, the reflux flow rate L and the boilup manipulator V are used to control the “primary” outputs associated with the product specifications. The liquid holdups in the reflux drum and in the column base, known as the “secondary” outputs, are usually controlled by distillate flow rate D and the bottoms flow rate B .

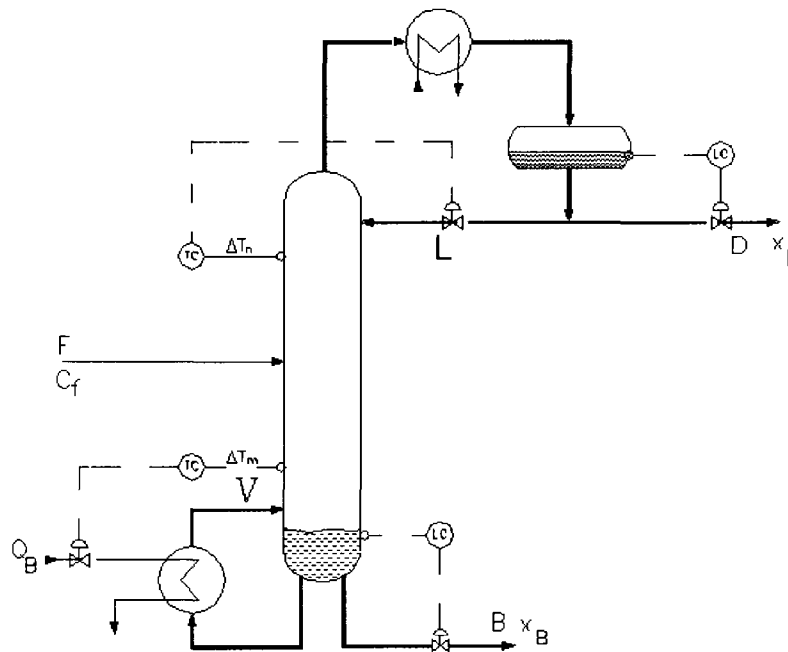


Figure A.4. Energy balance structure.

A.3.4. Material Balance Structure

Two other common control structures are the material balance structures $D-V$ and $L-B$. As shown in Figure A.5, the $D-V$ structure seems very similar to the $L-V$ structure. The only one difference between the $L-V$ and $D-V$ structures is that the roles of L and D are switched.

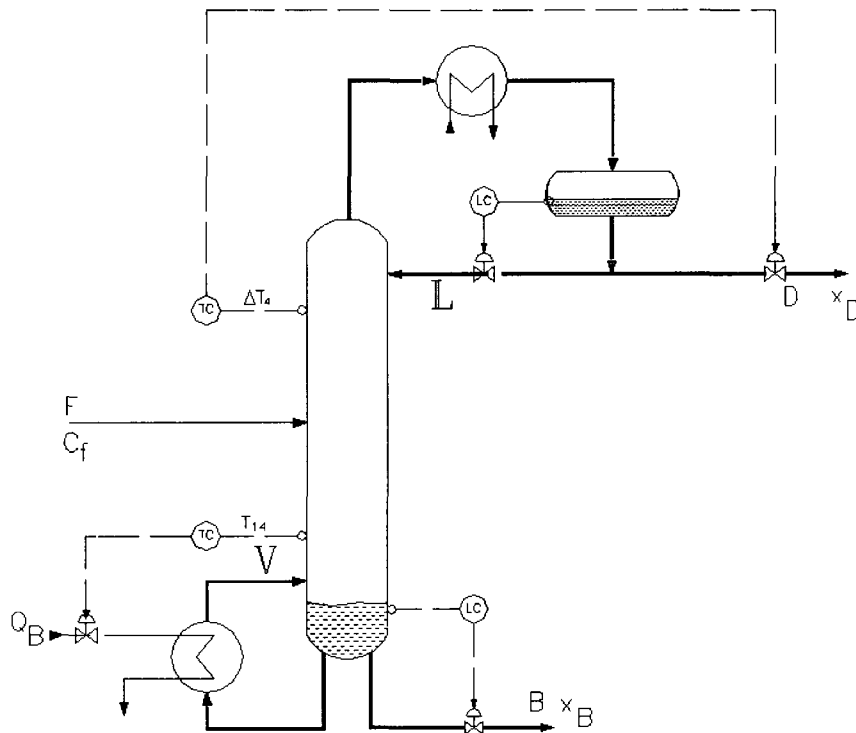


Figure A.5. $D-V$ control structure.

The $L-B$ structure is depicted in Figure A.6. There exists a possibility to occur an inverse response between reboiler liquid level and boilup flow, which causes difficulties for inventory control in the bottom section. This structure is very sensitive to disturbances in feed.

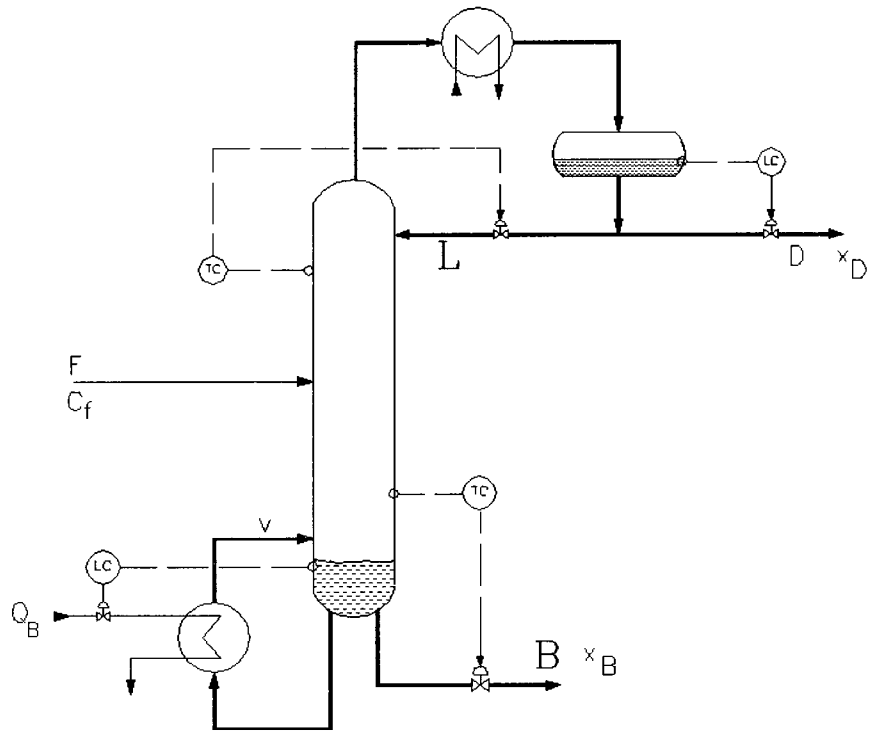


Figure A.6. L-B control structure.

APPENDIX B: PROCESS CALCULATION

B.1. Basic Engineering Data

The plant feed stock is condensate, whose actual composition always fluctuates around the average composition as shown in Table B.1 [44].

Table B.1. Condensate composition analyzed by gas chromatography.

Component	Mole %
Propane	0.01
Normal Butane	19.99
Isobutane	26.65
Isopentane	20.95
Normal Pentane	10.05
Hexane	7.26
Heptane	3.23
Octane	1.21
Nonane	0.0
Normal Decane	0.0
n-C ₁₁ H ₂₄	1.94
n-C ₁₂ H ₂₆	2.02
Cyclopentane	1.61
Methylclopentane	2.02
Benzene	1.61
Toluen	0.00
O-Xylene	0.00
E-Benzene	0.00

Table B.2. Distillation data.

Cut point (%)	TBP (°C)	ASTM D86 (°C)
0.00	-1.44	31.22
1.00	-0.80	31.63
2.00	1.61	32.94
5.00	10.56	37.72
7.50	18.02	40.29
10.00	24.67	45.29
15.00	29.57	47.84
20.00	31.58	48.86
25.00	33.59	49.89
30.00	35.99	51.09
35.00	39.12	52.92
40.00	43.94	55.83
45.00	50.00	59.64
50.00	58.42	65.19
55.00	66.23	70.38
60.00	69.51	72.55
65.00	70.77	73.34
70.00	75.91	76.68
75.00	86.06	84.11
80.00	98.63	94.20
85.00	100.57	95.91
90.00	115.54	109.54
95.00	131.07	124.024
98.00	148.30	140.20
99.00	159.91	146.78
100.00	168.02	156.75

Product specifications are given in Table B.3 [44].

Table B.3. Gasoline quality requirement.

Properties	Testing method	Requirements
1. Octane Number	ASTM D2699	min 87
2. Lead Content (g/l)	ASTM D3341	max 0.15
3. Distillation (deg C) :	ASTM D86	
IBP		-----
10% vol		max 70
50% vol		max 120
90% vol		max 190
FBP		max 210
Residue (% vol)		max 2.0
4. Corrosion. 3h/50°C	ASTM D130	max N-1
5. Existent Gum (mg/100ml)	ASTM D381	max 4.0
6. RVP @37.8 deg C (kPa)	ASTM D323	min 43
		max 80
7. Total sulfur content (%wt)	ASTM D1266	max 0.15
8. Oxidation stability (min)	ASTM D525	min 240
9. Density at 15 deg C (g/cm ³)	ASTM D1298	0.70 – 0.74

The plant's nominal capacity is 130,000 tons of raw condensate/year based on 24 operating hours per day and 350 working days per year. The plant capacity is quite low due to depending on upstream processes. The plant equipment is specified with a design margin of 10% above the nominal capacity and turndown ratio is 50% of the nominal capacity.

B.2. Distillation Process Calculation

B.2.1. Equilibrium Flash Vaporization Curves

According to W. C. Edmister method [45], the equilibrium flash vaporization (EFV) curve is converted from the data of true boiling point (TBP). For example, we perform calculation for the 50-percent point as follows:

$$t_{50\% \text{ (TBP)}} = 58.42 \text{ }^\circ\text{C}$$

$$t_{(30\% - 10\%) \text{ (TBP)}} = 35.99 - 24.67 = 11.32 \text{ }^\circ\text{C}.$$

Look up TBP-EFV chart, the temperature difference is determined as

$$t_{50\% \text{ (EFV-TBP)}} = 1.5 \text{ }^\circ\text{C}.$$

Therefore,

$$t_{50\% \text{ (EFV)}} = 58.42 + 1.5 = 59.62 \text{ }^\circ\text{C}.$$

Repeat the procedure above for all TBP temperatures to determine EFV (1 atm) temperatures. Then convert the EFV (1 atm) data into the EFV (4.6 atm) data by using the Cox chart [46]. The results are shown in Table B.4.

Table B.4. Relationship between ASTM, TBP, and EFV.

%vol.	TBP		EFV (1atm)		EFV (4.6 atm)
	t °C	Δt	Δt	t °C	t °C
I.B.P.	-1.44			41.62	93
		12	1.5		
5	10.56			43.12	95
		14.11	4		

10	24.67			47.12	102
		6.91	3		
20	31.58			50.12	106
		4.41	2.5		
30	35.99			52.62	110
		7.95	5		
40	43.93			57.62	116
		14.48	6		
50	58.42			63.62	125
		11.09	5.5		
60	69.51			69.12	132
		6.40	6.5		
70	75.91			75.62	141
		22.72	7.5		
80	98.63			83.12	150
		16.91	7		
90	115.54			90.12	158

B.2.2. Yield of Fractions

Based on the TBP data, the yield of fractions for the gasoline plant can be determined as shown in Figure B.1.

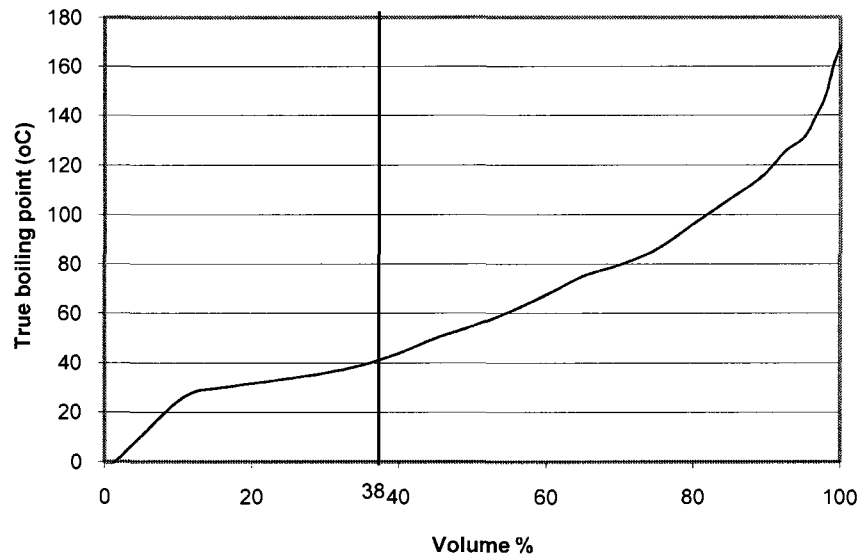


Figure B.1. Yield curve.

B.2.3. Operating Pressure

The column is designed 14 trays; and the pressure drop across each tray is estimated approximately 80 kPa. Therefore, the pressures at the feed section and the top section are 4.6 atm and 4 atm, respectively.

B.3. Calculation for the Feed Section

B.3.1. Description

The pre-heater rises the feed temperature towards the expected temperature at which the required phase equilibrium is established. As a result, the feed split specified by the yield curve is obtained.

The key parameters will be determined as follows:

- 1) Equilibrium phase flows into the feed section;

- 2) Material balance at the feed section;
- 3) The feed temperature.

B.3.2. Calculation

The feed has liquid-gas equilibrium with gas percentage of 38% volume. However, it is usually to deeply cut more 4% (the unexpected heavy component will be condensed and refluxed to the column). Thus there are two equilibrium phase flows: 1) vapor $V_F = 38+4 = 42(\%)$; and 2) liquid $L_F = 100 - 42 = 58(\%)$.

The phase equilibrium is depicted in Figure B.2.

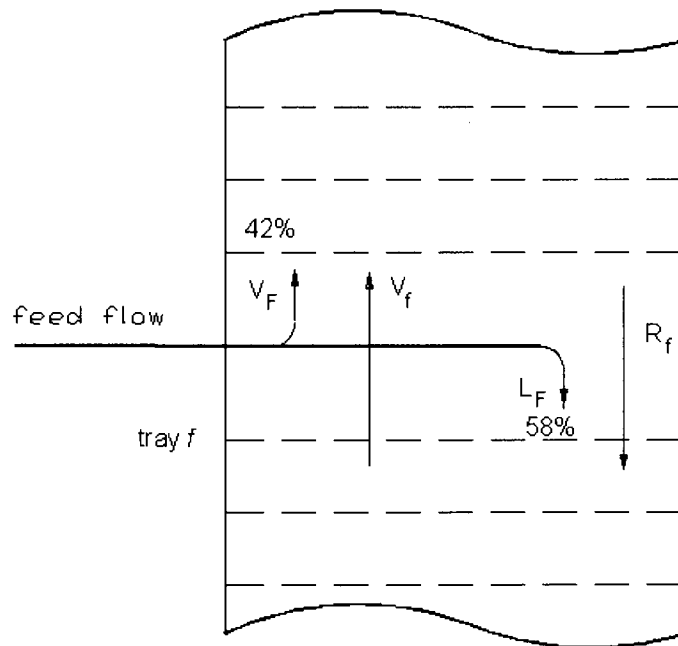


Figure B.2. Equilibrium phase flows at the feed section.

The heavy fraction flow L_F dissolved a small amount of light components is descending to the column bottom. These undesirable light components shall be caught by

the vapor flow V_f arising to the top column. The vapor flow V_f can be assigned as 28% vol.

The bottom product is determined by the yield curve as 62%vol; hence, the internal reflux across the feed section can be calculated:

$$R_f = B - L_F + V_f = 62 - 58 + 28 = 32\% \text{ vol.}$$

Look up the EFV curve (4.6 atm) of the feed section, the required feed temperature is 118°C corresponding to the point of 42% vol.

Table B.5. Material balances for the feed section.

Stream	Volume fraction vol%	Liquid flow rate m ³ /h	Liquid density ton/m ³	Mass flow rate ton/h
V_F	42	9.7015	0.591	5.7336
V_f	28	6.4677	0.598	3.8677
R_f	-32	-7.3916	0.615	-4.5458
Total light fractions $\sum S_D$	38	8.7775	0.577	5.0651
L_F	58	13.3973	0.726	9.7264
V_f	-28	-6.4677	0.598	-3.8677
R_f	32	7.3916	0.615	4.5458
Bottoms B	62	14.3213	0.727	10.4046

B.4. Calculation for the Stripping Section

B.4.1. Description

In the stripping section, liquid flows, which are descending from the feed section, include the equilibrium phase flow L_F and the internal reflux flow R_E . They are contacting with the arising vapor flow V_f for heat transfer and mass transfer. The result is

that all undesirable light components should be completely removed from the bottoms.

This process is accomplished with the aid of heat flow supplied by the reboiler.

The main parameters should be determined as follows:

- 1) The bottoms temperature;
- 2) The reboiler duty Q_B .

B.4.2. Calculation

The column base pressure is approximately the pressure at the feed section (4.6 bars) because the pressure drop across this section can be neglected.

The phase equilibrium is shown in Figure B.3.

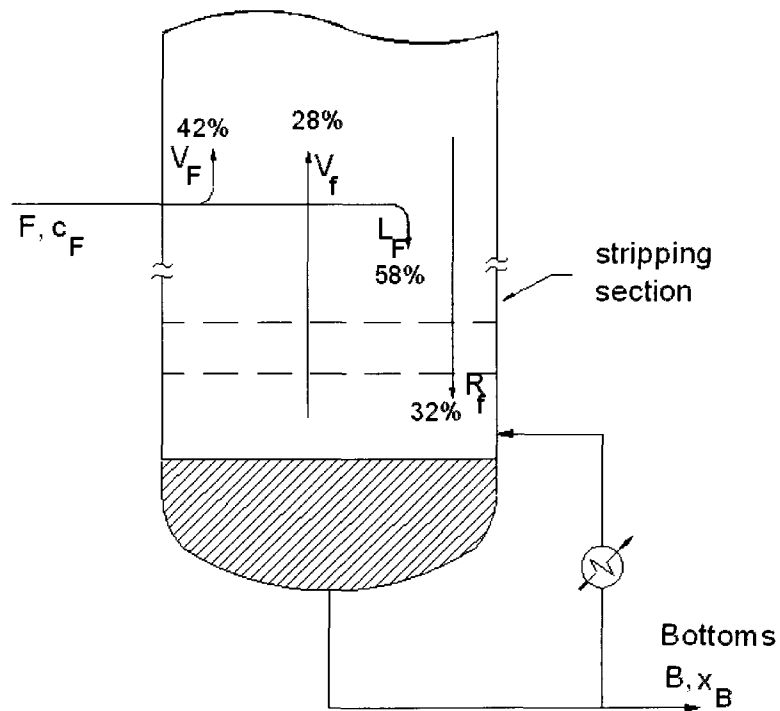


Figure B.3. Equilibrium phase flows at the stripping section.

Look up the EFV curve (1 atm) of the stripping section and the Cox chart, the equilibrium temperature at this section (4.6 atm) is 144 °C.

The reboiler duty is equal to the heat input, which generates boilup and increases the temperature of the stripping section by an increment of $144 - 118 = 26^\circ\text{C}$.

Table B.6. Material and energy balances of the stripping section.

INLET				
	ton/h	kcal/kg	kcal/h.10 ³	kJ/h.10 ³
L _f	9.73	68	661.40	2768.60
R _f	4.55	69	313.66	1313.00
Total	14.27		975.06	4081.61
OUTLET				
	ton/h	kcal/kg	kcal/h.10 ³	kJ/h.10 ³
V _f	3.87	165	638.16	2671.36
B	10.40	82	853.18	3571.41
Total	14.27		1491.34	6242.80

As a result, the reboiler duty is $Q_B = (6242.8 - 4081.61) 10^3 = 2161190 \text{ kJ/h}$.

B.5. Calculation for the Rectifying Section

B.5.1. Description

The overhead vapor flow, which includes V_F from the feed section and V_f from the stripping section, passes through the condenser (to remove heat) and then enter into the reflux drum. There exist two equilibrium phases: 1) liquid (butane); and 2) vapor (propane vapor and dry gas). The liquid from the reflux drum is partly pumped back into

the top tray as the reflux flow L and partly removed from the system as the distillate flow D .

The liquid is still dissolved a very small amount of light components. Therefore, the reflux flow whilst entering into the top tray will receive heat to vaporize completely all light component dissolved; and the liquid remained will be collected as the internal reflux flow.

B.5.2. Calculation

The top pressure is 4 atm due to pressure drop across the rectifying section.

The dew point of distillate is correspondingly the point 100% of the EFV curve of rectifying section. Based on the Cox chart, the top section temperature is determined as 46°C.

The equilibrium phase flows at the stripping section are display in Figure B.4.

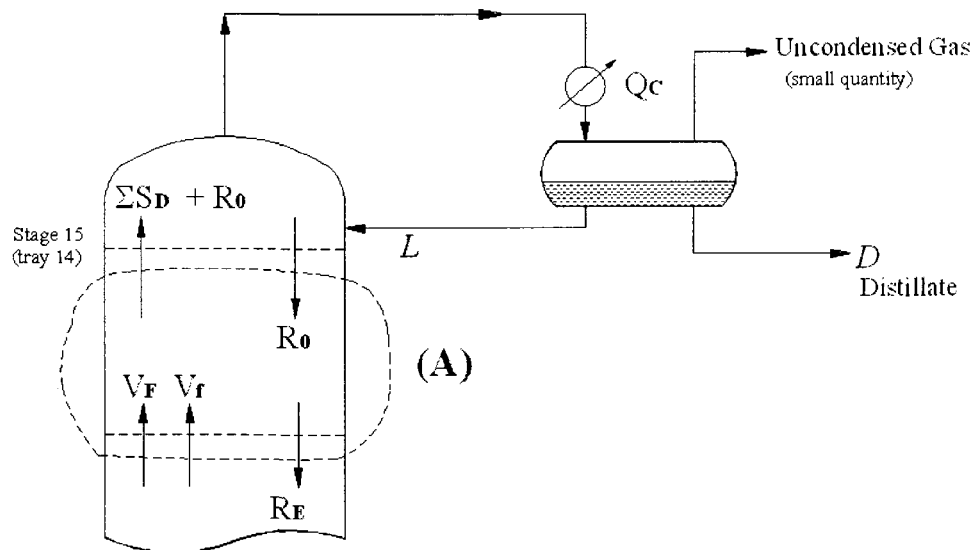


Figure B.4. Equilibrium phase flows at the rectifying section.

Table B.7. Material and energy balances around the boundary (A).

INLET				
	ton/h	kcal/kg	kcal/h.10 ³	kJ/h.10 ³
V _F +V _f	9.611	115	1105.3	4626.6
R ₀	R ₀	24	24 R ₀	100.5 R ₀
Total	9.611+ R ₀		1105.3+24 R ₀	4626.3+100.5 R ₀
OUTLET				
	ton/h	kcal/kg	kcal/h.10 ³	kJ/h.10 ³
ΣS _D +R ₀	5.065+R ₀	97	491.31+97R ₀	2056.65+406R ₀
R _f	4.546	16	72.73	304.46
Total	9.611+ R ₀		564.05+ 97R ₀	2361.11+406R ₀

Based on the energy balance, we find the solution of R_0 :

$$4626.3 + 100.5 R_0 = 2361.11 + 406 R_0$$

Therefore,

$$R_0 = 7.415 \text{ (ton/h)}.$$

We now calculate the (external) reflux flow L . Enthalpy data of the reflux flow L , looked up the experimental chart for petroleum's enthalpy, are corresponding to the liquid state of 40°C (liquid inlet at the top tray) and the vapor state of 46°C (vapor outlet at the column top).

$$L \text{ inlet at } 42^\circ\text{C: } H_{\text{liquid (inlet)}} = 22 \text{ kcal/kg}$$

$$L \text{ outlet at } 46^\circ\text{C: } H_{\text{vapor (outlet)}} = 106 \text{ kcal/kg}$$

We find the solution of the energy balance equation:

$$\Delta H_{R_0} \cdot R_0 = \Delta H_L \cdot L$$

$$(115-24)(7.42) = (106-22)L$$

Therefore,

$$L = 8.04 \text{ (ton/h)}.$$

B.6. Calculation Results

B.6.1. Raw Gasoline Property

The bottom product, named raw gasoline, is the major blend for manufacturing the finished gasoline. The distillation data of the raw gasoline is shown in Table B.8.

Table B.8. ASTM distillation curve of the raw gasoline.

% vol	Boiling point (°C)
0	41.68
5	44.51
10	46.23
20	48.10
30	50.14
40	55.72
50	66.24
60	72.13
70	80.98
80	95.07
90	122.81
100	161.34

B.6.2. Main Stream Property

The specification of the finished gasoline product is presented in Table B.9.

Table B.9. Main streams of the plant.

Stream	Condensate	LPG	Raw gasoline
Temperature (C)	118	46	144
Pressure (atm)	8.6	4.0	4.6
Density (kg/m ³)	670	585	727
Volume flow rate (m ³ /h)	227.6	8.78	21.88
Mass flow rate (kg/h)	15480	5061	10405
Mass flow rate (ton/year)	130000	43000	87000
Stream	Reformat	MTBE	Gasoline
Temperature (C)	30	30	30
RVP (kPa)	105	105	105
Volume flow rate (m ³ /h)	16.58	2.39	39.18
Density (kg/m ³)	789.8	746	752
Mass flow rate (kg/h)	12500	1800	29800
Mass flow rate (ton/year)	105000	15000	250000

B.7. Process Description

B.7.1. Simplified Process Flow Diagram

The simplified process flow diagram is shown in Figure B.5. The gasoline plant consists of many apparatuses and facilities, for instance, the distillation column, the mixer, and storage tanks.

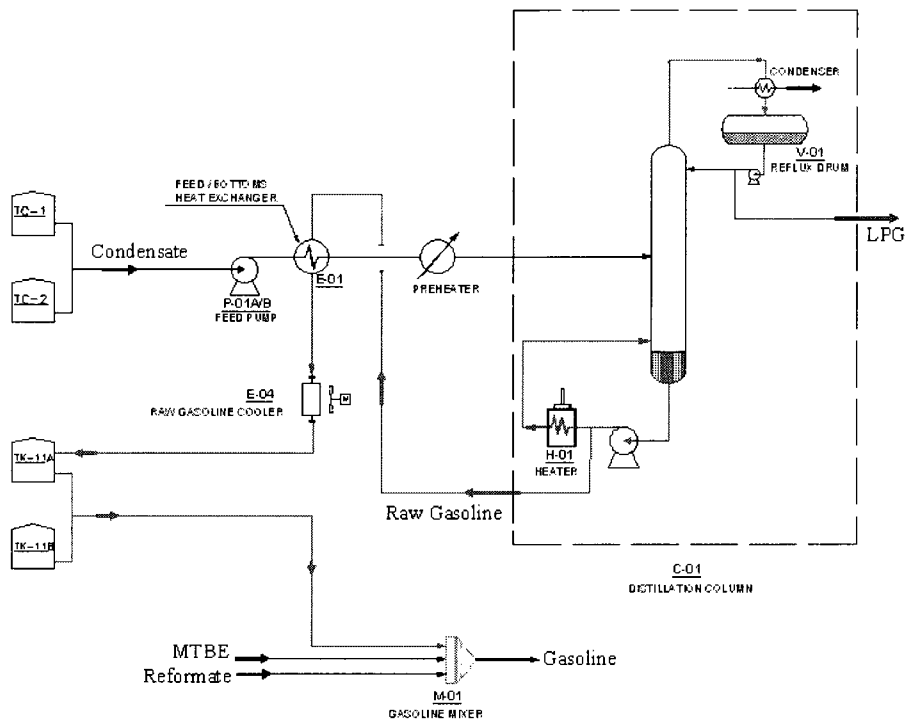


Figure B.5. Simplified process flow diagram of the gasoline plant.

B.7.2. Distillation Column

Condensate is fed using feed pumps (P-01 A/B) through the feed/bottoms heat exchanger (E-01) to the distillation column (C-01). The column (C-01) plays a very important role in the plant. Here, condensate after processing in the column (called naphtha or raw gasoline) is cut off the light fraction having a boiling point of less than 40°C. The raw gasoline then mixes with Reformate or MTBE to produce the finished gasoline.

The column has 24 actual trays (equivalent to 14 theoretical trays). Condensate is fed to the seventh tray and the raw gasoline is withdrawn off from the column base. The

operating pressure is 4.6 atm. The top temperature is 46°C; and the bottom temperature is 128°C.

The raw gasoline (naphtha) is sent to the raw gasoline tanks (TK-11 A/B). Its heat has been removed by the feed/bottoms heat exchanger and cooled by the raw gasoline cooler (E-04). A part of the bottom stream is heated up in the reboiler furnace (H-01) and returned to the distillation column to supply required heat for distillation.

The distillation column overhead vapor is cooled at the column overhead condenser (E-03) to produce the uncondensed gas, so called, off-gas, and the reflux liquid. The former is mainly burnt off at the reboiler furnace and the remaining amount for controlling the reflux drum pressure is burnt at flare. The latter accumulated at the reflux drum (V-01) is returned to the column at the top tray, under controlled flow rate, for maintaining stable operations and maximizing the recovery of naphtha.

B.7.3. Blending System and Product Distribution

The blending system consists of an in-line static mixer, an on-line multi-property analyzer, ratio control with DCS, and an off-line blend simulator.

The blending system will perform the following functions:

- 1) Continuous ratio control of blend header qualities to meet specification with minimum deviation from optimal recipe;
- 2) Continuous monitoring of blends using infra-red analyzers and tracking integrated blend quality;

- 3) Offline optimization of header quality control and recipe targets based on reconciled blend models and integrated blend results to optimally meet scheduler-specified blend order quality recipe and inventory targets.

Based on the quality requirements of gasoline as specified standard, the off-line simulator calculates precisely the necessary flow rate of octane booster to blend the whole volume of raw gasoline.

The other additives (detergent, color, anti-oxidation, and metal deactivator) are injected in a pre-determined amount directly into the raw gasoline stream just before the mixer.

Gasoline product blended at the gasoline mixer is analyzed with a multi-property analyzer and sent to the gasoline storage tanks (TK-13 A/B). In case of low quality, the gasoline will be pumped to off-spec storage tank and then returned to the mixer. The off-spec product tank is designed for 12 production hours.

From the gasoline storage tanks, gasoline after being checked the quality is pumped out the plant through the tank truck filling station or through the jetty to load in tankers.

B.7.4. Feed Control

The feed section has several components:

- 1) A local flow controller (FC-11) to control the feed flow rate;
- 2) A local controller for feed pumps;
- 3) A local pressure controller (PIC-13) to keep the pressure of the feed flow at 4.6 atm.

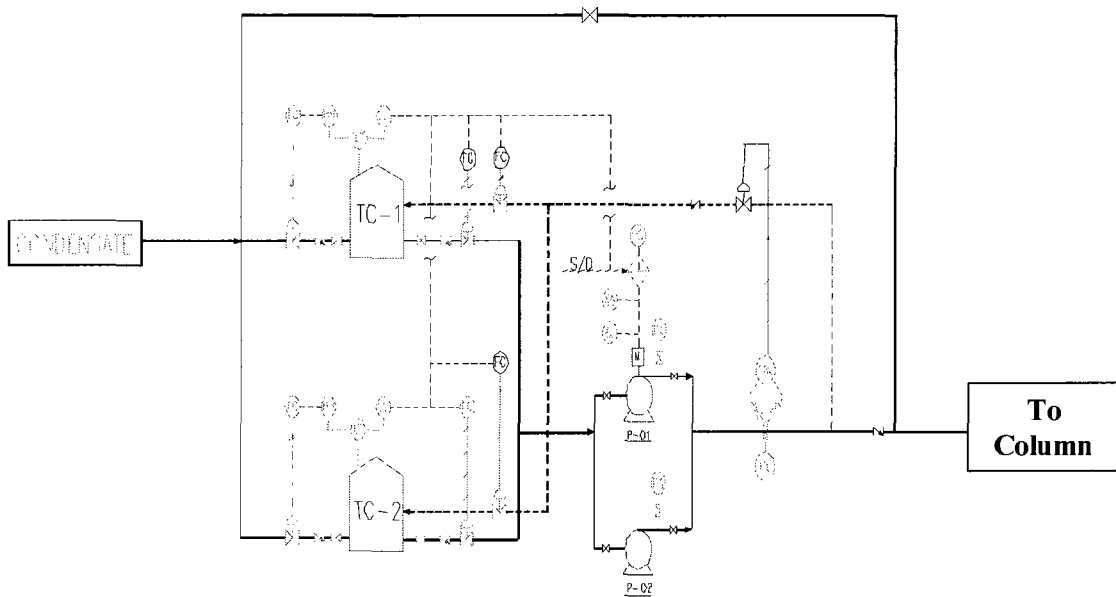


Figure B.6. A local control devices for feed pumps.

B.7.5. Top Column Section

Column pressure control includes: 1) vapor bypass line; and 2) vent line to flare. If there is a pressure drop in reflux drum (V-01), the local pressure controller commands to slightly open the control valve upstream (E-03) and gradually close the control valve in the vent line. If column pressure increases, the local pressure controller commands to gradually close the vapor bypass flow and open more flow to flare.

B.7.6. Bottom Section

The reboiler is a forced-circulation typed reboiler including pumps (P-08/09) and a heater (H-01). The discharge of the pump is split as two streams: 1) a part is withdrawn as raw gasoline; and 2) a part is heated up in the heater to generate vapor back to the column base. When the temperature in the bottom section is changed, the local temperature controller TIC-34 will adjust the rate of fuel gas into the heater.

APPENDIX C: MATHEMATICAL MODEL

C.1. Introduction

Distillation process is very complicated. So we develop its mathematical model to study its dynamics and then select appropriate control strategy. The mathematical model of distillation process is established on dynamic continuity equations of mass and energy for trays, condenser, reflux drum, and reboiler.

In general, the dynamic continuity equations state that the rate of accumulation of mass or energy in a system is equal to the mass or energy flows entered and generated, less the amount leaving and consumed within the system. The accumulation term is a first order time derivatives of the total mass or energy. The flow terms are algebraic. Therefore, the results are first order ordinary differential equations that are usually non-linear.

The liquid rates throughout the column will not be the same dynamically. They will depend on the fluid mechanics of the tray. Often a simple Francis weir formula relationship is used to relate the liquid holdup on a tray to the liquid flow rate L_n over the outlet weir:

$$M_n = f(L_n). \quad (C.1)$$

We now develop the state equations that will describe the dynamic behavior of a distillation column. The fundamental quantities are total mass and mass of the light component, which is the more volatile component.

C.2. Dynamic Study of Distillation Process

C.2.1. Generic Trays

A general tray is an n th stage such that $n = 1, 2, \dots, N$ and $n \neq f$.

The total mole holdup in the n th tray M_n is considered constant, but the imbalance in the input and output flows is accounted for in the component and heat balance equations:

$$\frac{d(M_n)}{dt} = L_{n+1} - L_n + V_{n-1} - V_n = L_{n+1} - L_n. \quad (\text{C.2})$$

Table C.1. Parameters of a generic tray.

	Phase	Flow rate	Concentration
INLET	Liquid	L_{n+1}	x_{n+1}
	Vapor	V_{n-1}	y_{n-1}
OULET	Vapor	V_n	y_n
	Liquid	L_n	x_n

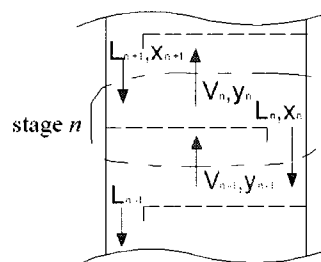


Figure C.1. A generic tray.

The rate of change of holdup in the n th tray results in the change of exit liquid flow after a hydraulic lag [38] or hydraulic time constant [35]:

$$\frac{dL_n}{dt} = \frac{1}{\tau} \frac{dM_n}{dt} \quad (\text{C.3})$$

where τ is hydraulic time constant.

The hydraulic lag can be treated as a liquid level problem, which is, complicated somewhat by the change in level across the plate. At the center of a tray, the average depth of clear liquid is usually less than at the either end and may even be less than the weir height, as shown in Figure C.2.

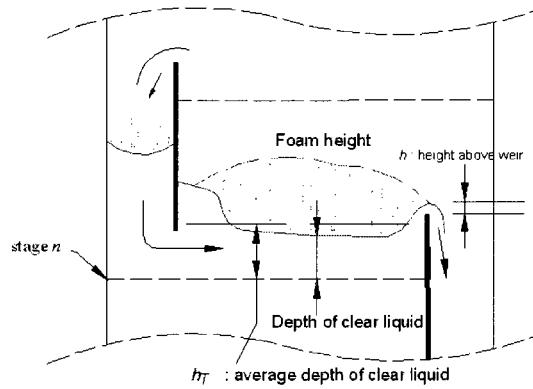


Figure C.2. Variation of liquid depth across a generic tray.

The hydraulic time constant can be calculated with the formula [38]:

$$\tau = A \frac{dh_f}{dL_{out}} = \frac{dM_n}{dL_n} \quad (\text{C.4})$$

where τ is hydraulic lag for 1 plate; M_n is holding of liquid per plate; and L_n is liquid rate.

Component balance is given by

$$\frac{d(M_n x_n)}{dt} = L_{n+1} x_{n+1} + V_{n-1} y_{n-1} - L_n x_n - V_n y_n. \quad (\text{C.5})$$

By differentiating and substituting for the $\frac{dM_n}{dL_n}$ term, we obtain:

$$\frac{dx_n}{dt} = \frac{L_{n+1}x_{n+1} + V_{n-1}y_{n-1} - (L_{n+1} + V_{n-1})x_n - V_n(y_n - x_n)}{M_n} \quad (\text{C.6})$$

Energy balance is given by

$$\frac{d(M_n h_n)}{dt} = h_{n+1}L_{n+1} + H_{n-1}V_{n-1} - h_n L_n - H_n V_n \quad (\text{C.7})$$

or,

$$M_n \frac{dh_n}{dt} + h_n \frac{dM_n}{dt} = h_{n+1}L_{n+1} + H_{n-1}V_{n-1} - h_n L_n - H_n V_n \quad (\text{C.8})$$

Because the $\frac{dh_n}{dt}$ term is approximately zero, substituting for the change of the

holdup $\frac{dM_n}{dt}$ term, we obtain:

$$V_n = \frac{h_{n+1}L_{n+1} + H_{n-1}V_{n-1} - (L_{n+1} + V_{n-1})h_n}{H_n - h_n} \quad (\text{C.9})$$

C.2.2. Feed Tray

The feed section includes the feed tray, as shown in Figure C.3.

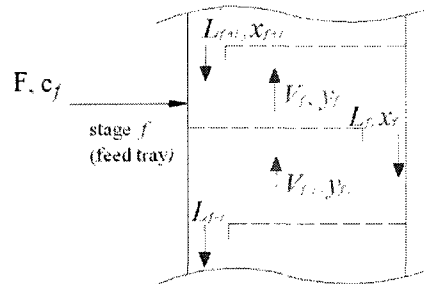


Figure C.3. Feed section.

Total mass balance is given by

$$\frac{d(M_f)}{dt} = F + L_{f+1} + V_{f-1} - L_f - V_f = F + L_{f+1} - L_f. \quad (\text{C.10})$$

Component balance is given by the following equations:

$$\frac{d(M_f x_f)}{dt} = F c_f + L_{f+1} x_{f+1} + V_{f-1} y_{f-1} - L_f x_f - V_f y_f$$

or,

$$\frac{dx_n}{dt} = \frac{L_{n+1} x_{n+1} + V_{n-1} y_{n-1} - (L_{n+1} + V_{n-1}) x_n - V_n (y_n - x_n)}{M_n}. \quad (\text{C.11})$$

Energy balance is given by

$$\frac{d(M_f h_f)}{dt} = h_f F + h_{n+1} L_{n+1} + H_{n-1} V_{n-1} - h_n L_n - H_n V_n$$

or,

$$V_n = \frac{h_f F + h_{n+1} L_{n+1} + H_{n-1} V_{n-1} - (L_{n+1} + V_{n-1}) h_n}{H_n - h_n}. \quad (\text{C.12})$$

C.2.3. Top Section

The top section consists of the top tray and the reflux drum, as shown in Figure

C.4.

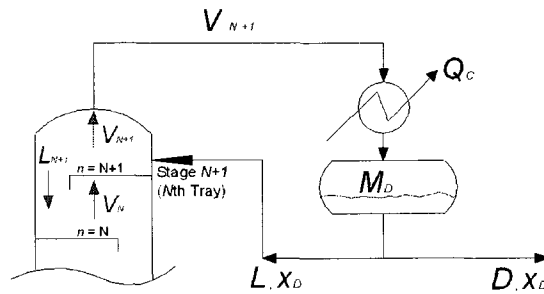


Figure C.4. Top section.

Total mass balance of the top tray is determined as

$$\frac{d(M_{N+1})}{dt} = L + V_N - L_{N+1} - V_{N+1}. \quad (\text{C.13})$$

Component balance of the top tray is characterized as

$$\frac{d(M_{N+1}x_{N+1})}{dt} = Lx_D + V_N y_N - L_{N+1} x_{N+1} - V_{N+1} y_{N+1}. \quad (\text{C.14})$$

Energy balance for the top tray is given by the following equations:

$$\frac{d(M_{N+1}h_{N+1})}{dt} = h_D L + H_N V_N - h_{N+1} L_{N+1} - H_{N+1} V_{N+1} \quad (\text{C.15})$$

or,

$$V_{N+1} = \frac{h_D L + H_N V_N - (L + V_N) h_{N+1}}{H_{N+1} - h_{N+1}}. \quad (\text{C.16})$$

Total mass balance for the reflux drum and condenser is given by

$$\frac{d(M_D)}{dt} = V_{N+1} - L - D. \quad (\text{C.17})$$

Component balance for the reflux drum and condenser is determined as

$$\frac{d(M_D x_D)}{dt} = V_{N+1} y_{N+1} - (L + D) x_D. \quad (\text{C.18})$$

We now define energy balance around condenser. The condenser duty Q_C is equal to the latent heat required to condense the overhead vapor to its bubble point:

$$Q_C = H_m V_m - h_{out} L_{out} = V_N (H_N - h_N). \quad (\text{C.19})$$

C.2.4. Bottom Section

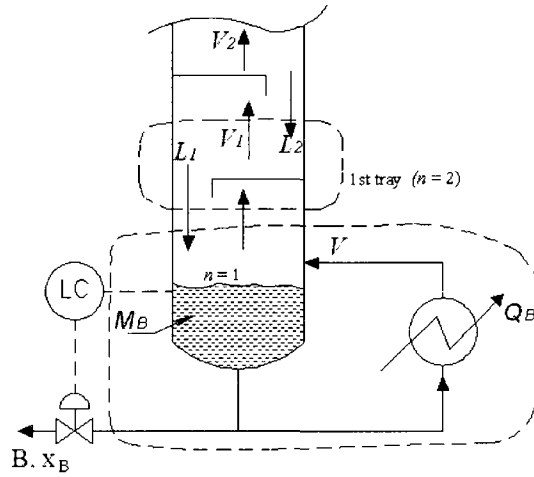


Figure C.5. Bottom section.

Total mass balance for the bottom tray is given by

$$\frac{d(M_2)}{dt} = L_3 - L_2 + V_B - V_2. \quad (\text{C.20})$$

Component balance for the bottom tray is given by

$$\frac{d(M_2 x_2)}{dt} = L_3 x_3 + V_B y_B - L_2 x_2 - V_2 y_2. \quad (\text{C.21})$$

Energy balance for the bottom tray is as follows:

$$\frac{d(M_B h_B)}{dt} = h_3 L_3 + H_B V_B - h_2 L_2 - H_2 V_2.$$

Therefore,

$$V_2 = \frac{h_3 L_3 + H_B V_B - (L_3 + V_B) h_2}{H_2 - h_2}. \quad (\text{C.22})$$

The base of the column has some particular characteristics as follows:

- 1) There is a reboiler heat flux Q_B to produce the boilup vapor flow V_B .

- 2) The holdup is a sensitive variable; hence, changes in sensible heat cannot be neglected.
- 3) The outflow of liquid from the bottoms B is determined externally.

Total mass balance for the column base is given by the following equation:

$$\frac{d(M_B)}{dt} = L_2 - V_B - B. \quad (\text{C.23})$$

Component balance for the column base is determined as

$$\frac{d(M_B x_B)}{dt} = L_2 x_2 - V_B y_B - B x_B. \quad (\text{C.24})$$

Energy balance for the column base is as follows:

$$\frac{d(M_B h_B)}{dt} = h_2 L_2 + Q_B - h_B B - H_B V_B \quad (\text{C.25})$$

or,

$$V_B = \frac{h_2 L_2 + Q_B - h_B B - M_B \frac{dh_B}{dt} - h_B \frac{dM_B}{dt}}{H_B}. \quad (\text{C.26})$$

All the equations above are state equations and describe the dynamic behavior of the distillation column. The state variables of the model are:

- 1) Liquid hold ups $M_1, M_2, \dots, M_f, \dots, M_{N+2}$;
- 2) Liquid concentrations $x_1, x_2, \dots, x_f, \dots, x_{N+1}$.

When all the equations above are resolved, we find how the flow rate and concentrations of the two product streams (distillate product, bottoms product) change with time, in the presence of changes in the various input variables.

C.3. Mathematical Model of Distillation Process

C.4. Simplified Model

To simplify the model, we make the following assumptions [35]:

- 1) The relative volatility α is constant throughout the column.
- 2) The vapor – liquid equilibrium relationship can be expressed by

$$y_n = \frac{\alpha x_n}{1 + (\alpha - 1)x_n} \quad (\text{C.27})$$

where x_n is the liquid composition on n th stage; y_n is the vapor composition on n th stage; and α is the relative volatility.

- 3) The overhead vapor is totally condensed in the condenser.
- 4) The liquid holdups on each tray, condenser, and the reboiler are constant and perfectly mixed (i.e., the same immediate liquid response, $dL_2 = dL_3 = \dots = dL_{N+2} = dL$).
- 5) The holdup of vapor is negligible throughout the system (i.e., the same immediate vapor response, $dV_1 = dV_2 = \dots = dV_{N+1} = dV$).
- 6) The molar flow rates of the vapor and liquid through the stripping and rectifying sections are constant:

$$V_1 = V_2 = \dots = V_{N+1};$$

$$L_2 = L_3 = \dots = L_{N+2}.$$

- 7) The column is numbered from bottom, e.g., $n = 1$ for reboiler, $n = 2$ for first tray, $n = f$ for feed tray, $n = N+1$ for top tray, and $n = N+2$ for condenser.

Under these assumptions, the dynamic model can be expressed by the following equations [12]:

1) Condenser ($n = N+2$):

$$M_D \dot{x}_n = (V + V_F)y_{n-1} - Lx_n - Dx_n. \quad (\text{C.28})$$

2) Tray n ($n = f+2, \dots, N+1$):

$$M \dot{x}_n = (V + V_F)(y_{n-1} - y_n) + L(x_{n+1} - x_n). \quad (\text{C.29})$$

3) Above feed location ($n = f+1$):

$$M \dot{x}_n = (V + V_F)(y_{n-1} - y_n) + L(x_{n+1} - x_n) + V_F y_F. \quad (\text{C.30})$$

4) Below feed location ($n = f$):

$$M \dot{x}_n = (V + V_F)(y_{n-1} - y_n) + L(x_{n+1} - x_n) + L_F x_F. \quad (\text{C.31})$$

5) Tray n ($n = 2, \dots, f-1$):

$$M \dot{x}_n = V(y_{n-1} - y_n) + (L + L_F)(x_{n+1} - x_n). \quad (\text{C.32})$$

6) Reboiler ($n=1$):

$$M_B \dot{x}_1 = (L + L_F)x_2 - Vy_1 - Bx_1. \quad (\text{C.33})$$

Flow rate are assumed as constant molar flows:

1) $L_F = q_F F$;

2) $V_F = F - L_F$;

3) $D = V_N - L = V + V_F - L$ (assuming condenser holdup constant);

4) $B = L_2 - V_1 = L + L_F - V$ (assuming boiler holdup constant);

Composition x_F and y_F in the liquid and vapor phase of the feed are obtained by solving the flash equations:

$$Fc_F = L_F x_F + V_F y_F \quad (\text{C.34})$$

$$y_F = \frac{\alpha x_F}{1 + (\alpha - 1)x_F} \quad (\text{C.35})$$

Although the model order is reduced, the representation of the distillation system is still nonlinear due to the vapor liquid equilibrium relationship between y_n and x_n in (C.27).

C.5. Mathematical Model of the Gasoline Refinery

In this section, we use some data obtained by process calculation described in Appendix B to plug in generic equations above. As the result, the mathematical model of the plant is completely defined.

C.5.1. Relative Volatility

Using the formula $\alpha_{ij} = K_i/K_j$ and looking up data in the handbook [39] for the operating range of temperature and pressure, the relative volatility is estimated as $\alpha = 5.68$.

C.5.2. Latent Heat and Boilup

The heat input of Q_B (reboiler duty) to the reboiler is to increase the temperature of stripping section and generate boilup V_0 [40]:

$$V_0 = \frac{Q_B - Bc_B(t_B - t_F)}{\lambda} \quad (\text{C.36})$$

where λ is the latent heat or heat of vaporization; B is the flow rate of bottom product (kg); c_B is the specific heat capacity (kJ/kg. $^{\circ}$ C); t_F is the inlet temperature ($^{\circ}$ C); and t_B is the outlet temperature (bottoms temperature, $^{\circ}$ C).

The latent heat at any temperature is described in terms of the latent heat at the normal boiling point [40]:

$$\lambda = \gamma \lambda_B \frac{T}{T_B} \quad (\text{C.37})$$

where L is the latent heat at absolute temperature T ($^{\circ}\text{R}$); L_B is the latent heat at absolute normal boiling point T_B ($^{\circ}\text{R}$); and γ is the correction factor obtained from the empirical chart.

The calculation results are summarized as follows:

$$\lambda = 730 \text{ (kJ/kg);}$$

$$V_0 = 3909.8 \text{ (kg/h) or } 66.8871 \text{ (kmole/h).}$$

The average vapor flow rate arising in the stripping section is calculated as

$$V = \frac{V_0 + V_f}{2} = 66.3407 \text{ (kmole/h).}$$

C.5.3. Liquid Holdups on Tray and Column Base

Liquid holdups are calculated with the methods proposed by McCabe [42] and Joshi [43].

Velocity of vapor phase arising in the column:

$$\omega_n = C \sqrt{\frac{\rho_L - \rho_G}{\rho_G}} \quad (\text{C.38})$$

where ρ_L is the density of liquid phase; ρ_V is the density of vapor phase; and C is a correction factor depending flow rates.

The actual velocity ω is normally selected that $\omega = (0.80 \div 0.85)\omega_n$ for paraffinic vapor.

The diameter of the column is calculated with the following formula:

$$D_k = \sqrt{\frac{4V_m}{3600\pi\omega}} \quad (\text{C.39})$$

where V_m is the mean flow of vapor in the column.

The height of the column is calculated on distance of trays. The tray distance is selected on basis of the column diameter.

The holdup in the column base is given by

$$M_B = \frac{\pi H_{NB} D_k^2 d_B}{4 W_B} \quad (\text{C.40})$$

where H_{NB} is normal liquid level in the column base (m); W_B is molar weight of the bottom product (kg/kmole); and d_B is density of the bottom product (kg/m³).

As a result, the holdup in the column base is calculated as

$$M_B = \frac{3.14(1.75)(1.4)^2}{4} \frac{726.5}{78.6} = 24.88 \text{ (kmole).}$$

The holdup on each tray is given by

$$M = \frac{0.95\pi h_T D_k^2 d_T}{4 W_T}$$

where h_T is average depth of clear liquid on a tray; W_T is molar weight of the liquid holdup on a tray; and d_T is the mean density of the liquid holdup on a tray.

Therefore, the holdup on each tray is calculated as

$$M = \frac{0.95(3.14)(0.28)(1.75)^2}{4} \frac{680}{75} = 5.80 \text{ (kmole).}$$

C.5.4. Liquid Holdup in Reflux Drum

The retention time of distillate in the reflux drum is selected as 5 minutes.

Liquid holdup M_D is equal to the quantity of distillate contained in the reflux drum:

$$M_D = \frac{5(L + D)}{60} \quad (\text{C.41})$$

where M_D is holdup in the reflux drum; L is the reflux flow rate; and D is the distillate flow rate.

As the result, the liquid holdup in reflux drum is calculated as

$$M_D = \frac{5(75.30 + 82.15)}{60} = 14.03 \text{ (kmole).}$$

C.6. Basic Mathematical Model of the Plant

Material balances for change in holdup of light component on each tray are as follows:

$$\text{Condenser } (n = 16): \quad M_D \dot{x}_{16} = (V + V_F)y_{15} - Lx_{16} - Dx_{16}$$

$$\text{Tray 14 } (n = 15): \quad M\dot{x}_{15} = (V + V_F)(y_{14} - y_{15}) + L(x_{16} - x_{15})$$

$$\text{Tray 13 } (n = 14): \quad M\dot{x}_{14} = (V + V_F)(y_{13} - y_{14}) + L(x_{15} - x_{14})$$

$$\text{Tray 12 } (n = 13): \quad M\dot{x}_{13} = (V + V_F)(y_{12} - y_{13}) + L(x_{14} - x_{13})$$

$$\text{Tray 11 } (n = 12): \quad M\dot{x}_{12} = (V + V_F)(y_{11} - y_{12}) + L(x_{13} - x_{12})$$

$$\text{Tray 10 } (n = 11): \quad M\dot{x}_{11} = (V + V_F)(y_{10} - y_{11}) + L(x_{12} - x_{11})$$

$$\text{Tray 9 } (n = 10): \quad M\dot{x}_{10} = (V + V_F)(y_9 - y_{10}) + L(x_{11} - x_{10})$$

$$\text{Tray 8 } (n = 9): \quad M\dot{x}_9 = V_F y_F + V y_8 - (V + V_F)y_9 + L(x_{10} - x_9)$$

$$\text{Tray 7 } (n = 8): \quad M\dot{x}_8 = V(y_7 - y_8) + Lx_9 + L_F x_F - (L + L_F)x_8$$

$$\text{Tray 6 } (n = 7): \quad M\dot{x}_7 = V(y_6 - y_7) + (L + L_F)(x_8 - x_7)$$

$$\text{Tray 5 } (n = 6): \quad M\dot{x}_6 = V(y_5 - y_6) + (L + L_F)(x_7 - x_6)$$

$$\text{Tray 4 } (n = 5): \quad M\dot{x}_5 = V(y_4 - y_5) + (L + L_F)(x_6 - x_5)$$

$$\text{Tray 3 } (n = 4): \quad M\dot{x}_4 = V(y_3 - y_4) + (L + L_F)(x_5 - x_4)$$

$$\text{Tray 2 } (n = 3): \quad M\dot{x}_3 = V(y_2 - y_3) + (L + L_F)(x_4 - x_3)$$

$$\text{Tray 1 } (n = 2): \quad M\dot{x}_2 = V(y_1 - y_2) + (L + L_F)(x_3 - x_2)$$

$$\text{Reboiler } (n = 1): \quad M_B \dot{x}_1 = (L + L_F)x_2 - Vy_1 - Bx_1.$$

Process data are summarized as follows:

- The liquid holdups: $M_D = 14.03$ (kmole), $M = 5.80$ (kmole), and $M_B = 24.88$ (kmole);
- Feed flow rates: $L_F = 104.2491$ (kmole/h) and $V_F = 98.5152$ (kmole/h);
- Flow rates above the feed location: $L_9 = \dots = L_{15} = L = 75.6380$ (kmole/h) and $V_9 = \dots = V_{15} = V + V_F = 66.1139 + 98.5152 = 164.6291$ (kmole/h);
- Flow rates below the feed location: $L_1 = \dots = L_8 = L + L_F = 75.6380 + 104.2491 = 179.8871$ (kmole/h) and $V_1 = \dots = V_8 = V = 66.1139$ (kmole/h);
- Distillate flow rate: $D = 92.7597$ (kmole/h);
- Bottoms flow rate: $B = 110.9235$ (kmole/h);
- Solving flash equations: $x_F = 0.2609$ and $y_F = 0.6672$.

In summary, the dynamic model is represented by a set of 31 nonlinear differential and algebraic equations:

$$\begin{aligned}
14.03 \dot{x}_{16} &= 164.6291 y_{15} - 75.6380 x_{16} - 92.7597 x_{16} \\
5.8 \dot{x}_{15} &= 164.6291(y_{14} - y_{15}) + 75.6380(x_{16} - x_{15}) \\
5.8 \dot{x}_{14} &= 164.6291(y_{13} - y_{14}) + 75.6380(x_{15} - x_{14}) \\
5.8 \dot{x}_{13} &= 164.6291(y_{12} - y_{13}) + 75.6380(x_{14} - x_{13}) \\
5.8 \dot{x}_{12} &= 164.6291(y_{11} - y_{12}) + 75.6380(x_{13} - x_{12}) \\
5.8 \dot{x}_{11} &= 164.6291(y_{10} - y_{11}) + 75.6380(x_{12} - x_{11}) \\
5.8 \dot{x}_{10} &= 164.6291(y_9 - y_{10}) + 75.6380(x_{11} - x_{10}) \\
5.8 \dot{x}_9 &= 66.1139 y_8 - 156.38 y_9 + 75.6380(x_{10} - x_9) + 59.95 \\
5.8 \dot{x}_8 &= 66.1139(y_7 - y_8) + 75.6380 x_9 - 188.59 x_8 + 33.99 \\
5.8 \dot{x}_7 &= 66.1139(y_6 - y_7) + 179.8871(x_8 - x_7) \\
5.8 \dot{x}_6 &= 66.1139(y_5 - y_6) + 179.8871(x_7 - x_6) \\
5.8 \dot{x}_5 &= 66.1139(y_4 - y_5) + 179.8871(x_6 - x_5) \\
5.8 \dot{x}_4 &= 66.1139(y_3 - y_4) + 179.8871(x_5 - x_4) \\
5.8 \dot{x}_3 &= 66.1139(y_2 - y_3) + 179.8871(x_4 - x_3) \\
5.8 \dot{x}_2 &= 66.1139(y_1 - y_2) + 179.8871(x_3 - x_2) \\
24.88 \dot{x}_1 &= 179.8871 x_2 - 110.9235 x_1 - 66.1139 y_1.
\end{aligned} \tag{C.42}$$

Vapor liquid equilibrium (VLE) relationship on each tray is given as

$$y_1 = \frac{5.68 x_1}{1 + 4.68 x_1}$$

$$y_2 = \frac{5.68x_2}{1 + 4.68x_2}$$

⋮

$$y_{15} = \frac{5.68x_{15}}{1 + 4.68x_{15}}.$$

(C.43)

APPENDIX D: DYNAMIC SIMULATION

D.1. Modular Decomposition of the Column

Modular decomposition of the column is depicted in Figure D.1. The column is divided into two groups: 1) rectifying section; and 2) stripping section.

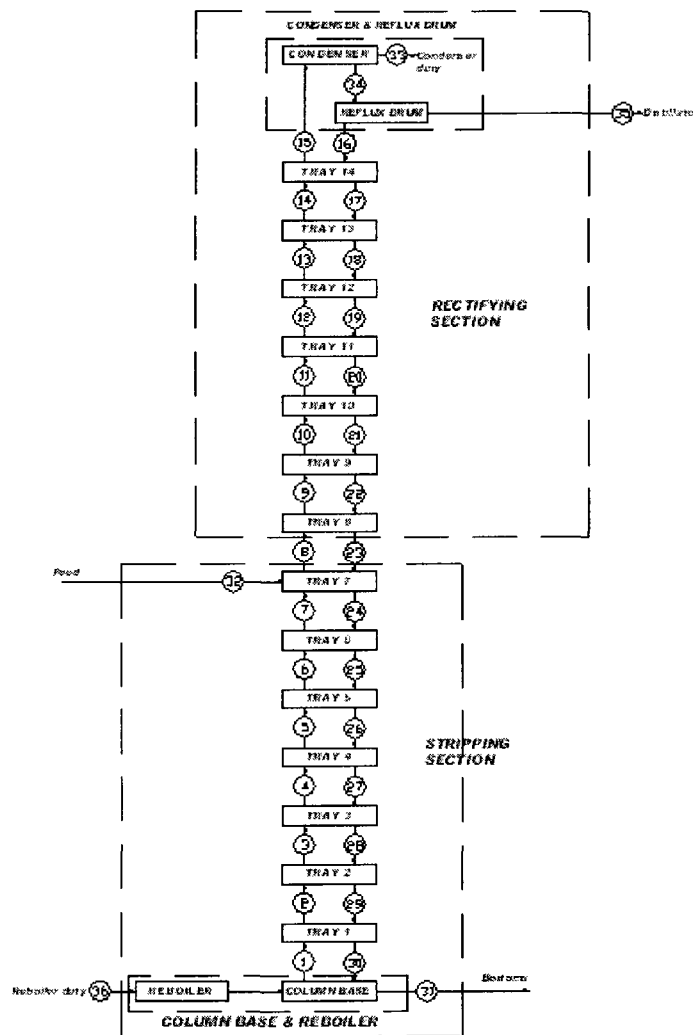


Figure D.1. Modular decomposition scheme for the distillation column.

D.2. Simulation with MATLAB Simulink

For convenience, the simulation program is organized as a hierarchical structure with three levels, as depicted in Figure D.2. The lowest-level modules actually represent differential and algebraic equations.

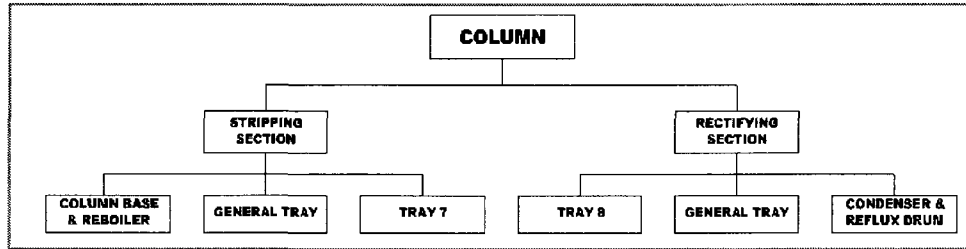


Figure D.2. Hierarchical structure of the simulation program.

The highest level of the simulation program in MATLAB Simulink is shown in Figure D.3.

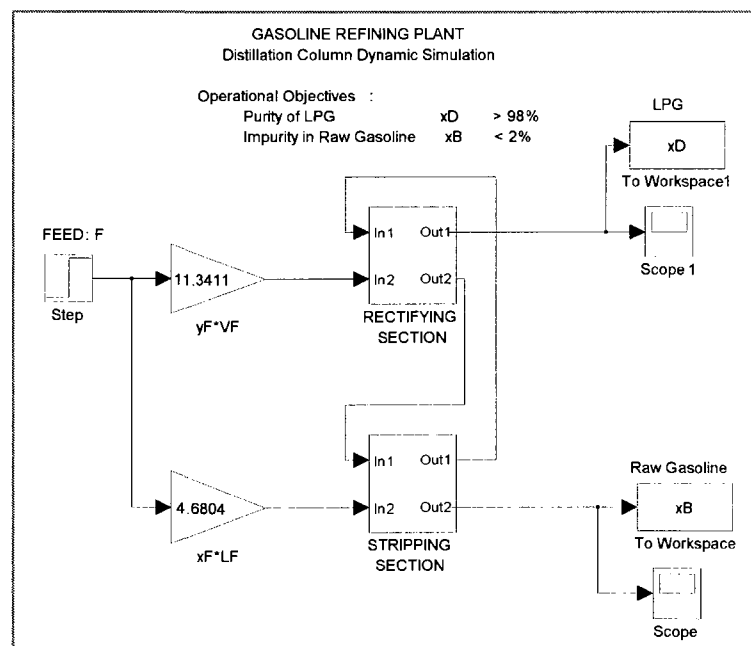


Figure D.3. Main program in MATLAB Simulink.

The second-level modules include the rectifying section, as shown in Figure D.4, and the stripping section, as shown in Figure D.5.

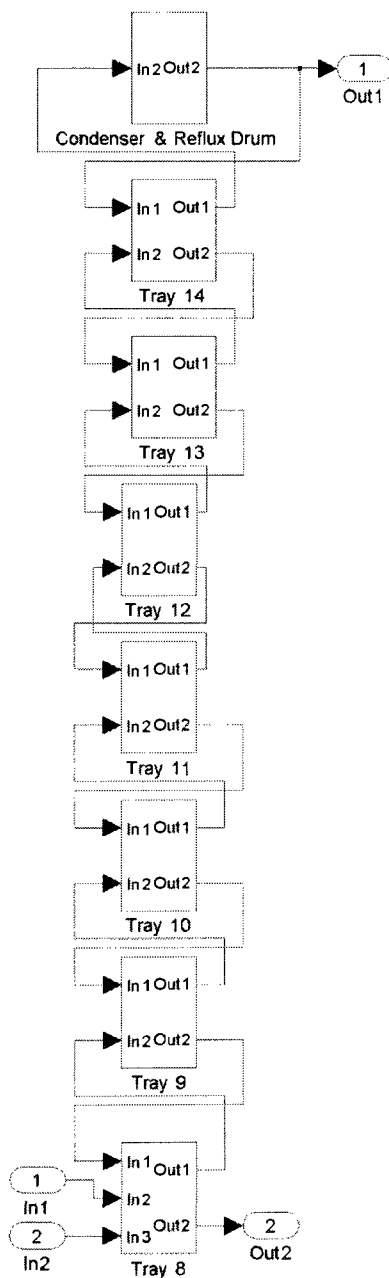


Figure D.4. Module of the rectifying section.

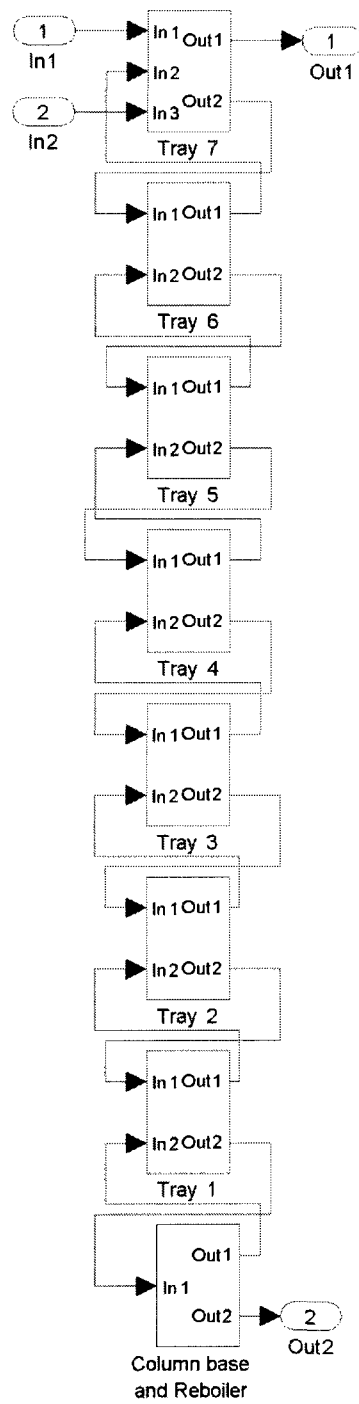


Figure D.5. Module of the stripping section.

The third-level modules include some special components and generic trays as depicted in the following figures.

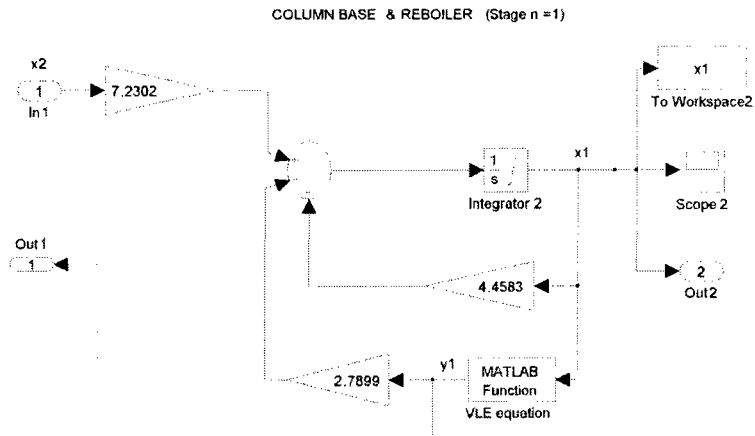


Figure D.6. Module of the column base and reboiler.

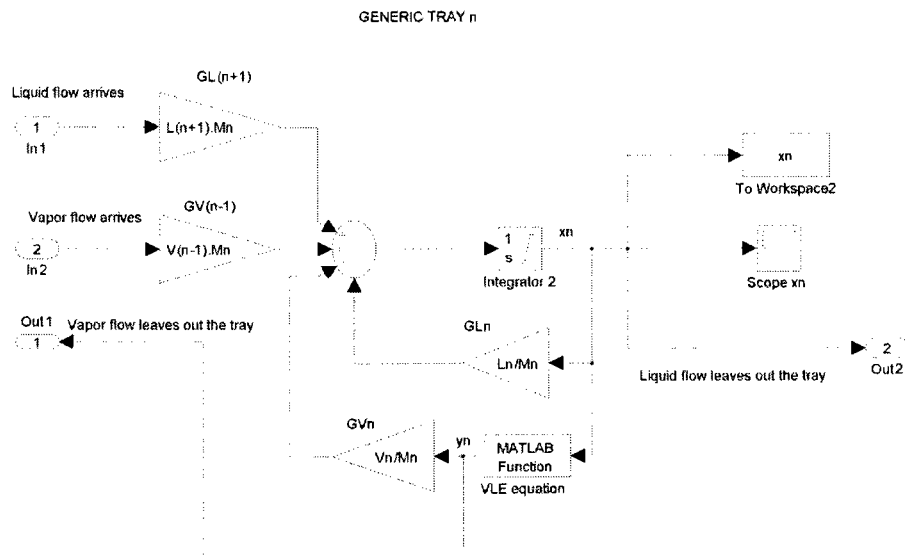


Figure D.7. Module of a generic tray.

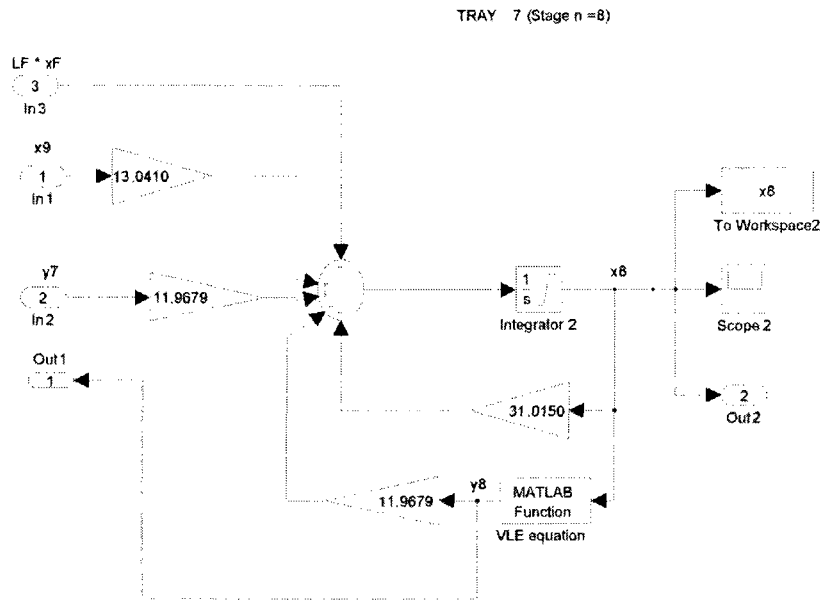


Figure D.8. Module of the feed tray.

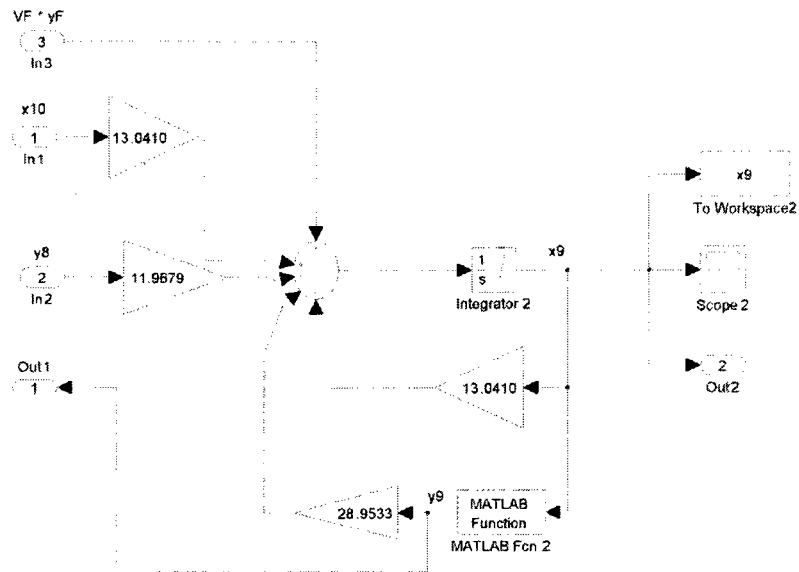


Figure D.9. Module of the eighth tray.

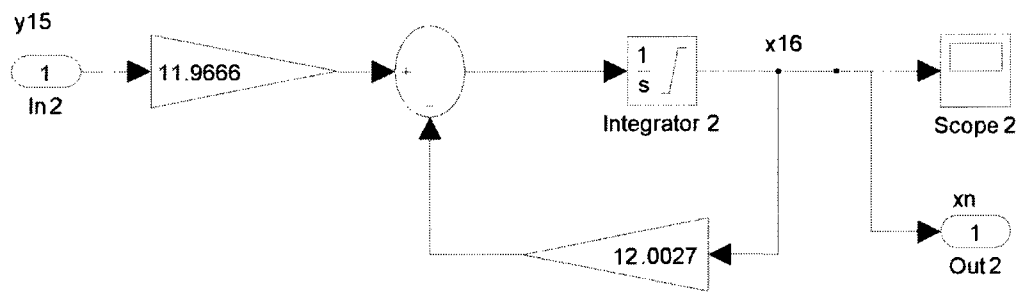


Figure D.10. Module of the condenser and reflux drum.

APPENDIX E: CONSTRUCTION OF REFERENCE MODEL

E.1. Model Construction

This section describes construction of full order model. The outputs of interest are the purity of overhead and bottom products. These quantities are desired to be kept within prescribed limits ($x_D \geq 98\%$ and $x_B \leq 2\%$) under disturbances of the feed streams or environment. The selected control structure is L - V structure, in which the reflux flow L at the column top and the boilup rate in the column bottom V are the manipulated inputs.

Consider the nonlinear equations represented for a generic tray:

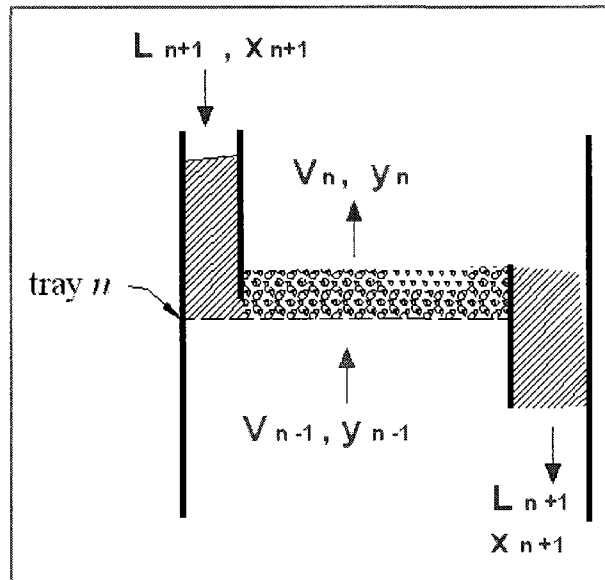


Figure E.1. Model of a generic tray.

Material balance is determined as follows:

$$\text{ACCUMULATION} = \text{INLET} - \text{OUTLET}$$

$$M_n \dot{X}_n = (V_{n-1} Y_{n-1} + L_{n+1} X_{n+1}) - (V_n Y_n + L_n X_n)$$

or,

$$\dot{X}_n = \frac{V_{n-1}}{M_n} Y_{n-1} - \frac{V_n}{M_n} Y_n + \frac{L_{n+1}}{M_n} X_{n+1} - \frac{L_n}{M_n} X_n \quad (\text{E.1})$$

where $L_1 = \dots = L_8 = L + L_F = L + 104.2491$; $L_9 = \dots = L_{15} = L$; $V_1 = \dots = V_8 = V$; and $V_9 = \dots = V_{15} = V + V_F = V + 98.5152$.

Vapor liquid equilibrium relationship at each tray is given by

$$Y_n = \frac{\alpha X_n}{1 + (\alpha - 1) X_n} = \frac{5.68 X_n}{1 + 4.68 X_n} \quad \forall n = 1, \dots, 15. \quad (\text{E.2})$$

Therefore, the concentrations of liquid on each tray are a vector function f of state vector x and manipulated input vector u :

$$\dot{X} = f(X, u, t)$$

where $X = (X_1, X_2, \dots, X_{16})^T$; and $u = (L, V)^T$.

The above nonlinear system can be linearized around the steady state value at the nominal operating point (X^*, u^*) . We define the perturbed states and control inputs as

$$dX = X - X^*$$

$$du = u - u^*$$

The linearized equations are given by:

$$d\dot{X} = \left[\frac{\partial f}{\partial x} \right] dX + \left[\frac{\partial f}{\partial u} \right] du$$

where $\partial f / \partial X$ stands for the Jacobian of the vector function f with respect to the state vector x ; and $\partial f / \partial u$ stands for the Jacobian of the vector function f with respect to the manipulated input vector u .

Make differentiation of (E.1) as follows:

$$\begin{aligned} d\dot{X}_n = & \frac{V_{n-1}^*}{M_n} dY_{n-1} + \frac{y_{n-1}^*}{M_n} dV_{n-1} - \frac{V_n^*}{M_n} dY_n - \frac{y_n^*}{M_n} dV_n \\ & + \frac{L_{n+1}^*}{M_n} dX_{n+1} + \frac{X_{n+1}^*}{M_n} dL_{n+1} - \frac{L_n^*}{M_n} dX_n - \frac{X_n^*}{M_n} dL_n. \end{aligned}$$

Substituting for terms of dY and regrouping the equation give the following result:

$$\begin{aligned} d\dot{X}_n = & \frac{L_{n+1}^*}{M_n} dX_{n+1} - \frac{L_n^* + K_n V_n^*}{M_n} dX_n + \frac{K_{n-1} V_{n-1}^*}{M_n} dX_{n-1} \\ & + \frac{X_{n-1}^* - X_n^*}{M_n} dL - \frac{Y_n^* - Y_{n-1}^*}{M_n} dV \end{aligned} \quad (\text{E.3})$$

where $dL_n = dL$, for all $n = 1, \dots, 15$; $dV_n = dV$, for all $n = 1, \dots, 15$; K_n is the linearized VLE constant; and y_n^* , x_n^* , L_n^* and V_n^* are the steady-state values at the nominal operating point.

In addition, we make linearization for some special stages as follows:

1) Reboiler ($n = 1$)

$$d\dot{X}_1 = \frac{L_2^*}{M_1} dX_2 + \frac{X_2^*}{M_1} dL - \frac{V_1^*}{M_1} dY_1 - \frac{Y_1^*}{M_1} dV - \frac{B}{M_1} dX_1.$$

2) Condenser ($n = 16$)

$$d\dot{X}_{16} = \frac{V_{15}^*}{M_{16}} dY_{15} + \frac{Y_{15}^*}{M_{16}} dV - \frac{L_{16}^*}{M_{16}} dX_{16} - \frac{X_{16}^*}{M_{16}} dL - \frac{D}{M_{16}} dX_{16}.$$

As a result, the model is represented in state space in terms of deviation variables:

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (\text{E.4})$$

and

$$y(t) = Cx(t) \quad (\text{E.5})$$

where $x = [dx_1 \ dx_2 \ \dots \ dx_{16}]^T$ is the vector of composition deviations; and $u = [dx_1 \ dx_2 \ \dots \ dx_{16}]^T$ is the vector of manipulated inputs; and $y = [dX_B \ dX_D]^T = [dX_1 \ dX_{16}]^T$ is the vector of controlled outputs.

The algorithm for calculating state matrix elements A (16x16) is described as follows:

- 1) Collect flow rates data:

$$L_1^* = \dots = L_8^* = 179.8871 \text{ (kmole/h)},$$

$$L_9^* = \dots = L_{15}^* = 75.6380 \text{ (kmole/h)},$$

$$V_1^* = \dots = V_8^* = 66.1139 \text{ (kmole/h)},$$

$$V_9^* = \dots = V_{15}^* = 164.6291 \text{ (kmole/h)}.$$

- 2) Collect liquid holdups data:

$$M_1 = M_B = 24.88 \text{ (kmole)},$$

$$M_2 = M_3 = \dots = M_{15} = 5.80 \text{ (kmole)},$$

$$M_{16} = M_B = 24.88 \text{ (kmole)}.$$

- 3) Calculate K_1, K_2, \dots, K_{16} .

- 4) Calculate the state matrix elements:

$$n=1: a_{1,1} = -\frac{B + K_1 V_1^*}{M_1}, \quad a_{1,2} = \frac{L_2^*}{M_1}$$

$$n=2: a_{2,1} = \frac{K_1 V_1^*}{M_2}, \quad a_{2,2} = -\frac{(L_2^* + K_2 V_2^*)}{M_2}, \quad a_{2,3} = \frac{L_3^*}{M_2}$$

$$\dots$$

$$n: \quad a_{n,n-1} = \frac{K_{n-1}V_{n-1}^*}{M_n}, \quad a_{n,n} = -\frac{(L_n^* + K_nV_n^*)}{M_n}, \quad a_{n,n+1} = \frac{L_{n+1}^*}{M_n}$$

$$\dots$$

$$n=16: \quad a_{16,15} = \frac{K_{15}V_{15}^*}{M_{16}}, \quad a_{16,16} = -\frac{L_{16}^* + D}{M_{16}}.$$

As a result, the state matrix A (16x16) is tri-diagonal:

$$A = \begin{pmatrix} -23.0 & 7.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 61.4 & -83.2 & 31.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 52.1 & -70.5 & 31.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 39.5 & -59.0 & 31.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 27.9 & -51.4 & 31.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 20.4 & -47.5 & 31.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16.5 & -45.7 & 31.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 14.7 & -44.9 & 13.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 13.9 & -44.9 & 13.0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 31.9 & -41.6 & 13.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 28.5 & -35.6 & 13.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 22.5 & -28.3 & 13.0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15.2 & -22.7 & 13.0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9.7 & -19.9 & 13.0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6.9 & -18.7 & 13.0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.4 & -12.0 \end{pmatrix}$$

The algorithm for calculating input matrix elements B (16x2) is as follows:

1) Collect data:

LPG product purity: $X_D^* = X_{16}^* = 0.9851 = 98.51\%$;

Raw Gasoline product purity: $1 - X_B^* = 1 - X_1^* = 0.989 = 98.9\%$.

2) Calculate the state matrix elements:

$$n=1: \quad b_{1,1} = \frac{X_2^*}{M_1}, \quad b_{1,2} = \frac{-Y_1^*}{M_1}$$

$$n=2: \quad b_{2,1} = \frac{(X_3 - X_2)}{M_2}, \quad b_{2,2} = \frac{-(Y_2 - Y_1)}{M_2}$$

...

$$n: \quad b_{n,1} = \frac{(X_{n+1} - X_n)}{M_n}, \quad b_{n,2} = \frac{-(Y_n - Y_{n-1})}{M_n}$$

...

$$n=16: \quad b_{16,1} = \frac{-X_{16}^*}{M_{16}}, \quad b_{16,2} = \frac{Y_{15}^*}{M_{16}}.$$

The input matrix B (16x2) is:

$B =$

$$\begin{bmatrix} 0.0012 & 0.0063 & 0.0091 & 0.0098 & 0.0076 & 0.0044 & 0.0022 & 0.0021 & 0.0048 \\ 0.0111 & 0.0216 & 0.0306 & 0.0285 & 0.0178 & 0.0086 & -0.0702; \\ -0.0024 & -0.0157 & -0.0237 & -0.0254 & -0.0196 & -0.0116 & -0.0058 & -0.0027 & -0.0024 \\ -0.0050 & -0.0097 & -0.0138 & -0.0128 & -0.0080 & -0.0039 & 0.0704 \end{bmatrix}^T.$$

The output matrix C (2x16) is

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

E.2. Stability Test

The stability of the system can be determined by using the Lyapunov direct method. Let us assume that the quadratic Lyapunov function is

$$V(x) = x^T P x \quad (\text{E.6})$$

where P is a symmetric, positive definite matrix.

The time derivative of V is

$$\dot{V}(x) = \dot{x}^T P x + x^T P \dot{x}. \quad (\text{E.7})$$

Since the system homogeneous differential equation $\dot{x} = Ax$ and $(Ax)^T = x^T A^T$, we have

$$\begin{aligned} \dot{V}(x) &= (Ax)^T P x + x^T P \dot{x} \\ \dot{V}(x) &= x^T A^T P x + x^T P A x = x^T (A^T P + P A) x. \end{aligned} \quad (\text{E.8})$$

If $A^T P + P A = -Q$ for some positive definite matrix Q , then the system is asymptotically stable.

We choose $Q = I$, where I is the identity (16x16) matrix. The symmetric matrix P is determined by solving the following equation:

$$A^T P + P A = -I \quad (\text{E.9})$$

where A is the state matrix. The system is asymptotically stable when matrix P is positive definite.

The system stability test can be done with the following MATLAB program:

```
% The condition V(0,t) = 0 is obviously satisfied.  
P = lyap(A,I);  
% Determinants of the principal minors:  
for i=1:16
```

```

    Pi = P(1:i,1:i);
    det(Pi);
    detPi=['det(P' int2str(i) ') = ' num2str(det(Pi))];
    disp(detPi);
end

```

The results show that all principal minors are positive:

```

det(P1) = 0.032831
det(P2) = 0.00090849
det(P3) = 2.4443e-005
det(P4) = 5.5476e-007
det(P5) = 1.0152e-008
det(P6) = 1.6208e-010
det(P7) = 2.3025e-012
det(P8) = 2.6183e-014
det(P9) = 4.1959e-016
det(P10) = 1.2886e-017
det(P11) = 6.7847e-019
det(P12) = 3.916e-020
det(P13) = 1.8308e-021
det(P14) = 7.089e-023
det(P15) = 2.4876e-024
det(P16) = 8.4612e-026

```

The symmetric matrix P is positive definite; hence, the system is asymptotically stable.

APPENDIX F: SOURCE CODE

F.1. Adaptive Mechanism

```
#include <iostream>
#include <stdlib.h>
#include "AdaptiveMech.h"
using namespace std;
AdaptiveMech::AdaptiveMech() {
}
AdaptiveMech::~AdaptiveMech() {
}
void AdaptiveMech::getAdapMechPkt(int k, Pkt x_kml, Pkt e_kml,
Pkt uc_kml, thetaPkt th_kml) {
    if(k != x_kml.k+1)
        printf("AdaptiveMech::getAdapMechPkt=>Wrong x(k-1) time
stamp!\n");
    if(k != e_kml.k+1)
        printf("AdaptiveMech::getAdapMechPkt=>Wrong e(k-1) time
stamp!\n");
    if(k != uc_kml.k+1)
        printf("AdaptiveMech::getAdapMechPkt=>Wrong uc(k-1) time
stamp!\n");
    this->x_kml = x_kml;
    this->e_kml = e_kml;
    this->uc_kml = uc_kml;
    this->th_kml = th_kml;
    return;
}
void AdaptiveMech::genAdapMechPkt(int k, thetaPkt &th_k) {
    // generate adaptive gains
    th_k.k = k;
```

```

    th_k.th1 = th_kml.th1 + T*gamma*(b11*e_kml.p1 +
b21*e_kml.p2)*x_kml.p1;
    th_k.th2 = th_kml.th2 + T*gamma*(b11*e_kml.p1 +
b21*e_kml.p2)*x_kml.p2;
    th_k.th3 = th_kml.th3 + T*gamma*(b12*e_kml.p1 +
b22*e_kml.p2)*x_kml.p1;
    th_k.th4 = th_kml.th4 + T*gamma*(b12*e_kml.p1 +
b22*e_kml.p2)*x_kml.p2;
    th_k.th5 = th_kml.th5 - T*gamma*(b11*e_kml.p1 +
b21*e_kml.p2)*uc_kml.p1;
    th_k.th6 = th_kml.th6 - T*gamma*(b11*e_kml.p1 +
b21*e_kml.p2)*uc_kml.p2;
    th_k.th7 = th_kml.th7 - T*gamma*(b12*e_kml.p1 +
b22*e_kml.p2)*uc_kml.p1;
    th_k.th8 = th_kml.th8 - T*gamma*(b12*e_kml.p1 +
b22*e_kml.p2)*uc_kml.p2;
    return;
}

```

F.2. Plant Model

```

#include <iostream>
#include <stdlib.h>
#include "PlantModel.h"
using namespace std;
Plant::Plant() {
    a11 = -6.7941 + 0.67*(rand()%2-0.5);
    a12 = -0.9095 + 0.09*(rand()%2-0.5);
    a21 = 1.4686 + 0.15*(rand()%2-0.5);
    a22 = -0.2497 + 0.02*(rand()%2-0.5);
    b11 = -0.1461 + 0.014*(rand()%2-0.5);
    b12 = 0.2073 + 0.02*(rand()%2-0.5);
    b21 = -0.0021 + 0.0002*(rand()%2-0.5);
    b22 = -0.0281 + 0.003*(rand()%2-0.5);
}

```

```

    c11 = -0.0624 + 0.006*(rand()%2-0.5);
    c12 = -0.0281 + 0.003*(rand()%2-0.5);
    c21 = 0.2458 + 0.025*(rand()%2-0.5);
    c22 = 0.0009 + 0.00009*(rand()%2-0.5);
    printf("A = %f\t %f\t %f\t %f\t\n", a11, a12, a21, a22);
    printf("B = %f\t %f\t %f\t %f\t\n", b11, b12, b21, b22);
    printf("B = %f\t %f\t %f\t %f\t\n", c11, c12, c21, c22);
}
Plant::~Plant() {
}
void Plant::getPlantPkt(int k, Pkt u_k, Pkt x_k) {
    if(k != u_k.k)
        printf("Plant::getPlantPkt => Wrong u(k) time stamp!\n");
    this->u_k = u_k;
    this->x_k = x_k;
    return;
}
void Plant::genPlantPkt(int k, Pkt &x_kp1) {
    //compute plant states
    x_kp1.p1 = (1+a11*T)*x_k.p1 + a12*T*x_k.p2 + b11*T*u_k.p1 +
b12*T*u_k.p2;
    x_kp1.p2 = a21*T*x_k.p1 + (1+a22*T)*x_k.p2 + b21*T*u_k.p1 +
b22*T*u_k.p2;
    x_kp1.k = k+1;
    return;
}

```

F.3. Reference Model

```

#include <iostream>
#include <stdlib.h>
#include "ReferenceModel.h"
using namespace std;
Refmdl::Refmdl() {

```

```

}
Refmdl::~Refmdl() {
}

void Refmdl::refmdlGetPkt(int k, Pkt uc_k, Pkt xm_k) {
    if(k != uc_k.k)
        printf("Refmdl::refmdlGetPkt => Wrong uc(k) time
stamp!\n");
    this->uc_k = uc_k;
    if(k != xm_k.k)
        printf("Refmdl::refmdlGetPkt => Wrong xm(k) time
stamp!\n");
    this->xm_k = xm_k;
    return;
}

void Refmdl::genRefPkt(int k, Pkt &xm_kp1) {
// compute reference states
    xm_kp1.k = k+1;
    xm_kp1.p1 = (1+am11*T)*xm_k.p1 + am12*T*xm_k.p2 +
bm11*T*uc_k.p1 + bm12*T*uc_k.p2;
    xm_kp1.p2 = am21*T*xm_k.p1 + (1+am22*T)*xm_k.p2 +
bm21*T*uc_k.p1 + bm22*T*uc_k.p2;
    return;
}

```

F.4. Linear Controller

```

#include <iostream>
#include <stdlib.h>
#include "LinearControl.h"
using namespace std;
Linctrl::Linctrl() {
}
Linctrl::~Linctrl() {
}

```

```

}
void Linctrl::getLinctrlPkt(int k, thetaPkt th_k, Pkt uc_k, Pkt
x_k) {
    if(th_k.k != k) printf("Linctrl::getLinctrlPkt => Wrong th(k)
time stamp!\n");
    this->th_k = th_k;
    if(k != uc_k.k) printf("Linctrl::getLinctrlPkt => Wrong uc(k)
time stamp!\n");
    this->uc_k = uc_k;
    if(k != x_k.k) printf("Linctrl::getLinctrlPkt => Wrong x(k)
time stamp!\n");
    this->x_k = x_k;
    return;
}
void Linctrl::genLinctrlPkt(int k, Pkt &u_k) {
//produce control signals
    u_k.p1 = th_k.th5*uc_k.p1 + th_k.th6*uc_k.p2 - th_k.th1*x_k.p1
- th_k.th2*x_k.p2;
    u_k.p2 = th_k.th7*uc_k.p1 + th_k.th8*uc_k.p2 - th_k.th3*x_k.p1
- th_k.th4*x_k.p2;
    u_k.k = k;
    return;
}

```

F.5. Comparator

```

#include <iostream>
#include <stdlib.h>
#include "Comparator.h"
using namespace std;
Comparator::Comparator() {
}
Comparator::~~Comparator() {
}

```

```

void Comparator::getCmprPkt(int k, Pkt x_k, Pkt xm_k) {
    if(k != x_k.k) printf("Comparator::Wrong x(k) time stamp!\n");
    this->x_k = x_k;
    this->xm_k = xm_k;
    return;
}

void Comparator::genCmprPkt(int k, Pkt &e_k) {
//compute errors
    e_k.p1 = x_k.p1 - xm_k.p1;
    e_k.p2 = x_k.p2 - xm_k.p2;
    e_k.k = k;
    return;
}

```

F.6. Controlled Output

```

#include <iostream>
#include <stdlib.h>
#include "ControlledOutput.h"
using namespace std;
ControlledOutput::ControlledOutput() {
}
ControlledOutput::~ControlledOutput() {
}
void ControlledOutput::getRefOutPkt(int k, Pkt x_k) {
    if(k != x_k.k) printf("ControlledOutput::Wrong x(k) time
stamp!\n");
    this->x_k = x_k;
    return;
}
void ControlledOutput::genRefOutPkt(int k, Pkt &y_k) {
//produce controlled outputs
    y_k.p1 = c11*x_k.p1 + c12*x_k.p2;
    y_k.p2 = c21*x_k.p1 + c22*x_k.p2;
}

```

```

    y_k.k = k;
    return;
}

```

F.7. Reference Outputs

```

#include <iostream>
#include <stdlib.h>
#include "ReferenceOutput.h"
using namespace std;
RefOutput::RefOutput() {
}
RefOutput::~RefOutput() {
}

void RefOutput::getRefOutPkt(int k, Pkt xm_k) {
    if(k != xm_k.k) printf("RefOutput::getRefOutPkt => Wrong xm(k)
time stamp!\n");
    this->xm_k = xm_k;
    return;
}

void RefOutput::genRefOutPkt(int k, Pkt &ym_k) {
    ym_k.p1 = cm11*xm_k.p1 + cm12*xm_k.p2;
    ym_k.p2 = cm21*xm_k.p1 + cm22*xm_k.p2;
    ym_k.k = k;
    ym_k.p1 = ym_k.p1;
    ym_k.p2 = ym_k.p2;
    return;
}

```

F.8. CGI Program

```

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <assert.h>

```

```

#include <errno.h>
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define DATA_PORT 2534
#define CLIENTIP "192.168.0.8"
#define LINELEN 1024

struct valinfo
{
    unsigned char *name;
    unsigned char *text;
};

struct thetaPkt
{
    int k;
    int th1;
    int th2;
    int th3;
    int th4;
    int th5;
    int th6;
    int th7;
    int th8;
};

```



```

struct Pkt
{
    int k;
    int p1;
    int p2;
};

static const int MAXSIZE = 2000;
//static const int T = 0.1;
static const int Tgamma = 1;
static const int MAXPKT = 11;

static const int uc1 = 500;
static const int uc2 = 100;

static const int am11 = -6794;
static const int am12 = -910;
static const int am21 = 1469;
static const int am22 = -2410;
static const int bm11 = -146;
static const int bm12 = 207;
static const int bm21 = -2;
static const int bm22 = -28;
static const int cm11 = -62;
static const int cm12 = -28;
static const int cm21 = 246;
static const int cm22 = 1;
static const int a11 = -6712;
static const int a12 = -897;
static const int a21 = 1559;
static const int a22 = -158;
static const int b11 = -83;
static const int b12 = 235;

```

```

static const int b21 = 8;
static const int b22 = 27;
static const int c11 = -62;
static const int c12 = -28;
static const int c21 = 246;
static const int c22 = 1;
static const int th0_1 = 371;
static const int th0_2 = -510;
static const int th0_3 = 280;
static const int th0_4 = -310;
static const int th0_5 = 1044;
static const int th0_6 = -42;
static const int th0_7 = 0;
static const int th0_8 = 1006;

unsigned char *getval(unsigned char *);

static int gotvals=0;
static int nvals=0;
static struct valinfo *vals;
static int hextobin(unsigned char);
static unsigned char *httpunescape(unsigned char *);
static int getvals(void);
static void getAdapMechPkt(int k, struct Pkt *x_kml, struct Pkt
*e_kml, struct Pkt *uc_kml, struct thetaPkt *th_kml);
static void fwritethpkt(char* fname, struct thetaPkt* th);
static void fwritepkt(char* fname, struct Pkt* pkt);
void itoch(int x1, unsigned char* c, int* len, unsigned char*
sign);

unsigned char *getval(unsigned char *name)
{
    int i;

```

```

if (!gotvals)
{
    nvals = getvals();
    gotvals = 1;
}
if (nvals == 0) return NULL;

for (i=0;i<nvals;i++)
{
    if(strcmp((const char *)name, (const char
*)vals[i].name)==0)
        return vals[i].text;
}
return NULL;
}

/* ===== httpunescape ===== */
static unsigned char *httpunescape(unsigned char *sis)
{
    unsigned char *siptr;
    unsigned char *soptr;
    unsigned char *sos;
    sos = (unsigned char *) calloc (strlen((char *)sis)+1,sizeof
(unsigned char));
    if(sos == NULL) return NULL;
    soptr = sos;
    siptr = sis;
    while (*siptr)
    {
        if(*siptr == '%')
        {
            int c = 0, i;

```

```

        for (i=0;i<2;i++)
        {
            int h;
            siptr++;
            if (*siptr == '\0') break;
            if ((h=hexctobin(*siptr)) == -1) break;
            c = c<<4 + h;
        }
        if (i != 2)
        {
            free(sos);
            return NULL;
        }
        *soptr++ = (unsigned char) c;
        } else if (*siptr == '+')
        *soptr++ = ' ';
    else
        *soptr++ = *siptr;
        siptr++;
    }
    *soptr = '\0';

    strcpy ((char *)sis, (const char *)sos);
    free (sos);
    return sis;
}

/* ===== hexctobin ===== */
static int  hexctobin(unsigned char c)
{
    if(isdigit(c))
        return c-'0';
    else if (isxdigit(c))

```

```

        return tolower(c) - 'a' + 10;
    else
        return -1;
}

/* ===== getvals ===== */
static int  getvals(void)
{
    int i;
    int vcnt = 0;
    unsigned char *vstr;
    unsigned char *vptr;
    unsigned char *eptr;
    unsigned char *aptr;

    vstr = (unsigned char *) getenv ( "REQUEST_METHOD");
    if(vstr == NULL)    return 0;
    if(strcmp((const char *)vstr,"POST") == 0)
    {
        int l, cl;
        vstr = (unsigned char *) getenv("CONTENT_LENGTH");
        if (vstr == NULL || strlen((const char *)vstr) == 0)
            return 0;
        if ((cl = atoi((const char *)vstr)) == 0)
            return 0;
        vstr = (unsigned char *)malloc(cl+2);
        if(vstr == NULL)
            return 0;
        fgets((char *)vstr,cl+1,stdin);
        l = strlen((const char *)vstr);
        if(vstr[l-1] == '\n')
            vstr[l-1]='\0';
    }
}

```

```

else
{
    vstr = (unsigned char *) getenv("QUERY_STRING");

    if(vstr == NULL) return 0;
}
vptr = vstr;

while (*vptr)
    if(*vptr++ == '&') vcnt++;

vcnt++;
vals = (struct valinfo *)calloc(vcnt,sizeof (struct
valinfo));
if(vals == NULL) return 0;
vptr = vstr;

for (i=0;i<vcnt;i++)
{
    eptr = (unsigned char *)strchr((const char *)vptr, '=');
    aptr = (unsigned char *)strchr((const char *)vptr, '&');

    if (eptr == NULL)
        return 0;
    *eptr = '\0';
    vals[i].name = httpunescape(vptr);
    if (vals[i].name == NULL) return 0;
    if (aptr)
    {
        *aptr = '\0';
        vptr = aptr+1;
    }
    vals[i].text = httpunescape(eptr+1);
}

```

```

        if(vals[i].text == NULL) return 0;
    }
    return vcnt;
}

/* ===== write adaptive gains ===== */
static void fwritethpkt(char* fname, struct thetaPkt* th){
    unsigned char *c;
    unsigned char *cs;
    int fd, len;

    fd=open(fname,O_WRONLY | O_CREAT, 0666);
    if((fd==-1)){
        printf("Can't write %s\n", fname);
        exit(1);
    }

    c = (unsigned char*) malloc(20);
    cs = (unsigned char*) malloc(2);
    itoch(th->k, c, &len, cs);
    write(fd, c, len);
    write(fd, "\t", 1);

    free((void*) c);
    free((void*) cs);
    c = (unsigned char*) malloc(20);
    cs = (unsigned char*) malloc(1);
    itoch(th->th1, c, &len, cs);
    write(fd, cs, 1);
    write(fd, c, len);
    write(fd, "\t", 1);

    free((void*) c);

```

```
free((void*) cs);
c = (unsigned char*) malloc(20);
cs = (unsigned char*) malloc(1);
itoch(th->th2, c, &len, cs);
write(fd, cs, 1);
write(fd, c, len);
write(fd, "\t", 1);
```

```
free((void*) c);
free((void*) cs);
c = (unsigned char*) malloc(20);
cs = (unsigned char*) malloc(1);
itoch(th->th3, c, &len, cs);
write(fd, cs, 1);
write(fd, c, len);
write(fd, "\t", 1);
```

```
free((void*) c);
free((void*) cs);
c = (unsigned char*) malloc(20);
cs = (unsigned char*) malloc(1);
itoch(th->th4, c, &len, cs);
write(fd, cs, 1);
write(fd, c, len);
write(fd, "\t", 1);
```

```
free((void*) c);
free((void*) cs);
c = (unsigned char*) malloc(20);
cs = (unsigned char*) malloc(1);
itoch(th->th5, c, &len, cs);
write(fd, cs, 1);
write(fd, c, len);
```



```

write(fd, "\t", 1);

free((void*) c);
free((void*) cs);
c = (unsigned char*) malloc(20);
cs = (unsigned char*) malloc(1);
itoch(th->th6, c, &len, cs);
write(fd, cs, 1);
write(fd, c, len);
write(fd, "\t", 1);

free((void*) c);
free((void*) cs);
c = (unsigned char*) malloc(20);
cs = (unsigned char*) malloc(1);
itoch(th->th7, c, &len, cs);
write(fd, cs, 1);
write(fd, c, len);
write(fd, "\t", 1);

free((void*) c);
free((void*) cs);
c = (unsigned char*) malloc(20);
cs = (unsigned char*) malloc(1);
itoch(th->th8, c, &len, cs);
write(fd, cs, 1);
write(fd, c, len);
write(fd, "\t", 1);
free((void*) c);
free((void*) cs);

close(fd);
return;

```

```

}
/* ===== write signals ===== */
static void fwritepkt(char* fname, struct Pkt* pkt){
    unsigned char *c;
    unsigned char *cs;
    int fd, len;

    fd=open(fname,O_WRONLY | O_CREAT, 0666);
    //fd=open("adap.out",O_WRONLY, 0666);
    if((fd==-1)){
        printf("Can't write %s\n", fname);
        exit(1);
    }

    c = (unsigned char*) malloc(20);
    cs = (unsigned char*) malloc(2);
    itoch(pkt->k, c, &len, cs);
    write(fd, c, len);
    write(fd, "\t", 1);
    free((void*) c);
    free((void*) cs);

    c = (unsigned char*) malloc(20);
    cs = (unsigned char*) malloc(1);
    itoch(pkt->p1, c, &len, cs);
    write(fd, cs, 1);
    write(fd, c, len);
    write(fd, "\t", 1);
    free((void*) c);
    free((void*) cs);

    c = (unsigned char*) malloc(20);
    cs = (unsigned char*) malloc(1);

```

```

    itoch(pkt->p2, c, &len, cs);
    write(fd, cs, 1);
    write(fd, c, len);
    write(fd, "\t", 1);
    free((void*) c);
    free((void*) cs);

    close(fd);

    return;
}
/* ===== write a signal ===== */
static void fwritesig(char* fname, int sig){
    unsigned char *c;
    unsigned char *cs;
    int fd, len;

    fd=open(fname,O_WRONLY | O_CREAT, 0666);
    //fd=open("adap.out",O_WRONLY, 0666);
    if((fd==-1)){
        printf("Can't write %s\n", fname);
        exit(1);
    }

    c = (unsigned char*) malloc(20);
    cs = (unsigned char*) malloc(2);
    itoch(sig, c, &len, cs);
    write(fd, cs, 1);
    write(fd, c, len);
    write(fd, "\t", 1);
    free((void*) c);
    free((void*) cs);
}

```

```

    close(fd);

    return;
}

/* ===== getsignal ===== */
static void getsig(char* fname, int* signal){
    int fn, s, i, m, sn, l;
    unsigned char* buf;
    buf = (unsigned char*) malloc(20);
    //printf("<p> Opening the input file\n");
    fn = open(fname, O_RDONLY);
    //make sure it was really opened
    if(fn== -1)
    {
        printf("<p> Cannot open the fn.dat file.\n");
        //exit(1);
        return;
    }
    read(fn, buf, 20);
    s = atoi((const char*) buf);
    *signal = s;
    free((void*) buf);
    close(fn);
    return;// s;
}

/* ===== get ascii signal ===== */
static int getcsig(char* fname, int* signal){
    int fn, s, i, m, sn, l;
    unsigned char* buf;
    buf = (unsigned char*) malloc(20);
    //printf("<p> Opening the input file\n");
    fn = open(fname, O_RDONLY);

```

```

    //make sure it was really opened
    if(fn==-1)
    {
        printf("<p> Cannot open the input file %s.\n", fname);
        //exit(1);
        return 0;
    }
    read(fn, buf, 20);
    s = atoi((const char*) buf);
    *signal = s;
    free((void*) buf);
    close(fn);
    return 1;
}

/* ===== settime ===== */
static void settime(char* fname, int k){
    int f;
    int t=0;
    unsigned char *c;
    unsigned char *cs;
    int len;

    f=open(fname, O_WRONLY | O_CREAT, 0666);
    // if((f==-1)){
        // printf("Can't write %s\n", fname);
        // exit(1);
    // }
    while(f==-1 && t<10000){
        printf("<p>t=%d: cannot open %s\n", t++, fname);
    }
    if(t>=10000) exit(1);

    c = (unsigned char*) malloc(20);

```

```

    cs = (unsigned char*) malloc(2);
    itoch(k, c, &len, cs);
    write(f, c, len);
    write(f, "\t", 1);
    free((void*) c);
    free((void*) cs);

    close(f);
    return;
}

/* ===== wait ===== */
static void wait(int n){
    int i=0;
    while(i++<n);
}

/* ===== gettime ===== */
static void gettime(int* kT){
    int k;
    //Use for testing CGI running on ARM
    getsig("/var/tmp/fkt.dat", &k);
    //clear signals
    //clearsig("/var/tmp/fkt.dat");
    *kT = k;
    //printf("<p> x(k-1).k = %d\n", k);
    //Write to files
    fwritesig("/var/tmp/chk_kT.dat", k);
    return;
}

/* ===== clear signal ===== */
static void clearsig(char* fname){
    int fn;
    //printf("<p> Opening the input file\n");

```

```

    fn = open(fname, O_WRONLY, 0666);
    //make sure it was really opened
    if(fn===-1)
    {
        printf("<p> Cannot open the input file %s.\n", fname);
        exit(1);
        return;
    }
    write(fn,"",20);
    close(fn);
    return;
}

/*===== integer to character ===== */

void itoch(int x1, unsigned char* c, int* len, unsigned char*
sign){
    int i;
    unsigned char* ch;

    ch = (unsigned char*) malloc(128);

    i=-1;
    if(x1==0)
    {
        ch[0] = '0';
        *len = 1;
    }
    if(x1<0){
        x1 = -x1;
        sign[0] = '-';
    } else {
        sign[0] = '+';
    }
}

```

```

while(x1>0)
{
    i++;
    ch[i] = '0' + x1%10;
    x1=x1/10;

}
*len=i+1;
for(i=0; i<*len; i++)
{
    c[i] = (const char) ch[*len-i-1];
}
free((void*) ch);

return;
}
/* ===== getAdapMechPkt ===== */
static void getAdapMechPkt(int k, struct Pkt *x_kml, struct Pkt
*e_kml, struct Pkt *uc_kml, struct thetaPkt *th_kml){
    int kx, x1, x2, ke, e1, e2, kuc, uc1, uc2, kth, th1, th2, th3,
th4, th5, th6, th7, th8;
    int outfile;

    //Receive signals
    getsig("/var/tmp/fkx.dat", &kx);
    getsig("/var/tmp/fx1.dat", &x1);
    getsig("/var/tmp/fx2.dat", &x2);
    getsig("/var/tmp/fke.dat", &ke);
    getsig("/var/tmp/fe1.dat", &e1);
    getsig("/var/tmp/fe2.dat", &e2);
    getsig("/var/tmp/fkuc.dat", &kuc);
    getsig("/var/tmp/fuc1.dat", &uc1);

```



```

    getsig("/var/tmp/fuc2.dat", &uc2);
    getsig("/var/tmp/fkth.dat", &kth);
    getsig("/var/tmp/fth1.dat", &th1);
    getsig("/var/tmp/fth2.dat", &th2);
    getsig("/var/tmp/fth3.dat", &th3);
    getsig("/var/tmp/fth4.dat", &th4);
    getsig("/var/tmp/fth5.dat", &th5);
    getsig("/var/tmp/fth6.dat", &th6);
    getsig("/var/tmp/fth7.dat", &th7);
    getsig("/var/tmp/fth8.dat", &th8);

    //Display on HTML page
    printf("<h3> Embedded Adaptive Controller: input signals
received.</h3>\n");
    printf("<h3> Time = %dT </h3>\n", k);
    printf("<p> x(%d).k = %d\n", k-1, kx);
    printf("<p> x1(%d) = %d\n", k-1, x1);
    printf("<p> x2(%d) = %d\n", k-1, x2);
    printf("<p> e(%d).k = %d\n", k-1, ke);
    printf("<p> e1(%d) = %d\n", k-1, e1);
    printf("<p> e2(%d) = %d\n", k-1, e2);
    printf("<p> uc(%d).k = %d\n", k-1, kuc);
    printf("<p> uc1(%d) = %d\n", k-1, uc1);
    printf("<p> uc2(%d) = %d\n", k-1, uc2);
    printf("<p> th1(%d).k = %d\n", k-1, kth);
    printf("<p> th1(%d) = %d\n", k-1, th1);
    printf("<p> th2(%d) = %d\n", k-1, th2);
    printf("<p> th3(%d) = %d\n", k-1, th3);
    printf("<p> th4(%d) = %d\n", k-1, th4);
    printf("<p> th5(%d) = %d\n", k-1, th5);
    printf("<p> th6(%d) = %d\n", k-1, th6);
    printf("<p> th7(%d) = %d\n", k-1, th7);
    printf("<p> th8(%d) = %d\n", k-1, th8);

```

```

x_kml->k = kx;
x_kml->p1 = x1;
x_kml->p2 = x2;
e_kml->k = ke;
e_kml->p1 = e1;
e_kml->p2 = e2;
uc_kml->k = kuc;
uc_kml->p1 = uc1;
uc_kml->p2 = uc2;
th_kml->k = kth;
th_kml->th1 = th1;
th_kml->th2 = th2;
th_kml->th3 = th3;
th_kml->th4 = th4;
th_kml->th5 = th5;
th_kml->th6 = th6;
th_kml->th7 = th7;
th_kml->th8 = th8;

//Write to files
fwritepkt("/var/tmp/chk_x.dat", x_kml);
fwritepkt("/var/tmp/chk_e.dat", e_kml);
fwritepkt("/var/tmp/chk_uc.dat", uc_kml);
fwritethpkt("/var/tmp/chk_th.dat", th_kml);
return;
}
/* ===== genAdapMechPkt ===== */
void genAdapMechPkt(int k, struct Pkt* x_kml, struct Pkt* e_kml,
struct Pkt* uc_kml, struct thetaPkt* th_kml, struct thetaPkt
*th_k){

th_k->k = k;

```

```

    th_k->th1 = th_kml->th1 + Tgamma*(b11*e_kml->p1 + b21*e_kml-
>p2)*x_kml->p1/10000000;
    th_k->th2 = th_kml->th2 + Tgamma*(b11*e_kml->p1 + b21*e_kml-
>p2)*x_kml->p2/10000000;
    th_k->th3 = th_kml->th3 + Tgamma*(b12*e_kml->p1 + b22*e_kml-
>p2)*x_kml->p1/10000000;
    th_k->th4 = th_kml->th4 + Tgamma*(b12*e_kml->p1 + b22*e_kml-
>p2)*x_kml->p2/10000000;
    th_k->th5 = th_kml->th5 - Tgamma*(b11*e_kml->p1 + b21*e_kml-
>p2)*uc_kml->p1/10000000;
    th_k->th6 = th_kml->th6 - Tgamma*(b11*e_kml->p1 + b21*e_kml-
>p2)*uc_kml->p2/10000000;
    th_k->th7 = th_kml->th7 - Tgamma*(b12*e_kml->p1 + b22*e_kml-
>p2)*uc_kml->p1/10000000;
    th_k->th8 = th_kml->th8 - Tgamma*(b12*e_kml->p1 + b22*e_kml-
>p2)*uc_kml->p2/10000000;
    // Write to files
    fwritethpkt("/var/tmp/adapout.dat", th_k);
    settime("/var/tmp/fka.dat", k);

    return;
}

/* ===== main ===== */
main ()
{
    unsigned char *str1,*str2, *cmd;
    int valid;
    int stime, gamm;
    int i=0;
    int k, kT;
    int kmax=2;

```

```

struct Pkt x_kml;
struct Pkt e_kml;
struct Pkt uc_kml;
struct thetaPkt th_kml;
struct thetaPkt th_k;
int getkt=0;
int getkmax=0;

cmd = getval((unsigned char *) "cmd");
printf("Content-type: text/html\n\n");
printf("<html> <head> <title> Web Server of Embedded Adaptive
Controller </title>
                                </head>\n");
if(strcmp((const char *)cmd, "run") == 0){

    str1 = getval((unsigned char*)"sptime");
    str2 = getval((unsigned char*)"gamma");
    printf("<body><h1>Embedded Adaptive Controller</h1>\n");
    k=1;
    while(!getkmax){
        getkmax=getcsig("/var/tmp/fkmax.dat", &kmax);
        wait(5000);
    }
    printf("<p> Maximal step size kmax = %d\n", kmax);
    while(k< kmax){
        printf("<h2> Time = %dT:</h2>\n", k);
        //Waiting for Plant Simulator sending signals
        printf("<p> Waiting for Plant Simulator sending
signals.\n");
        while(!getkt){
            getkt = getcsig("/var/tmp/fkt.dat", &kT);
            wait(5000);
        }

```

```

printf("<p> kT = %d, k = %d\n", kT, k);

while(k!=kT){
    getsig("/var/tmp/fkt.dat", &kT);
    wait(10000);
}
printf("<p> Adaptive Mech receiving input
signals...\n");
getAdapMechPkt(k, &x_kml, &e_kml, &uc_kml, &th_kml);
printf("<p> Adaptive Mech synthesizing adaptive
gains...\n");
genAdapMechPkt(k, &x_kml, &e_kml, &uc_kml, &th_kml,
&th_k);
    k++;
}
} else {
    printf ("<p>Sorry, the request is invalid.\n");
}
printf ("</body></html>\n");
exit (0);
}

```

APPENDIX G: MATLAB/C++ SIMULATION OUTPUT FILES

The simulation programs written in MATLAB and C++ give the same results, which are stored in output files. The results are shown in the following tables.

G.1. State Variables Files

The content of state variables data files is shown in Table G.1.

Table G.1. Simulation result of state variables.

k	t	x1	x2	xm1	xm2
0	0.000000	0.000000	0.000000	0.000000	0.000000
1	0.100000	-0.005938	-0.000402	-0.005232	-0.000386
2	0.200000	-0.007274	-0.001596	-0.006874	-0.001531
3	0.300000	-0.007525	-0.002967	-0.007297	-0.002888
4	0.400000	-0.006703	-0.004377	-0.007309	-0.004274
5	0.500000	-0.006552	-0.005628	-0.007186	-0.005626
6	0.600000	-0.007393	-0.006885	-0.007024	-0.006927
7	0.700000	-0.007877	-0.008274	-0.006854	-0.008172
8	0.800000	-0.006929	-0.009549	-0.006686	-0.009360
9	0.900000	-0.006522	-0.010673	-0.006524	-0.010494
10	1.000000	-0.006511	-0.011823	-0.006369	-0.011576
11	1.100000	-0.005794	-0.012844	-0.006221	-0.012609
12	1.200000	-0.005655	-0.013834	-0.006080	-0.013594
13	1.300000	-0.005802	-0.014738	-0.005945	-0.014533
14	1.400000	-0.005829	-0.015578	-0.005816	-0.015429
15	1.500000	-0.005804	-0.016366	-0.005693	-0.016284
16	1.600000	-0.005700	-0.017279	-0.005576	-0.017099
17	1.700000	-0.006102	-0.018029	-0.005464	-0.017877
18	1.800000	-0.006161	-0.018812	-0.005358	-0.018620

19	1.900000	-0.005840	-0.019639	-0.005256	-0.019327
20	2.000000	-0.005030	-0.020340	-0.005159	-0.020003
21	2.100000	-0.005135	-0.020948	-0.005067	-0.020647
22	2.200000	-0.005074	-0.021660	-0.004979	-0.021262
23	2.300000	-0.004855	-0.022270	-0.004894	-0.021848
24	2.400000	-0.005031	-0.022872	-0.004814	-0.022407
25	2.500000	-0.004512	-0.023498	-0.004737	-0.022941
26	2.600000	-0.004009	-0.023983	-0.004664	-0.023449
27	2.700000	-0.004133	-0.024360	-0.004595	-0.023935
28	2.800000	-0.004671	-0.024769	-0.004528	-0.024398
29	2.900000	-0.004958	-0.025263	-0.004465	-0.024840
30	3.000000	-0.004323	-0.025755	-0.004404	-0.025261
31	3.100000	-0.003983	-0.026125	-0.004346	-0.025663
32	3.200000	-0.004546	-0.026483	-0.004291	-0.026047
33	3.300000	-0.004544	-0.026923	-0.004239	-0.026413
34	3.400000	-0.004603	-0.027330	-0.004189	-0.026762
35	3.500000	-0.004297	-0.027685	-0.004141	-0.027095
36	3.600000	-0.004450	-0.027977	-0.004095	-0.027412
37	3.700000	-0.004369	-0.028329	-0.004052	-0.027715
38	3.800000	-0.003873	-0.028643	-0.004010	-0.028004
39	3.900000	-0.004316	-0.028964	-0.003971	-0.028280
40	4.000000	-0.003581	-0.029283	-0.003933	-0.028543
41	4.100000	-0.003195	-0.029457	-0.003897	-0.028794
42	4.200000	-0.004056	-0.029607	-0.003863	-0.029033
43	4.300000	-0.004004	-0.029793	-0.003830	-0.029261
44	4.400000	-0.003677	-0.029938	-0.003798	-0.029479
45	4.500000	-0.003060	-0.030105	-0.003769	-0.029687
46	4.600000	-0.002967	-0.030178	-0.003740	-0.029885

47	4.700000	-0.003945	-0.030245	-0.003713	-0.030074
48	4.800000	-0.003632	-0.030477	-0.003687	-0.030254
49	4.900000	-0.003741	-0.030604	-0.003662	-0.030426
50	5.000000	-0.003254	-0.030814	-0.003639	-0.030590
51	5.100000	-0.004091	-0.030945	-0.003616	-0.030747
52	5.200000	-0.003652	-0.031179	-0.003595	-0.030896
53	5.300000	-0.004092	-0.031310	-0.003574	-0.031039
54	5.400000	-0.004512	-0.031532	-0.003555	-0.031175
55	5.500000	-0.003289	-0.031857	-0.003536	-0.031304
56	5.600000	-0.002858	-0.031948	-0.003519	-0.031428
57	5.700000	-0.002967	-0.031971	-0.003502	-0.031546
58	5.800000	-0.003991	-0.032005	-0.003485	-0.031659
59	5.900000	-0.003572	-0.032227	-0.003470	-0.031766
60	6.000000	-0.003271	-0.032303	-0.003455	-0.031868
61	6.100000	-0.003674	-0.032298	-0.003441	-0.031966
62	6.200000	-0.003566	-0.032383	-0.003428	-0.032059
63	6.300000	-0.003464	-0.032482	-0.003415	-0.032148
64	6.400000	-0.003407	-0.032554	-0.003403	-0.032233
65	6.500000	-0.002696	-0.032590	-0.003391	-0.032314
66	6.600000	-0.002674	-0.032534	-0.003380	-0.032391
67	6.700000	-0.003472	-0.032468	-0.003370	-0.032465
68	6.800000	-0.003254	-0.032440	-0.003360	-0.032535
69	6.900000	-0.003053	-0.032512	-0.003350	-0.032602
70	7.000000	-0.003501	-0.032535	-0.003341	-0.032666
71	7.100000	-0.003466	-0.032586	-0.003332	-0.032727
72	7.200000	-0.003658	-0.032618	-0.003324	-0.032785
73	7.300000	-0.002994	-0.032758	-0.003316	-0.032840
74	7.400000	-0.003796	-0.032787	-0.003308	-0.032893

75	7.500000	-0.004422	-0.032935	-0.003301	-0.032944
76	7.600000	-0.003688	-0.033168	-0.003294	-0.032992
77	7.700000	-0.002866	-0.033314	-0.003287	-0.033038
78	7.800000	-0.003432	-0.033233	-0.003281	-0.033082
79	7.900000	-0.003142	-0.033228	-0.003275	-0.033124
80	8.000000	-0.002693	-0.033288	-0.003269	-0.033164
81	8.100000	-0.003564	-0.033206	-0.003264	-0.033202
82	8.200000	-0.003913	-0.033185	-0.003259	-0.033238
83	8.300000	-0.004231	-0.033216	-0.003254	-0.033272
84	8.400000	-0.004049	-0.033382	-0.003249	-0.033306
85	8.500000	-0.003128	-0.033535	-0.003244	-0.033337
86	8.600000	-0.002355	-0.033510	-0.003240	-0.033367
87	8.700000	-0.002156	-0.033344	-0.003236	-0.033396
88	8.800000	-0.002358	-0.033200	-0.003232	-0.033423
89	8.900000	-0.002603	-0.033080	-0.003228	-0.033449
90	9.000000	-0.003386	-0.032950	-0.003225	-0.033474
91	9.100000	-0.004014	-0.032966	-0.003221	-0.033498
92	9.200000	-0.004132	-0.033023	-0.003218	-0.033520
93	9.300000	-0.003349	-0.033146	-0.003215	-0.033542
94	9.400000	-0.003703	-0.033219	-0.003212	-0.033563
95	9.500000	-0.003246	-0.033293	-0.003209	-0.033582
96	9.600000	-0.003219	-0.033271	-0.003207	-0.033601
97	9.700000	-0.003219	-0.033284	-0.003204	-0.033619
98	9.800000	-0.002317	-0.033355	-0.003202	-0.033636
99	9.900000	-0.002500	-0.033271	-0.003199	-0.033652
100	10.000000	-0.003393	-0.033212	-0.003197	-0.033668

G.2. Reference and Controlled Outputs

The content of simulation data files for reference and controlled outputs is shown in Table G.2.

Table G.2. Simulation result of reference and controlled outputs.

k	t	y1	y2	ym1	ym2
0	0.000000	-0.000000	0.000000	-0.000000	0.000000
1	0.100000	0.000365	-0.001534	0.000337	-0.001286
2	0.200000	0.000479	-0.001880	0.000472	-0.001691
3	0.300000	0.000535	-0.001946	0.000536	-0.001796
4	0.400000	0.000528	-0.001735	0.000576	-0.001800
5	0.500000	0.000556	-0.001697	0.000607	-0.001771
6	0.600000	0.000643	-0.001915	0.000633	-0.001733
7	0.700000	0.000713	-0.002042	0.000657	-0.001692
8	0.800000	0.000694	-0.001798	0.000680	-0.001652
9	0.900000	0.000703	-0.001694	0.000702	-0.001613
10	1.000000	0.000737	-0.001692	0.000723	-0.001576
11	1.100000	0.000724	-0.001508	0.000742	-0.001540
12	1.200000	0.000745	-0.001472	0.000761	-0.001507
13	1.300000	0.000781	-0.001511	0.000779	-0.001474
14	1.400000	0.000807	-0.001519	0.000796	-0.001443
15	1.500000	0.000829	-0.001513	0.000813	-0.001414
16	1.600000	0.000850	-0.001487	0.000828	-0.001386
17	1.700000	0.000896	-0.001592	0.000843	-0.001359
18	1.800000	0.000923	-0.001608	0.000858	-0.001334
19	1.900000	0.000928	-0.001525	0.000871	-0.001309
20	2.000000	0.000901	-0.001317	0.000884	-0.001286

21	2.100000	0.000925	-0.001344	0.000896	-0.001264
22	2.200000	0.000943	-0.001329	0.000908	-0.001243
23	2.300000	0.000948	-0.001273	0.000919	-0.001223
24	2.400000	0.000976	-0.001319	0.000930	-0.001203
25	2.500000	0.000964	-0.001185	0.000940	-0.001185
26	2.600000	0.000948	-0.001056	0.000950	-0.001168
27	2.700000	0.000967	-0.001088	0.000959	-0.001151
28	2.800000	0.001011	-0.001228	0.000968	-0.001135
29	2.900000	0.001042	-0.001302	0.000977	-0.001120
30	3.000000	0.001019	-0.001139	0.000985	-0.001105
31	3.100000	0.001010	-0.001051	0.000992	-0.001091
32	3.200000	0.001054	-0.001197	0.001000	-0.001078
33	3.300000	0.001067	-0.001197	0.001007	-0.001066
34	3.400000	0.001082	-0.001212	0.001013	-0.001054
35	3.500000	0.001075	-0.001134	0.001020	-0.001042
36	3.600000	0.001092	-0.001173	0.001026	-0.001031
37	3.700000	0.001098	-0.001153	0.001032	-0.001021
38	3.800000	0.001078	-0.001025	0.001037	-0.001011
39	3.900000	0.001114	-0.001139	0.001042	-0.001001
40	4.000000	0.001080	-0.000950	0.001047	-0.000992
41	4.100000	0.001062	-0.000851	0.001052	-0.000984
42	4.200000	0.001117	-0.001073	0.001057	-0.000976
43	4.300000	0.001120	-0.001060	0.001061	-0.000968
44	4.400000	0.001105	-0.000975	0.001065	-0.000960
45	4.500000	0.001073	-0.000816	0.001069	-0.000953
46	4.600000	0.001070	-0.000792	0.001073	-0.000946
47	4.700000	0.001130	-0.001045	0.001077	-0.000940

48	4.800000	0.001118	-0.000964	0.001080	-0.000934
49	4.900000	0.001128	-0.000992	0.001084	-0.000928
50	5.000000	0.001105	-0.000867	0.001087	-0.000922
51	5.100000	0.001159	-0.001083	0.001090	-0.000917
52	5.200000	0.001140	-0.000970	0.001093	-0.000911
53	5.300000	0.001170	-0.001084	0.001095	-0.000907
54	5.400000	0.001201	-0.001192	0.001098	-0.000902
55	5.500000	0.001138	-0.000877	0.001100	-0.000897
56	5.600000	0.001115	-0.000766	0.001103	-0.000893
57	5.700000	0.001123	-0.000794	0.001105	-0.000889
58	5.800000	0.001184	-0.001058	0.001107	-0.000885
59	5.900000	0.001166	-0.000950	0.001109	-0.000882
60	6.000000	0.001150	-0.000873	0.001111	-0.000878
61	6.100000	0.001174	-0.000977	0.001113	-0.000875
62	6.200000	0.001170	-0.000949	0.001115	-0.000871
63	6.300000	0.001167	-0.000922	0.001116	-0.000868
64	6.400000	0.001166	-0.000908	0.001118	-0.000865
65	6.500000	0.001125	-0.000724	0.001120	-0.000863
66	6.600000	0.001122	-0.000718	0.001121	-0.000860
67	6.700000	0.001167	-0.000925	0.001123	-0.000857
68	6.800000	0.001153	-0.000868	0.001124	-0.000855
69	6.900000	0.001144	-0.000816	0.001125	-0.000853
70	7.000000	0.001171	-0.000932	0.001126	-0.000851
71	7.100000	0.001170	-0.000923	0.001128	-0.000848
72	7.200000	0.001183	-0.000973	0.001129	-0.000846
73	7.300000	0.001147	-0.000801	0.001130	-0.000845
74	7.400000	0.001196	-0.001009	0.001131	-0.000843

75	7.500000	0.001238	-0.001170	0.001132	-0.000841
76	7.600000	0.001201	-0.000981	0.001133	-0.000839
77	7.700000	0.001156	-0.000769	0.001134	-0.000838
78	7.800000	0.001188	-0.000915	0.001134	-0.000836
79	7.900000	0.001170	-0.000840	0.001135	-0.000835
80	8.000000	0.001145	-0.000724	0.001136	-0.000833
81	8.100000	0.001195	-0.000949	0.001137	-0.000832
82	8.200000	0.001215	-0.001039	0.001137	-0.000831
83	8.300000	0.001235	-0.001121	0.001138	-0.000830
84	8.400000	0.001229	-0.001074	0.001139	-0.000829
85	8.500000	0.001178	-0.000837	0.001139	-0.000827
86	8.600000	0.001132	-0.000637	0.001140	-0.000826
87	8.700000	0.001115	-0.000585	0.001140	-0.000825
88	8.800000	0.001123	-0.000637	0.001141	-0.000825
89	8.900000	0.001134	-0.000701	0.001141	-0.000824
90	9.000000	0.001176	-0.000903	0.001142	-0.000823
91	9.100000	0.001214	-0.001065	0.001142	-0.000822
92	9.200000	0.001223	-0.001095	0.001143	-0.000821
93	9.300000	0.001180	-0.000893	0.001143	-0.000820
94	9.400000	0.001203	-0.000985	0.001144	-0.000820
95	9.500000	0.001178	-0.000867	0.001144	-0.000819
96	9.600000	0.001176	-0.000860	0.001144	-0.000818
97	9.700000	0.001176	-0.000860	0.001145	-0.000818
98	9.800000	0.001125	-0.000627	0.001145	-0.000817
99	9.900000	0.001133	-0.000674	0.001145	-0.000817
100	10.000000	0.001185	-0.000905	0.001146	-0.000816

G.3. Plant Error

The content of simulation data file for plant error is shown in Table G.3.

Table G.3. Simulation result of plant errors.

k	t	e_1	e_2
0	0.000000	0.000000	0.000000
1	0.100000	-0.000706	-0.000016
2	0.200000	-0.000400	-0.000065
3	0.300000	-0.000229	-0.000079
4	0.400000	0.000606	-0.000103
5	0.500000	0.000634	-0.000002
6	0.600000	-0.000368	0.000042
7	0.700000	-0.001023	-0.000102
8	0.800000	-0.000243	-0.000189
9	0.900000	0.000002	-0.000179
10	1.000000	-0.000142	-0.000247
11	1.100000	0.000427	-0.000236
12	1.200000	0.000425	-0.000241
13	1.300000	0.000143	-0.000205
14	1.400000	-0.000013	-0.000149
15	1.500000	-0.000110	-0.000082
16	1.600000	-0.000124	-0.000180
17	1.700000	-0.000638	-0.000152
18	1.800000	-0.000803	-0.000193
19	1.900000	-0.000584	-0.000312
20	2.000000	0.000129	-0.000337
21	2.100000	-0.000068	-0.000301

22	2.200000	-0.000096	-0.000399
23	2.300000	0.000040	-0.000422
24	2.400000	-0.000217	-0.000465
25	2.500000	0.000226	-0.000557
26	2.600000	0.000656	-0.000534
27	2.700000	0.000462	-0.000425
28	2.800000	-0.000143	-0.000371
29	2.900000	-0.000493	-0.000423
30	3.000000	0.000081	-0.000494
31	3.100000	0.000364	-0.000462
32	3.200000	-0.000255	-0.000436
33	3.300000	-0.000305	-0.000510
34	3.400000	-0.000414	-0.000569
35	3.500000	-0.000157	-0.000590
36	3.600000	-0.000355	-0.000565
37	3.700000	-0.000318	-0.000614
38	3.800000	0.000137	-0.000639
39	3.900000	-0.000345	-0.000684
40	4.000000	0.000352	-0.000740
41	4.100000	0.000702	-0.000663
42	4.200000	-0.000194	-0.000574
43	4.300000	-0.000175	-0.000531
44	4.400000	0.000122	-0.000459
45	4.500000	0.000709	-0.000418
46	4.600000	0.000773	-0.000293
47	4.700000	-0.000232	-0.000171
48	4.800000	0.000055	-0.000223

49	4.900000	-0.000078	-0.000177
50	5.000000	0.000385	-0.000224
51	5.100000	-0.000474	-0.000198
52	5.200000	-0.000057	-0.000283
53	5.300000	-0.000518	-0.000271
54	5.400000	-0.000957	-0.000357
55	5.500000	0.000247	-0.000553
56	5.600000	0.000660	-0.000520
57	5.700000	0.000534	-0.000425
58	5.800000	-0.000505	-0.000347
59	5.900000	-0.000102	-0.000461
60	6.000000	0.000184	-0.000434
61	6.100000	-0.000232	-0.000332
62	6.200000	-0.000138	-0.000324
63	6.300000	-0.000049	-0.000333
64	6.400000	-0.000004	-0.000321
65	6.500000	0.000696	-0.000276
66	6.600000	0.000707	-0.000143
67	6.700000	-0.000103	-0.000003
68	6.800000	0.000106	0.000095
69	6.900000	0.000297	0.000090
70	7.000000	-0.000160	0.000130
71	7.100000	-0.000134	0.000141
72	7.200000	-0.000334	0.000167
73	7.300000	0.000322	0.000082
74	7.400000	-0.000488	0.000106

75	7.500000	-0.001121	0.000008
76	7.600000	-0.000394	-0.000176
77	7.700000	0.000422	-0.000276
78	7.800000	-0.000151	-0.000152
79	7.900000	0.000133	-0.000104
80	8.000000	0.000576	-0.000124
81	8.100000	-0.000300	-0.000005
82	8.200000	-0.000655	0.000053
83	8.300000	-0.000977	0.000056
84	8.400000	-0.000800	-0.000076
85	8.500000	0.000116	-0.000198
86	8.600000	0.000885	-0.000143
87	8.700000	0.001080	0.000052
88	8.800000	0.000874	0.000223
89	8.900000	0.000625	0.000369
90	9.000000	-0.000161	0.000524
91	9.100000	-0.000792	0.000532
92	9.200000	-0.000914	0.000498
93	9.300000	-0.000134	0.000396
94	9.400000	-0.000491	0.000343
95	9.500000	-0.000037	0.000289
96	9.600000	-0.000012	0.000330
97	9.700000	-0.000015	0.000335
98	9.800000	0.000884	0.000281
99	9.900000	0.000699	0.000381
100	10.000000	-0.000196	0.000456

G.4. Adaptive Gains

The content of simulation data file for adaptive gains is shown in Table G.4.

Table G.4. Simulation result of adaptive gains.

k	Time	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8
0	0.000	0.372	-0.511	0.281	-0.310	1.044	-0.043	0.000	1.007
1	0.100	0.372	-0.511	0.281	-0.310	1.044	-0.043	0.000	1.007
2	0.200	0.372	-0.511	0.281	-0.310	1.039	-0.044	0.007	1.008
3	0.300	0.372	-0.511	0.281	-0.310	1.036	-0.044	0.011	1.009
4	0.400	0.371	-0.511	0.281	-0.310	1.035	-0.045	0.013	1.009
5	0.500	0.372	-0.511	0.281	-0.310	1.039	-0.044	0.007	1.008
6	0.600	0.372	-0.511	0.281	-0.310	1.044	-0.043	0.001	1.007
7	0.700	0.372	-0.511	0.281	-0.310	1.041	-0.043	0.005	1.007
8	0.800	0.371	-0.511	0.281	-0.310	1.034	-0.045	0.015	1.009
9	0.900	0.371	-0.511	0.281	-0.310	1.032	-0.045	0.017	1.010
10	1.000	0.371	-0.511	0.281	-0.310	1.032	-0.045	0.016	1.010
11	1.100	0.371	-0.511	0.281	-0.310	1.031	-0.045	0.017	1.010
12	1.200	0.371	-0.511	0.281	-0.310	1.034	-0.045	0.013	1.009
13	1.300	0.371	-0.511	0.281	-0.310	1.037	-0.044	0.008	1.008
14	1.400	0.371	-0.511	0.281	-0.310	1.038	-0.044	0.007	1.008
15	1.500	0.371	-0.511	0.281	-0.310	1.038	-0.044	0.007	1.008
16	1.600	0.371	-0.511	0.281	-0.310	1.037	-0.044	0.008	1.008
17	1.700	0.371	-0.511	0.281	-0.310	1.036	-0.044	0.008	1.008
18	1.800	0.371	-0.511	0.281	-0.310	1.032	-0.045	0.015	1.009
19	1.900	0.371	-0.511	0.281	-0.310	1.026	-0.046	0.022	1.011
20	2.000	0.371	-0.511	0.281	-0.309	1.022	-0.047	0.027	1.012
21	2.100	0.371	-0.511	0.281	-0.309	1.023	-0.047	0.026	1.012
22	2.200	0.371	-0.511	0.281	-0.309	1.022	-0.047	0.026	1.012

23	2.300	0.371	-0.511	0.281	-0.309	1.022	-0.047	0.026	1.012
24	2.400	0.371	-0.511	0.281	-0.309	1.022	-0.047	0.025	1.012
25	2.500	0.371	-0.511	0.281	-0.309	1.020	-0.047	0.027	1.012
26	2.600	0.371	-0.511	0.281	-0.310	1.022	-0.047	0.024	1.011
27	2.700	0.371	-0.511	0.281	-0.310	1.026	-0.046	0.016	1.010
28	2.800	0.371	-0.511	0.281	-0.310	1.030	-0.046	0.011	1.009
29	2.900	0.371	-0.511	0.281	-0.310	1.028	-0.046	0.012	1.009
30	3.000	0.371	-0.511	0.281	-0.310	1.025	-0.047	0.016	1.010
31	3.100	0.371	-0.511	0.281	-0.310	1.025	-0.046	0.015	1.009
32	3.200	0.371	-0.511	0.281	-0.310	1.028	-0.046	0.011	1.009
33	3.300	0.371	-0.511	0.281	-0.310	1.026	-0.046	0.012	1.009
34	3.400	0.371	-0.511	0.281	-0.310	1.024	-0.047	0.015	1.009
35	3.500	0.371	-0.511	0.281	-0.310	1.021	-0.047	0.018	1.010
36	3.600	0.371	-0.511	0.281	-0.310	1.020	-0.048	0.019	1.010
37	3.700	0.371	-0.511	0.281	-0.310	1.017	-0.048	0.021	1.011
38	3.800	0.371	-0.512	0.281	-0.309	1.015	-0.049	0.023	1.011
39	3.900	0.371	-0.512	0.281	-0.310	1.016	-0.048	0.021	1.011
40	4.000	0.371	-0.512	0.281	-0.309	1.013	-0.049	0.024	1.011
41	4.100	0.371	-0.512	0.281	-0.310	1.016	-0.048	0.019	1.010
42	4.200	0.371	-0.511	0.281	-0.310	1.021	-0.047	0.011	1.009
43	4.300	0.371	-0.511	0.281	-0.310	1.019	-0.048	0.012	1.009
44	4.400	0.371	-0.511	0.281	-0.310	1.018	-0.048	0.013	1.009
45	4.500	0.371	-0.511	0.281	-0.310	1.019	-0.048	0.011	1.009
46	4.600	0.371	-0.511	0.281	-0.311	1.024	-0.047	0.004	1.007
47	4.700	0.371	-0.511	0.281	-0.311	1.029	-0.046	-0.004	1.006
48	4.800	0.371	-0.511	0.281	-0.311	1.027	-0.046	-0.002	1.006
49	4.900	0.371	-0.511	0.281	-0.311	1.028	-0.046	-0.003	1.006
50	5.000	0.371	-0.511	0.281	-0.311	1.027	-0.046	-0.003	1.006

51	5.100	0.371	-0.511	0.281	-0.311	1.030	-0.046	-0.007	1.005
52	5.200	0.371	-0.511	0.281	-0.311	1.026	-0.046	-0.003	1.006
53	5.300	0.371	-0.511	0.281	-0.311	1.026	-0.046	-0.002	1.006
54	5.400	0.371	-0.511	0.281	-0.311	1.022	-0.047	0.002	1.007
55	5.500	0.371	-0.512	0.281	-0.310	1.016	-0.048	0.011	1.009
56	5.600	0.371	-0.511	0.281	-0.310	1.017	-0.048	0.008	1.008
57	5.700	0.371	-0.511	0.281	-0.311	1.022	-0.047	0.001	1.007
58	5.800	0.371	-0.511	0.281	-0.311	1.026	-0.046	-0.005	1.005
59	5.900	0.371	-0.511	0.281	-0.311	1.022	-0.047	-0.001	1.006
60	6.000	0.371	-0.511	0.281	-0.311	1.021	-0.047	-0.000	1.006
61	6.100	0.371	-0.511	0.281	-0.311	1.023	-0.047	-0.003	1.006
62	6.200	0.371	-0.511	0.281	-0.311	1.021	-0.047	-0.001	1.006
63	6.300	0.371	-0.511	0.281	-0.311	1.020	-0.048	-0.000	1.006
64	6.400	0.371	-0.511	0.281	-0.311	1.020	-0.048	-0.000	1.006
65	6.500	0.371	-0.511	0.281	-0.311	1.019	-0.048	-0.001	1.006
66	6.600	0.371	-0.511	0.281	-0.311	1.024	-0.047	-0.008	1.005
67	6.700	0.371	-0.511	0.281	-0.312	1.029	-0.046	-0.015	1.003
68	6.800	0.371	-0.511	0.281	-0.312	1.028	-0.046	-0.014	1.004
69	6.900	0.371	-0.511	0.281	-0.312	1.029	-0.046	-0.015	1.004
70	7.000	0.371	-0.511	0.281	-0.312	1.031	-0.045	-0.018	1.003
71	7.100	0.371	-0.511	0.281	-0.312	1.030	-0.046	-0.016	1.003
72	7.200	0.371	-0.511	0.281	-0.312	1.029	-0.046	-0.014	1.004
73	7.300	0.371	-0.511	0.281	-0.312	1.027	-0.046	-0.011	1.004
74	7.400	0.371	-0.511	0.281	-0.312	1.029	-0.046	-0.014	1.004
75	7.500	0.371	-0.511	0.281	-0.311	1.026	-0.046	-0.009	1.005
76	7.600	0.371	-0.511	0.281	-0.311	1.018	-0.048	0.002	1.007
77	7.700	0.371	-0.512	0.281	-0.310	1.015	-0.048	0.006	1.008
78	7.800	0.371	-0.511	0.281	-0.311	1.018	-0.048	0.001	1.007

79	7.900	0.371	-0.511	0.281	-0.311	1.017	-0.048	0.002	1.007
80	8.000	0.371	-0.511	0.281	-0.311	1.018	-0.048	0.001	1.007
81	8.100	0.371	-0.511	0.281	-0.311	1.022	-0.047	-0.005	1.006
82	8.200	0.371	-0.511	0.281	-0.311	1.020	-0.048	-0.002	1.006
83	8.300	0.371	-0.512	0.281	-0.311	1.015	-0.048	0.005	1.007
84	8.400	0.371	-0.512	0.281	-0.310	1.009	-0.050	0.014	1.009
85	8.500	0.371	-0.512	0.281	-0.309	1.003	-0.051	0.022	1.011
86	8.600	0.371	-0.512	0.281	-0.309	1.004	-0.051	0.021	1.011
87	8.700	0.371	-0.512	0.281	-0.310	1.010	-0.050	0.012	1.009
88	8.800	0.371	-0.511	0.281	-0.311	1.017	-0.048	0.001	1.007
89	8.900	0.371	-0.511	0.281	-0.311	1.024	-0.047	-0.007	1.005
90	9.000	0.371	-0.511	0.281	-0.312	1.028	-0.046	-0.013	1.004
91	9.100	0.371	-0.511	0.281	-0.312	1.027	-0.046	-0.010	1.004
92	9.200	0.371	-0.511	0.281	-0.311	1.021	-0.047	-0.002	1.006
93	9.300	0.371	-0.512	0.281	-0.310	1.015	-0.049	0.008	1.008
94	9.400	0.371	-0.512	0.281	-0.310	1.014	-0.049	0.010	1.008
95	9.500	0.371	-0.512	0.281	-0.310	1.011	-0.049	0.015	1.010
96	9.600	0.371	-0.512	0.281	-0.310	1.011	-0.049	0.016	1.010
97	9.700	0.371	-0.512	0.281	-0.310	1.011	-0.049	0.017	1.010
98	9.800	0.371	-0.512	0.281	-0.310	1.011	-0.049	0.017	1.010
99	9.900	0.371	-0.512	0.281	-0.310	1.017	-0.048	0.009	1.008
100	10.000	0.371	-0.511	0.281	-0.311	1.022	-0.047	0.003	1.007

APPENDIX H: HARDWARE VALIDATION OUTPUT FILES

H.1. Validation Output Files of State Variables

The content of state variable data files is shown in Table H.1.

Table H.1. Validation result of state variables.

k	t	x_1	x_2	x_{m1}	x_{m2}
0	0.000000	0.000000	0.000000	0.000000	0.000000
1	0.100000	-0.005213	-0.000372	-0.005232	-0.000386
2	0.200000	-0.006672	-0.001506	-0.006874	-0.001531
3	0.300000	-0.008169	-0.002906	-0.007297	-0.002888
4	0.400000	-0.007644	-0.004473	-0.007309	-0.004274
5	0.500000	-0.007699	-0.005796	-0.007186	-0.005626
6	0.600000	-0.008154	-0.007132	-0.007024	-0.006927
7	0.700000	-0.007384	-0.008622	-0.006854	-0.008172
8	0.800000	-0.006498	-0.009855	-0.006686	-0.009360
9	0.900000	-0.006647	-0.010913	-0.006524	-0.010494
10	1.000000	-0.006411	-0.012023	-0.006369	-0.011576
11	1.100000	-0.006230	-0.013130	-0.006221	-0.012609
12	1.200000	-0.006451	-0.014033	-0.006080	-0.013594
13	1.300000	-0.005944	-0.015028	-0.005945	-0.014533
14	1.400000	-0.005865	-0.015859	-0.005816	-0.015429
15	1.500000	-0.005507	-0.016666	-0.005693	-0.016284
16	1.600000	-0.005139	-0.017478	-0.005576	-0.017099
17	1.700000	-0.005865	-0.018174	-0.005464	-0.017877
18	1.800000	-0.006413	-0.019033	-0.005358	-0.018620
19	1.900000	-0.005215	-0.019821	-0.005256	-0.019327
20	2.000000	-0.004796	-0.020508	-0.005159	-0.020003

21	2.100000	-0.005636	-0.021153	-0.005067	-0.020647
22	2.200000	-0.005499	-0.021902	-0.004979	-0.021262
23	2.300000	-0.005389	-0.022509	-0.004894	-0.021848
24	2.400000	-0.005468	-0.023112	-0.004814	-0.022407
25	2.500000	-0.005280	-0.023668	-0.004737	-0.022941
26	2.600000	-0.004964	-0.024270	-0.004664	-0.023449
27	2.700000	-0.005034	-0.024759	-0.004595	-0.023935
28	2.800000	-0.004301	-0.025247	-0.004528	-0.024398
29	2.900000	-0.004719	-0.025581	-0.004465	-0.024840
30	3.000000	-0.004414	-0.026025	-0.004404	-0.025261
31	3.100000	-0.004911	-0.026411	-0.004346	-0.025663
32	3.200000	-0.005111	-0.026873	-0.004291	-0.026047
33	3.300000	-0.003898	-0.027315	-0.004239	-0.026413
34	3.400000	-0.004461	-0.027557	-0.004189	-0.026762
35	3.500000	-0.003833	-0.027813	-0.004141	-0.027095
36	3.600000	-0.003692	-0.028126	-0.004095	-0.027412
37	3.700000	-0.003811	-0.028324	-0.004052	-0.027715
38	3.800000	-0.004348	-0.028620	-0.004010	-0.028004
39	3.900000	-0.004405	-0.028935	-0.003971	-0.028280
40	4.000000	-0.004391	-0.029157	-0.003933	-0.028543
41	4.100000	-0.004473	-0.029522	-0.003897	-0.028794
42	4.200000	-0.003726	-0.029797	-0.003863	-0.029033
43	4.300000	-0.004176	-0.029945	-0.003830	-0.029261
44	4.400000	-0.004041	-0.030135	-0.003798	-0.029479
45	4.500000	-0.003734	-0.030351	-0.003769	-0.029687
46	4.600000	-0.004398	-0.030524	-0.003740	-0.029885
47	4.700000	-0.004528	-0.030781	-0.003713	-0.030074

48	4.800000	-0.004364	-0.031089	-0.003687	-0.030254
49	4.900000	-0.003851	-0.031392	-0.003662	-0.030426
50	5.000000	-0.004275	-0.031608	-0.003639	-0.030590
51	5.100000	-0.003488	-0.031867	-0.003616	-0.030747
52	5.200000	-0.004129	-0.031972	-0.003595	-0.030896
53	5.300000	-0.003415	-0.032127	-0.003574	-0.031039
54	5.400000	-0.002908	-0.032175	-0.003555	-0.031175
55	5.500000	-0.003426	-0.032197	-0.003536	-0.031304
56	5.600000	-0.002905	-0.032248	-0.003519	-0.031428
57	5.700000	-0.002945	-0.032252	-0.003502	-0.031546
58	5.800000	-0.002959	-0.032197	-0.003485	-0.031659
59	5.900000	-0.003857	-0.032162	-0.003470	-0.031766
60	6.000000	-0.004283	-0.032284	-0.003455	-0.031868
61	6.100000	-0.003904	-0.032490	-0.003441	-0.031966
62	6.200000	-0.003717	-0.032604	-0.003428	-0.032059
63	6.300000	-0.003901	-0.032713	-0.003415	-0.032148
64	6.400000	-0.004434	-0.032914	-0.003403	-0.032233
65	6.500000	-0.003717	-0.033116	-0.003391	-0.032314
66	6.600000	-0.003381	-0.033278	-0.003380	-0.032391
67	6.700000	-0.002788	-0.033267	-0.003370	-0.032465
68	6.800000	-0.003621	-0.033253	-0.003360	-0.032535
69	6.900000	-0.003745	-0.033342	-0.003350	-0.032602
70	7.000000	-0.003136	-0.033419	-0.003341	-0.032666
71	7.100000	-0.003398	-0.033342	-0.003332	-0.032727
72	7.200000	-0.003197	-0.033439	-0.003324	-0.032785
73	7.300000	-0.003819	-0.033512	-0.003316	-0.032840
74	7.400000	-0.003593	-0.033652	-0.003308	-0.032893

75	7.500000	-0.003756	-0.033687	-0.003301	-0.032944
76	7.600000	-0.004296	-0.033717	-0.003294	-0.032992
77	7.700000	-0.003310	-0.033822	-0.003287	-0.033038
78	7.800000	-0.003600	-0.033808	-0.003281	-0.033082
79	7.900000	-0.003684	-0.033813	-0.003275	-0.033124
80	8.000000	-0.004076	-0.033915	-0.003269	-0.033164
81	8.100000	-0.003503	-0.034075	-0.003264	-0.033202
82	8.200000	-0.003989	-0.034090	-0.003259	-0.033238
83	8.300000	-0.004075	-0.034174	-0.003254	-0.033272
84	8.400000	-0.004365	-0.034355	-0.003249	-0.033306
85	8.500000	-0.003148	-0.034481	-0.003244	-0.033337
86	8.600000	-0.004010	-0.034367	-0.003240	-0.033367
87	8.700000	-0.003440	-0.034446	-0.003236	-0.033396
88	8.800000	-0.003184	-0.034452	-0.003232	-0.033423
89	8.900000	-0.003502	-0.034402	-0.003228	-0.033449
90	9.000000	-0.004142	-0.034367	-0.003225	-0.033474
91	9.100000	-0.004063	-0.034495	-0.003221	-0.033498
92	9.200000	-0.003750	-0.034588	-0.003218	-0.033520
93	9.300000	-0.003078	-0.034652	-0.003215	-0.033542
94	9.400000	-0.003062	-0.034570	-0.003212	-0.033563
95	9.500000	-0.003270	-0.034511	-0.003209	-0.033582
96	9.600000	-0.004051	-0.034490	-0.003207	-0.033601
97	9.700000	-0.003045	-0.034669	-0.003204	-0.033619
98	9.800000	-0.003074	-0.034609	-0.003202	-0.033636
99	9.900000	-0.003973	-0.034624	-0.003199	-0.033652
100	10.000000	-0.003841	-0.034750	-0.003197	-0.033668

H.2. Reference and Controlled Outputs

The content of simulation data files for reference and controlled outputs is shown in Table H.2.

Table H.2. Simulation result of reference and controlled outputs.

k	time	y ₁	y ₂	y _{m1}	y _{m2}
0	0.000000	-0.000000	0.000000	-0.000000	0.000000
1	0.100000	0.000320	-0.001217	0.000337	-0.001286
2	0.200000	0.000436	-0.001558	0.000472	-0.001691
3	0.300000	0.000563	-0.001908	0.000536	-0.001796
4	0.400000	0.000573	-0.001787	0.000576	-0.001800
5	0.500000	0.000611	-0.001802	0.000607	-0.001771
6	0.600000	0.000674	-0.001909	0.000633	-0.001733
7	0.700000	0.000668	-0.001731	0.000657	-0.001692
8	0.800000	0.000648	-0.001525	0.000680	-0.001652
9	0.900000	0.000685	-0.001561	0.000702	-0.001613
10	1.000000	0.000701	-0.001507	0.000723	-0.001576
11	1.100000	0.000719	-0.001466	0.000742	-0.001540
12	1.200000	0.000756	-0.001518	0.000761	-0.001507
13	1.300000	0.000753	-0.001401	0.000779	-0.001474
14	1.400000	0.000770	-0.001383	0.000796	-0.001443
15	1.500000	0.000770	-0.001300	0.000813	-0.001414
16	1.600000	0.000770	-0.001215	0.000828	-0.001386
17	1.700000	0.000832	-0.001386	0.000843	-0.001359
18	1.800000	0.000887	-0.001514	0.000858	-0.001334
19	1.900000	0.000837	-0.001235	0.000871	-0.001309
20	2.000000	0.000830	-0.001138	0.000884	-0.001286
21	2.100000	0.000897	-0.001335	0.000896	-0.001264

22	2.200000	0.000909	-0.001304	0.000908	-0.001243
23	2.300000	0.000919	-0.001279	0.000919	-0.001223
24	2.400000	0.000940	-0.001298	0.000930	-0.001203
25	2.500000	0.000943	-0.001254	0.000940	-0.001185
26	2.600000	0.000940	-0.001181	0.000950	-0.001168
27	2.700000	0.000958	-0.001198	0.000959	-0.001151
28	2.800000	0.000927	-0.001027	0.000968	-0.001135
29	2.900000	0.000961	-0.001125	0.000977	-0.001120
30	3.000000	0.000954	-0.001054	0.000985	-0.001105
31	3.100000	0.000994	-0.001171	0.000992	-0.001091
32	3.200000	0.001018	-0.001218	0.001000	-0.001078
33	3.300000	0.000958	-0.000935	0.001007	-0.001066
34	3.400000	0.000998	-0.001067	0.001013	-0.001054
35	3.500000	0.000968	-0.000921	0.001020	-0.001042
36	3.600000	0.000967	-0.000888	0.001026	-0.001031
37	3.700000	0.000980	-0.000916	0.001032	-0.001021
38	3.800000	0.001020	-0.001041	0.001037	-0.001011
39	3.900000	0.001031	-0.001055	0.001042	-0.001001
40	4.000000	0.001036	-0.001052	0.001047	-0.000992
41	4.100000	0.001051	-0.001071	0.001052	-0.000984
42	4.200000	0.001014	-0.000897	0.001057	-0.000976
43	4.300000	0.001045	-0.001002	0.001061	-0.000968
44	4.400000	0.001042	-0.000971	0.001065	-0.000960
45	4.500000	0.001029	-0.000900	0.001069	-0.000953
46	4.600000	0.001073	-0.001055	0.001073	-0.000946
47	4.700000	0.001088	-0.001086	0.001077	-0.000940
48	4.800000	0.001086	-0.001048	0.001080	-0.000934

49	4.900000	0.001064	-0.000928	0.001084	-0.000928
50	5.000000	0.001095	-0.001027	0.001087	-0.000922
51	5.100000	0.001055	-0.000844	0.001090	-0.000917
52	5.200000	0.001096	-0.000993	0.001093	-0.000911
53	5.300000	0.001057	-0.000827	0.001095	-0.000907
54	5.400000	0.001029	-0.000709	0.001098	-0.000902
55	5.500000	0.001060	-0.000830	0.001100	-0.000897
56	5.600000	0.001030	-0.000708	0.001103	-0.000893
57	5.700000	0.001033	-0.000718	0.001105	-0.000889
58	5.800000	0.001032	-0.000721	0.001107	-0.000885
59	5.900000	0.001085	-0.000930	0.001109	-0.000882
60	6.000000	0.001113	-0.001030	0.001111	-0.000878
61	6.100000	0.001096	-0.000941	0.001113	-0.000875
62	6.200000	0.001088	-0.000898	0.001115	-0.000871
63	6.300000	0.001102	-0.000941	0.001116	-0.000868
64	6.400000	0.001139	-0.001066	0.001118	-0.000865
65	6.500000	0.001102	-0.000899	0.001120	-0.000863
66	6.600000	0.001086	-0.000820	0.001121	-0.000860
67	6.700000	0.001050	-0.000682	0.001123	-0.000857
68	6.800000	0.001100	-0.000876	0.001124	-0.000855
69	6.900000	0.001109	-0.000905	0.001125	-0.000853
70	7.000000	0.001075	-0.000763	0.001126	-0.000851
71	7.100000	0.001089	-0.000824	0.001128	-0.000848
72	7.200000	0.001079	-0.000777	0.001129	-0.000846
73	7.300000	0.001118	-0.000923	0.001130	-0.000845
74	7.400000	0.001109	-0.000870	0.001131	-0.000843

75	7.500000	0.001119	-0.000908	0.001132	-0.000841
76	7.600000	0.001152	-0.001034	0.001133	-0.000839
77	7.700000	0.001096	-0.000804	0.001134	-0.000838
78	7.800000	0.001113	-0.000872	0.001134	-0.000836
79	7.900000	0.001118	-0.000892	0.001135	-0.000835
80	8.000000	0.001144	-0.000983	0.001136	-0.000833
81	8.100000	0.001114	-0.000849	0.001137	-0.000832
82	8.200000	0.001144	-0.000963	0.001137	-0.000831
83	8.300000	0.001151	-0.000983	0.001138	-0.000830
84	8.400000	0.001173	-0.001051	0.001139	-0.000829
85	8.500000	0.001104	-0.000767	0.001139	-0.000827
86	8.600000	0.001152	-0.000968	0.001140	-0.000826
87	8.700000	0.001121	-0.000835	0.001140	-0.000825
88	8.800000	0.001106	-0.000775	0.001141	-0.000825
89	8.900000	0.001123	-0.000849	0.001141	-0.000824
90	9.000000	0.001160	-0.000999	0.001142	-0.000823
91	9.100000	0.001159	-0.000980	0.001142	-0.000822
92	9.200000	0.001143	-0.000908	0.001143	-0.000821
93	9.300000	0.001105	-0.000751	0.001143	-0.000820
94	9.400000	0.001101	-0.000747	0.001144	-0.000820
95	9.500000	0.001112	-0.000796	0.001144	-0.000819
96	9.600000	0.001158	-0.000978	0.001144	-0.000818
97	9.700000	0.001103	-0.000743	0.001145	-0.000818
98	9.800000	0.001103	-0.000750	0.001145	-0.000817
99	9.900000	0.001157	-0.000960	0.001145	-0.000817
100	10.000000	0.001153	-0.000929	0.001146	-0.000816

H.3. Plant Error

The content of simulation data file for plant error is shown in Table H.3.

Table H.3. Simulation result of plant errors.

k	t	e_1	e_2
0	0.000000	0.00000	0.00000
1	0.100000	0.00002	0.00001
2	0.200000	0.00020	0.00003
3	0.300000	-0.00087	-0.00002
4	0.400000	-0.00033	-0.00020
5	0.500000	-0.00051	-0.00017
6	0.600000	-0.00113	-0.00021
7	0.700000	-0.00053	-0.00045
8	0.800000	0.00019	-0.00049
9	0.900000	-0.00012	-0.00042
10	1.000000	-0.00004	-0.00045
11	1.100000	-0.00001	-0.00052
12	1.200000	-0.00037	-0.00044
13	1.300000	0.00000	-0.00049
14	1.400000	-0.00005	-0.00043
15	1.500000	0.00019	-0.00038
16	1.600000	0.00044	-0.00038
17	1.700000	-0.00040	-0.00030
18	1.800000	-0.00106	-0.00041
19	1.900000	0.00004	-0.00049
20	2.000000	0.00036	-0.00050
21	2.100000	-0.00057	-0.00051

22	2.200000	-0.00052	-0.00064
23	2.300000	-0.00050	-0.00066
24	2.400000	-0.00065	-0.00071
25	2.500000	-0.00054	-0.00073
26	2.600000	-0.00030	-0.00082
27	2.700000	-0.00044	-0.00082
28	2.800000	0.00023	-0.00085
29	2.900000	-0.00025	-0.00074
30	3.000000	-0.00001	-0.00076
31	3.100000	-0.00057	-0.00075
32	3.200000	-0.00082	-0.00083
33	3.300000	0.00034	-0.00090
34	3.400000	-0.00027	-0.00080
35	3.500000	0.00031	-0.00072
36	3.600000	0.00040	-0.00071
37	3.700000	0.00024	-0.00061
38	3.800000	-0.00034	-0.00062
39	3.900000	-0.00043	-0.00065
40	4.000000	-0.00046	-0.00061
41	4.100000	-0.00058	-0.00073
42	4.200000	0.00014	-0.00076
43	4.300000	-0.00035	-0.00068
44	4.400000	-0.00024	-0.00066
45	4.500000	0.00004	-0.00066
46	4.600000	-0.00066	-0.00064
47	4.700000	-0.00082	-0.00071
48	4.800000	-0.00068	-0.00083
49	4.900000	-0.00019	-0.00097

50	5.000000	-0.00064	-0.00102
51	5.100000	0.00013	-0.00112
52	5.200000	-0.00053	-0.00108
53	5.300000	0.00016	-0.00109
54	5.400000	0.00065	-0.00100
55	5.500000	0.00011	-0.00089
56	5.600000	0.00061	-0.00082
57	5.700000	0.00056	-0.00071
58	5.800000	0.00053	-0.00054
59	5.900000	-0.00039	-0.00040
60	6.000000	-0.00083	-0.00042
61	6.100000	-0.00046	-0.00052
62	6.200000	-0.00029	-0.00055
63	6.300000	-0.00049	-0.00056
64	6.400000	-0.00103	-0.00068
65	6.500000	-0.00033	-0.00080
66	6.600000	0.00000	-0.00089
67	6.700000	0.00058	-0.00080
68	6.800000	-0.00026	-0.00072
69	6.900000	-0.00040	-0.00074
70	7.000000	0.00021	-0.00075
71	7.100000	-0.00007	-0.00061
72	7.200000	0.00013	-0.00065
73	7.300000	-0.00050	-0.00067
74	7.400000	-0.00029	-0.00076
75	7.500000	-0.00046	-0.00074
76	7.600000	-0.00100	-0.00072
77	7.700000	-0.00002	-0.00078

78	7.800000	-0.00032	-0.00073
79	7.900000	-0.00041	-0.00069
80	8.000000	-0.00081	-0.00075
81	8.100000	-0.00024	-0.00087
82	8.200000	-0.00073	-0.00085
83	8.300000	-0.00082	-0.00090
84	8.400000	-0.00112	-0.00105
85	8.500000	0.00010	-0.00114
86	8.600000	-0.00077	-0.00100
87	8.700000	-0.00020	-0.00105
88	8.800000	0.00005	-0.00103
89	8.900000	-0.00027	-0.00095
90	9.000000	-0.00092	-0.00089
91	9.100000	-0.00084	-0.00100
92	9.200000	-0.00053	-0.00107
93	9.300000	0.00014	-0.00111
94	9.400000	0.00015	-0.00101
95	9.500000	-0.00006	-0.00093
96	9.600000	-0.00084	-0.00089
97	9.700000	0.00016	-0.00105
98	9.800000	0.00013	-0.00097
99	9.900000	-0.00077	-0.00097
100	10.000000	-0.00064	-0.00108

H.4. Adaptive Gains

The content of simulation data file for adaptive gains is shown in Table G.4.

Table H.4. Simulation result of adaptive gains.

k	time	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8
0	0.000	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0004	1.0066
1	0.100	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0004	1.0066
2	0.200	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0004	1.0066
3	0.300	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0004	1.0066
4	0.400	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0004	1.0066
5	0.500	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0004	1.0066
6	0.600	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0066
7	0.700	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0066
8	0.800	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0066
9	0.900	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0004	1.0066
10	1.000	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0066
11	1.100	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0066
12	1.200	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0066
13	1.300	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0066
14	1.400	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0067
15	1.500	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0067
16	1.600	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0066
17	1.700	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0066
18	1.800	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0066
19	1.900	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0066
20	2.000	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0066
21	2.100	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0066
22	2.200	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0066
23	2.300	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0067
24	2.400	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0067
25	2.500	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0005	1.0067
26	2.600	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0427	0.0004	1.0067
27	2.700	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0428	0.0004	1.0067
28	2.800	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0428	0.0005	1.0067

29	2.900	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0428	0.0005	1.0067
30	3.000	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0428	0.0004	1.0067
31	3.100	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0428	0.0005	1.0067
32	3.200	0.3716	-0.5106	0.2806	-0.3100	1.0441	-0.0428	0.0004	1.0067
33	3.300	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0428	0.0005	1.0068
34	3.400	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0428	0.0005	1.0068
35	3.500	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0428	0.0005	1.0068
36	3.600	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0428	0.0005	1.0068
37	3.700	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0428	0.0005	1.0068
38	3.800	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0428	0.0005	1.0068
39	3.900	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0428	0.0005	1.0068
40	4.000	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0428	0.0005	1.0068
41	4.100	0.3716	-0.5106	0.2806	-0.3100	1.0442	-0.0428	0.0005	1.0068
42	4.200	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0068
43	4.300	0.3716	-0.5106	0.2806	-0.3099	1.0442	-0.0428	0.0005	1.0068
44	4.400	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0068
45	4.500	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0068
46	4.600	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0068
47	4.700	0.3716	-0.5106	0.2806	-0.3099	1.0442	-0.0428	0.0005	1.0068
48	4.800	0.3716	-0.5106	0.2806	-0.3099	1.0442	-0.0428	0.0005	1.0069
49	4.900	0.3716	-0.5106	0.2806	-0.3099	1.0442	-0.0428	0.0005	1.0069
50	5.000	0.3716	-0.5106	0.2806	-0.3099	1.0442	-0.0428	0.0005	1.0069
51	5.100	0.3716	-0.5106	0.2806	-0.3099	1.0442	-0.0428	0.0005	1.0069
52	5.200	0.3716	-0.5106	0.2806	-0.3099	1.0442	-0.0428	0.0005	1.0069
53	5.300	0.3716	-0.5106	0.2806	-0.3099	1.0442	-0.0428	0.0005	1.0069
54	5.400	0.3716	-0.5106	0.2806	-0.3099	1.0442	-0.0428	0.0005	1.0069
55	5.500	0.3716	-0.5106	0.2806	-0.3099	1.0442	-0.0428	0.0005	1.0069
56	5.600	0.3716	-0.5106	0.2806	-0.3099	1.0442	-0.0428	0.0005	1.0069
57	5.700	0.3716	-0.5106	0.2806	-0.3099	1.0442	-0.0428	0.0005	1.0069
58	5.800	0.3716	-0.5106	0.2806	-0.3099	1.0442	-0.0428	0.0005	1.0069
59	5.900	0.3716	-0.5106	0.2806	-0.3099	1.0442	-0.0428	0.0005	1.0069

60	6.000	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0069
61	6.100	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0069
62	6.200	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0069
63	6.300	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0069
64	6.400	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0069
65	6.500	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0069
66	6.600	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0069
67	6.700	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0069
68	6.800	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0069
69	6.900	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0069
70	7.000	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0069
71	7.100	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0069
72	7.200	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0069
73	7.300	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0069
74	7.400	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0069
75	7.500	0.3716	-0.5106	0.2806	-0.3099	1.0441	-0.0428	0.0005	1.0070
76	7.600	0.3716	-0.5106	0.2806	-0.3099	1.0440	-0.0428	0.0005	1.0070
77	7.700	0.3716	-0.5106	0.2806	-0.3099	1.0440	-0.0428	0.0005	1.0070
78	7.800	0.3716	-0.5106	0.2806	-0.3099	1.0440	-0.0428	0.0005	1.0070
79	7.900	0.3716	-0.5106	0.2806	-0.3099	1.0440	-0.0428	0.0005	1.0070
80	8.000	0.3716	-0.5106	0.2806	-0.3099	1.0440	-0.0428	0.0005	1.0070
81	8.100	0.3716	-0.5106	0.2806	-0.3099	1.0440	-0.0428	0.0005	1.0070
82	8.200	0.3716	-0.5106	0.2806	-0.3099	1.0440	-0.0428	0.0005	1.0070
83	8.300	0.3716	-0.5106	0.2806	-0.3098	1.0440	-0.0428	0.0006	1.0070
84	8.400	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0428	0.0005	1.0070
85	8.500	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0428	0.0006	1.0070
86	8.600	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0428	0.0006	1.0070
87	8.700	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0428	0.0006	1.0070
88	8.800	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0428	0.0006	1.0070
89	8.900	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0428	0.0006	1.0070
90	9.000	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0428	0.0006	1.0070

91	9.100	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0428	0.0006	1.0070
92	9.200	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0428	0.0006	1.0070
93	9.300	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0429	0.0005	1.0070
94	9.400	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0428	0.0005	1.0070
95	9.500	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0428	0.0005	1.0070
96	9.600	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0428	0.0006	1.0070
97	9.700	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0429	0.0005	1.0070
98	9.800	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0429	0.0005	1.0070
99	9.900	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0428	0.0005	1.0070
100	10.000	0.3716	-0.5106	0.2806	-0.3098	1.0441	-0.0429	0.0006	1.0070

REFERENCES

- [1] ARM Ltd. (2009, Oct. 19). ARM processor instruction set architecture. [Online]. Available: <http://www.arm.com/products/CPUs/architecture.html>
- [2] R. N. Shreve and G. T. Austin, "Petroleum Processing," in *Shreve's Chemical Process Industries*, 5th ed. New York: McGraw-Hill, 1984, pp. 713–725.
- [3] R. C. Darton, "Distillation and absorption technology: Current market and new developments," *Trans. IChemE*, vol. 70, pp. 435–437, Sep. 1992.
- [4] T. J. McAvoy, *Interaction Analysis Principles and Applications*. Durham, NC: Instrument Society of America, 1983.
- [5] J. L. Humphrey, A. F. Seibert, and R. R. Koort, *Separation Technologies – Advances and Priorities*, Washington. DC: US DOE Report, 1991.
- [6] B. Dahho, H. Youlal, A. Hmamed, and A. Majdoul, "Identification and control of a distillation column," in *IFAC 9th Triennial World Congress*, Budapest, Hungary, vol. 3, pp. 1719–1724, 1984.
- [7] G. C. Shen and W. K. Lee, "Adaptive inferential control for chemical processes with intermittent measurements," *Ind. Eng. Chem. Res.*, vol. 28, no. 5, pp. 557-563, May 1989.
- [8] D. E. Seborg, T. F. Edgar, and S. L. Shah, "Adaptive control strategies for process control: A Survey". *AIChE J.*, vol. 32, no. 6, pp. 881-913, Jun. 1986.
- [9] M. Agarwal and D. E. Seborg, "A multivariable nonlinear self-tuning controller," *AIChE J.*, vol. 33, no. 8, pp. 1379-1386, Aug. 1987.

- [10] H. T. Nguyen and N. Afzulpurkar, "Application of multivariable adaptive control design to natural gasoline plant," in *Proc. 3rd Asian Conf. Industrial Automation and Robotics*, Bangkok, Thailand, 2003, pp. 90–95.
- [11] K. S. Narendra and A. M. Annaswamy, *Stable Adaptive Systems*. Englewood Cliff, N.J.: Prentice Hall, 1989.
- [12] G. Stephanopoulos, *Chemical Process Control*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
- [13] Wikipedia. (2009, Sep. 12). *MATLAB* [Online]. Available: <http://en.wikipedia.org/wiki/MATLAB>
- [14] K. J. Astrom and B. Wittenmark, *Adaptive Control*. New York, NY: Addison-Wesley Publishing Company, 1995.
- [15] G. O. Mutambara, *Design and Analysis of Control Systems*. Boca Raton, FL: CRC Press, 1999.
- [16] *S3C44B0X RISC Microprocessor Datasheet*, Samsung Corporation, 1998.
- [17] C. D. Holland, *Fundamentals of Multicomponent Distillation*. New York: McGraw-Hill, 1981.
- [18] H. Kehlen and M. T. Ratzsch, "Complex multicomponent distillation calculations by continuous thermodynamics," *Chem. Eng. Sci.*, 42, pp. 221–232, 1987.
- [19] R. J. Fuentes and M. J. Balas, "Robust model reference adaptive control with disturbance rejection," in *2002 Proc. American Control Conf.*, Anchorage, AK, 2002, pp. 4003-4008.

- [20] G. Kreisselmeier and K. Narendra, "Stable model reference adaptive control in the presence of bounded disturbances," *IEEE Trans. Autom. Control*, vol. 27, no. 6, pp. 1169-1175, Dec. 1982.
- [21] B. Peterson and K. Narendra, "Bounded error adaptive control," *IEEE Trans. Autom. Control*, vol. 27, no. 6, pp. 1161-1168, Dec. 1982.
- [22] K. J. Astrom and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*. Englewood Cliffs, NJ: Prentice-Hall International, 1990.
- [23] G. A. McNeill and J. D. Sachs, "High Performance Column Control," *Chem. Eng. Prog.*, vol. 65, no. 3, pp. 33-39, Mar. 1969.
- [24] J. S. Moczek, R. E. Otto, and T. J. Williams, *Approximation Model for the Dynamic Response of Large Distillation Units*. Amsterdam: Elsevier, 1965.
- [25] S. Skogestad and M. Morari, "Understanding the dynamic behavior of distillation columns," *Ind. Eng. Chem. Res.*, vol. 27, pp. 1848-1862, Jun. 1987.
- [26] H. T. Nguyen and N. Afzulpurkar, "Symbolic Algebra Approach for Adaptive Controller Design," in *Proc. 4th Asian Conf. Industrial Automation and Robotics*, Bangkok, Thailand, 2005.
- [27] C. L. Phillips and H. Troy, *Digital Control System Analysis and Design*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
- [28] D. Hung, *Lecture Notes and Handouts of CMPE 264*. San Jose, CA: Computer Engineering Department, San Jose State University, 2008.

- [29] EventHelix Inc. (2009, Oct. 2). *NFS Protocol Diagrams* [Online]. Available: http://www.eventhelix.com/RealtimeMantra/Networking/NFS_Protocol_Sequence_Diagram.pdf
- [30] H. Li, *Lecture Notes and Handouts of CMPE 244*. San Jose, CA: Computer Engineering Department, San Jose State University, 2009.
- [31] *ARM Software Development Toolkit Version 2.50 User Guide*, ARM Ltd., 1998.
- [32] M. Felton, *CGI Internet Programming with C++ and C*. Upper Saddle River, NJ: Prentice Hall, 1997.
- [33] D. E. Seborg, T. F. Edgar, and S. L. Shah, "Adaptive control strategies for process control: A survey," *AIChE J.*, vol. 32, no. 6, pp. 881-913, Jun. 1986.
- [34] V. M. Popov, *Hyperstability of Automatic Control Systems*. New York: Springer, 1973.
- [35] W. L. Luyben, *Process Modeling Simulation and Control for Chemical Engineers*, 2nd ed. Auckland: McGraw-Hill, 1990.
- [36] R. Richardson, *Lehigh Distillation Control Short Course*. Lehigh, PA: Lehigh University, 1990.
- [37] W. L. Luyben, *Practical Distillation Control*. New York: Van Nostrand Reinhold, 1993.
- [38] P. Harriott, *Process Control*. New York: McGraw-Hill, 1964.
- [39] J. H. Perry, Ed., *Chemical Engineers' Handbook*, 6th ed. New York: McGraw-Hill, 1984.

- [40] R. G. E. Franks, *Modeling and Simulation in Chemical Engineering*. New York: Wiley-Interscience, 1972.
- [41] W. L. Nelson, *Petroleum Refinery Engineering*. Auckland: McGraw-Hill International Book Company, 1982.
- [42] W. L. McCabe and J. C. Smith, *Unit Operations of Chemical Engineering*. New York: McGraw-Hill, 1976.
- [43] M. V. Joshi, *Process Equipment Design*. New Delhi: Macmillan Company of India, 1979.
- [44] *Feasibility Study of Condensate Processing Plant*, Petrovietnam Corp., 1998.
- [45] W. C. Edmister, "Hydrocarbon adsorption and fractionation process design methods," *Petroleum Engineer*, vol. 21, pp. 45–50, 1949.
- [46] B. I. Lee, J. H. Erbar, and W. C. Edmister, "Properties for low temperature hydrocarbon process calculation," *AIChE J.*, vol. 19, pp. 349–356, Mar. 1973.