Loughborough University
Institutional Repository

Remote maintenance of real time controller software over the internet

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

- A Doctoral Thesis. Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University.

Metadata Record: https://dspace.lboro.ac.uk/2134/12484

Publisher: © Chengwei Dai

Please cite the published version.

This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

C O M M O N S   D E E D

Attribution-NonCommercial-NoDerivs 2.5

You are free:

- to copy, distribute, display, and perform the work

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor.

Noncommercial. You may not use this work for commercial purposes.

No Derivative Works. You may not alter, transform, or build upon this work.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the Legal Code (the full license).

Disclaimer

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/

Loughborough University

University Library

Author/Filing Title ....DAI, C.

Class Mark ....T

Please note that fines are charged on ALL overdue items.

FOR REFERENCE ONLY

0403270642

Remote Maintenance of Real Time Controller Software over the Internet

by

Chengwei Dai

A Doctoral Thesis

Submitted in partial fulfilment of the requirements
for the award of
Doctor of Philosophy of Loughborough University

October 2005

© by Chengwei Dai, 2005

Loughborough University
Pilkington Library

Date SEPT. 2006

Class T

Acc No. 0403270642

Abstract

The aim of the work reported in this thesis is to investigate how to establish a standard platform for remote maintenance of controller software, which provides remote monitoring, remote fault identification and remote performance recovery services for geographically distributed controller software over the Internet.

A Linear Quadratic Gaussian (LQG) controller is used as the benchmark for the control performance assessment; the LQG benchmark variances are estimated based on the Lyapunov equation and subspace matrices. The LQG controller is also utilized as the reference model of the actual controller to detect the controller failures. Discrepancies between control signals of the LQG and the actual controller are employed to a General Likelihood Ratio (GLR) test and the controller failure detection is characterized to detect sudden jumping points in the mean or variance of the discrepancies. To restore the degraded control performance caused by the controller failures, a compensator is designed and inserted into the post-fault control loop, which serially links with the faulty controller and recovers the degraded control performance into an acceptable range.

Techniques of controller performance monitoring, controller failure detection and maintenance are extended into the Internet environment. An Internet-based maintenance system for controller software is developed, which provides remote control performance assessment and recovery services, and remote fault identification service over the Internet for the geographically distributed controller software. The integration between the mobile agent technology and the controller software maintenance is investigated. A mobile agent based controller software maintenance system is established; the mobile agent structure is designed to be flexible and the travelling agents can be remotely updated over the Internet. Also, the issue of heavy data process and transfer over the Internet is probed and a novel data process and transfer scheme is introduced. All the proposed techniques are tested on simulations or a process control unit. Simulation and experimental results illustrate the effectiveness of the proposed techniques.

Acknowledgements

This work was carried out under the supervision of Dr Shuang-Hua Yang.
Firstly, I would like to express my gratitude towards my supervisor, Dr Shuang-Hua Yang, for his research guidance and introducing the subject of remote maintenance of controller software. I am grateful to him, for his diligent efforts to guide my PhD project, and valuable comments on my papers and this thesis.

I would like to thank my family, particularly, my mother, her encouragements gave me confidence and strength to pass through the tough time during my three years' foreign life, and my nephew YangYang, his naiveté and lovely smiles brought a lot happy memories to me. I greatly appreciate everything they have done for me.

I would also like to thank my friends, Mark, XiaoBing, Dezhi and Hongmei for their assistance in the past three years and in my thesis writing.

Last but not least, I would like to thank the Faculty of Science, Loughborough University to offer her generous financial aids to fund my PhD study.

Publications from this Research Work

1. **Dai, C.** and Yang, S., "An Approach for Control Software Supervision", Proceedings of the 9th Chinese Automation & Computing Society Conference in the UK, Hu, H., Yue, Y. and Zhao, Z. X. (eds), Pacilantic International Ltd, Colchester, England, 2003, pp. 41-46, ISBN: 0953389065.
2. **Dai, C.** and Yang, S., "Fault Detection in Model Predictive Controller", The 5th Asian Control Conference, IEEE, Melbourne, Australia, 2004, pp. 942-946, ISBN: 0734030169.
3. **Dai, C.** and Yang, S., "An Approach for Controller Fault Detection", Proceedings of the 5th World Congress on Intelligent Control and Automation, IEEE, Hangzhou, China, 2004, pp. 1637-1641, ISBN: 0-7803-8273-0.
4. **Dai, C.** and Yang, S., "Controller Performance Assessment with LQG Benchmark Obtained under Subspace Method ", UK Control 2004 Proceedings, Sahinkaya, M. N. and Edge, K. A. (eds), UKControl\'04, Bath, UK, 2004, ID-066, ISBN: 0861971302, [CD-ROM].
5. Yang, S. and **Dai, C.**, "Multi-rate Control in Internet Based Control System", UK Control 2004 Proceedings, Sahinkaya, M.N. and Edge, K.A. (eds), UKControl\'04, Bath, UK, 2004, ID-053, ISBN: 0861971302, [CD-ROM].
6. **Dai, C.** and Yang, S., **"**Maintaining Control Performance in Faulty Control Systems**"**, IEEE International Conference on Systems, Man & Cybernetics , W. Thissen, W., Wieringa, P., Pantic, M. and M. Ludema, M. (eds), IEEE, IEEE International Conference on SMC , The Hague, The Netherlands, 2004, pp. 5074-5078, ISBN: 0780385675 .
7. **Dai, C.** and Yang, S., "A Data Transfer Mechanism for Internet Based Application", Proceedings of the 10th Chinese Automation & Computing Society Conference in the UK, Liverpool, England, 2004, pp. 199-204.
8. **Dai, C.** and Yang, S., "An Approach for Controller Failure Detection and Maintenance" , Proceedings of the 11th Chinese Automation & Computing Society Conference in the UK, 2005, pp. 117-122 .

9. **Dai, C.** and Yang, S., "A Multi-Agent System for Control Performance Remote Monitoring", Proceeding of International Mediterranean Modeling Multiconference - I3M 2005, Marseilles, France, 2005, pp. 117-122.
10. **Dai, C.** and Yang, S., "Mobile Agent-Based Remote Monitoring and Maintenance of Process Controller Performance", *Measurement and Control* , 38(10), 2005, pp. 310-313, 319.

Table of Contents

Abstract ii

Acknowledgements iv

Publications from this Research Work v

Table of Contents vii

List of Tables xiii

List of Figures xiii

Chapter 1. Introduction 1

1.1 Introduction 1

1.2 Motivation 1

1.3 Objectives 2

1.4 Organization of this Thesis 4

Chapter 2. Overview of Remote Maintenance of Controller Software over the Internet 6

2.1 A Brief History of the Internet 6

2.2 Internet's Tomorrow 8

2.3 Software in Computer Aided Process Control Systems 11

2.3.1 General System Structure 11

2.3.2 System Software 12

2.4 Controller Software Maintenance 14

2.4.1 Control Performance Assessment 15

2.4.1.1 Performance Assessment Standards 15

2.4.1.2 Performance Assessment Approaches 16

2.4.2 Fault Detection in Control Systems 18

2.4.3 Failure Recovery in Control Systems 20

2.5 Remote Maintenance over the Internet 21

2.5.1 History 23

2.5.2 Structure of Remote Maintenance System 24

2.5.3 System Infrastructure 25

2.5.3.1 Communication Protocol 25

2.5.3.2 Control Modes in Remote Maintenance Systems 25

2.6 Issues Arising from Internet Based Maintenance Systems 26

2.6.1 Internet Traffic Time Delay 26

2.6.2 User Interface Design 27

2.6.3 Data Process Model 29

2.6.4 Distributed Object Transfer Mechanism 32

2.6.5 Security Measures 35

2.7 Conclusions 39

Chapter 3. Controller Performance Monitoring and Assessment **40**

3.1 Introduction 40

3.2 LQG Control as a Benchmark 40

3.2.1 LQG Control 40

3.2.2 LQG Benchmark vs. MVC Benchmark 42

3.3 LQG Solution via Subspace Approach 43

3.3.1 Subspace Identification 43

3.3.2 LQG Design 46

3.4 Controller Performance Assessment Based on LQG Benchmark 47

3.4.1 Calculation of LQG Benchmark Variances 47

3.4.2 Performance Measure 49

3.4.3 Calculation of LQG Benchmark via Classical Approach vs. Subspace Approach 50

3.5 Simulation 52

3.6 Conclusions 55

Chapter 4. Controller Fault Detection 57

4.1 Introduction 57

4.2 Challenges of Controller Fault Detection 58

4.3 Controller Fault Detection 59

4.3.1 LQG Control as a Benchmark for Controller Fault Detection 59

4.3.2 Change Detection Algorithm 60

4.3.3 Applying GLR Test to Controller Fault Detection 62

4.3.3.1 Decision Rule Based on Mean μ 63

4.3.3.2 Decision Rule Based on Variance σ 63

4.3.4 Data Collection 64

4.3.5 Controller Fault Detection Structure 65

4.4 Case Studies 66

4.4.1 SISO Simulation 66

4.4.2 MIMO Simulation 71

4.5 Conclusions 73

Chapter 5. Controller Failure Recovery 74

5.1 Introduction 74

5.2 Design of a Fault Shift-Based Compensator 74

5.2.1 Equivalent Fault Transfer 74

5.2.2 Compensator Design 75

5.2.3 Extension to MIMO Systems 77

5.2.4 Simulation Results 78

5.2.4.1 SISO Case Study 78

5.2.4.2 MIMO Case Study 80

5.3 Design of a Model Reference-Based Compensator 84

5.3.1 Compensator Design 84

5.3.2 Calculation of S_{11} 87

5.4 Controller Failure Detection and Maintenance Scheme 88

5.5 Case Studies 89
5.5.1 Experiment Description 90
5.5.2 SISO Case 90
5.5.2.1 Experimental Results 91
5.5.2.2 Calculation of Compensator Model 93
5.5.3 MIMO Case 94
5.5.3.1 Experimental Results 95
5.5.3.2 Compensation 99
5.6 Conclusions 100

Chapter 6. Data Process and Transfer over the Internet for Real time Applications 101
6.1 Introduction 101
6.2 Enabling Technologies 102
6.3 Real Time Data Process 103
6.3.1 Real Time Web Based Applications 103
6.3.2 Real Time Scientific Data in Web-based Applications 104
6.3.3 XML and HDF 106
6.3.4 Heterogeneous Scientific Data Processed by XML and HDF 108
6.4 Wrapping HDF by XML 109
6.4.1 HDF Objects Access from Java Applications 109
6.4.2 Wrapping HDF with XML 111
6.4.2.1 Structure Mapping 111
6.4.2.2 Data Mapping 113
6.5 Data Object Transfer Mechansim 114
6.5.1 RMI Based Data Transfer Structure 114
6.5.2 Data Object Priority 116
6.6 Case Study 119
6.6.1 System Description 120
6.6.2 Data Priority Establishment 123

6.6.3 Implementation 125
6.6.4 Simulation Results and Analysis 128
6.6 Conclusions 129

Chapter 7. Real Time Remote Maintenance of Controller Software over the Internet 131
7.1 Introduction 131
7.2 Architecture of Remote Maintenance System 132
7.2.1 Remote Maintenance Platform 132
7.2.2 Framework of Remote Maintenance System 137
7.3 Communcation Infrastructure 138
7.4 Implementation of Monitoring and Detection Level 139
7.4.1 Interaction among System Components 139
7.4.2 Parallel Computing 140
7.4.3 Parameter Tuner 141
7.5 Implementation of the Maintenance Level 142
7.5.1 Compensator Tcst 142
7.5.2 Compensator Implementation 144
7.6 Case Study 145
7.6.1 System Description 146
7.6.2 Remote Data Process and Transfer 148
7.6.3 Performance Monitoring and Fault Detection Levels 149
7.6.4 Compensation Taken Place 152
7.7 Conclusions 153

Chapter 8. Mobile Agent Based Remote Maintenance Of Controller Software 154
8.1 Introduction 154
8.2 Mobile Agent and Remote Maintenance of Controller Software 155
8.3 Architecture of the Agent Based Maintenance System 156

8.3.1 Agent Host 157
8.3.2 Agent Client 160
8.3.3 Agent Implementation on the Agent Host 164
8.3.4 Communication among Agents through Agent Server 165
8.4 Dynamic Updating of Mobile Agents 166
8.4.1 Requirements of Dynamic Updating 166
8.4.2 Agent Replacement 167
8.4.3 Reflection 168
8.4.4 Reflection in Mobile Agents 170
8.5 Case Study 173
8.5.1 System Description 174
8.5.2 Experiment Design and Result Analysis 175
8.5 Conclusions 180

9. Conclusions and Perspectives **182**
9.1 Conclusions of this thesis 182
9.2 Perspectives 184

References **186**

List of Tables

Table 2.1. Comparison between CORBA and RMI 35

Table 3.1. Controller tuning parameters 52

Table 3.2. Performance index 53

Table 5.1. System and reference models in SISO case 91

Table 5.2. System and reference models in MIMO case 99

Table 5.3. S_{ij} Matrices 99

Table 6.1. Data priority 125

Table 6.2. Heavy data XML file 126

Table 6.3. PID parameter XML file 128

Table 7.1. PID XML file 148

List of Figures

Figure 2.1. Control hierarchy of computer aided process control software 11

Figure 2.2. Control performance assessment standards 15

Figure 2.3. Structure of the remote system 24

Figure 2.4. Internet time delay 26

Figure 2.5. HDF5 structure 29

Figure 2.6. CORBA architecture 32

Figure 2.7. Java RMI architecture overview 33

Figure 3.1. Tradeoff curve 41

Figure 3.2. Classic approach vs. subspace approach 50

Figure 3.3. Process under MPC control with different parameters 54

Figure 3.4. Performance assessment based on the tradeoff curve 55

Figure 4.1. Potential faults in a control system 57
Figure 4.2. Controller fault detection structure 65
Figure 4.3. Water heater process 66
Figure 4.4. Process output 68
Figure 4.5. Control outputs of the PID and LQG controllers 69
Figure 4.6. Likelihood ratio results for testing the varied mean 69
Figure 4.7. Likelihood ratio for testing the varied variance 70
Figure 4.8. Process output under the disturbance impact 70
Figure 4.9. Likelihood ratio value under the disturbance impact 71
Figure 4.10. The discrete-time system controlled by decentralized controllers 71
Figure 4.11. The MIMO process governed by decentralized controllers 72
Figure 4.12. Likelihood ratio 73

Figure 5.1. Normal and abnormal control loops 75
Figure 5.2. Serial link compensator 76
Figure 5.3. Transferring controller fault 77
Figure 5.4. Serial link compensator for a PID controller 79
Figure 5.5. Faulty PID controller with and without compensation 79
Figure 5.6. Step response 82
Figure 5.7. Faulty control process with and without compensation 83
Figure 5.8. Compensator in the faulty control loop 85
Figure 5.9. Controller failure detection and maintenance scheme 89
Figure 5.10. PCU rig 90
Figure 5.11. Water level 92
Figure 5.12. Performance index 93
Figure 5.13. Possibilities density function (PDF) of the collected residuals 94
Figure 5.14. GLR test results 94
Figure 5.15. Water level 96

Figure 5.16. Water temperature 96
Figure 5.17. The monitored control performance 97
Figure 5.18. The mean value of the likelihood ratio test
(for the control signal for water level) 97
Figure 5.19. The variance value of the likelihood ratio test
(for the control signal for water level) 98
Figure 5.20. The mean value of the likelihood ratio test
(for the control signal for water temperature) 98
Figure 5.21. The mean value of the likelihood ratio test
(for the control signal for water temperature) 99

Figure 6.1. XML data structure 108
Figure 6.2. HDF data storage structure 109
Figure 6.3. Accessing HDF in the Java environment 110
Figure 6.4. Converting HDF structure into XML structure 112
Figure 6.5. HDF data mapping into JDOM 113
Figure 6.6. Data transfer structure based on RMI 115
Figure 6.7. Procedure of looking up the remote objects 117
Figure 6.8. A local reactor process 118
Figure 6.9. PID controller 119
Figure 6.10. PID controller configure setting 119
Figure 6.11. Architecture of remote monitoring system 121
Figure 6.12. HDF structure 121
Figure 6.13. Valve work status 121
Figure 6.14. PID parameters 122
Figure 6.15. Output flow rate and temperature 122
Figure 6.16. Output pressure and concentration 123
Figure 6.17. Data priority setting panel 123
Figure 6.18. Structure of HDF Wrapper class 128

Figure 7.1. Platform for controller software maintenance 132
Figure 7.2. Data flow 136
Figure 7.3. Structure of the remote maintenance system 137
Figure 7.4. XML/HDF object transfer via RMI 138
Figure 7.5. Interaction between the system components 140
Figure 7.6. Remote tuning parameters of the local control software 142
Figure 7.7. Components in the compensation level 143
Figure 7.8. Compensator mobility 144
Figure 7.9. Experimental system layout 146
Figure 7.10. Local control system 147
Figure 7.11. The interface of the remote maintenance system 147
Figure 7.12. Parameter change 150
Figure 7.13. Water level 150
Figure 7.14. Performance index 151
Figure 7.15. GLR test 151
Figure 7.16. Test result of the compensator 152
Figure 7.17. Compensator frame 153

Figure 8.1. Structure of the remote maintenance system 157
Figure 8.2. Structure of the agent manager 158
Figure 8.3. Structure of the LQGLoop agent 161
Figure 8.4. Interaction among agents 161
Figure 8.5. Compensator life cycle 164
Figure 8.6. Agent replacement 167
Figure 8.7. Service management unit 169
Figure 8.8. Procedures of implementing the agent services 170
Figure 8.9. Experitmental system layout 173
Figure 8.10. User interface of the remote agent based system 173
Figure 8.11. Agent management unit replacements 174
Figure 8.12. Measurement results 176

Figure 8.13. Record during updating operation 177
Figure 8.14. GLR test results 177
Figure 8.15. Compensator class replacement 180

Chapter 1. Introduction

1.1 Introduction

Control performance usually refers how well a controller and the controlled process work together in a single loop, which is of importance to control specialists and to a large number of process engineers who have inherited control systems within their plants. Estimates of the percentage of industrial process controllers with performance problems are surprisingly high; studies (Hugo, 2000; Barbara, 1999) indicate that anywhere from 66% to 80% of controllers are not performing as well as they should. This might be surprising in the light of the fact that about 95% feedback controllers in the process industry are relative simple linear controllers. Although the use of advanced model controls is now widespread, Proportional-Integral-Derivative (PID) controllers are by far dominant feedback control algorithms. It's fair to say that with the amount of research efforts put into control theories in the past decades, such as H_2 and H_∞, techniques about designing and commissioning controllers are matured enough to handle these simple linear controllers.

The problem is that the performance of the controller once worked well the associated software is prone to degrade over time due to external or internal unexpected factors (Pekilis and Seviora, 1997). With widely incorporating advanced control strategies into computer based control systems, the control software in the controller becomes more and more complex. Study (Zulkernine and Seviora, 2001) has shown that some faults, which elude detection efforts, do not surface until the software is operational although rigorous use of model checking, testing, and other technological innovations; building large fault-free software systems has proven nearly impossible in practice. On the other hand, running controller software needs to be on-line monitored and maintained in response to unpredictable work situations. When raw materials and/or workload change, the related controller software needs to be revisited to tune the parameters again. In some cases, it is required to re-design or update control strategies. Therefore, what is lacking is the maintenance of the controller software after commissioning.

1.2 Motivation

Last two decades have witnessed that Internet technology is becoming one of the most popular and promising research topics. The emerging Internet technology offers unprecedented interconnection capability and ways of distributing collaborative work and opens up a completely new and low-cost channel for selling, marketing, monitoring and manufacturing products. Past research on e-Automation (Wu and Buse, 2002), Internet-based control (Yang, et al., 2003a, 2003b) and e-Diagnostics (Sematech, 2002) has investigated the integration of control systems and Internet technologies to design Internet-enabled systems and demonstrated the great future for the next generation of control systems. However, few of research explore how to remotely maintain the controller software over the Internet in order to keep a healthy control performance. Actually, there are a number of benefits for remote maintenance of controller software over the Internet, including:

(1) The Internet can provide a worldwide information platform for sharing all kinds of information. Experts can remotely monitor and diagnose control systems, and remotely re-tune controller software units over the Internet;

(2) The Internet enables collaboration between skilled operators situated in geographically diverse locations; experts can cooperate with local operators to tune and maintain the running contorller software and do not need to travel to the real sites;

(3) Control system design and maintenance centres can be created, which operate on the web and provide central supports to all the authorized industrial processes. This web-based maintenance and analysis system is ideal for rapid prototyping, instructional use, and practical engineering applications.

1.3 Objectives

The aim of this project is to investigate how to carry out the remote maintenance of controller software in a systematic way via the Internet, and how to establish a standard platform prototype to remotely maintain the geographically distributed controller software. In details, an Internet-based controller software maintenance system will be established, which should provide the following services for the geographically distributed controller software:

1. Remotely monitoring control performance. The maintenance system on-line retrieves process variables transmitted from local control processes and assesses the control performance in terms of a certain performance index which compares the actually achieved objective function values with a benchmark values.

2. Remotely detecting controller software fault. When the control performance is inadequate, the maintenance system is to on-line identify the state of the running controller software. Early warning on urgent problems can provide invaluable benefits with appropriate actions taken to avoid serious process upsets.

3. Remotely recovering the degraded control performance. A compensator is designed in the remote maintenance system to recover the degraded performance caused by the faulty controller software. The compensator will be tested in the associated simulation before installed into the local processes, and then dispatched and implemented in the local process site to maintain the degraded control performance.

4. Heavy data transfer over the Internet. Remote maintenance of controller software involves heavy data transfer exchanging among the remote maintain system and local control systems. The bulky data transfer is significantly complicated by heterogeneity and limited traffic resources of the Internet. It is important for the maintenance system to have an effective data transfer mechanism to process and transfer the heavy and heterogeneous data.

As more and more applications are put on-line, and under the general competitive pressure to cut costs, control and optimization manpower at sites are generally scarce and overloaded; therefore, remote maintenance for the running controller software becomes an attractive option. Companies with multiple sites in remote locations can access, analyze, and react to information from the controller software faster and more efficiently through the maintenance centre over the Internet. Software suppliers can maintain the geographically distributed controller software for those without the internal expertise. It also virtually eliminates the need for any controller software supplier's expert to conduct on-site maintenance. Therefore time and money can both be significantly saved.

1.4 Organization of This Thesis

The organization of the thesis is as follows:
First of all, an overview of Internet technology implemented in controller software maintenance is presented in Chapter 2. Basic ideas of controller software maintenance are explained, and relevant challenges and new trends in the remote maintenance of controller software are discussed in details.

Techniques of controller performance monitoring, controller failure detection and maintenance are then reported in Chapters 3, 4 and 5. In Chapter 3, controller performance assessment is investigated in depth. This chapter uses a Linear Quadratic Gaussian (LQG) control as the performance assessment benchmark; a "model-free" method to estimate the LQG benchmark variances via the subspace method is proposed.

In Chapter 4, the issue of controller failure detection is investigated. Discrepancies between control signals of the LQG and the actual controller are employed to a General Likelihood Ratio (GLR) test and the controller failure detection is characterized to detect sudden jumping points of the discrepancies.

Chapter 5 explores the maintenance of degraded control performance caused by the faulty controller. A compensator is designed, which is to insert the post fault process and serially links with the faulty controller to recover the degraded control performance.

Chapters 6, 7 and 8 present the implementation of the techniques of controller monitoring, detection and maintenance into remote environments; issues arising from the implementation are investigated.

Chapter 6 discusses the heavy data process and transfer over the Internet. A novel data format is proposed, which incorporates the eXtensible Markup Language (XML) and

Hierarchical Data Format (HDF) to process the heavy and heterogeneous sampled process variables. The processed HDF objects and XML objects are sent across computer networks with the support of the Java Remote Method Invocation (RMI) data transfer infrastructure.

Chapter 7 proposes a methodology for the design of a controller software remote maintenance system and an Internet-based remote maintenance system for controller software is developed.

Chapter 8 incorporates the mobile agent technology into the controller software maintenance; a mobile agent based controller software maintenance system is established. The mobile agent structure is designed to be flexible, original services in traveling mobile agents can be upgraded and new services can be on-line added into and implemented.

The significant contributions of this thesis are outlined in Chapter 9 and future research directions are discussed. It is hoped that the work included in this thesis will help to stimulate further research in the field of remote maintenance of controller software and their applications.

Chapter 2. Overview of Remote Maintenance of Controller Software over the Internet

2.1 A Brief History of the Internet

"Internet" refers to the global information system that -- (i) is logically linked together by a globally unique address space based on the Internet Protocol (IP) or its subsequent extensions/follow-ons; (ii) is able to support communications using the Transmission Control Protocol/Internet Protocol (TCP/IP) suite or its subsequent extensions/follow-ons, and/or other IP-compatible protocols; and (iii) provides, uses or makes accessible, either publicly or privately, high level services layered on the communications and related infrastructure described herein.

---------- *Federal Networking Council (FNC), 1995.*

The Internet has revolutionized the computer and communications world. The Internet is a collection of individual networks, connected by intermediate networking devices, and functions as a single large network that information can be sent and received in computers anywhere and anytime. It also enables collaboration and interaction between individuals and their computers without regard for geographic locations.

The Internet was the result of some visionary thinking by researchers in the early 1960s who saw great potential in allowing computers to share information on research and development in scientific and military fields (Internet History, 2005). Researchers at the Massachusetts Institute of Technology (MIT) played an exploring role in theoretical development of the Internet. In 1960, J.C.R. Licklider of MIT developed the concept of “Galactic Network” in his groundbreaking work “Man Computer Symbiosis”, where people and computers would be able to cooperate in making decisions and controlling complex situations, the user interface would be friendly and simple to interact and the computing speed of computers would be enough fast to implement users’ decisions (Internet Pioneers, 2000). The theory packet switching underpinning the Internet connection was proposed by Leonard Kleinrock of MIT in July 1961. To explore this, a network connection between Massachusetts and California was built by Lawrence Roberts of MIT in 1965. This experiment demonstrated that computers could work together well and running programs could retrieve data as necessary on the remote machine.

The Internet was incubated by the creation of the Advanced Research Projects Agency Network (ARPANET) which was a large wide-area network proposed by Leonard Roberts in 1967 and developed by the United States Defense Advanced Research Project Agency in 1969 (ARPA History, 1999). The ARPANET embodied the basic principle of the Internet design, the idea of the open architecture networking, where the inter-connection of independent networks was voluntary and flexible; different types of computer with different transmission schemes could be freely selected to connect each other and integrate into networks. Teams from MIT, the National Physics Laboratory (UK) and RAND Corporation developed a distributed mesh topology network model which enabled messages and data freely sent and received by computers, known as an interface message processor (IMP) (Internet History, 2005). The first IMP went online in 1969 and provided communication services between computers at University of California, Los Angeles (UCLA) and Stanford. By December 1969 APRANET comprised four host computers as with the addition of research centres in Santa Barbara and Utah, USA. More and more computer networks were linked to the ARPANET net; by December 1971, 23 host computers were incorporated into the APRANET and UCLA students could 'login' to Stanford's computer, access its databases and try to send data.

The TCP/IP architecture proposed by researchers of Stanford University in the 70's 阿 accelerated the development of the Internet technology, and made the Internet step out of the labs and come into commercial applications. TCP/IP became the core Internet protocol and replaced the earlier Network Control Protocol which was proved to be incapable of keeping up with the growing network traffic load. In 1976, Robert M. Metcalfe invented what has come to be known as Ethernet, the local area networking (LAN) technology that turned PCs into communication tools by linking them together. Eventually, the technology was used to link more than 50 million PCs worldwide. Tim Berners-Lee made fundamental contributions of the web technology. In 1989, he proposed a new protocol for information distribution; in 1991, he built the first web site http://info.cern.ch/, which was first put online on August 6, 1991; in 1994, he founded the World Wide Web Consortium (W3C) to improve and build the standard of the Internet technology (Tim Berners-Lee, 2005).

The first really friendly Internet-based interface called a Gopher was developed by researchers of University of Minnesota in 1991 (Gopher, 1998). Gopher was a distributed document search and retrieval network protocol designed for the Internet. In 1992, the National Science Foundation issued a solicitation for the "InterNIC" which provided specific Internet services for users: directory and database services (by AT&T), registration services (by Network Solutions Inc.), and information services (by General Atomics/CERFnet). Marc Andersen and the University of Illinois developed a graphical user interface to the WWW, called "Mosaic for X". Later, Netscape Corp produced the most successful graphical type of browser and server, "Netscape". In 1990, the first Internet search-engine for finding and retrieving computer files, Archie, was developed at McGill University, Montreal. In 1991 the World Wide Web was released to the public.

Delphi was the first national commercial online service to offer Internet access to its subscribers; it opened up an email connection in July 1992 and full Internet services in November 1992 (Internet History, 2005). Microsoft's full scale entry into the browser, server, and Internet Service Provider market completed the major shift over to a commercially based Internet. The release of Windows 98 in June 1998 with the Microsoft browser well integrated into the desktop encouraged the enormous growth of the Internet.

2.2 Internet's Tomorrow

Internet technology has developed enough in recent decades that good facilities are available for developments and communication in most widely used languages("Internet", 2006) The technology is changing our traditional concepts about time and distance, and providing a great deal of benefits for process industries. It offers unprecedented interconnection capabilities and ways of distributing collaborative work and opens up a completely new channel for controlling, monitoring and maintaining control processes. Now, the Internet has evolved beyond its anarchic nature and haphazard interfaces, and become the basis for an international information infrastructure. The key issue to discuss the future Internet is not how the technology will change, but how the process of change and evolution itself will be managed. The Internet is now accessible to anyone who can afford a simple terminal, a modem and a phone line. As legions of new users come on-line, the Internet will continue to evolve and improve in the following possible directions:

Networks: The future network will expand its backbone capacity and will be able to carry morc information at faster speeds as more and more providers enter the market to strengthen the infrastructure. Strands of fibre, as thin as human hair, will carry information at the speed of light to homes and businesses. Coaxial cables that currently carry cable TV to many people's homes will be enabled to carry Internet traffic. This information will not only include textual data, but sound, graphics, and full-motion video, which will merge with the traditional programming on cable TV.

Internet Protocol: The new standard, Internet Protocol Version 6 (IPv6) was first devised in 1995. The migration to IPv6 from current Internet Protocol Version 4 (IPv4) is expected to be gradual in the future; and may take place over the next 5-10 years. A prime use of IPv6 addressing capability will be embedded systems--placing microprocessors in every device imaginable, from refrigerators to gas pumps, and linking them to the Internet for control and information gathering purposes. Thus the Internet will become the network for millions upon millions of intelligent devices.

Wireless Access: Both the Internet and wireless communications continue to grow at a fantastic rate, and a lot of work is underway to combine them in a more powerful and convenient fashion. The third generation (3G) wireless services promise to bring the Internet access anytime and anywhere, with a connection speed faster than most of us have at home right now and let users browse Web pages from cell phones, personal digital assistants (PDAs), and browser-equipped mobile telephones.

Services, Products & Variety: As a result of intelligent interfaces and faster links, commercial applications will become widespread on the Internet. Users will be able to wander through libraries and shopping malls at their homes or offices. Services may range from videophone applications to virtual reality environments where participants will interact with each other in imaginary worlds. In addition, it will be in the interest of the commercial organisations funding the network to support these activities, as it will increase the usage and utility of the network for everyone. The Internet will continue to evolve and serve society in ways that we could not have imagined several years ago, and in ways we can barely imagine today.

High Performance: The future of the Internet looks towards the commodity Internet and towards a high-performance academic and research Internet. High-performance Internet development today will feed through into future products exploiting higher speeds, more advanced switches, and better protocols. Super networks are key architectural elements of the IT infrastructure (cyber infrastructure), which will be installed worldwide for the advancement of scientific research and commodity applications. High-performance computing and communication technologies will allow for new levels of persistent collaboration over continental and transoceanic distances, coupled with the ability to process, disseminate, and share information on unprecedented scales, immediately benefiting the scientific and commercial communities.

Flexible Architecture: The future Internet architecture should support flexible, efficient, highly-dynamic mobility and should provide auto-configuration of end systems and routers. Furthermore, the Internet architecture should give users and network administrators the ability to allocate capacity among users and applications. In today's Internet, allocation occurs implicitly as a result of congestion control. The goal has generally been some approximation of "fairness"; all slow down together, but this is not always the right model. In the future Internet architecture, the allocation of the traffic resources can be more intelligent and flexible; the architecture can judge the user ID and the actual working situation, and the important users and emergent situations are prioritized to allocate network resources.

2.3 Software in Computer Aided Process Control Systems

Rapid advances in computer hardware and software technologies and advancement in the field of science and technology have resulted in a phenomenal growth in the applications of computer control to industrial processes. Over the last 30 years, the application of digital computer control to industrial processes has changed from the exceptional to the commonplace. Today, computers have become a normal component of process control systems.

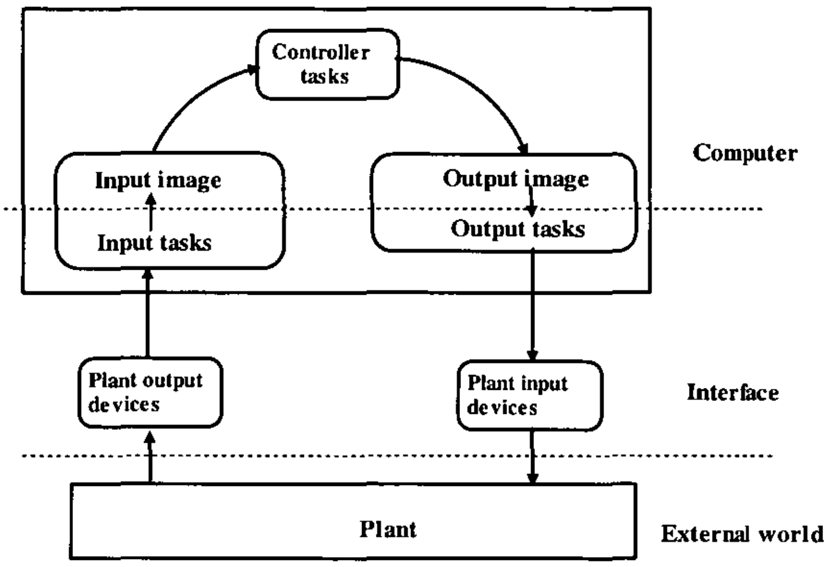


Figure 2.1. Generalized computer control system (Bennett, 1988)

2.3.1 General System Structure

The layout of the computer control system is shown in Figure 2.1. The input devices plus the input software provide the information to create an input image to the plant. The input image is a snapshot of the status of the plant and this snapshot is renewed at specified intervals. The external information is collected from the process of obtaining the snapshot and sent to the computer (Bennett, 1988).

The output image represents the current set of outputs generated by the control calculation. The output image will be updated periodically by the control tasks; it is the job of the output task to convey the output image to the plant. The control tasks can thus be considered as operating on an internal image of the plant. The communication with the operator is treated as a part of the plant input and plant output tasks. Control of the system can be extended to be shared between several computers on the distributed sites, and information is transmitted among computers. The overall operation of the system is sequential: the tasks, the plant input, the control, the plant output and communication being carried out in turn, with the sequence then being repeated indefinitely.

2.3.2 System Software

Computer aided control process software can be categorized into the following parts in terms of functions (Singh, 2004):

1. **Executive software**. The executive software supervises the operation of the computer system. It also assists the application software to perform its tasks. The main functions performed by executive software can be summarized as follows: (1) Setting time task to schedule the execution of the application programs; (2) Managing the operation of the hardware of the computer system, allocating specific functions with appropriate memories and storing data from external bulk memory into disks; (3) Overseeing input/output operations; (4) Serving the priority interrupt system; (5) Loading analog and digital inputs into memory.

2. **Application software**. Application software performs the main functions of the system. It handles all the specialized functions required for that particular installation or process. The important functions performed by application software are: (1) Data conversion between external devices and system software; (2) Scheduling optimization and control computations; (3) Running system functions; (4) Logging and associated managements.

3. **Support software**. The system support software aids operators to run the application program. In details, the support software provides the following

services: (1) Assemblers that convert programs written in a specific assembly language or a higher-level language into the machine language of the computer; (2) Editors, linking loaders and similar packages that allow segments of programs written separately to be incorporated into one program in the computer; (3) Programs that help debug applications programs. These programs may not be carried in the computer's memory at all times but may be entered when needed from external storage.

Some important features of the computer aided control process software are (Bennett, 1988):

- It executes at the scheduled sample intervals (with some specified tolerance) and that associated software components, which interact with the sensors and actuators, have critical time constraints.
- It allows the process control computer hardware system to perform time operations that are governed by a real time clock.
- It helps the hardware in responding to other externally generated occurrences through an external or priority interrupt system.
- It helps the hardware to read the values of external variables and transmit signals to external devices, including human interpretable system such as an operator's console.

Also, process control computer applications should be operated in real time and must be able to respond to interrupts from external devices. The nature and difficulty of producing a real time process control application depend on the complexity and time constraints of the problem. The tighter the time constraints with respect to the computer's clock speed, and the more the things that need to be serviced simultaneously (at least simultaneously from the view point of the control object), the more difficult will it be to complete the software design and implementation successfully. These interrupts may come from the real time clock input and output devices, the operator terminal, the process interface or from other devices on the data network.

2.4 Controller Software Maintenance

Software maintenance is the lifecycle phase that keeps an operational software system functioning even though the systems requirements are changing. Basically, software maintenance occurs for one of three reasons:

- to fix existing defects;
- to add new functionalities to a systems existing functionality;
- to tune parameters of the software to adapt the new working situation.

Different from the software development, the software maintenance has the following characteristics (Abran and Nguyenkim, 1993):

- The size and complexity of each maintenance work request are such that it can usually be handled by one or two resources;
- Maintenance work requests come in more or less randomly and cannot be accounted for individually in the annual budget-planning process;
- Minor enhancements (adaptive) work requests of the enhancement category are reviewed with customers and can be assigned priorities;
- The maintenance workload is not managed using project management techniques, but rather with queue management techniques;
- Maintenance has a broader scope of configuration management with more operational considerations.

Controller software maintenance is to keep the controller software in an optimal state for the purpose of maintaining a healthy performance of control systems. In details, it consists of three stages (Dai and Yang, 2003):

- Controller performance assessment. It is mainly concerned with quantification of controller performance. Usually, one uses some measures – whether achievable or ideal, subjective or objective – to determine control performance. Some of these measures are the steady-state offset, integrated absolute error (IAE), integrated squared error (ISE), and mean squared error (MSE). Performance measure is used to decide if the performance of a control system is satisfactory or not.
- Controller fault detection. When the controller performance is inadequate, it is important to identify the running state of the controller software. If the degraded performance is caused by a failure in controller software, the maintenance

operation should be taken.

- Controller failure maintenance. If the degradation of the performance is caused by a failure in the controller software, the faulty controller software will be compensated and the degraded control performance needs to be recovered.

In the following sections, past research on the above three fields is reviewed; achieved theoretical research and applications will be investigated and assessed.

2.4.1 Control Performance Assessment

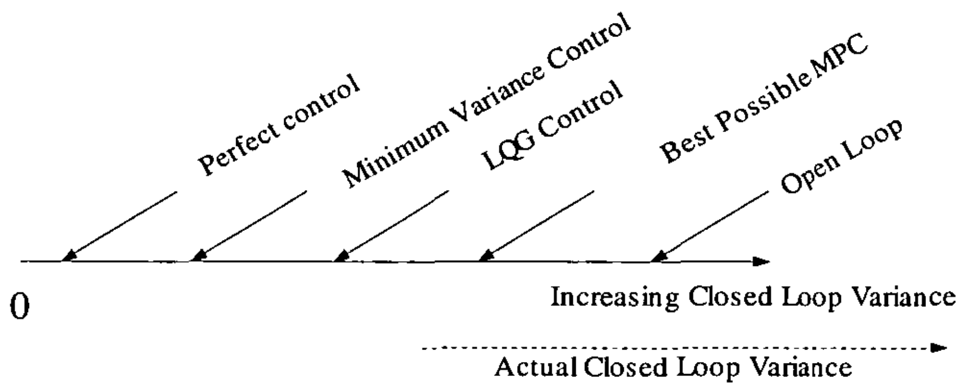


Figure 2.2. Control performance assessment standards (Hugo, 2000)

2.4.1.1 Performance Assessment Standards

There are many ways to assess the quality of process controllers, but in general they explicitly or implicitly involve a comparison of the current quality of control to some standards. Various standards can be ranked based on the tightness of control (Hugo, 2000) as shown in Figure 2.2.

Perfect Control: Perfect control has unrealistic requirements which are almost impossible to meet in actual applications, such as without noise, without model mismatch, and the controller having enough power to achieve control tasks. However, perfect control is often implicitly implemented in the control performance assessment; for example, in the output variance measurement, the actual output variance is implicitly compared to the perfect control with a zero variance.

Minimum Variance Control: Minimum Variance Control (MVC) is a popular benchmark for control performance assessment and was first proposed by Åström

(Åström, 1967). Minimum variance control gives the lowest achievable output variance and complete cancellation of the error (other than measurement noise) one sampling time after the process deadtime; however, MVC requires a perfect process model and a perfect disturbance model which are hard to meet in actual applications.

Linear Quadratic Gaussian (LQG) Control: The LQG controller is an unconstrained controller and provides a useful lower bound on the achievable performance of a linear controller. The infinite prediction horizon of the LQG algorithm endows the algorithm with powerful stabilizing properties.

Best Possible Model Predictive Controller (MPC): The main idea of MPC is to choose the control action by repeatedly solving on line an optimal control problem. This aims at minimizing a performance criterion over a future horizon, possibly subject to constraints on the manipulated inputs and outputs, where the future behavior is computed according to a model of the plant. Issues arise for guaranteeing closed-loop stability, to handle model uncertainty, and to reduce on-line computations. MPC response is generally not as tight as a Minimum Variance Controller, so the MPC performance index represents a more attainable standard.

Open Loop: An open-loop control system doesn't have or doesn't use feedback; it is mainly used for determining whether any control should be applied. Comparing closed loop control, the open loop control can only counteract against disturbances, for which it has been designed and become stable as long as the controlled object is stable.

2.4.1.2 Performance Assessment Approaches

Conventional estimation procedures compare the existing controller to a theoretical benchmark such as the Minimum Variance Controller (MVC). Harris (1989) laid theoretical foundations for performance assessment of the single loop from routine operating data. Time series analysis of the output error is used to determine the minimum variance control for the process. A comparison of the output variance term with the minimum achievable variance reveals how well the controller is currently doing. In his further work (Harris 1996, 1999), Harris proposed a performance index, which is a ratio of the achievable variance under the ideal MVC to the actual variance, and he also used the analysis of variance for feedforward/feedback control loops to indicate whether a poor performance is due to the feedforward/feedback controller.

Based on idea of MVC, Stanfelj et al. (1993) proposed the use of cross-correlation analysis to assess performance of feedforward/feedback systems. He also found that the poor feedback controller performance could be attributed to modeling error or poor controller tuning or external perturbations in the feedback systems. Normal operating data from a feedback system without any measured external perturbations cannot provide information for such a diagnosis.

Harris et al. (1996) and Huang et al. (1997a) generalized the minimum variance benchmark to the multivariate case based on the multivariate interpretation of the delay term, known as the interactor matrix. The interactor matrix plays a crucial role in determining the control invariant or the minimum variance control for a Multi-Input-Multi-Output (MIMO) process. However, estimation of the interactor matrix requires some closed loop excitations and cannot be based on routine operating data alone. Huang (1997b) showed that the interactor matrix obtained under closed loop conditions is the same as the open loop interactor. Huang and Shah (1999) gave a detailed exposition of univariate and multivariate, feedback and feedforward performance assessment for the case of both stochastic and deterministic inputs. The reviews by Qin (1998) and Harris et al. (1999) presented the state of research in controller performance assessment.

Some researchers adopt other measures to evaluate the control performance. Xia and Howell (2003) introduced using a loop performance index to evaluate the loop status, which is used as a qualitative measure of current loop performance. Juricek et al. (2001) introduced an approach based on process variables predictions to determine if an emergency limit will be violated in the future. The proposed approach complements existing fault diagnosis techniques and has prediction function. If diagnosis is not informative, e.g. the fault has not been seen before, the approach can predict future outputs to help to determine the impact of the fault and provide insights to the corrective action. Swanda and Seborg (1997) used dimensionless settling time and the dimensionless integral of the absolute values of the error to assess the performance of Proportional-Integral (PI) controllers from closed-loop response data for a setpoint step change. The benchmark values of several different closed loop systems are independent of the process model. Feonard (1997) proposed an ideal controller with respect to an integral of the absolute value of the error (IAE) criterion; the controller can be used as the assessment standard to evaluate the actual control performance. Isaksson (1996) developed a whole set of performance indices based on control structure limitations and intended control tasks (stochastic control, setpoint tracking or disturbance rejection). The measure of the achievable performance for a particular controller type can be obtained as a ratio between the actual integrated squared error (ISE) and the optimal ISE for the unrestricted controller structure. The indices take into account intended control tasks and controller structure constraints. The calculation of the indices requires knowledge of the controller and an estimation of the plant. In most cases one has to use extraneous test signals in order to estimate the plant model.

2.4.2 Fault Detection in Control Systems

There has been an increasing interest in fault detection in recent years, as a result of the increased degree of automation and the growing demand for higher performance, efficiency, reliability and safety in industrial systems (Angeli and Chatzinikolaou, 2004). Classical fault detection methods are model based detections. The basic idea of the detection method is that the parameters of the actual process are repeatedly estimated and the results are compared with the parameters of the reference model initially obtained under fault free conditions. Any substantial discrepancy indicates a change in the process and may be interpreted as a fault (Isermann et al., 1983; Kumamaru et al.1986., 1989).

In the case of very complex time-varying and non-linear systems, where reliable measurements are very complicated and valid mathematical models do not exist, a number of different methods have been proposed by researchers. These methods come from the area of Artificial Intelligence and allow the development of new approaches to fault detection in dynamical systems (Scherer and White, 1989; Tzafestas, 1989;

Bykat, 1991). Basila, Stefanek, and Cinar (1990) developed a supervisory expert system that uses object based knowledge representation to represent heuristic and model-based knowledge. Chen and Modarres (1992) developed an expert system, called FAX, to address the determination of the root cause of process malfunctions and suggestions for corrective action(s) to avert abnormal situations. Becraft and Lee (1993) have proposed an integrated framework comprising of a neural network and an expert system.

Trend modelling can also be used to explain the various important events happening in the process and to do malfunction diagnosis and predict future states. Cheung and Stephanopoulos (1990) have built a formal framework for the representation of process trends. They introduced triangulation to represent trends. A series of triangles constitute a process trend. Through this method, the actual trend always lies within the bounding triangle, which illustrates the maximum error in the representation of the trend. Janusz and Venkatasubramanian (1991) identified a comprehensive set of primitives using which any trend can be represented. They used a finite difference method to calculate the first and second derivative of the process trend changes and based on these values, the primitives are identified. Konstantinov and Yoshida (1992) proposed a qualitative analysis procedure with the help of an expandable shape library that stores shapes like decreasing concavely, decreasing convexly and so on.

Multivariate Statistical Process Control (MSPC) is originally developed for the monitoring of continuous and batch processes (Kresta et al., 1991; Nomikos and MacGregor, 1995). The name reflects an association with (univariate) statistical process control methods, and much of the technology finds some counterpart therein. MSPC has traditionally employed static linear analysis, and has avoided giving individual attention to all possible faults, preferring to look at deviations from normal operation in a generic sense. By exploiting the presence of highly correlated sensors in a typical process plant to offset the limitations of this simplicity, a very powerful address to the problem is arrived at. At present the practical technology remains firmly based on two techniques (or their close analogues): principal component analysis (PCA) and partial least squares (PLS), also known as projection to latent structures (Geladi and Kowalski, 1986). Advances are concentrated on a number of important fronts, including the non-linear and classification issues (Kramer, 1992, Wold, 1989, Dong and McAvoy 1996); the dynamic nature of plant data (Dayal and MacGregor, 1996), and configuration within a feedback setting (Chen et al., 1998).

2.4.3 Failure Recovery in Control Systems

Failure may occur in any system, especially in complex systems. Defects in sensors, actuators, in the process itself, or within the controller, can be amplified by the closed-loop control systems, and faults can develop into malfunctions of the loop (Blanke et al, 2001). The malfunctions can cause disastrous damages; the damages range from those affecting the qualities of final products to those involving dangers to property or loss of human life. The malfunctions must be mitigated and control systems should be maintained under ant fault scenario.

In the last decade, Fault Tolerant Controls (FTCs) have enjoyed tremendous successes in effectively accommodating defects in sensors or actuators. A number of theoretical results as well as application examples (Rizzoni and Min, 1991; Niemann and Stoustrup, 2003) have been described in the literature. The FTCs are usually based on a reconfigurable controller. The purpose of controller reconfiguration is to compensate for the effects of the failed component. The existing methods for the reconfigurable controller design such as the quadratic regulator (Zhang and Jiang, 2001) often assume that a perfect fault detection and diagnosis (FDD) scheme is available and the post-fault model of the system is known completely, and the reconfiguring algorithm is then developed according to the post fault model. Karsai et al (2002) used the fault adaptive control to manage the fault recovery, in which the plant is modelled as hybrid bond graph. An observer is developed to track system behaviour and the controller reconfiguration relies on a pre-developed controller library. Zhang and Jiang (2003) proposed an active fault tolerant control system, which explicitly incorporates an allowable system performance degradation in the event of partial actuator faults in the design process. This control system is based on model-following and command input management techniques and the degradation in dynamic performance is accounted for through a degraded reference model.

An eigen-structure assignment (EA) based algorithm was proposed in (Jiang, 1994).

In this approach, the post-fault eignvalues are assigned in an optimal way such that performance recovery of the original system is maximized. An extension to integrated FDI and reconfigurable control design using the EA algorithm has been developed in (Zhang and Jiang, 2001). An approach that is related to Pseudo-Inverse-Method (PIM) is the control distribution concept reported in (Celi, Huang and Shih, 1996). In this method, a control distribution matrix is computed based on the minimization of the H_2 normal of a quantity that provides a measure of the difference between failed and healthy systems. An on-line reconfiguration method that does not require the use FDI algorithm is the hybrid adaptive linear quadratic control proposed in (Ahmed-Zaid et al, 1991). Even though this design method does not need explicit fault information, it has an on-line fault accommodation capability. A hybrid approach which designs a fault tolerant control system utilizing system redundancy has been employed.

2.5 Remote Maintenance over the Internet

The relationship between information technologies and process control has reached a new stage (Valera et al., 2005), encouraging the creation of applications such as monitoring, control and maintenance over the Internet, such as tele-working(1997), tele-medicine(Lau et al., 2001), and tele-robotics(Urbancsek and Vajda, 2003), E-Maintenance(Koç and Lee, 2001). The emerging Internet technologies offer unprecedented interconnection capability and ways of distributing collaborative work, and there have great potential to bring the advantages of these ways of working to the high-level maintenance of control systems. These advantages include that:

- plants are remotely monitored and adjusted; as a result, plants can retrieve data and react to plant fluctuations from anywhere around the world at any time;
- skilled operators in geographically diverse locations are able to collaborate to execute control tasks; experts on the control center can remotely provide the consulting services for the operators;
- companies operate the maintenance of its equipment out of the hands of the users, sale its service with the products, and achieve near-zero-downtime ability to its products and systems.

2.5.1 History

The concept of Web-based remote sensing and collaborative diagnostic and maintenance system was initially proposed for medical care in 1988(Wang et al., 2004). The system was designed to enable doctors or small clinics in rural areas to obtain instantaneous consultations from specialists in urban hospitals (Sacile, 2000). Stefano et al. (1997) adopted a similar concept for industry. Wang et al. (2004) developed a Web-based maintenance system based on virtual instruments, to provide remote sensing, monitoring, and on-line fault diagnosis for equipment, together with a collaborative maintenance platform for international experts to interactively share their experiences in maintenance.

Lee (1997) laid out the framework for remote diagnostics and maintenance for manufacturing equipment. It has become a standard for others to follow. The remote diagnostic system includes video-conferencing and remote measurements delivered as close to real-time as possible. It also provides on-line fault diagnosis and supports a multi-user kind of collaborative consultation. E-maintenance (Lee et al., 2000) is transforming manufacturing companies to a service business to support their customers anywhere and anytime. E-maintenance provides companies with predictive intelligence tools (such as a watchdog agent) to monitor their assets (equipment, products, process, etc.) through Internet wireless communication systems to prevent them from unexpected breakdown. E-maintenance systems can be used to monitor, analyze, compare, reconfigure, and sustain the system via the Internet. In addition, these intelligent decisions can be harnessed through web-enabled agents and connect them to e-business tools (such as customer relation management systems) to achieve smart e-service solutions.

Chu and Ahn (2002) built an Internet-based advisory service (Repair Advisory Service, RAS) for composite repair, which is proposed to increase the efficiency of the repair process. Based on the client–server architecture, RAS offers time-effective sharing of engineering knowledge between remote locations, easy maintenance and update of the application software, and ubiquitous access to the services. The web browser is used as the user interface, which provides an easy access to the service and software friendly environment.

An Internet-based machine condition monitoring system has been developed by Orsagh et al. (2000) that allows off-site maintenance personnel to access machinery diagnostic information in real-time. The Internet provides an inexpensive vehicle for the collection and assessment of condition data from machines in distant plants that leads to improved maintenance scheduling. The on-line health monitoring system is designed to efficiently and reliably transfer acquired data to a remote server for processing and analysis then publish the results on a secure Web-site on a minute by minute basis.

E-service and Tele-service engineering (Lee, 1998) have been developed and applied in industrial processes (machine manufacture, nuclear power plants, robotics), as well as in biomedicine. Pro-Active Maintenance researches in the framework of industrial partnership are investigated by (Iung and Morel, 2000). The main idea is to develop the concept of Intelligent Manufacturing System tested at Business and Shop Floor levels. The research is completed within both a Chinese-European and industrial–academic cooperation sponsored by the European Community (Iung, 2003). The action aim is to better monitor and anticipate the industrial process degradation and failures by developing an evolution of the platform towards a new one supporting e-Maintenance. Kussel et al. (2000) proposed to characterise Tele-service (vs. E-service) using three main criteria:

- Geographical distance between the customer and the supplier. It implies that a supplier who is spatially separated from the customer provides the service;
- Use of information technology required carrying out the service in terms of remote information processing, storing and communication;
- Industrial service: The services have to be in the field of industrial services (maintenance, diagnosis, etc.).

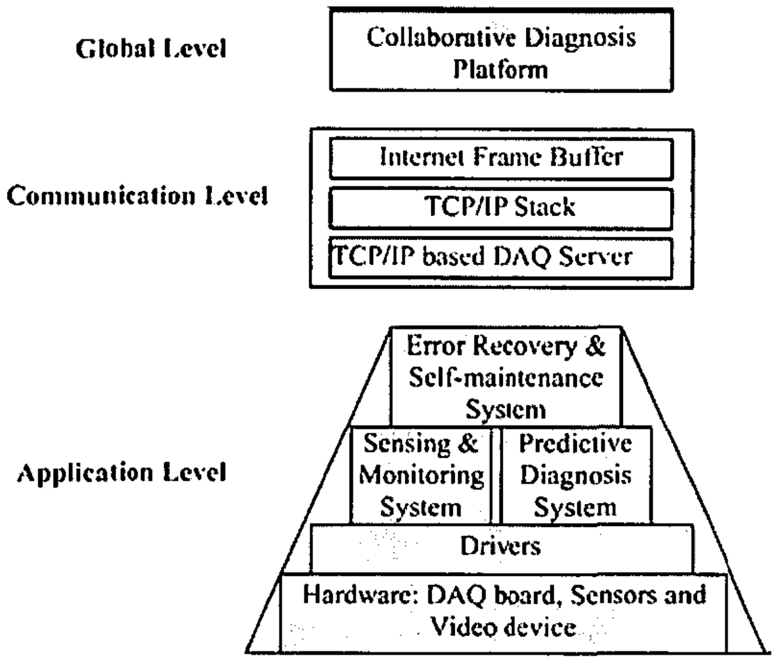

Global Level
Collaborative Diagnosis Platform
Communication Level
Internet Frame Buffer
TCP/IP Stack
TCP/IP based DAQ Server
Application Level
Error Recovery & Self-maintenance System
Sensing & Monitoring System
Predictive Diagnosis System
Drivers
Hardware: DAQ board, Sensors and Video device

Figure 2.3. Structure of the remote system (Wang et al, 2004)

2.5.2 Structure of Remote Maintenance System

A practical Web-based collaborative maintenance system proposed by (Wang et al, 2004) has three levels: the global level, the communication level, and the application level. Figure 2.3 shows the hierarchy of the three levels. The first level is the application level. It consists of the Data Acquisition (DAQ) server and various types of sensors, their interfaces and drivers, and DAQ cards. Major purpose of the application level is to eliminate the need for a struggle with distributed application development and deployment issues and to allow industry engineers to focus on application functionality, machine monitoring and fault diagnosis. The second level is the communication level; it sets up the necessary communication channels between the monitored machines and the Web browsers. Its roles are to receive users' commands from the Web browsers and then forward them either to the main Web server or to its embedded DAQ server. The third level is the global level which consists of the platform for collaborative maintenance.

2.5.3 System Infrastructure

2.5.3.1 Communication Protocol

The Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) protocols are used to establish the communication channel between the client and the server applications. The TCP is one of the core protocols of the Internet protocol suite. Using TCP, programs on networked computers can create connections to one another, over which they can send data. The protocol guarantees that data sent by one endpoint will be received in the same order by the other without any piece missing. TCP mechanism makes it a robust and reliable protocol, well suitable to transfer bulk data. However, this protocol has a drawback of having unpredictable arrival time of data (Bishop et al., 2004). Under UDP, programs on networked computers can be send short messages known as datagrams to another. UDP does not provide the reliability and ordering guarantees that TCP does; as a result, UDP is faster and more efficient for many lightweight or time-sensitive real time application (Bishop et al., 2005); however, datagrams transferred by UDP is unreliable; the datagrams may arrive out of order or go missing without notice (Serjantov et al., 2001).

2.5.3.2 Control Modes in Remote Maintenance Systems

Different levels of control modes in the remote maintenance system are implemented as follows (Bellamine et al., 2004):

- Continuous manual control mode: In this mode, remote users monitor the local running system and send direct primitive commands over the Internet to the local system. The remote users take full control responsibility for the local system; however, this mode is easy to be damaged by the Internet time delay, which can lead to irregular data transmission and data loss; in the worst case time delay can make the whole system unstable.
- Supervisory control mode: This mode can attack the disadvantage of continuous manual control mode caused by the time delay. The low-level control is located in the local site and is executed; the remote operators send supervisory commands and higher level instructions over the Internet to the local site.

- Observation mode: In this mode, the remote operators only monitor the running system over the Internet and can not send commands to control the operation of the local system.

2.6 Issues Arising from Internet-Based Maintenance Systems

Internet incorporated into maintenance systems has introduced many new features such as web-related traffic delay, web-based interfaces, uncertainty about who the users are, concurrent user access and web-related safety which should be considered in the design of Internet-based maintenance systems. These new features make the design methodology for remote maintenance systems be different from those for local maintenance systems and some issues are raised from the design of the Internet-based maintenance system, discussed as follows:

2.6.1 Internet Traffic Time Delay

In contrast to a typical distributed maintenance management system where the system load has been determined from conception, the Internet-based maintenance system has a variable working load. The Internet-based maintenance system has to encounter the uncertain transmitting time-delay and data-loss problems.

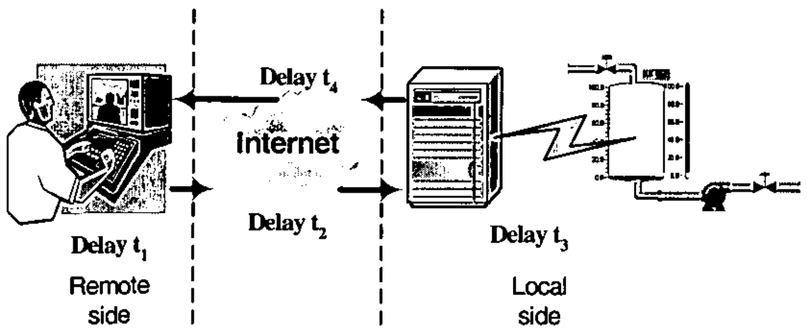


Figure 2.4. Internet Time delay (Yang et al., 2003a)

A block diagram of Internet-based systems is drawn in Figure 2.4. The total time of performing an operation (a control action) per cycle is $t_1 + t_2 + t_3 + t_4$; where the four types of time delay are:

- t_1 time delaying making control decision by a remote operator.
- t_2 time delaying transmitting a control command from the remote operator to the local system (the web server).

- t_3 execution time of the local system to perform the control action.
- t_4 time delaying transmitting the information from the local system to the remote operator.

A study (Luo and Chen, 2000) has shown that the time delay is temporal and variable. The Internet time delay increases with distance, but the delay depends also on the number of nodes traversed and the Internet load (Han et al., 2001). In detail, the Internet time delay is characterized by the processing speed of nodes, the load of nodes, the connection bandwidth, the amount of data, the transmission speed, etc. The Internet time delay $T_d(k)$ (i.e. t_2 and t_4) at instant k can be described as follows:

$$T_d(k)=\sum_{i=0}^{n}[\frac{l_i}{C}+t_i^R+t_i^L(k)+\frac{M}{b_i}]=\sum_{i=0}^{n}[\frac{l_i}{C}+t_i^R+\frac{M}{b_i}]+\sum_{i=0}^{n}t_i^L(k)=d_N+d_L(k) \qquad (2.1)$$

where l_i is the ith length of link, C is the speed of light, t_i^R the routing speed of the ith node, $t_i^L(k)$ the delay caused by the ith node's load, M the amount of data and b_i the bandwidth of the ith link, d_N is a term, which is independent of time, and $d_L(k)$.

2.6.2 User Interface Design

To some extent designing a web-based system is just like designing any other computer system and designers need to consider the following issues:

- **User**. The transactional nature of HTTP means that it is hard to know where a particular user has been before or when they've done. The most widely preached and important UI design principle is to understand who users are and what they want to do (Dix, 1999).
- **Space**. Good layout is no easy task at the best of times. Graphical toolkits such as X-Motif and Java Abstract Windowing Toolkit (AWT) often have complex layout rules to allow dynamic window resizing and there has been endless work on pretty icons and automatic text layout. It is probably fair to say that no existing solutions are perfect.
- **Flowchart**. Flowcharts are needed to indicate current process status and historical trend displays. In the control panel, the GUI displays the industrial process and makes it possible for the user to adjust the settings. The monitor

panel should provide a video and chatting system to reflect the current status of the process and to talk with other users respectively (Warnier et al., 2003).

- **Dynamic image.** The dynamic image is required to provide real time information about the current system status. The dynamic image is regularly generated by the server according to the system status, sent to the clients, and is automatically refreshed after a certain period of time. In order to achieve the above functions, the server push mechanism has been used. The basic principle of the server push mechanism is that the information sending action is based upon information changes, monitored by the server, rather than on the client request. This not only speeds up client information updating, but also reduces the server loading.
- **Suitable media.** It is required to choose suitable media for different interface tasks and minimize the amount of irrelevant information in the interface. The irrelevant information may obscure important information by attracting the attention of a user.

There are two popular tools to implement the user interface: LabVIEW and Java applet. LabVIEW is a highly productive graphical programming language for building data acquisition and instrumentation systems. It is a general purpose programming system with extensive libraries of functions for many programming tasks (Kasimir et al., 2001). LabVIEW includes libraries for data acquisition, data analysis, data presentation and data storage. A LabVIEW program is called a virtual instrument (VI) because its appearance and operation can imitate an actual instrument. LabVIEW also supports network communication protocols such as TCP and UDP. However, LabVIEW is complicated to use; for real industrial processes it could be better to write applets in Java (Yeung and Huang, 2001).

Java is a widely accepted networked programming language developed by Sun Microsystems. An Applet (a Java program loaded through the network from a Web server to a browser) creates a window that resides within a Web document. HTML reserves an area on the page for the Applet just as it handles a GIF file. The applet environment is completely isolated from the HTML. It depends on HTML for its distribution. But once it is loaded, a Java applet stands on its own and communicates only through a low level Internet programming service back to the server machine from which it was loaded. Only a Java-capable browser is required which will download the applets from the web server for running them on the client system; no specific software is required to be installed at the remote site.

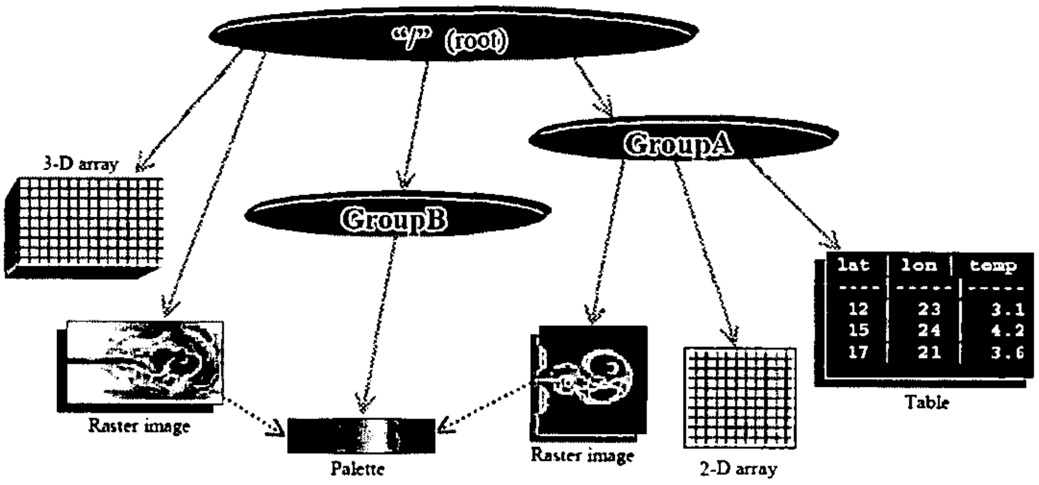


Figure 2.5. HDF5 Structure (HDF5, 2002)

2.6.3 Data Process Model

Internet-based applications in control systems involve heavy data exchange between the local site and the remote site. The data often collected and organized into collections that are distributed and are stored on heterogeneous storage systems. It is important for the Internet-based applications to efficiently manage and process these data and provide widely accepted services for clients to access, retrieve and manipulate the data in standard formats. The popular data formats used in a web environment include:

HDF 5

Hierarchical Data Format version 5 (HDF5) is a data format and an associated software library designed to store, access, manage, exchange, and archive diverse, complex data in continuously evolving heterogeneous computing and storage environments. HDF5 supports any type of data suitable for digital storage, regardless of its origin or size. The HDF5 format and library provide a powerful means of organizing and accessing data in a manner that allows scientists to share, process, and manipulate data in today's heterogeneous and quickly-evolving high-performance computational environment. HDF5 groups and links allow the creation of complex data dependencies reflecting the nature and/or intended usage of the stored data. The grouping mechanism combines related objects together; the linking mechanism, which is similar to the hard and soft links of UNIX environments, allows the sharing of objects between different groups.

HDF5 is extensively used with scientific research, engineering development, and other data formats. For example, petabytes of remote sensing data received from satellites (Almeida, 2003), terabytes of computational results from weather or nuclear testing models(Michelson et al., 2003) , and megabytes of high-resolution brain scans are stored in HDF5 files along with additional information necessary for efficient data exchange, data processing, visualization and archiving (NIFTI, 2003). Barkstrom (2001) described the use of the Ada95 language to bind HDF4 and HDF5. Bhattacharyya et al. (1998) proposed algorithms for determining the transmission rate associated with each multicast channel, based on static resource constraints, e.g. network bandwidth bottlenecks. Nam and Sussman (2003) implemented HDF format to store National Aeronautics and Space Administration (NASA) remote sensing data with a specific schema.

XML

eXtensible Markup Language (XML) is a mark-up language for documents containing structured information, which contains both content (words, pictures, etc.) and some indications of what role that content plays. With a tree structure to express the data, XML forms a standard for storing structured content in text files and for data exchange via the Internet. The main benefit of XML over other web-based document structures, such as most notably HTML, is that each "tag" in XML represents a data field, with the data held within this tag while HTML only deals with web presentation (e.g. defining text on a web page) and carries no information about the context of the data itself.

Some data formats based on XML have been proposed and applied, such as eXtensible Data Format (XDF, 2002), which is a common scientific data format based on XML and general mathematical principles that can be used throughout the scientific disciplines. Widener (2001) used message formats wrapped by the eXtensible Mark-up Language (XML) to provide flexible run-time metadata definition facilities for an efficient binary communication mechanism. Lehti and Fankhauser (2004) explored how the Web Ontology Language (OWL) can be used as a more abstract modelling layer on top of XML data sources, described by an XML Schema, to which extent the semantic relationships provided by OWL can be used for mapping heterogeneous data sources to a common global schema, and how the inference mechanisms of OWL can be used to check the consistency of such mappings.

ScadaOnWeb

Based on XML and Semantic Web technologies, ScadaOnWeb (2001) applied Semantic Web technologies to the domain of engineering data by making available an ontology for Supervisory Control And Data Acquisition (SCADA) applications, which integrates Resource Description Framework (RDF, 2001), Web Ontology Language and Mathematical Markup Language into an XML file to provide a general model for SCADA data. The XML file that defines the semantics can be combined with a binary file for effective handling of large amounts of structured numeric data inherent into most SCADA applications. The semantics wrapped around the structured numeric data enables the user to understand and to interpret the numbers. As a GIF or JPEG picture can be viewed by clicking on it, engineering data defined with this new Web data type can be displayed in any browser with a ScadaOnWeb plug-in. Standard transaction templates supporting the ScadaOnWeb technology enable the user to view and extract data as well as to download subsets from any *n*-dimensional dataset.

The ScadaOnWeb technology is widely applied into the flood warning system based upon remote sensors, the balance group energy management, flexible metering of domestic and small industrial consumers, condition based maintenance for electrical power systems and production sites (Dreyer, 2003).

2.6.4 Distributed Object Transfer Mechanism

An important characteristic of large computer networks such as the Internet or large corporate intranets is that they are heterogeneous. Distributed object transfer methodologies aim to allow application interoperability and independence of platform, operating system, programming language and even of network and protocol. In a distributed object system, each of these distributed object components interoperate as a unified whole. These objects may be distributed on different computers throughout a network, living within their own address space outside of an application, and yet appear as though they were local to an application. There are two popular distributed object transfer mechanisms discussed as follows:

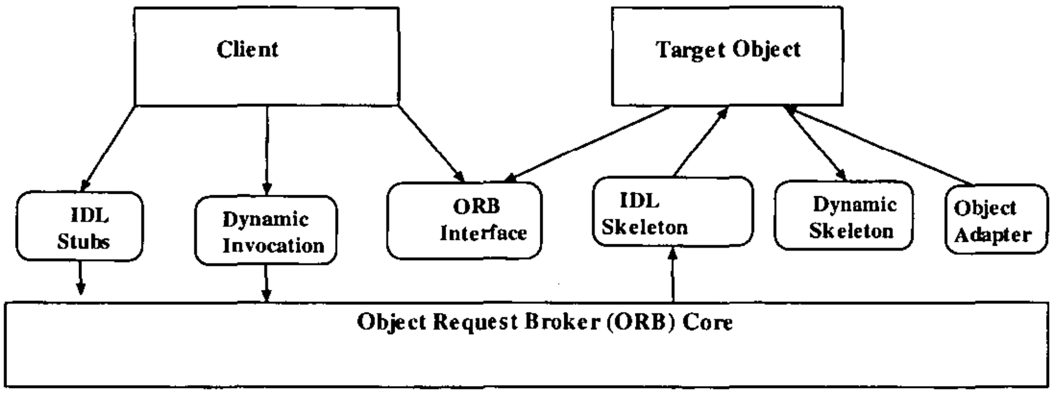


Figure 2.6. CORBA architecture

CORBA

The Common Object Request Broker Architecture (CORBA) is a specification that defines how distributed objects can interoperate. CORBA combines the benefits of object-orientation and distributed computing and is a middle-ware architecture that provides an object bus which allows components/ objects to communicate with each other across networks (Wang et al., 2001). The main components of CORBA are shown in Figure 2.6. The central component of CORBA is the Object Request Broker (ORB). The ORB Core is the most crucial part of the Object Request Broker; it is responsible for communication of requests. It encompasses the entire communication infrastructure necessary to identify and locate objects, handle connection management and deliver data. In general, the ORB is not required to be a single component; it is simply defined by its interfaces.

The basic functionality provided by the ORB consists of passing the requests from clients to the object implementations on which they are invoked. In order to make a request the client can communicate with the ORB Core through the Interface Definition Language (IDL) stub or through the Dynamic Invocation Interface (DII). The stub represents the mapping between the language of implementation of the client and the ORB core. Thus the client can be written in any language as long as the implementation of the ORB supports this mapping. The ORB Core then transfers the request to the object implementation which receives the request as an up-call through either an IDL skeleton, or a dynamic skeleton.

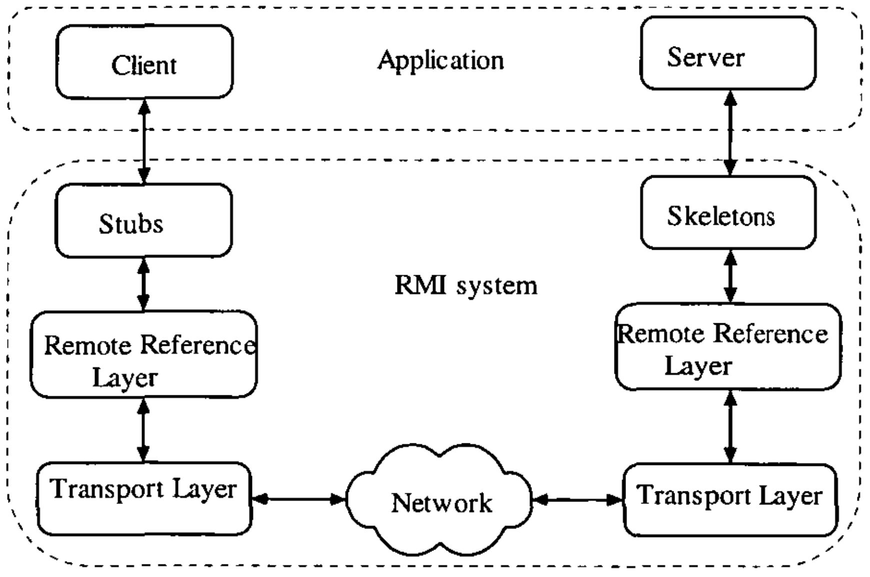


Figure 2.7. Java RMI architecture overview

RMI

Remote method invocation (RMI) is a popular middleware platform for Internet-based applications and enables remote communication between programs written in Java. The RMI allows Java developers to invoke object methods, and execute them on remote Java Virtual Machines (JVMs). Under the RMI, entire objects can be passed to and returned from a remote method as parameters and any Java object can be passed as a parameter and be sent across a network and dynamically loaded at a run-time by foreign virtual machines.

A high level architecture of the RMI framework is shown in Figure 2.7. All objects invoked remotely are implemented in an interface, which is a tag used to distinguish remote objects from normal objects. A client makes an invocation on a remote object by making a call to the stub object. The remote reference layer is responsible for carrying out the invocation. The transport layer is responsible for connection setup, connection management and keeping track of and dispatching to the remote objects. The skeleton for a remote object is a server-side entity that contains a method, which dispatches calls to the actual remote object implementation.

CORBA vs. RMI

Table 2.1 presents a comparison of the crucial features of CORBA ORB and RMI. CORBA's most important features are its language independent wire protocol, dynamic acquiring of object interfaces and the ability to compose and execute method invocations at run-time. They are supported by IIOP, interface repository and dynamic invocation interface. Other features include different parameter passing modes, persistent naming and persistent object references. RMI supports some features not found in CORBA/Java combination. These include dynamic class and stub downloads, object passing by value and URL based object naming. It can be seen that CORBA is a more complex architecture with more features.

A study (Hencko et al., 1998) shows that CORBA/Java is more suitable for large fully or partially web-enabled applications where legacy support is needed and good performances under heavy client load are crucial. Java RMI on the other hand is suitable for smaller fully web-enabled applications where legacy support can be managed by already existing bridges or is not necessary at all and where the ease of learning and the ease of use are more critical than performances are. In the single client scenario RMI is faster than CORBA to transfer the basic data types. The difference is especially noticeable when the client and the server are located on the same computer. A test employed by (Juric et al., 2000) suggests that when comparing CORBA's IDL string with RMI, RMI is 18% faster on single computer, but CORBA is 12% faster when client and server are on separate computers. It can be seen that CORBA handles network communication better than RMI.

Table 2.1 Comparison between CORBA and RMI (Patil et al., 1999)

Features supported only by CORBA ORB	Features supported only by RMI
Dynamic method invocations (Dynamic invocation interface)	Dynamic class downloading
Dynamic acquiring of object interfaces (Interface repository)	Dynamic stub downloading
Language independent wire-protocol (IIOP)	Object passing by value

Parameter passing modes (in, out, inout);
Persistent naming;
Persistent object references;
	URL based object naming

2.6.5 Security Measures

The difference between a normal remote maintenance system and a local maintenance system is that the communication medium is the Internet rather than any other private medium. The public Internet possesses security risks by nature of its open environment. Remote operation extending the scope of the local management system can be falsified by outsiders through the Internet. The strong need for information security on the Internet is attributable to several factors, including the massive interconnection of heterogeneous and distributed systems, the availability of high volumes of sensitive information at the end systems maintained by corporations and government agencies, easy distribution of automated malicious software by misfeasors, the ease with which computer crimes can be committed anonymously from across geographic boundaries, and the lack of forensic evidence in computer crimes, which makes the detection and prosecution of criminals extremely difficult.

There are various security measures implemented to protect the Internet applications from hostile attacks, including:

1. Authentication

Authentication establishes the identity of one party to another. Most commonly authentication establishes the identity of a user to some parts of the system, typically by means of a password. Passwords are generally combinations of a number of alphanumerical characters. Because most protocols pass them on as plain text over the network, they are easily broken by wire-tapping. Therefore it is highly recommended using encryption in combination with a one-time password protocol (Sandhu et al., 1996).

Public key infrastructure (PKI) is an arrangement which allows the third party to vet and vouch for user identities (Wing and O'Higgins, 1999). It also allows binding of public keys to users. PKI works as follows (Branched, 1997): (1) A password encrypted with the target's public key on the user side is authenticated on the target side; (2) A challenge code given by the target side is digitally signed with the user's private key on the user side, and the signed code is authenticated on the target side. This procedure is performed both on the user side and the operated side. Protecting the operation equipment can be realized by using a single firewall for the whole network or by giving each unit of operated equipment its own individual protection. However, in an actual application, this last case is not recommended because the amount of traffic on the network cannot be controlled and most of these systems are designed for compactness in consideration of limited resources (Furuya, Kato and Sekozawa, 2001).

2. Access Control

Access control determines what one party will allow another to do with respect to resources and objects mediated by the former. Access control usually requires authentication as a prerequisite. Access control is usually applied after authentication has been established. Access control can take several forms (Sandhu and Samarati 1997), including:

- **Discretionary Access Control**. Discretionary access control (DAC) is based on the idea that the owner of data should determine who has access to it. DAC allows data to be freely copied from object to object, so even if access to the original data is denied, access to a copy can be obtained (Bugliesi et al., 2004).

- **Mandatory Access Control**. In a Mandatory Access Control (MAC) Model, all subjects and objects are classified based on predefined sensitivity levels that are used in the access decision process. An important goal of a MAC model is to control the information flow in order to ensure confidentiality and integrity of the information, which is not addressed by DAC models (Bertino and Sandhu,2001).

- **Role-based Access Control:** Role-based access control (RBAC) requires that access rights be assigned to roles rather than to individual users (as in DAC) (Xue and Huai, 2005). Users obtain these rights by virtue of being assigned membership in appropriate roles. This simple idea greatly eases the administration authorizations.

- **Agent-based Approach**: With the increase of Internet applications, software agents are becoming popular as an emerging system-building paradigm. This paradigm can be effectively used to provide security features for Web applications. An agent is a process characterized by adaptation, cooperation, autonomy, and mobility. Some agent communication languages can be used to negotiate policies during conflicts for secure interoperation among participating policy domains. Agents can be assigned security enforcement tasks at the servers and client machines (James et al., 2001).

3. Intrusion Detection System

When a user of an information system takes an action which he or she is not legally allowed to take, it is called intrusion. The intruder may come from outside, or the intruder may be an insider who exceeds his limited authority to take action. Whether or not the action is detrimental, it is of concern because it might be detrimental to the health of the system, or to the service provided by the system. An intrusion detection system (IDS) inspects all inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system. There are several ways to categorize IDS (Kemmerer and Vigna, 2002):

- **Misuse detection vs. anomaly detection**: the misuse detection involves characterizing known ways to penetrate a system (Lunt, 1993). Each one is usually described as a pattern. The misuse detection system monitors for explicit patterns. The anomaly detection is to define and characterize the correct static form and/or acceptable dynamic behaviours of the system, and then to detect wrongful changes or wrongful behaviour (Konno and Tateoka, 2005). It relies on being able to define the desired form or behavior of the system and then to distinguish between that and undesired or anomalous behavior. The boundary between acceptable and anomalous form of stored code and data is precisely definable.

- **Network-based vs. host-based systems**: In a network-based system, or Network-based Intrusion Detection System (NIDS), the individual packets flowing through a network are analyzed. The NIDS can detect malicious packets that are designed to be overlooked by a firewall's simplistic filtering rules. A Host-based Intrusion Detection System consists of an agent on a host which identifies intrusions by analyzing system calls, application logs, file-system modifications (binaries, password files, capability/acl databases) and other host activities and states. The host based detection system looks at system logs for evidence of malicious or suspicious application activity in real time. It also monitors key system files for evidence of tampering (Vigna et al., 2004).

- **Passive system vs. reactive system**: In a passive system, the IDS detects a potential security breach, logs the information and signals an alert. In a reactive system, the IDS responds to the suspicious activity by logging off a user or by reprogramming the firewall to block network traffic from the suspected malicious source (Wang et al., 2001).

2.7 Conclusions

Internet technology offers unprecedented interconnection capability and ways of distributing collaborative work and opens up a completely new and low-cost channel for remote monitoring and maintaining for control processes. There are many topics relating to Internet-based maintenance systems. But in this thesis we will focus on remotely monitoring and maintaining the controller software over Internet. Theoretical research work and case studies in this area will be reported in the remainder of this thesis.

Chapter 3. Controller Performance Monitoring and Assessment

3.1 Introduction

A vast majority of industrial control loops have performance problems. Numerous investigations (Barbara, 1999) have shown that feedback controllers perform far less than optimally; 80% of loops have performance problems. And, as discovered at Techmation Inc and others when testing thousands of control loops in hundreds of operating plants, 30% of loops actually increase variability in the short term over manual control (Ingimundarson, 2003). In addition, pulp and paper mill auditing has shown that a large percentage of control loops actually de-stabilize product uniformity.

There are considerable commercial and academic interests in methods for analyzing the performance of control systems. These interests are motivated by the significant importance that control systems have enabled companies to achieve goals related to quality, safety and asset utilization, such as ensuring that controllers do not degrade to the point where they jeopardize final product quality; and determining whether or not advanced controls are required.

In this chapter, a novel method to assess the control performance is proposed; a Linear Quadratic Gaussian (LQG) control is used as the benchmark to assess the controller performance. Subspace approaches are adopted to design the LQG control and estimate the LQG benchmark variances. The only priori knowledge required is the system input and output data and the estimated Kalman filter gain. The rest of this paper is organized as follows. A brief description of LQG control is presented in Section 3.2. Section 3.3 discusses a subspace approach to design a LQG control. In Section 3.4, the estimation of the LQG benchmark variance is investigated. A simulation is presented in Section 3.5. Section 3.6 gives the conclusions.

3.2 LQG Control as a Benchmark

3.2.1 LQG Control

The LQG optimal control, also known as H_2 optimal control, was developed during the 1960s largely for aerospace applications, with refinement continuing into the 1970s.

The design method has been successfully used in a wide range of applications both in aerospace and in many other areas. The LQG algorithm endows the algorithm with powerful stabilizing properties and provides with the "limit of performance"for any linear controller in terms of the input and output variance. Figure 3.1 shows a typical performance curve for a LQG controller. The performance curve is sensitive to the ratio of the input to measurement noise variances. Uncertainty in the process and noise models also results in an uncertain performance curve. This curve can be obtained by solving the LQG problem, where the LQG objective function is defined by $J(\lambda) = E(y^2) + \lambda E(u^2)$.

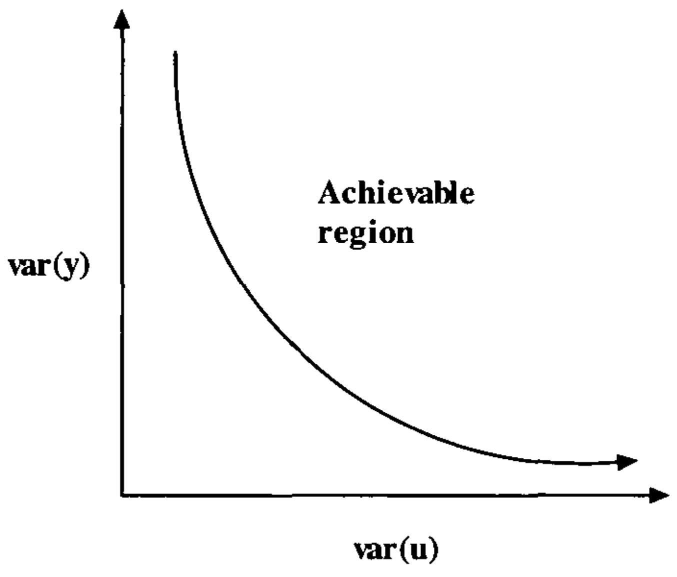


Figure 3.1. Tradeoff curve

By varying λ, various optimal solutions of $E(y^2)$ and $E(u^2)$ can be calculated. Thus a curve with the optimal output variance as ordinate and the manipulative variable variance as the abscissa can be plotted from these calculations. This curve defines the limit of performance of all linear controllers, as applied to a linear time invariant process. Five optimal controllers may be identified from the tradeoff curve summarized by Shah et al. (2001):

- Minimum cost control: The minimum cost control is online computing the minimum of cost function over time horizon and generating optimal control signals; it offers an offset-free control performance with the minimum possible control effort. This optimal control is located at the left end of the tradeoff curve.

- Least cost control: Facing the same process output error, the least cost control offers the least control efforts and makes the actuator take the least manipulative actions for a given output variance.
- Least error control: Slightly different from the least cost control, the least error control generates least output errors under the same control efforts as the existing controller. If the input variance is acceptable but the output variance has to be reduced then this represents the lowest achievable output variance for the given input variance.
- Tradeoff control: This optimal controller can be identified by drawing the shortest line to the tradeoff curve from the existing controller; the intersection is the tradeoff control. Clearly, this tradeoff controller has performance between the least cost control and the least error control. It offers a tradeoff between reductions of the output error and the control effort (Shah et al., 2001).
- Minimum variance control: The minimum variance control represents the theoretical lower bound on the achievable output variance; it can be identified at the right end of the tradeoff curve.

3.2.2 LQG Benchmark vs. MVC Benchmark

The area of performance assessment is concerned with the analysis of operating controllers. Performance assessment aims at evaluating controller performance from routine data. The classic methods use a Minimum Variance Controller (MVC) as a benchmark to compare the existing controller. Harris (1989) laid theoretical foundations for performance assessment of the single loop from routine operating data. The main advantage of using MVC as the benchmark to monitor the controller performance is that it requires little process information. However, the minimum variance control is not usually the control algorithm of choice in most practical situations owing to its demand for excessive control actions and poor robustness although performance assessment or optimal H_2 normal control as the benchmark does provide us with such useful information as a global lower bound of process variance or the H_2 normal measure. In (Kozub and Garcia, 1993), it is noted that often the minimum variance control is neither desirable nor practically achievable. What is considered to be a well functioning loop in the process industry will frequently have variance well above the minimum variance. The minimum variance benchmarking is therefore often too pessimistic and gives false alarms. A controller indicates which performs poorly according to MVC has potential to improve, but there is no guarantee that the performance will be improved by retuning it.

The LQG controller is an unconstrained controller and provides a useful lower bound on the achievable performance of a linear controller. The infinite prediction horizon of the LQG algorithm endows the algorithm with powerful stabilizing properties. Comparing to MVC, the LQG criterion is more practical. It takes both the control effort and output performance into account and the LQG benchmark represents the limits of performance for a linear system. LQG benchmark has been proposed in the literature to assess controller performance. However, the traditional method based on LQG (Huang and Shah, 1999) requires an explicit model identified from the inputs and outputs to design the LQG controller and then implements the controller into actual applications to calculate the benchmark variances. In the following sections, we will present a "model-free" method to design a LQG controller and estimate the LQG benchmark variances. The only priori knowledge required is the system input and output data and the estimated Kalman filter gain.

3.3 LQG Solution via Subspace Approach

3.3.1 Subspace Identification

Consider the following state space representation of a linear time-invariant system as:

$$\begin{aligned} x_{k+1} &= \mathbf{A}x_k + \mathbf{B}u_k + \mathbf{K}e_k \\ y_k &= \mathbf{C}x_k + \mathbf{D}u_k + e_k \end{aligned}$$

where the symbols are the input $u_k \in \Re^l$, the output $y_k \in \Re^m$, the state $x_k \in \Re^h$, $\mathbf{K} \in \Re^{h\times m}$ is the Kalman filter gain and the sequence $e_k \in \Re^m$ is a zero-mean Gaussian white noise. $\mathbf{A} \in \Re^{h\times h}$, $\mathbf{B} \in \Re^{h\times l}$, $\mathbf{C} \in \Re^{m\times h}$ and $\mathbf{D} \in \Re^{m\times l}$ are the matrices of the state space system.

With the subspace transformation, the above equation can be rewritten as:

$$\mathbf{Y}_f = \mathbf{\Gamma}_N \mathbf{X}_f + \mathbf{H}_N \mathbf{U}_f + \mathbf{H}_N^s \mathbf{E}_f \tag{3.1}$$

where

$$\mathbf{Y}_f = \begin{bmatrix} y_N & y_{N+1} & \cdots & y_{j-N+1} \\ y_{N+1} & y_{N+2} & \cdots & y_{j-N+2} \\ \cdots & \cdots & \cdots\cdot & \cdots\cdot \\ y_{2N-1} & y_{2N} & \cdots & y_j \end{bmatrix} \in \Re^{mN\times(j-2N+2)}$$

$$\mathbf{U}_f = \begin{bmatrix} u_N & u_{N+1} & \cdots & u_{j-N+1} \\ u_{N+1} & u_{N+2} & \cdots & u_{j-N+2} \\ \cdots & \cdots & \cdots\cdot & \cdots\cdot \\ u_{2N-1} & u_{2N} & \cdots & u_j \end{bmatrix} \in \Re^{lN\times(j-2N+2)}$$

$$\mathbf{E}_f = \begin{bmatrix} e_N & e_{N+1} & \cdots & e_{j-N+1} \\ e_{N+1} & e_{N+2} & \cdots & e_{j-N+2} \\ \cdots & \cdots & \cdots\cdot & \cdots\cdot \\ e_{2N-1} & e_{2N} & \cdots & e_j \end{bmatrix} \in \Re^{mN\times(j-2N+2)}$$

$$\mathbf{X}_f = [x_N \quad x_{N+1} \quad \cdots \quad x_{j-N+1}] \in \Re^{h\times(j-2N+2)}$$

$$\mathbf{\Gamma}_N = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \cdots \\ \mathbf{CA}^{N-1} \end{bmatrix} \in \Re^{mN\times h} \quad \mathbf{H}_N = \begin{bmatrix} \mathbf{D} & 0 & \cdots & 0 \\ \mathbf{CB} & \mathbf{D} & \cdots & 0 \\ \cdots & \cdots & \cdots\cdot & \cdots \\ \mathbf{CA}^{N-2}\mathbf{B} & \mathbf{CA}^{N-3}\mathbf{B} & \cdots & \mathbf{D} \end{bmatrix} \in \Re^{mN\times lN}$$

$$\mathbf{H}_N^s = \begin{bmatrix} \mathbf{I} & 0 & \cdots & 0 \\ \mathbf{CK} & \mathbf{I} & \cdots & 0 \\ \cdots & \cdots & \cdots\cdot & \cdots \\ \mathbf{CA}^{N-2}\mathbf{K} & \mathbf{CA}^{N-3}\mathbf{K} & \cdots & \mathbf{I} \end{bmatrix} \in \Re^{mN\times mN}$$, $\boldsymbol{I}$ is an identity matrix.

Set $\mathbf{W}_p = \begin{bmatrix} \mathbf{Y}_p \\ \mathbf{U}_p \end{bmatrix}$, where

$$\mathbf{Y}_p = \begin{bmatrix} y_0 & y_1 & \cdots & y_{j-2N+1} \\ y_1 & y_2 & \cdots & y_{j-2N+2} \\ \cdots & \cdots & \cdots\cdot & \cdots\cdot \\ y_{N-1} & y_N & \cdots & y_{j-N} \end{bmatrix} \in \Re^{mN\times(j-2N+2)}$$

$$\mathbf{U}_p = \begin{bmatrix} u_0 & u_1 & \cdots & u_{j-2N+1} \\ u_1 & u_2 & \cdots & u_{j-2N+2} \\ \cdots & \cdots & \cdots\cdot & \cdots\cdot \\ u_{N-1} & u_N & \cdots & u_{j-N} \end{bmatrix} \in \Re^{lN\times(j-2N+2)}$$

The indices p and f stand for past and future.

In the subspace identification, we often use the orthogonal projection of the row space matrices. The projection of the row spaces of $\mathbf{M} \in \Re^{p \times j}$ into the row space of $\mathbf{N} \in \Re^{q \times j}$ denoted by $\mathbf{M}/\mathbf{N}$ is defined as (Favoreel et al., 1998b):

$$\mathbf{M}/\mathbf{N} = \mathbf{M}\mathbf{N}^{\downarrow}\mathbf{N}$$

where $^{\downarrow}$ denotes the Moore-Penrose pseudo inverse.

The subspace identification can be considered to find the optimal prediction of the future outputs $\hat{\mathbf{Y}}_f$ based on the past and present inputs and outputs $\mathbf{W}_p$ and the future inputs $\mathbf{U}_f$ with the following linear predictor:

$$\hat{\mathbf{Y}}_f = \mathbf{L}_w \mathbf{W}_p + \mathbf{L}_u \mathbf{U}_f \tag{3.2}$$

where $\mathbf{L}_w$ and $\mathbf{L}_u$ are subspace matrices.

The prediction $\hat{\mathbf{Y}}_f$ can be obtained through solving a least square problem.

$$\min_{L_w, L_u} \left\| \mathbf{Y}_f - (\mathbf{L}_w, \mathbf{L}_u) \begin{pmatrix} \mathbf{W}_p \\ \mathbf{U}_f \end{pmatrix} \right\|_F^2$$

where $\|\bullet\|_F^2$ denotes the Frobenius-norm of a matrix.

The solution to this problem is the orthogonal projection of the row space $\mathbf{Y}_f$ into the row space spanned by $\mathbf{W}_p$ and $\mathbf{U}_f$.

$$\hat{\mathbf{Y}}_f = \mathbf{Y}_f / \begin{pmatrix} \mathbf{W}_p \\ \mathbf{U}_f \end{pmatrix} = \underbrace{\mathbf{Y}_f /_{\mathbf{U}_f} \mathbf{W}_p}_{\mathbf{L}_w \mathbf{W}_p} + \underbrace{\mathbf{Y}_f /_{\mathbf{W}_p} \mathbf{U}_f}_{\mathbf{L}_u \mathbf{U}_f}$$

Set the number of columns in the data matrices infinity ($j \rightarrow \infty$), then we have (Favoreel and De Moor, 1998a):

$$\hat{\mathbf{Y}}_f = \mathbf{\Gamma}_N \mathbf{X}_f + \mathbf{H}_N \mathbf{U}_f \tag{3.3}$$

and

$$\mathbf{\Gamma}_N \mathbf{X}_f = \mathbf{L}_w \mathbf{W}_p$$
$$\mathbf{H}_N = \mathbf{L}_u$$

With the singular value decomposition (SVD) of $\mathbf{L}_w$, we have:

$$\mathbf{L}_w = (\mathbf{U}_1 \quad \mathbf{U}_2)\begin{pmatrix} \mathbf{S}_1 & 0 \\ 0 & \mathbf{S}_2 \end{pmatrix}\begin{pmatrix} \mathbf{V}_1^T \\ \mathbf{V}_2^T \end{pmatrix}$$

where $\mathbf{U}_1$, $\mathbf{U}_2$, $\mathbf{V}_1$ and $\mathbf{V}_2$ are square matrices, and $\mathbf{S}_1$ and $\mathbf{S}_2$ are diagonal matrices.

$\mathbf{\Gamma}_N$ can be obtained by(Favoreel et al., 1998a) :

$$\mathbf{\Gamma}_N = \mathbf{U}_1 \mathbf{S}_1^{\ 1/2}$$

Therefore, the subspace matrices $\mathbf{\Gamma}_N$ and $\mathbf{H}_N$ in Equation 3.1 can be identified based on the system input and output data without knowing the system model.

3.3.2 LQG Design

Consider a LQG control law:

$$J_{LQG} = \sum_{k=N}^{2N-1} \{ (\hat{y}_k - r_k)^T Q_k (\hat{y}_k - r_k) + u_k^{\ T} R_k u_k \}$$

where $\hat{y}_k$ is the predicted output, u_k the future input, r_k is the reference trajectory, Q_k and R_k are user defined weight matrices.

Set the past/future input sequence $\mathbf{u}_p / \mathbf{u}_f$ is the first column of past/future matrix, $\mathbf{U}_p / \mathbf{U}_f$; correspondingly, past/future output sequence $\mathbf{y}_p / \hat{\mathbf{y}}_f$ is the first column of past/future predict matrix, $\mathbf{Y}_p / \hat{\mathbf{Y}}_f$;

$$\mathbf{u}_p = \begin{bmatrix} u_0 \\ u_1 \\ \cdots \\ u_{N-1} \end{bmatrix}, \ \mathbf{u}_f = \begin{bmatrix} u_N \\ u_{N+1} \\ \cdots \\ u_{2N-1} \end{bmatrix}, \ \mathbf{y}_p = \begin{bmatrix} y_0 \\ y_1 \\ \cdots \\ y_{N-1} \end{bmatrix}, \text{and } \hat{\mathbf{y}}_f = \begin{bmatrix} \hat{y}_N \\ \hat{y}_{N+1} \\ \cdots \\ \hat{y}_{2N-1} \end{bmatrix}.$$

and the sequence $\mathbf{w}_p$ is defined as:

$$\mathbf{w}_p = \begin{pmatrix} \mathbf{y}_p \\ \mathbf{u}_p \end{pmatrix}$$

Suppose the reference trajectory is zero, the optimal control criterion can be re-written as:

$$J_{LQG} = \hat{\mathbf{y}}_f^T \mathbf{Q}_f \hat{\mathbf{y}}_f + \mathbf{u}_f^T \mathbf{R}_f \mathbf{u}_f \tag{3.4}$$

where

$$\mathbf{Q}_f = \begin{bmatrix} Q_N & 0 & 0 & 0 \\ 0 & Q_{N+1} & 0 & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & Q_{2N-1} \end{bmatrix} \qquad \mathbf{R}_f = \begin{bmatrix} R_N & 0 & 0 & 0 \\ 0 & R_{N+1} & 0 & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & R_{2N-1} \end{bmatrix}$$

Based on Equation 3.2, the performance criterion J_{LQG} can be written as:

$$J_{LQG} = (\mathbf{L}_w \mathbf{w}_p + \mathbf{L}_u \mathbf{u}_f)^T \mathbf{Q}_f (\mathbf{L}_w \mathbf{w}_p + \mathbf{L}_u \mathbf{u}_f) + \mathbf{u}_f^T \mathbf{R}_f \mathbf{u}_f$$

The LQG problem can be solved by putting the trace of the derivative of J_{LQG} with respect to the input sequence $\mathbf{u}_f$ to zero:

$$\frac{\partial J_{LQG}}{\partial u_f} = 0$$

The subspace based LQG control law is calculated as (Favoreel et al., 1998b):

$$\mathbf{u}_f = -(\mathbf{R}_f + \mathbf{L}_u^T \mathbf{Q}_f \mathbf{L}_u)^{-1} \mathbf{L}_u^T \mathbf{Q}_f \mathbf{L}_w \mathbf{w}_p$$

3.4 Controller Performance Assessment Based on LQG Benchmark

3.4.1 Calculation of LQG Benchmark Variances

According to Equation 3.1,

$$\mathbf{Y}_f = \mathbf{\Gamma}_N \mathbf{X}_f + \mathbf{H}_N \mathbf{U}_f + \mathbf{H}_N^s \mathbf{E}_f$$

the output sequence $\mathbf{y}_f$, the optimal output sequence $\hat{\mathbf{y}}_f$ can be expressed as:

$$\mathbf{y}_f = \mathbf{\Gamma}_N \mathbf{x}_f + \mathbf{H}_N \mathbf{u}_f + \mathbf{H}_N^s \mathbf{e}_f \tag{3.5}$$
$$\hat{\mathbf{y}}_f = \mathbf{\Gamma}_N \mathbf{x}_f + \mathbf{H}_N \mathbf{u}_f \tag{3.6}$$

where $\mathbf{x}_f$ is the first column of $\mathbf{X}_f$, $\mathbf{x}_f = [x_N] \in \Re^{h \times 1}$, and $\mathbf{y}_f$ / $\mathbf{e}_f$ is the first column of $\mathbf{Y}_f / \mathbf{E}_f$; $\mathbf{y}_f = \begin{bmatrix} y_N \\ y_{N+1} \\ \dots \\ y_{2N-1} \end{bmatrix}$ and $\mathbf{e}_f = \begin{bmatrix} e_N \\ e_{N+1} \\ \dots \\ e_{2N-1} \end{bmatrix}$.

Substituting $\hat{\mathbf{y}}_f$ of Equation 3.6 into the optimal control criterion:

$$J_{LQG} = \hat{\mathbf{y}}_f^T \mathbf{Q}_f \hat{\mathbf{y}}_f + \mathbf{u}_f^T \mathbf{R}_f \mathbf{u}_f$$

we have:

$$J_{LQG} = (\mathbf{\Gamma}_N \mathbf{x}_f + \mathbf{H}_N \mathbf{u}_f)^T \mathbf{Q}_f (\mathbf{\Gamma}_N \mathbf{x}_f + \mathbf{H}_N \mathbf{u}_f) + \mathbf{u}_f^T \mathbf{R}_f \mathbf{u}_f$$

The minimum of J_{LQG} can be achieved by setting the derivative of J_{LQG} with respect to the input sequence $\mathbf{u}_f$ to zero, and $\mathbf{u}_f$ is then obtained:

$$\mathbf{u}_f = \underbrace{-(\mathbf{R}_f + \mathbf{H}_N^T \mathbf{Q}_f \mathbf{H}_N)^{-1} \mathbf{H}_N^T \mathbf{Q}_f \mathbf{\Gamma}_N}_{Z} \mathbf{x}_f = \mathbf{Z}\mathbf{x}_f \tag{3.7}$$

Replacing $\mathbf{u}_f$ in Equation 3.7 with Equation 3.6, we have:

$$\hat{\mathbf{y}}_f = \mathbf{\Gamma}_N \mathbf{x}_f + \mathbf{H}_N \mathbf{u}_f = (\mathbf{\Gamma}_N + \mathbf{H}_N \mathbf{Z})\mathbf{x}_f$$

The optimal control criterion of LQG can be computed as:

$$\begin{aligned} J_{LQG} &= \lim_{N\to\infty} E\{\hat{\mathbf{y}}_f{}^T \mathbf{Q}_f \hat{\mathbf{y}}_f + \mathbf{u}_f{}^T \mathbf{R}_f \mathbf{u}_f\} \\ &= \lim_{N\to\infty} E\{\mathbf{x}_f{}^T (\mathbf{\Gamma}_N + \mathbf{H}_N \mathbf{Z})^T \mathbf{Q}_f (\mathbf{\Gamma}_N + \mathbf{H}_N \mathbf{Z})\mathbf{x}_f + \mathbf{x}_f{}^T \mathbf{Z}^T \mathbf{R}_f \mathbf{Z} \mathbf{x}_f\} \\ &= \lim_{N\to\infty} E\{\mathbf{x}_f{}^T \underbrace{(\mathbf{\Gamma}_N + \mathbf{H}_N \mathbf{Z})^T \mathbf{Q}_f (\mathbf{\Gamma}_N + \mathbf{H}_N \mathbf{Z})}_{\mathbf{F}_y}\mathbf{x}_f + \mathbf{x}_f{}^T \underbrace{\mathbf{Z}^T \mathbf{R}_f \mathbf{Z}}_{\mathbf{F}_u} \mathbf{x}_f\} \\ &= \lim_{N\to\infty} \{E[\mathbf{x}_f{}^T (\mathbf{F}_y + \mathbf{F}_u)\mathbf{x}_f]\} \end{aligned} \tag{3.8}$$

According to Equations 3.7, Equations 3.5 can be re-written as:

$$\begin{aligned} \mathbf{y}_f &= \mathbf{\Gamma}_N \mathbf{x}_f + \mathbf{H}_N \mathbf{u}_f + \mathbf{H}_N^s \mathbf{e}_f = \underbrace{(\mathbf{\Gamma}_N + \mathbf{H}_N \mathbf{Z})}_{\mathrm{T}} \mathbf{x}_f + \mathbf{H}_N^s \mathbf{e}_f \\ &= \mathbf{T}\mathbf{x}_f + \mathbf{H}_N^s \mathbf{e}_f \end{aligned} \tag{3.9}$$

Back to the state-space expression, Equation 3.9 is re-written as:

$$\begin{aligned} x_{N+1} &= \tilde{\mathbf{A}} x_N + \tilde{\mathbf{K}} e_N \\ y_N &= \tilde{\mathbf{C}} x_N + e_N \end{aligned} \tag{3.10}$$

where $\tilde{\mathbf{A}}, \tilde{\mathbf{C}}$ and $\tilde{\mathbf{K}}$ are proper matrices;

$$y_N = \tilde{C}x_N + e_N$$
$$y_{N+1} = \tilde{C}x_{N+1} + e_{N+1} = \tilde{C}\tilde{A}x_N + \tilde{C}\tilde{K}e_N + e_{N+1}$$
....
$$y_{2N-1} = \tilde{C}x_{2N-1} + e_{2N-1} = \tilde{C}\tilde{A}^{N-1}x_N + \tilde{C}\tilde{A}^{N-2}\mathbf{K}e_N + \ldots + \tilde{C}\tilde{K}e_{2N-2} + e_{2N-1}$$

$$\begin{bmatrix} y_N \\ y_{N+1} \\ \ldots \\ y_{2N-1} \end{bmatrix} = \begin{bmatrix} \tilde{C} \\ \tilde{C}\tilde{A} \\ \ldots \\ \tilde{C}\tilde{A}^{N-1} \end{bmatrix} x_N + \begin{bmatrix} \mathbf{I} & 0 & \ldots & 0 \\ \tilde{C}\tilde{K} & \mathbf{I} & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ \tilde{C}\tilde{A}^{N-2}\tilde{K} & \tilde{C}\tilde{A}^{N-3}\tilde{K} & \ldots & \mathbf{I} \end{bmatrix} \begin{bmatrix} e_N \\ e_{N+1} \\ \ldots \\ e_{2N-1} \end{bmatrix}$$

According to the definition of $\mathbf{y}_f$, $\mathbf{x}_f$ and $\mathbf{e}_f$, we have

$$\mathbf{y}_f = \mathbf{T}\mathbf{x}_f + \mathbf{H}_N^s\mathbf{e}_f$$

$$\mathbf{T} = \begin{bmatrix} \tilde{C} \\ \tilde{C}\tilde{A} \\ \ldots \\ \tilde{C}\tilde{A}^{N-1} \end{bmatrix} \qquad \mathbf{H}_N^s = \begin{bmatrix} \mathbf{I} & 0 & \ldots & 0 \\ \tilde{C}\tilde{K} & \mathbf{I} & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ \tilde{C}\tilde{A}^{N-2}\tilde{K} & \tilde{C}\tilde{A}^{N-3}\tilde{K} & \ldots & \mathbf{I} \end{bmatrix}$$

If the designed LQG controller makes the system stable, we have the steady state covariance, $\tilde{\mathbf{P}}$:

$$\tilde{\mathbf{P}} = \lim_{N\to\infty} E\{\mathbf{x}_N{}^T\mathbf{x}_N\} \tag{3.11}$$

$\tilde{\mathbf{P}}$ is the solution of the following discrete-time Lyapunov equation.

$$\tilde{\mathbf{P}} = \tilde{\mathbf{A}}\tilde{\mathbf{P}}\tilde{\mathbf{A}}^T + \tilde{\mathbf{K}}\tilde{\mathbf{W}}\tilde{\mathbf{K}}^T \tag{3.12}$$

where $\tilde{\mathbf{W}} = E[e_N e_N^T]$, $\tilde{\mathbf{W}}$ is the measurement noise density.

Therefore, the optimal LQG cost function, J_{LQG} can be calculated as:

$$\begin{aligned} J_{LQG} &= trace(\mathbf{F}_y\tilde{\mathbf{P}}) + trace(\mathbf{F}_u\tilde{\mathbf{P}}) \\ &= J_{LQG\_y} + J_{LQG\_u} \end{aligned} \tag{3.13}$$

3.4.2 Performance Measure

A simple performance index measurement can be obtained by taking a comparison between the LQG benchmark and the actual achieved objective function.

$$\eta = \frac{J_{LQG}}{J_{act}} \tag{3.14}$$

The LQG controller is an unconstrained controller and provides the best achievable nominal performance amongst the class of all stabilizing controllers. The achieved function cost by actual controller should be greater than the LQG's function cost and the index is therefore $0 \le \eta \le 1$. The index has a value of 1 at the perfect performance and 0 at the worst.

We can also compare the actual input and output variances with their counterparts to get complete information about which part contributes the most to a degraded performance.

$$\begin{cases} \eta_y = \dfrac{J_{LQG\_y}}{J_{act\_y}} \\ \eta_u = \dfrac{J_{LQG\_u}}{J_{act\_u}} \end{cases}$$

J_{act}, J_{act_y} and J_{act_u} are defined by

$$\begin{aligned} J_{act} &= \frac{1}{K}\{\sum_{j=1}^{K}[\tilde{y}_{t+j} - \tilde{r}_{t+j}]^2\} + \frac{1}{K}\{\sum_{j=1}^{K}[\tilde{u}_{t+j+1} - \tilde{u}_{t+j}]^2\} \\ &= J_{act\_y} + J_{act\_u} \end{aligned}$$

where $\tilde{y}_t$, $\tilde{r}_t$ and $\tilde{u}_t$ are output, setpoint and input of the actual process , and K is the sample data length respectively .

3.4.3 Calculation of LQG Benchmark via Classical Approach vs. Subspace Approach

The classical way of using the LQG as benchmark to assess the controller performance consists of the following steps:

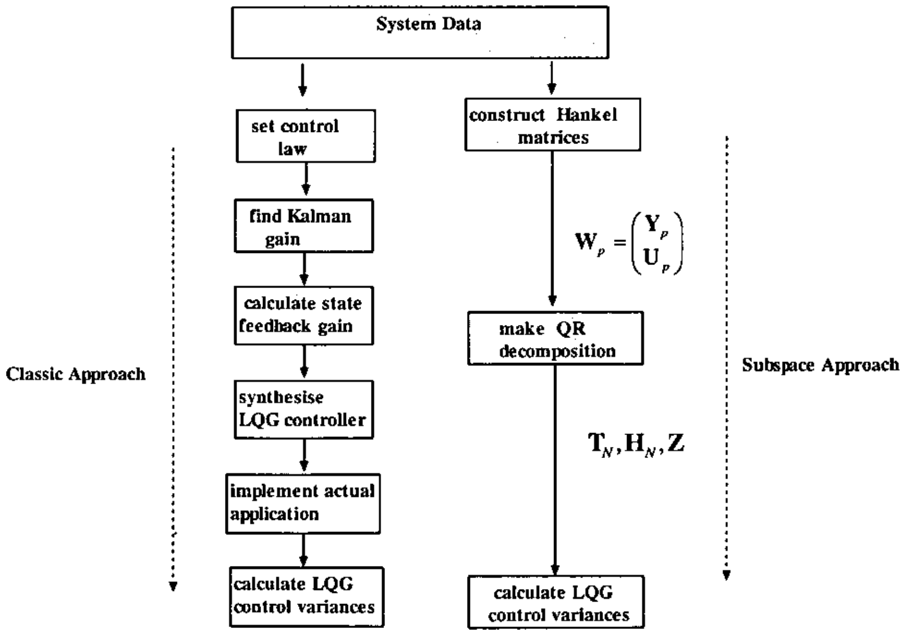

System Data
set control law
find Kalman gain
calculate state feedback gain
synthesise LQG controller
implement actual application
calculate LQG control variances
construct Hankel matrices
$W_p = \begin{pmatrix} Y_p \\ U_p \end{pmatrix}$
make QR decomposition
T_N, H_N, Z
calculate LQG control variances
Classic Approach
Subspace Approach

Figure 3.2. Classic approach vs. subspace approach

(1) System identification: Given measurements of input u_k and the outputs y_k of the unknown system, the system model is identified;

(2) Finding a Kalman gain $\mathbf{K}_f$ such that the estimate of x_k is optimal. The solution to this problem is through a steady state Kalman filter gain equation;

(3) Calculating the state feedback gain $\mathbf{K}_c$, then a feedback compensator is finally synthesised as a series connection of the Kalman filter and the optimal state-feedback controller;

(4) Implementing the designed LQG control into actual applications;

(5) Collecting process variables of the LQG control and calculating LQG control variances.

The first three steps are to design a LQG control via traditional state space model; details see (Naeem et al., 2003).

The subspace based LQG benchmark approach simplifies the assessment procedure of the classical approach and consists of the following steps:

(1) Collecting the process outputs and control signals from the discrete–time system and construct the data block Hankel matrices $\mathbf{Y}_f, \mathbf{U}_f$ and $\mathbf{W}_p = \begin{pmatrix} \mathbf{Y}_p \\ \mathbf{U}_p \end{pmatrix}$.

(2) Making the following QR-decomposition:

$$\begin{pmatrix} \mathbf{W}_p \\ \mathbf{U}_f \\ \mathbf{Y}_f \end{pmatrix} = \begin{pmatrix} \mathbf{R}_{11} & 0 & 0 \\ \mathbf{R}_{21} & \mathbf{R}_{22} & 0 \\ \mathbf{R}_{31} & \mathbf{R}_{32} & \mathbf{R}_{33} \end{pmatrix} \begin{pmatrix} \mathbf{Q}_1^T \\ \mathbf{Q}_2^T \\ \mathbf{Q}_3^T \end{pmatrix}$$

setting

$$\mathbf{L} = \begin{pmatrix} \mathbf{R}_{31} & \mathbf{R}_{32} \end{pmatrix} \begin{pmatrix} \mathbf{R}_{11} & 0 \\ \mathbf{R}_{21} & \mathbf{R}_{22} \end{pmatrix}^{\dagger}$$

and deriving the LQG controller parameters:

$$\mathbf{L}_w = \mathbf{L}(:,1:N(m+l))$$
$$\mathbf{L}_u = \mathbf{L}(:,1:N(m+l)+1:end)$$
$$\mathbf{L}_w = (\mathbf{U}_1 \quad \mathbf{U}_2) \begin{pmatrix} \mathbf{S}_1 & 0 \\ 0 & \mathbf{S}_2 \end{pmatrix} \begin{pmatrix} \mathbf{V}_1^T \\ \mathbf{V}_2^T \end{pmatrix}$$

then calculating matrices $\mathbf{\Gamma}_N, \mathbf{H}_N,\ \mathbf{Z}$:

$$\mathbf{Z} = -(\mathbf{R}_f + \mathbf{H}_N^T \mathbf{Q}_f \mathbf{H}_N)^{-1} \mathbf{H}_N^T \mathbf{Q}_f \mathbf{\Gamma}_N$$

$$\mathbf{\Gamma}_N = \mathbf{U}_1 \mathbf{S}_1^{1/2}$$

$$\mathbf{H}_N = \mathbf{L}_u$$

(3) Calculating LQG benchmark variance J_{LQG_y} and J_{LQG_u} via Lyapunov equation.

Observing the comparison shown in Figure 3.2, we can make some important remarks:

1. Comparing with the classic method, the subspace method to calculate the LQG benchmark variance is much shortened and simple; the only requirement is system inputs/outputs; system model parameters are never explicitly identified.
2. The classical LQG framework only calculates the first control input while the subspace approach computes the control input over the whole forward horizon. Although only the first input is implemented, the knowledge of the inputs over this horizon is useful for system analysis (Favoreel et al., 1998b).
3. The LQG problem can be solved by the unconstrained model predictive control (MPC), which controls the future behavior of a plant through the use of an explicit process model. At each control interval the MPC algorithm computes an open-loop sequence of manipulated variable adjustments in order to optimize the future plant behavior. The subspace method also can be implemented to design the MPC control.

Table 3.1. Controller tuning parameters

Controller	prediction horizon	control horizon	Relative weight
MPC#1	3	3	0.5
MPC#2	5	5	0.5
MPC#3	15	15	1

3.5 Simulation

To illustrate the proposed measure, we consider a linear discrete-time system described by:

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}u_k + \mathbf{K}e_k$$
$$y_k = \mathbf{C}x_k + \mathbf{D}u_k + e_k$$
$$\mathbf{A} = \begin{bmatrix} -0.313 & 56.7 & 0 \\ -0.0139 & -0.426 & 0 \\ 0 & 56.7 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0.232 \\ 0.0203 \\ 0 \end{bmatrix}$$
$$\mathbf{C}^T = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \text{and } \mathbf{D} = 0$$

and e_k is a Gaussian zero mean white noise sequence with $E[e_k e_k^T] = 0.35$.

Three Model Predictive Controllers (MPC) with different tuning parameters are implemented on this process. The tuning parameters are shown in Table 3.1.

The process governed by the three MPCs respectively is shown in Figures 3.3, which indicates that the performance gradually improves when the prediction and control horizon increase.

Table 3.2. Performance index

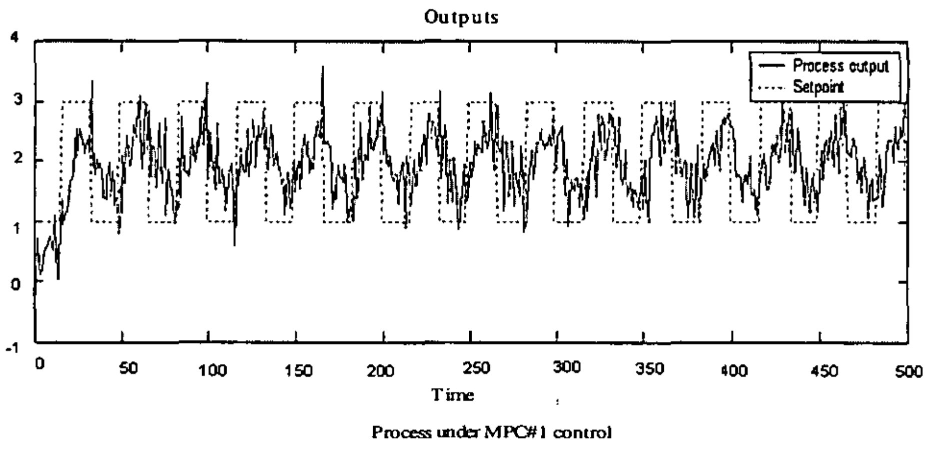
Controller	$\eta = \frac{J_{LQG}}{J_{act}}$	$\eta_u = \frac{J_{LQG\_u}}{J_{act\_u}}$	$\eta_y = \frac{J_{LQG\_y}}{J_{act\_y}}$
MPC#1	0.790	0.778	0.793
MPC#2	0.811	0.792	0.82
MPC#3	0.842	0.808	0.856

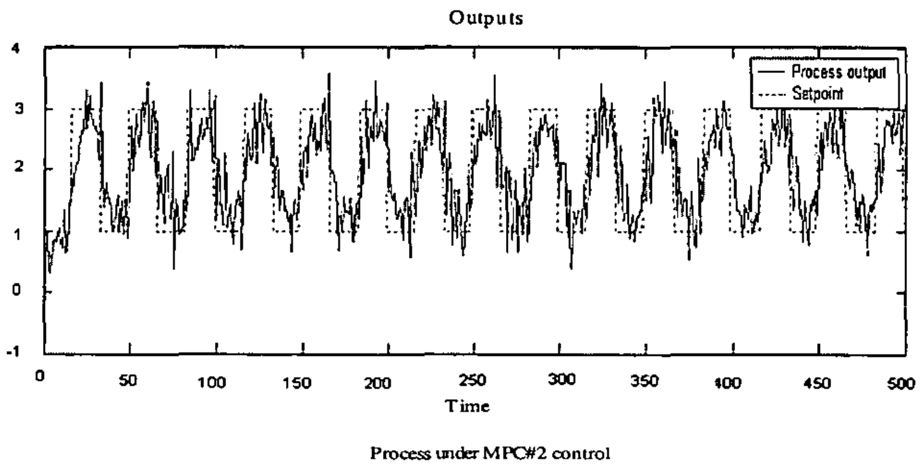
Collecting the system inputs and outputs to construct the data block Hankel matrices $\mathbf{Y}_f$, $\mathbf{U}_f$ and $\mathbf{W}_p = \begin{pmatrix} \mathbf{Y}_p \\ \mathbf{U}_p \end{pmatrix}$ and then making QR-decomposition, matrices $\mathbf{\Gamma}_N$, $\mathbf{H}_N$ and $\mathbf{Z}$ can be identified following the steps in Section 3.4.3. Taking the future horizon in the LQG criterion N =10, the closed loop of the process under the LQG control is calculated as follows:

$$x_{k+1} = \tilde{\mathbf{A}}x_k + \tilde{\mathbf{K}}e_k$$
$$y_k = \tilde{\mathbf{C}}x_k + e_k$$

$$\tilde{\mathbf{A}} = \begin{bmatrix} -0.2261 & 58.4843 & 0 \\ -0.0063 & -0.2705 & 0 \\ 0 & 56.7000 & 0 \end{bmatrix}$$

$$\tilde{\mathbf{C}}^T = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \tilde{\mathbf{K}} = \begin{bmatrix} 2.9781 \\ -0.0011 \\ 1 \end{bmatrix}$$





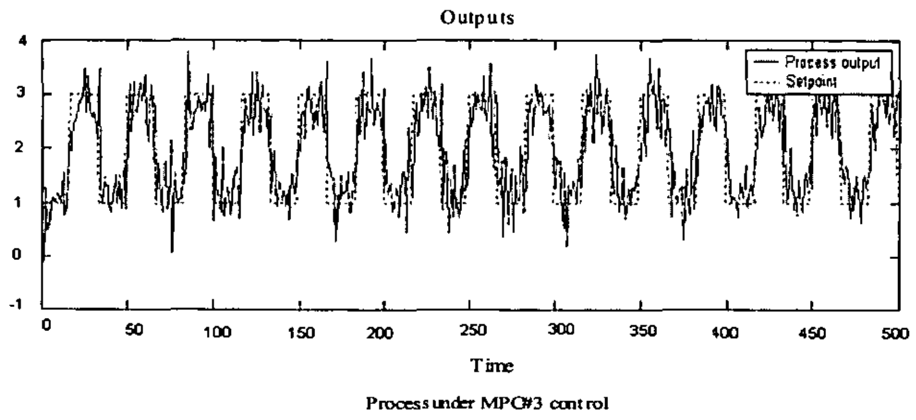


Figure 3.3. Process under MPC control with different parameters

Based on Equation 3.12 and Equation 3.8, we can calculate respectively $\tilde{\mathbf{P}}$, $\mathbf{F}_y$ and $\mathbf{F}_u$, and optimal LQG benchmark values. The performance of the three MPC controllers relative to the LQG control is shown in Tables 3.2.

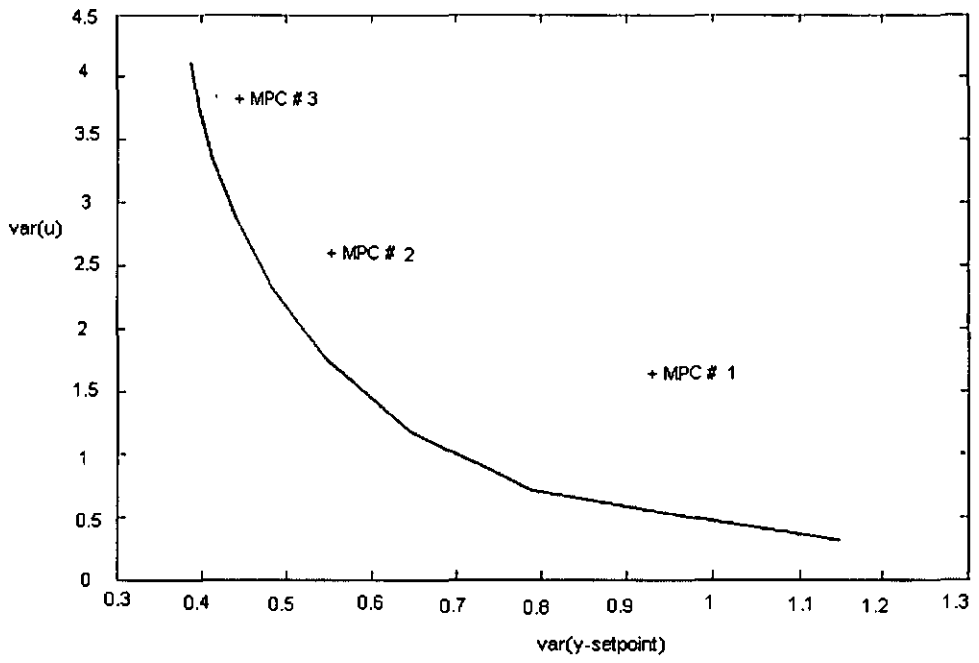


Figure 3.4. Performance assessment based on the tradeoff curve

The tradeoff curve is shown in Figure 3.4. The LQG control provides a useful lower bound on the achievable performance of a linear controller. Of the three MPC controllers, MPC#3 is the closest to the LQG curve as it is expected. MPC#2 has a better performance than MPC#1 because MPC#2 takes longer prediction and control horizons. From this curve, one can see clearly the potential for improving the performance of MPC#1 and MPC#2. This simulation also provides insight into how the various MPC controllers may be tuned to give the desired performance.

3.6 Conclusions

This chapter discusses algorithms for designing the LQG control and calculating the LQG control variances via the subspace approach to assess the control performance. The subspace approach enables a "model-free" method to design a LQG controller and estimate the LQG benchmark variances; the subspace matrices are determined by system input/output data. The only priori knowledge required is the system input and output data and the estimated Kalman filter gain. The estimation of benchmark variances is obtained directly from the solution of a Lyapunov equation and subspace matrices, which are determined by system input/output data. This study does not claim that the proposed subspace approach requires less priori knowledge than the traditional methods. The subspace method actually implicitly buries the process model and leaps the intermediate model identification. However, avoiding the use of the system model and actual implementations can bring an advantage of applicability by reducing unnecessary model identification calculation and the high accuracy of performance assessment.

Chapter 4. Controller Fault Detection

4.1 Introduction

A fault is defined as un-permitted deviation of at least one characteristic property or variable of the system from acceptable/usual/standard behaviour (Nyberg, 1999). The fault is a state that may lead to a malfunction or a failure in the system. In control systems, faults may take place in any system component (controller, sensor, actuator, plant components or any combination.) as shown in Figure 4.1.

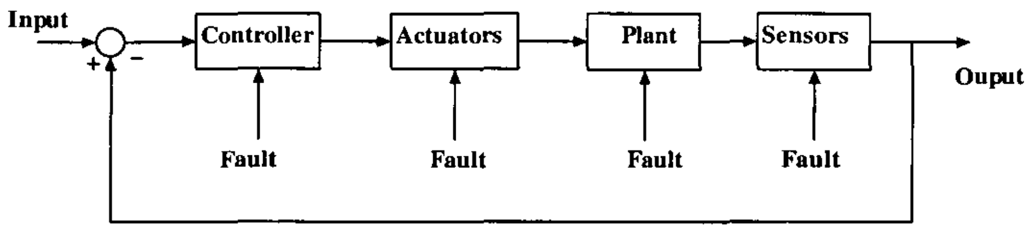


Figure 4.1. Potential faults in a control system

Controller faults are mainly caused by physical or logical factors of the controller. The physical factors refer to faults occurred in the associated hardware of the controller. For example, in a feedback control system, which is composed of a system and a feedback controller, controller faults will occur when signals are not transmitted perfectly on the route between the system and the controller, or the communication channels fail to send signals to the controller within the required time range.

The logical factors refer to internal faults in the associated control software of the controller. Controller software is becoming more and more complicated especially with advanced control strategies being widely incorporated into industrial computer based control systems to solve actual difficult multiple input, multiple output (MIMO) control problems. The steadily growing size and complexity of controller software makes it extremely difficult to exhaustively test the software to ensure that it will adequately perform its specified function. It has been shown that software complexity is a determinant factor leading to error-proneness of software (Zulkernine and Seviora, 2001). In actual applications, internal/external factors can contribute to the abrupt or gradual performance deterioration later on even if the software initially performs well.

Controller faults range from loss of partial control functions to a complete loss of control. Due to the controller's crucial role in a control system, even minor controller faults can lead to serious and expensive damage. The problems may range from those affecting the quality of final products to those involving danger to property or loss of human life.

In this chapter, an algorithm for controller fault detection is proposed. A Linear Quadratic Gaussian (LQG) controller is utilized as a benchmark or a reference model for the actual controller. Discrepancies between the benchmark and actual controller outputs are shown to be in an approximate Gaussian distribution. A baseline of the controller in the normal operation is treated as the asymptotic distribution of the discrepancies. Fault detection characterized by sudden jumps in the parameters of the discrepancy is formulated as an online change point identification problem. A detector based on a general likelihood ratio test is employed to online monitor the change point. The rest of this chapter is arranged as follows. Section 4.2 gives a brief description of the challenges of the controller fault detection. Section 4.3 presents the controller fault detection based on the LQG benchmark. Section 4.4 implements the proposed controller fault detection algorithm into the simulation applications. Section 4.5 gives the conclusions.

4.2 Challenges of Controller Fault Detection

There are two challenges for controller fault detection. One is the lack of a generally accepted description of what constitutes desired behaviors of black-box controllers. The desired behaviours if available will be adopted as the benchmark in the detection procedure to monitor the observed behaviors of the actual controller. Traditional fault detection methods are often based on an explicit black-box mathematical model (Bloch et al, 1994). The black-box model uses a standard flexible structure to approximate varieties of different systems. The fundamental principle of the model based fault detection is based on the assumption that the faults are reflected in the physical parameters such as friction, viscosity, capacitance, inductance, etc. Parameters of the model are repeatedly on-line estimated and the results are compared with the parameters of the reference model obtained initially under fault free conditions. Any substantial discrepancy indicates a change in the process and may be interpreted as a fault. However, different from plants or machines, some advanced controls, such as Model Predictive Controls (MPCs) are linear time-varying controllers, and their parameters are on-line updated in response to the controlled process dynamics to minimize a time-varying objective function (Patwardhan and Shah, 2002); therefore, it is hard to establish an accurate mathematical model as the reference model to precisely describe controller behaviors in practice.

The other is how to detect the fault as early as possible. Early warning on urgent problems can provide invaluable benefits with appropriate actions taken to avoid serious process upsets. As a result, it is important to detect the controller faults and raise alarms as early as the faults occur. The following sections will address the above two issues.

4.3 Controller Fault Detection

4.3.1 LQG Control as a Benchmark for Controller Fault Detection

Besides being used as a benchmark for control performance assessment, distinguishing features of the LQG control make it be suitable as a benchmark for controller fault detection:

1. The LQG control provides the best achievable nominal performance amongst the class of all stabilizing controllers. The LQG controller is an unconstrained controller and provides a useful lower bound on the achievable performance of a linear controller. The infinite prediction horizon of the LQG algorithm endows the algorithm with powerful stabilizing properties.
2. The LQG control can be designed via various methods. A number of modifications can be made to the LQG control methodology to tailor the LQG control design for specific requirements. Feedforward control, error feedback and integral control can be implemented within the LQG framework.
3. The steady-state LQG control problem is equivalent to an H_2 optimization control. The LQG cost function can be written in terms of the closed loop system 2-norm:

$$
\begin{aligned}
J &= \lim_{t\to\infty} E[x^T(t)\mathbf{Q}x^T(t) + u^T(t)\mathbf{R}u^T(t)] \\
&= \lim_{t\to\infty} E[\mathbf{Q}^{1/2}x^T(t) \mid \mathbf{R}^{1/2}x^T(t)]\begin{bmatrix}\mathbf{Q}^{1/2}x^T(t) \\ \mathbf{R}^{1/2}x^T(t)\end{bmatrix} \\
&=\| \mathbf{G}_{cl} \|_2^2
\end{aligned}
$$

where x is the system state, u is the control signal, $\mathbf{Q}$ and $\mathbf{R}$ are semi-positive weight matrices , and $\mathbf{G}_{cl}$ is an appropriate matrix.

4. The poles of the closed loop system under the LQG control are shown to be the union of the poles of the Linear Gaussian Regulator (LQR) and the Kalman filter (Burl, 1999), which implies that the closed loop system is stable since both LQR and Kalman filter are stable.

However, the LQG control is mainly implemented in the theoretical analysis and limitedly used in actual applications (Ougrinovski and Petersen, 2003). The LQG control suffers from a major disadvantage that the LQG control does not take into account the issue of robustness. The LQG control design methodology heavily relies on the use of an accurate linear dynamical model for the process being controlled. In practice, although it is possible to obtain process models either from principles or from experimental measurements, these models will always be subject to errors. Thus, the control system needs to be designed to be robust against these modelling errors. Since the issue of robustness is usually critical in any control system design, the absence of any systematic way of dealing with robustness has severely limited the usefulness of the LQG control method.

4.3.2 Change Detection Algorithm

The problem of change detection is understood as the detection of abrupt changes in some characteristic properties of a system. Generally, such characteristics are not observed directly (e.g., through corresponding sensors) and can only be inferred indirectly from the measurements available. Change detection is a common but difficult signal processing task. Typical applications are vibration monitoring or music transcription (Basseville and Nikiforov, 1999). Strong theoretical results hold in the case of known statistical models of the data (Laurent and Doncarli, 1998), however, some real-life systems are difficult to model accurately and model-based statistical techniques are useless in such cases. One needs then to resort to nonparametric data analysis techniques.

The likelihood-ratio test is a statistical test relying on a test statistic computed by taking the ratio of the maximum value of the likelihood function to the maximum with that constraint relaxed. The likelihood ratio test statistic is obtained by replacing the unknown parameters under each hypothesis with their maximum-likelihood estimators. Taking a simple likelihood ratio test, a general likelihood ratio test (GLR) is designed and employed to the controller fault detection. The GLR test is to detect the abrupt non-stationary changes in a signal, where independent and identically distributed samples are drawn from two known probability density functions (PDF).

Consider a sequence of independent random variables y_k with a probability density $p_\theta(y)$ depending on only one scalar parameter, θ. Before the unknown change time t_0, the parameter θ is equal to θ_0 and after the change it is equal to $\theta_1 \neq \theta_0$, where θ_1 is the up-to-date scalar parameter. Samples with fixed size N are taken, and at the end of each sample a decision rule is computed to test between the two following hypotheses about the parameter θ:

$$\begin{aligned} &H_0 : \theta = \theta_0, \text{ where } 0 \le t \le t_0 \\ &H_1 : \theta = \theta_1, \text{ where } t > t_0 \end{aligned}$$

The log-likelihood of H_1 relative to H_0 based on this single observation is

$$s(y) = \ln \frac{p_{\theta_1}(y)}{p_{\theta_0}(y)} \tag{4.1}$$

A decision rule is given by

$$\begin{aligned} &\text{If } s < h;\ H_0 \text{ is chosen} \\ &\text{If } s \ge h;\ H_1 \text{ is chosen} \end{aligned}$$

where h is a threshold.

As long as the decision is taken in favor of H_0, sampling and test are continued. An alarm is given immediately after the sample of observation for which the decision is taken in favor of H_1.

4.3.3 Applying GLR Test to Controller Fault Detection

Differences of control signals between the LQG control and the actual controller are used to detect the controller failure. Theoretically, if the actual controller is in a healthy work state and tuned well, the difference between outputs of the actual controller and the LQG control will be stable when both controllers control the same process (the outputs of both controllers will be similar when the actual controller is also designed to minimize the same or similar performance index with the LQG controller). Any fault occurring in the system will make the differences or residuals vary significantly. Consider the observed signals $r(t)$:

$$\begin{aligned} r(t) &= u_{act}(t) - u_{LQG}(t) + r_0(t) \\ &= r_1(t) + r_0(t) \end{aligned} \tag{4.2}$$

$r_0(t)$ is the measurement noise following a Gaussian distribution with zero mean, $r_1(t)$ is the difference between the control signals of the LQG controller, u_{LQG}, and the actual controller u_{act}.

The Gaussian distribution has a very important property that the distribution of a sum of a large number of independent variables is approximately normal as the number of observations becomes large (Gaussian distribution, 2000). Therefore, for a stable control process, the distribution of independent random residual $r_1(t)$ in Equation 4.2 takes on a Gaussian distribution if an enough large number of residuals are sampled; $r(t)$ in Equation 4.2 can be assumed to approximately follow the Gaussian distribution with the mean value μ and variance σ.

$$p(r(t)) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{-(r(t)-\mu)^2}{2\sigma^2}\right) \tag{4.3}$$

where $p(.)$ denotes the density function of the signals $r(t)$.

Because of the random nature of the observed signal $r(t)$, the characteristics of observed signal $r(t)$ are represented by both mean value μ and variance σ. Therefore, the abrupt change point of $r(t)$ can be identified based on the parameters μ or σ. Two decision rules are given in the following based on μ and σ respectively.

4.3.3.1 Decision Rule Based on Mean μ

Suppose a fault occurs at the instant t_0, before the instant t_0, $r(k)$ follows a Gaussian distribution with mean μ_0 and variance σ_0. After the instant t_0, $r(k)$ shifts to a new distribution with a different mean denoted by μ_1 and the variance σ_0 keeps unchanged. Therefore, the early fault detection can be transferred into the identification of the change point in the mean value μ (from μ_0 to μ_1). Based on the GLR test, a detector can be designed as follows. Two hypotheses about the parameter μ are:

$$\begin{cases} H_0 : \mu = \mu_0 & \text{; normal work state} \\ H_1 : \mu = \mu_1 & \text{; fault work state} \end{cases}$$

The mean value μ is calculated within a moving window.

$$\mu = \frac{\sum_{i=1}^{k} r(i)}{k} \tag{4.4}$$

The likelihood ratio at the instant k, s_k can take a simple form given by (Basseville and Nikiforov, 1998):

$$s_k = \frac{\mu_1 - \mu_0}{\sigma_0^{\ 2}}\left(r(k) - \frac{\mu_0 + \mu_1}{2}\right) \tag{4.5}$$

where s_k is the likelihood ratio.

The decision rule of identifying a fault by using a pre-defined threshold λ_1 is as follows:

$s_k > \lambda_1 : H_1$ is assumed i.e. a fault is detected.

$s_k \le \lambda_1 : H_0$ is assumed i.e. no fault is detected.

4.3.3.2 Decision rule based on Variance σ

It might be a case that there is no significant change in the mean value μ , but significant change in the variance σ . After some kinds of faults occur, suppose a fault occurs at the instant t_0, before the instant t_0, $r(k)$ has a variance σ_0; after t_0, it has a new variance σ_1. Two hypotheses about the parameter σ are:

$$H_0 : \sigma = \sigma_0 \quad \text{; normal work state}$$
$$H_1 : \sigma = \sigma_1 \quad \text{; faulty work state}$$

The variance σ is calculated within a moving window

$$\sigma = \left(\frac{\sum_{i=1}^{N} (r(i) - \mu)^2}{N} \right)^{1/2}$$

The likelihood ratio at the instant k, s_k is given by (Basseville and Nikiforov, 1998):

$$s_k = \ln \frac{\sigma_0}{\sigma_1} + \left(\frac{1}{\sigma_0^2} - \frac{1}{\sigma_1^2}\right) \frac{(r(k) - \mu)^2}{2} \tag{4.6}$$

The decision rule of identifying a fault is used a pre-defined threshold λ_2:

$s_k > \lambda_2 : H_1$ is assumed i.e. a fault is detected.

$s_k \leq \lambda_2 : H_0$ is assumed i.e. no fault is detected.

4.3.4 Data Collection

The amount of time in which quality data are collected from industrial processes during the normal operation and the faulty operation is usually limited. Typically, only a small portion of the operation time exists where it can be determined with confidence that the data are not somehow corrupted and no faults occurred in the process. Also, the process supervisors do not generally allow faults to remain in the process for long periods of time for the purpose of producing data used in fault detection algorithms.

It is desirable to detect the fault as soon as possible. Given a fixed time $T = n\Delta t$, it is beneficial to sample the data as fast as possible ($\Delta t \to 0, n \to \infty$). In terms of process monitoring, however, there are three possible problems with the sampling as fast as possible. Firstly, for the amount of data produced, the computational requirements may exceed the computational power available. Secondly, the model fit may exceed the computational frequencies, where measurement noise is predominant. Finally, statistics that ignore serial correlation will generally perform more poorly for short sampling times (Chiang, Russell and Braatz, 2001).

A suitable approach of the sampling interval is to use a moving window, which collects the monitored data on-line over a period of time before using the data in the process monitoring algorithms. This moving window technique can generally reduce normal process variability and hence produce a more sensitive process monitoring method. The sampled data collected during both normal and faulty operations are stored in historical databases, which are used to fit the distribution model and diagnose the fault after it occurs.

4.3.5 Controller Fault Detection Structure

The structure of controller fault detection is shown in Figure 4.2, which consists of two stages: the residual generation stage and the residual evaluation stage. In the residual generation stage, the benchmark controller, a LQG control, is designed to run in parallel to the actual controller. The LQG controller can be treated as a relative measurement of the actual controller behavior. The residuals are generated from the difference between the control signals of the actual controller and the LQG controller.

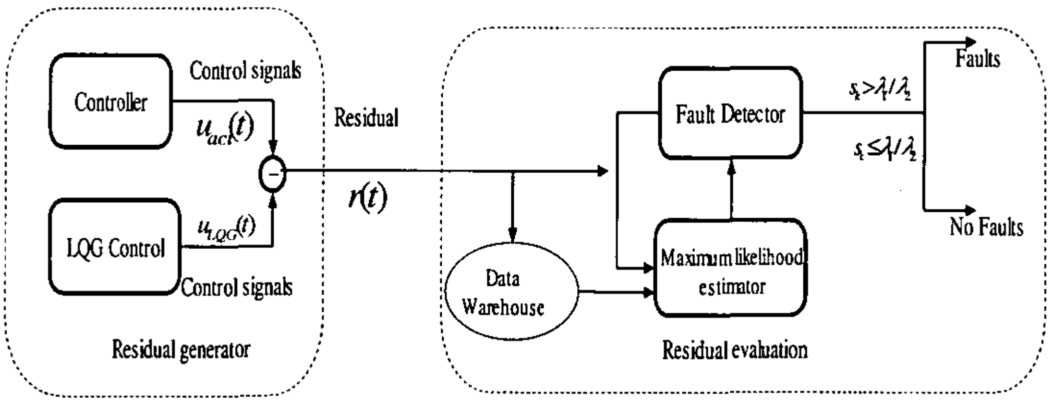


Figure 4.2. Controller fault detection structure

The data warehouse collects and stores the generated residuals on-line for future diagnosis analysis. The residual evaluation collects the residuals and generates the detection decision. It consists of three parts, a threshold, a maximum likelihood estimator and a fault detector. The maximum likelihood estimator computes the mean/ variance of the moving residuals and sends the result to the fault detector. Based on the updated mean/variance, the detector tests the observed residual data by comparing whether or not the likelihood ratio is within the predefined threshold range. Once the observed residuals are out of the range, an alarm will raise to show the monitored controller is in a faulty work state.

4.4 Case Studies

4.4.1 SISO Simulation

In order to illustrate and evaluate how the proposed fault detection approach monitors the behaviors of target controllers; a water heater process has been used as a case study. The water heater process is shown in Figure 4.3. In this case study, the water temperature in the pipe is controlled by adjusting the flow of the stream.

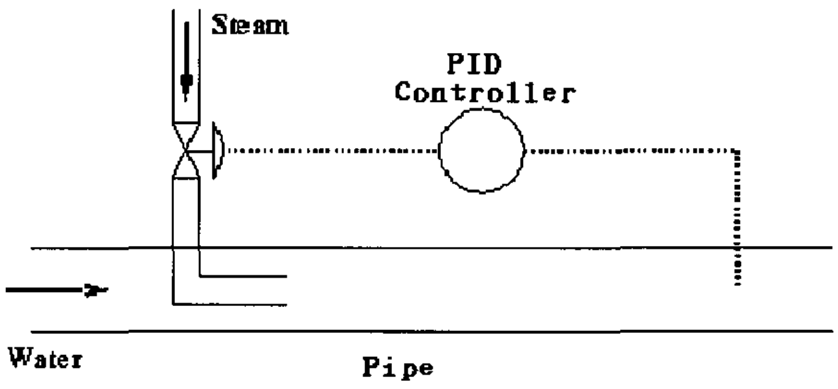


Figure 4.3. Water heater process

A PID controller is designed here to regulate the flow rate of stream to maintain a desired water temperature. The transfer function of the water heater process is $\frac{0.2965}{s+0.1802}$, where s is the Laplace transfer symbol. A sine wave noise and a white noise with zero mean are adopted in the water heater process as the process noise, $\mathbf{w}_{proc}$, and the measurement noise, $\mathbf{v}_{sensor}$, respectively. Their noise intensities are given respectively by:

$$\begin{cases} E(\mathbf{v}^T\mathbf{v}) = 0.005 \\ E(\mathbf{w}^T\mathbf{w}) = 0.998 \end{cases}$$

A LQG controller is designed for the water heater process as a benchmark controller using the cost function to specify the trade-off between regulation performance and cost of control.

$$J(u) = \int_0^{\infty}(y^2 + 1.5u^2)dt$$

The LQ-optimal gain K_f and estimator gain K_{est}, obtained via the Control Toolbox in MATLAB ,are given by:

$$\begin{cases} K_f = 0.2433 \\ K_{est} = 6.7737 \end{cases}$$

The state estimator is represented by a matrix of transfer functions as follows.

$$\begin{bmatrix} \dfrac{0.2965}{s+4.197} & \dfrac{0.5}{s+4.197} \\ \dfrac{4.017}{s+4.197} & \dfrac{6.774}{s+4.197} \end{bmatrix}$$

Connecting the estimator and the optimal state-feedback gain, the final LQG controller gain is given by: $K(s) = -\dfrac{1.647}{s+4.319}$. The controller fault detection scheme shown in Figure 4.2 is implemented for the water heater to monitor the PID controller behaviors. At the instant 500, a designed fault is introduced into the actual controller by attaching the outlet of the PID controller with a transfer function model, $\dfrac{0.1}{s+0.1}$. The fault leads to significant oscillations in the process output as shown in Figure 4.4. The outputs of the PID and LQG controller considerably deviate from each other as shown in Figure 4.5.

In the normal work state, 400 observed residuals are collected to fit the Gaussian distribution parameters and the identified mean $\mu_0$and variance σ_0 are -0.1067 and 0.0121 respectively. The fault detector uses a moving observation window to monitor the residuals. The detector detects and calculates the varied mean according to Equation 4.5. The threshold is set as 0.2. The likelihood ratio result s_k is shown in Figure 4.6. Detecting the varied value of the variance according to Equation 4.6, the likelihood ratio results s_k are shown in Figure 4.7. It is clear that as shown in Figures 4.6 and 4.7, the likelihood ratio increases significantly from the instant 500; both indicate the fault has taken place around the instant 500.

The proposed detector is robust to the impact from the failure of the other components and disturbances. In this simulation, an extra disturbance is introduced into the water heater process at the instant 500, which is a white noise with the variance 0.5. Figure 4.8 shows that the control process works under the disturbance. The detector is implemented to monitor the PID control signals; the mean of the difference of the PID and LQG controller outputs is detected according to Equation 4.5. The likelihood ratio values are shown in Figure 4.9, which indicates that the test value is still under the threshold under the impact of the disturbance.

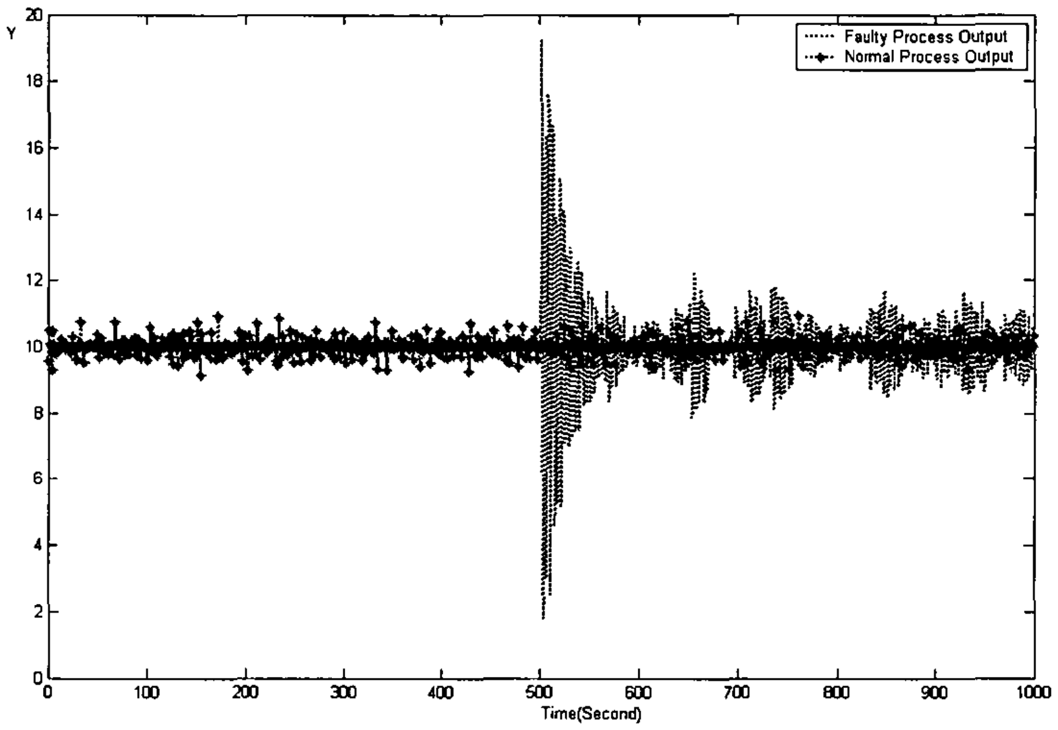


Figure 4.4. Process output

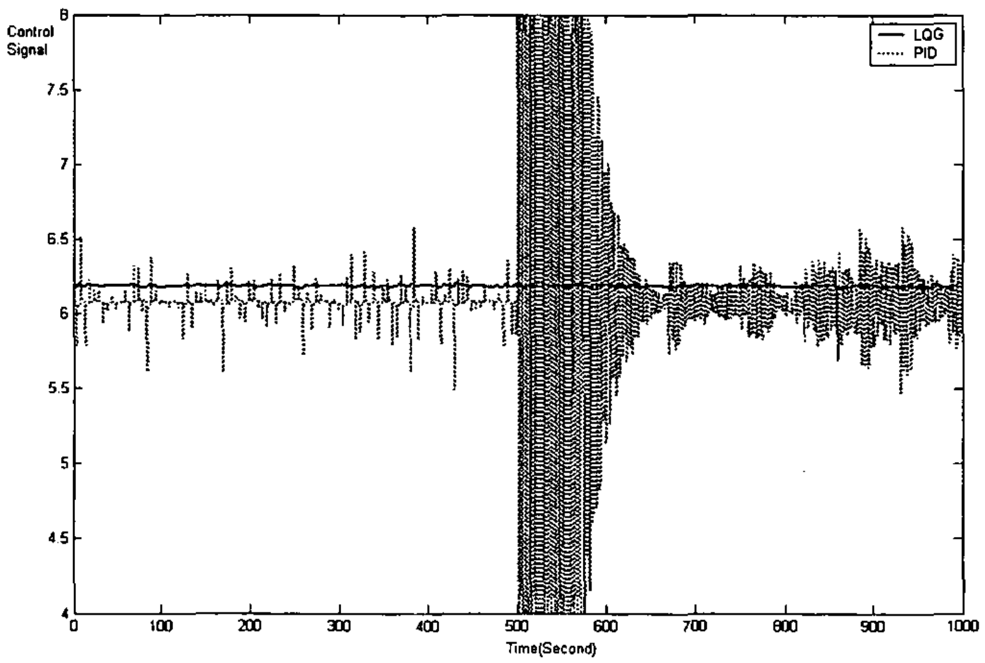


Figure 4.5. Control outputs of the PID and LQG controllers

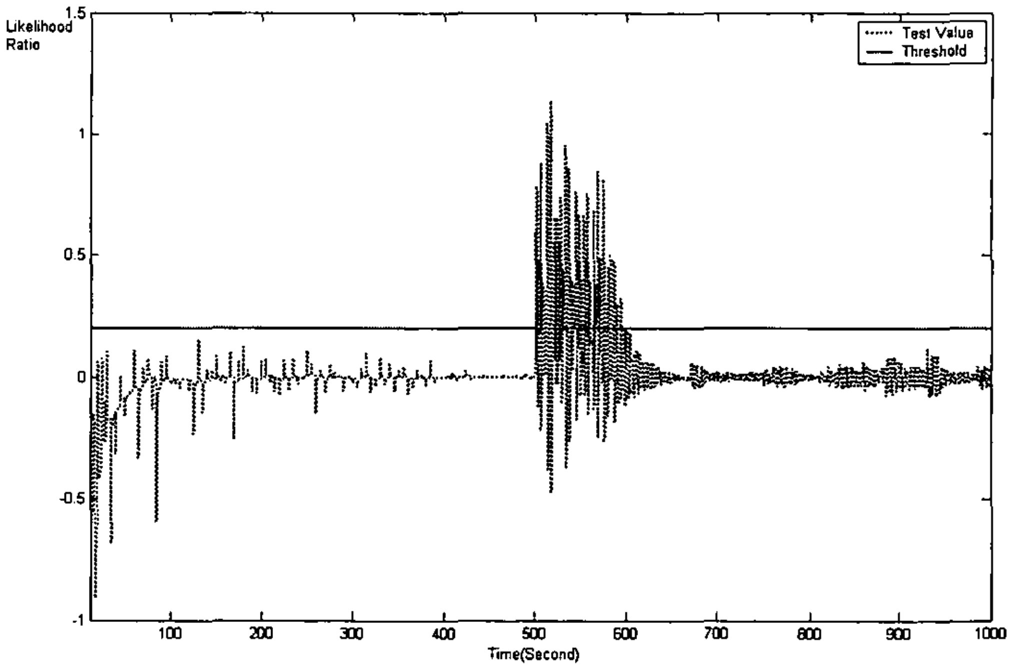


Figure 4.6. Likelihood ratio results for testing the varied mean

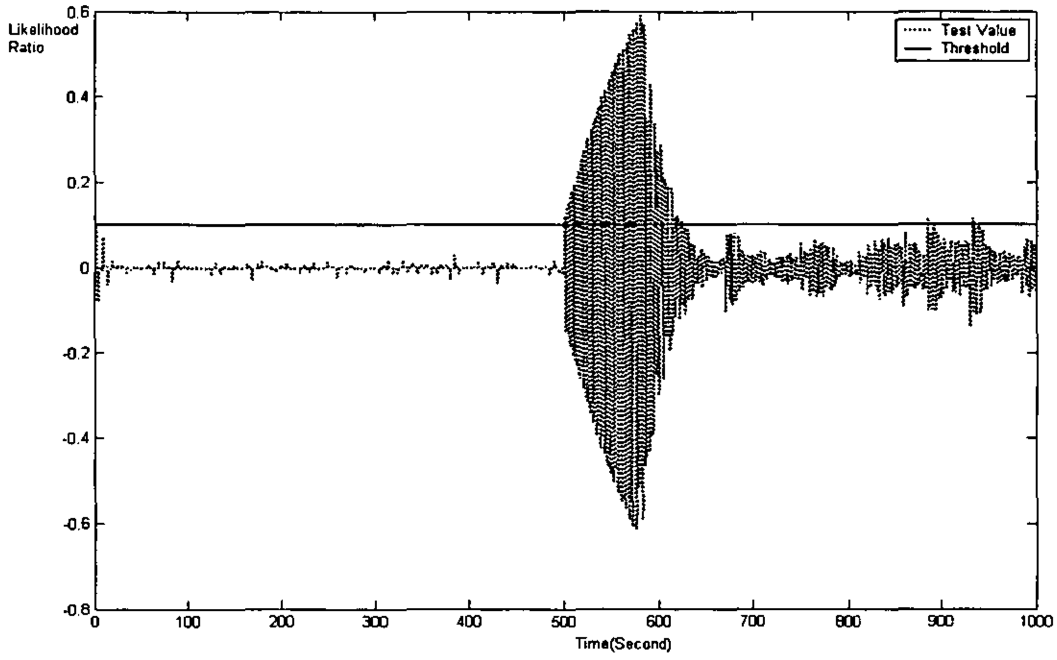


Figure 4.7. Likelihood ratio results for testing the varied variance

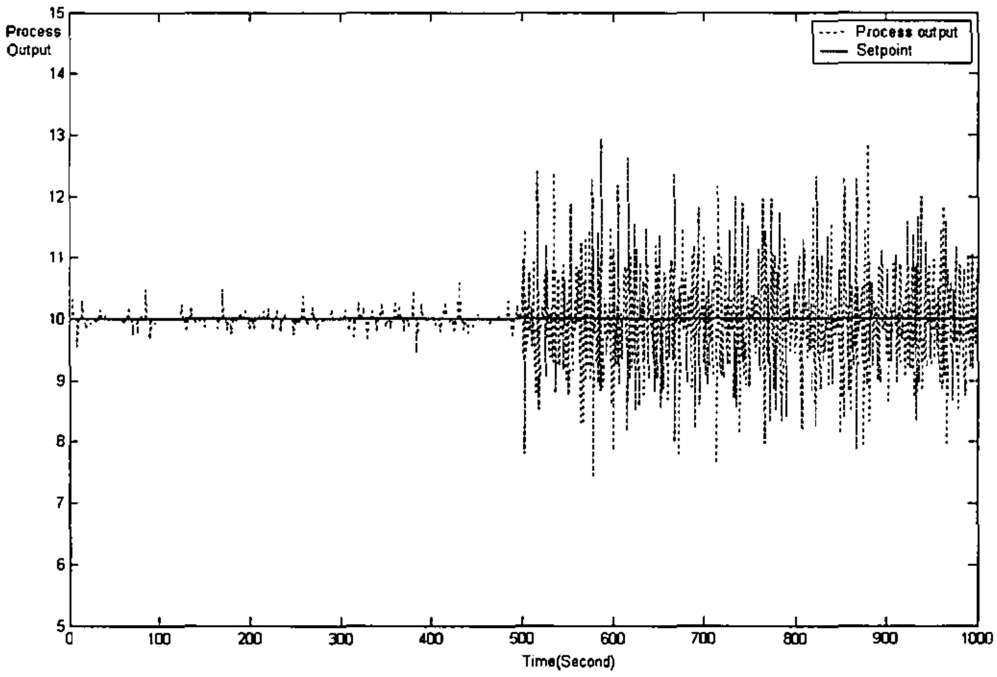


Figure 4.8. Process output under the disturbance

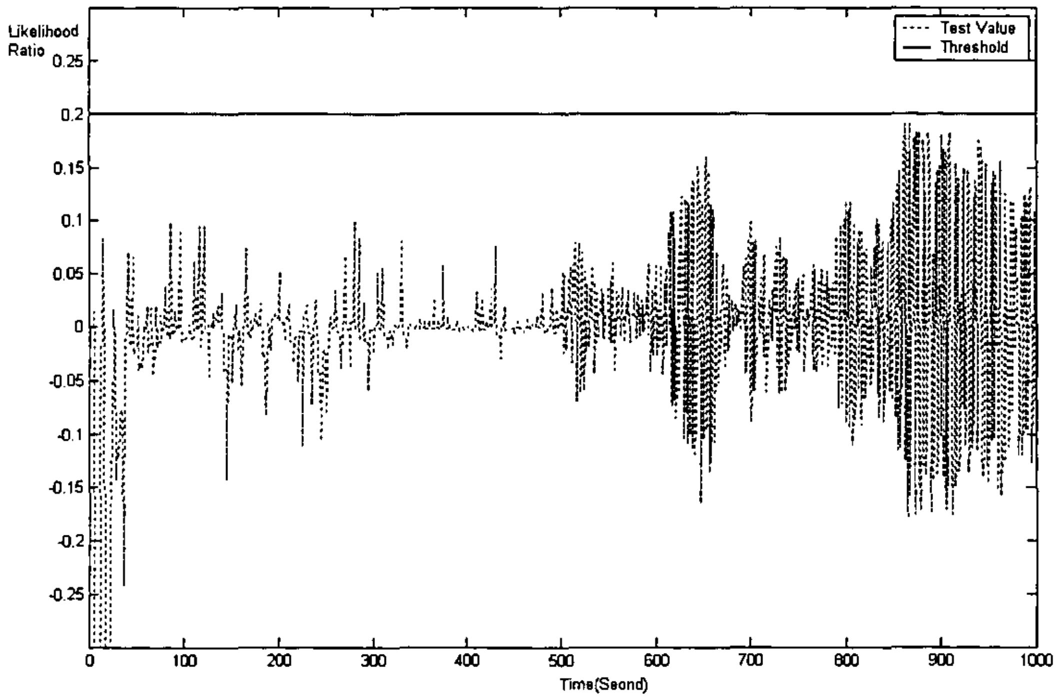


Figure 4.9. Likelihood ratio result of testing the varied mean under the disturbance impact

4.4.2 MIMO Simulation

Taking an example in (Goodwin, 2000), this linear continuous-time system described by:

$$G_p = \begin{bmatrix} \dfrac{1}{s^2+3s} & -\dfrac{1}{s+1} \\ \dfrac{0.5}{s^2+2s+3} & \dfrac{6}{s^2+5s+6} \end{bmatrix}$$

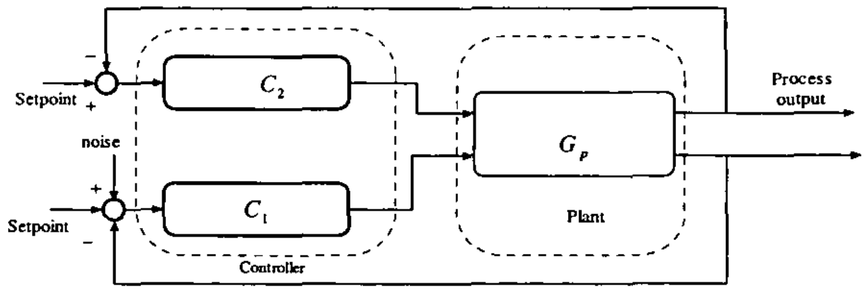


Figure 4.10. The discrete-time system controlled by decentralized controllers

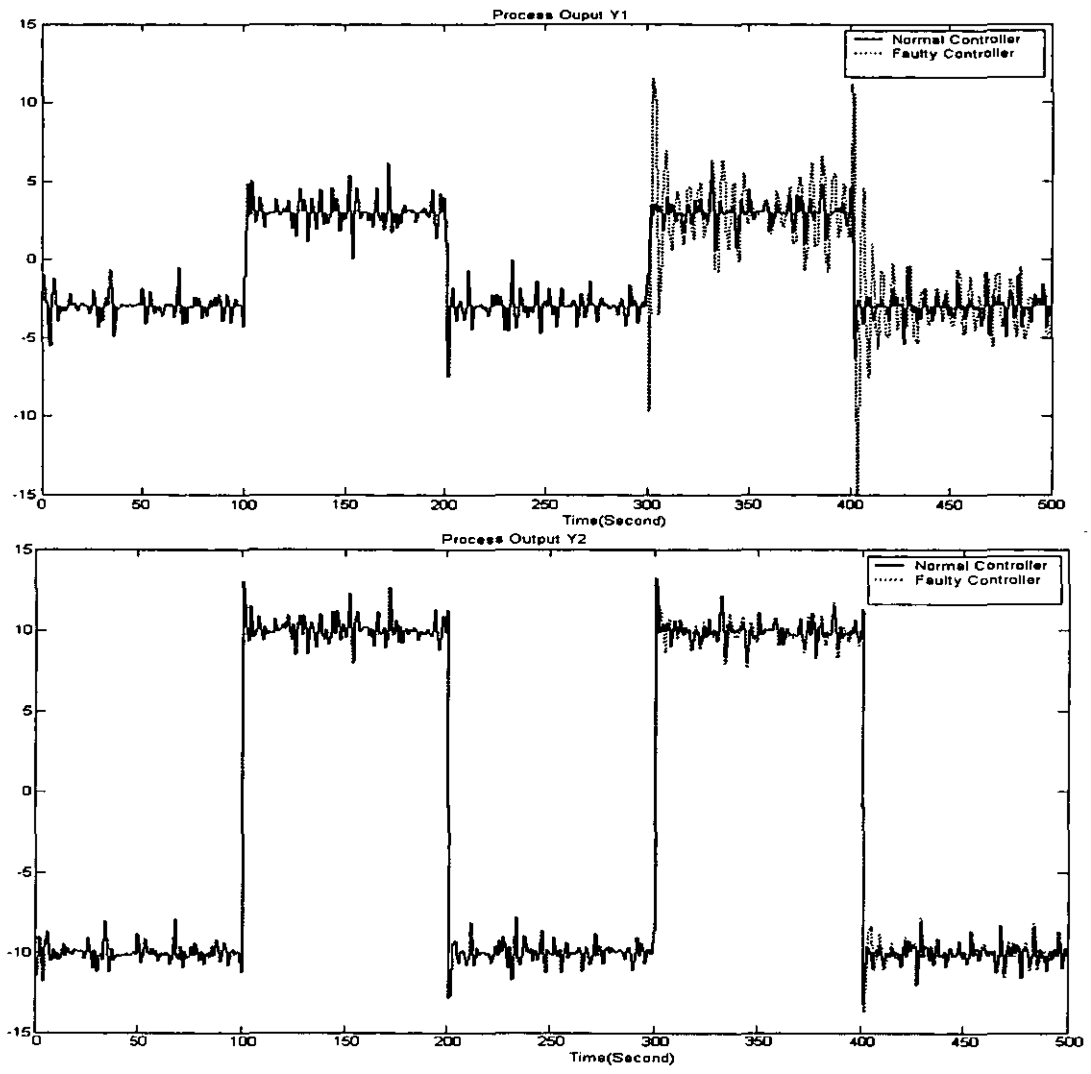


Figure 4.11. The MIMO process governed by decentralized controllers

Two decentralized controllers are designed to govern the process; the controller models are:

$$C_1 = \frac{4.5s^2 + 13.5s + 9}{s^2 + 4s}$$

$$C_2 = \frac{1.5s^2 + 7.5s + 9}{s^2 + 4s}$$

The structure of MIMO process with the decentralized controllers is shown in Figure 4.10. The disturbance in the setpoint is a white noise. The process outputs and control signals are shown in Figure 4.11. The LQG control is designed via MPC solution (Huang and Shah, 1999). The prediction horizon and control horizon of the MPC are set as 20 and the weight factor is set as 1. A fault is designed to take place at the instant 300, which changes C_1 model into $\tilde{C}_1 = \dfrac{2.5s^2 + 3.5s + 9}{s^2 + 4s}$. In this decentralized loop structure of the control process, the fault results in considerable oscillation of the output Y1 from the desired range while slightly impacts on the output Y2 as shown in Figure 4.11. The detector detects the mean of the difference between the decentralized controllers and the LQG controller according to Equation 4.5. The likelihood ratio results are shown in Figure 4.12 and the test value for control signal U1 crosses the threshold, 20, which implies the decentralized controllers are faulty.

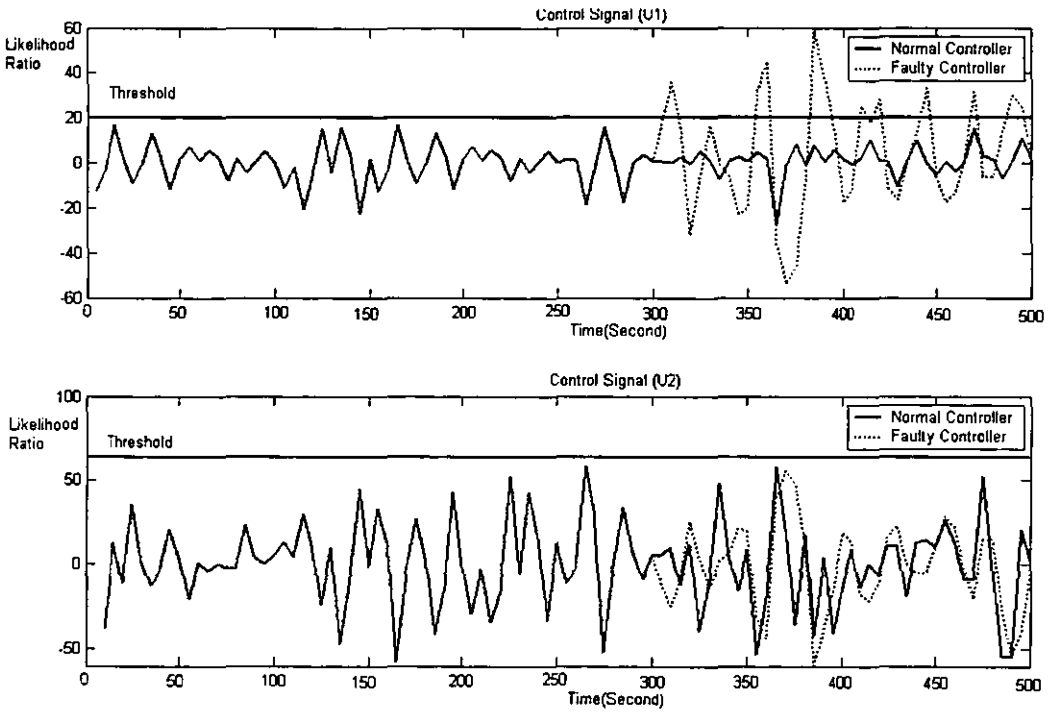


Figure 4. 12. Likelihood Ratio

4.5 Conclusions

Monitoring and maintenance of controllers is an important issue in control system management. In this chapter, we developed an online detection technique for controller monitoring. A LQG is used as a benchmark for the controller failure detection. Discrepancies between the benchmark and actual controller outputs are shown to be in an approximate Gaussian distribution. A baseline of controller in the normal operation is treated as the asymptotic distribution of the discrepancies. Fault detection characterized by sudden jumps in the parameters of the discrepancy is formulated as an online change point identification problem. A detector based on the GLR test is employed to online monitor the change point.

Chapter 5. Controller Failure Recovery

5.1 Introduction

Failure is defined as a permanent interruption of a system's ability to perform a required function under specified operating conditions. Failure may occur in any system, especially in complex systems. With the wide implementation of advanced control strategies in industries, more and more attentions have been paid to control system maintenance. Due to the controller's crucial role, any minor failure in the controller might cause serious consequences; the consequent damage ranges from that reduces the quality of final products to that involving loss of property or loss of human life. As a result, the controller failures must be detected and corrected as soon as possible, and control systems should be maintained in a good working condition.

Controller failure maintenance presents a challenging research problem. Conventional Fault Tolerant Controls (FTCs) are infeasible and alternative techniques have to be developed. In this chapter, two such methods designing a compensator to maintain the controller failure are proposed: one is the use of the fault shift-based method to design a compensator; the other is the use of the model following technique to design a compensator. The rest of this chapter is arranged as follows. Section 5.2 presents a fault shift based compensator design approach. Section 5.3 proposes a model reference based compensator design. Section 5.4 introduces a controller failure detection and maintenance scheme. Section 5.5 presents experimental results. Section 5.6 gives the conclusions.

5.2 Design of a Fault Shift-Based Compensator

5.2.1 Equivalent Fault Transfer

Consider a feedback control system shown in Figure 5.1, in which G_c represents a controller model, and G_p is a plant model. Assume that a fault occurs in the controller and the fault model can be described as G_f; set m as the identified model for the normal open loop between the controller input, $e(t)$, and the process output $y(t)$ and m_f as the identified model for the faulty open loop.

$$\begin{cases} m(s) \approx G_p(s)G_c(s) \\ m_f(s) \approx G_p(s)G_f(s)G_c(s) \end{cases} \tag{5.1}$$

$G_f(s)$ is calculated by

$$G_f(s) = \frac{m_f(s)}{m(s)} \tag{5.2}$$

Working on the transfer function of the open control loop, there is no difference whether a fault occurred in the controller or in the plant, and the model of the open loop remains $G_p(s)G_f(s)G_c(s)$. Therefore, when a controller failure happens, we can suppose the controller is still in a normal state, the fault part G_f is "shifted" to the plant and the equivalent faulty plant model is:

$$G_{new}(s) = G_p(s)G_f(s) \tag{5.3}$$

This idea also can extend to other components in the control systems; faults in these components can also be transferred to the plant with the model G_{new}.

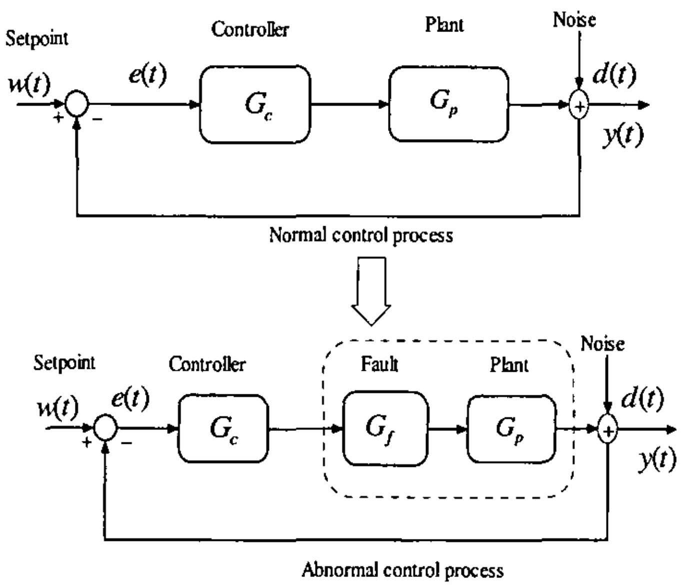


Figure 5.1. Normal and abnormal control loops

5.2.2 Compensator Design

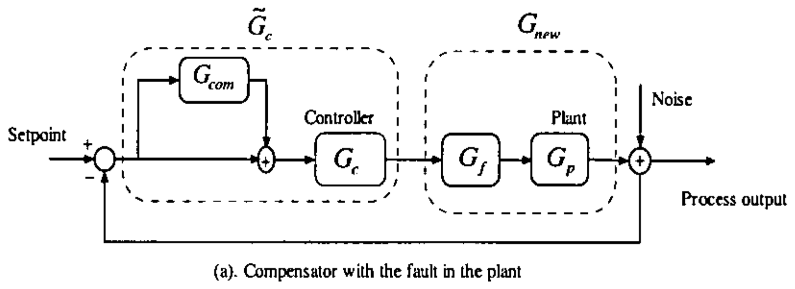
To maintain the degraded control performance in the case of the controller failure, a serial link compensator is designed here. The objective of the compensator is to work together with the faulty controller to stabilize the control process and restore the degraded performance. The design of the compensator consists of two steps:

1. Designing a new controller. Based on the post-fault process model G_{new}, we can design a new controller $\tilde{G}_c$ for the equivalent faulty plant to achieve a satisfactory control performance.
2. Calculating the compensator model. Assume the compensator G_{com} is in a serial link with the existing controller, as shown in Figure 5.2. Then, the designed compensator works with the existing faulty controller to take the place of the new controller $\tilde{G}_c$ in order to maintain the degraded performance.

The model of the compensator is given as follows:

$$\begin{cases} [1+G_{com}(s)]\times G_c(s)=\tilde{G}_c(s) \\ G_{com}(s)=\dfrac{\tilde{G}_c(s)-G_c(s)}{G_c(s)} \end{cases} \tag{5.4}$$

and the transfer function of the open loop is $G_p(s)G_f(s)G_c(s)[1+G_{com}(s)]$.



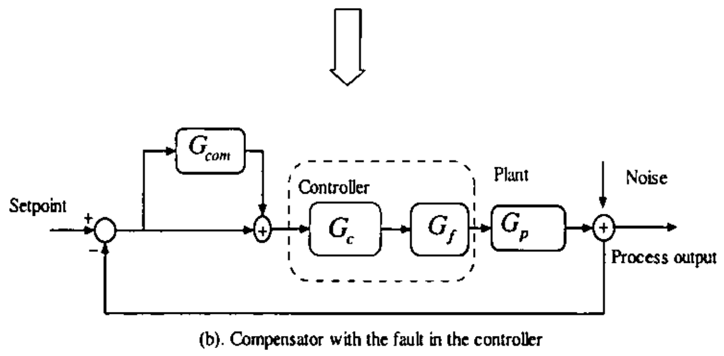


Figure 5.2. Serial link compensator

It is obvious that the original controller $G_c(s)$ and the faulty part of the controller $G_f(s)$ can be reunified as Figure 5. 2 (b). It means that the compensator designed for the equivalent faulty plant $G_{new}(s)$ in Figure 5.2 (a) can be adopted directly for the faulty controller to maintain its performance.

5.2.3 Extension to MIMO Systems

Consider a faulty control system shown in Figure 5.3 (a) and set $\mathbf{M}_f$ as the estimated model for the abnormal open loop. In the case of a controller failure, we treat the faulty controller being in the normal state, but transfer the controller faults into the plant. Suppose there is a model $\tilde{\mathbf{G}}_f$ so that:

$$\tilde{\mathbf{G}}_f\mathbf{G}_p = \mathbf{G}_p\mathbf{G}_f \tag{5.5}$$

Therefore, the faulty control loop can be reconstructed as shown in Figure 5.3 and $\mathbf{M}_f$ can be rewritten as:

$$\mathbf{M}_f \approx \tilde{\mathbf{G}}_f\mathbf{G}_p\mathbf{G}_c \tag{5.6}$$

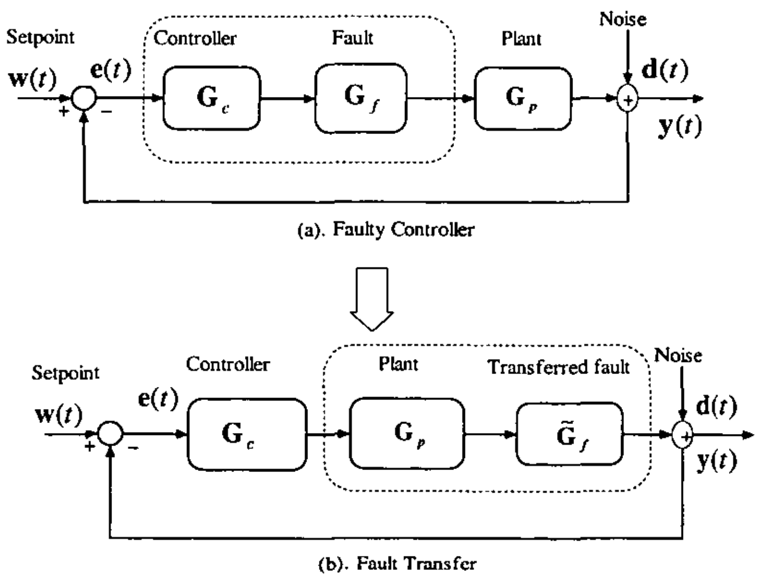


Figure 5.3. Transferring controller fault

The fault model $\mathbf{G}_f$ in fact doesn't exist in the actual control system; it is only an alternative way to mathematically describe the fault. Therefore transferring the model $\mathbf{G}_f$ into $\tilde{\mathbf{G}}_f$ is reasonable.

$\tilde{\mathbf{G}}_f$ can be obtained as follows: assume there are two group data available, one for the normal state of the open loop, the other for the faulty state. $\mathbf{E}_n$ and $\mathbf{Y}_n$ are the nominal inputs and outputs of the open loop, $\mathbf{E}_f$ and $\mathbf{Y}_f$ are the faulty inputs and outputs. Set $\mathbf{M}_1^{-1}$ as the estimated inverse model for the normal open loop, which can be identified by this data group: $\{\mathbf{E}_n, \mathbf{Y}_n\} \rightarrow \mathbf{M}_1^{-1}$

$$\mathbf{M}^{-1}{}_1 \approx [\mathbf{G}_p \mathbf{G}_c]^{-1} \tag{5.7}$$

$\mathbf{M}_f$ can be estimated by identifying the group of data: $\{\mathbf{Y}_f, \mathbf{E}_f\} \rightarrow \mathbf{M}_f$; therefore we have

$$\tilde{\mathbf{G}}_f = \mathbf{M}_f \mathbf{M}_1^{-1} \tag{5.8}$$

$$\mathbf{G}_{new} = \tilde{\mathbf{G}}_f \mathbf{G}_p \tag{5.9}$$

Based on the equivalent faulty process model, $\mathbf{G}_{new}$, we can design a new controller $\tilde{\mathbf{G}}_c$ to govern the equivalent plant. Therefore, the compensator model in MIMO can be obtained according to Equation 5.4:

$$\mathbf{G}_{com} = \frac{\tilde{\mathbf{G}}_c}{\mathbf{G}_c} - \mathbf{I} = \mathbf{G}_p \tilde{\mathbf{G}}_c \mathbf{M}_1^{-1} - \mathbf{I} \tag{5.10}$$

where $\mathbf{I}$ is a unit matrix.

5.2.4 Simulation Results

5.2.4.1 SISO Case Study

The simulation of a water tank process is implemented by using MATLAB. The transfer function of the water tank is $G_p = \frac{1}{s^2 + 10s + 20}$. The process is governed by a PD (Proportional-Derivative) controller with the parameters K_P=300 and K_D=10. The transfer function of the PD controller is: $G_c = 300 + 10s$.

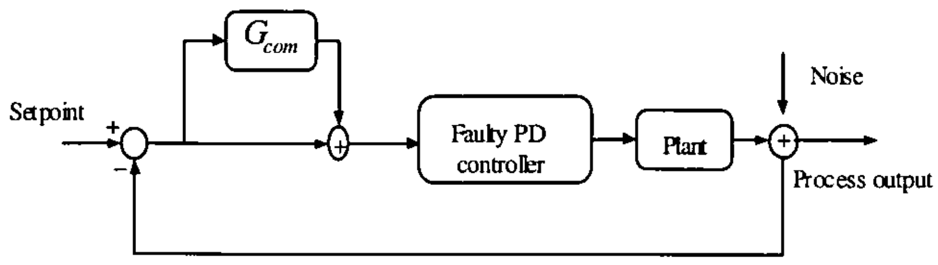


Figure 5.4. Serial link compensator for a PD controller

A fault occurs at the instant 200 where a constant gain is attached in the PD output outlet, $k = 0.1$, and makes the output of the PD controller $u_{PD\_f} = u_{PD} \times 0.1$, where u_{PD} is the control signals of the PD controller in the normal state, u_{PD_f} is the controller signal in the faulty case.

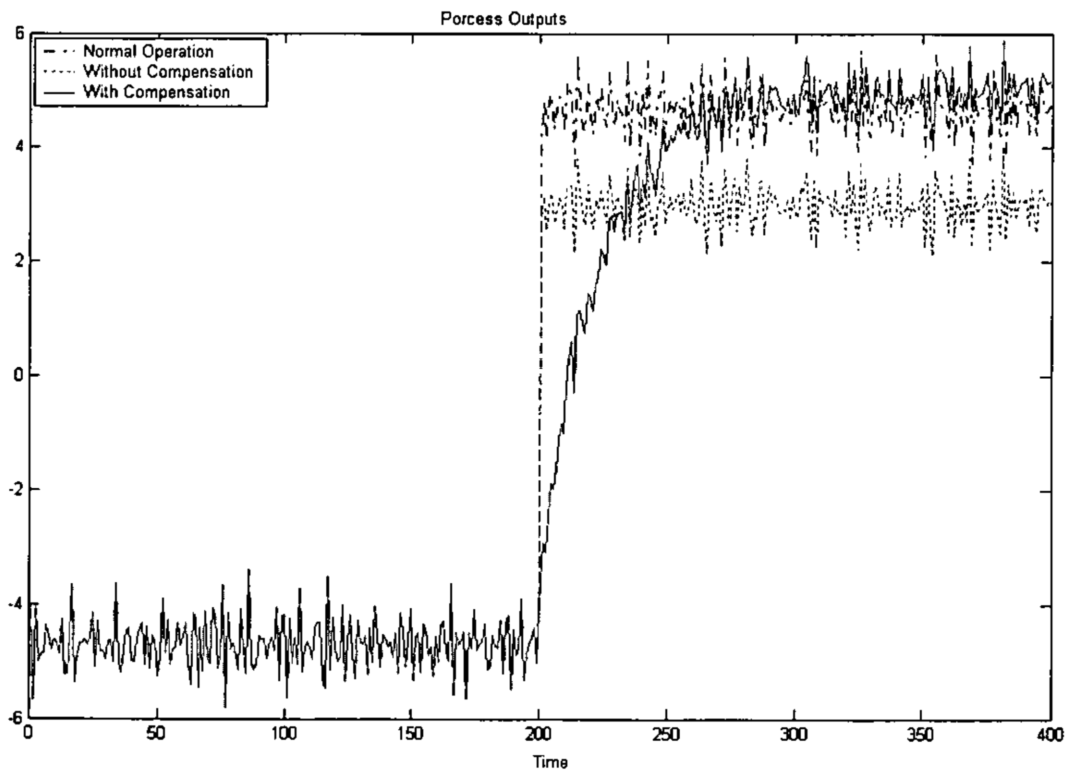


Figure 5.5. Faulty PD controller with and without compensation

To be simple, in this simulation, we directly use the faulty model $G_f = 0.1$ and transfer the fault to the plant; the equivalent faulty process model G_{new} is obtained:

$$G_{new} = \frac{0.1}{s^2 + 10s + 20}$$

According to G_{new}, we can design a new PI (Proportional-Integral) controller to govern the process with the parameters: K_p=30, K_I=10. The transfer function of the PI controller is:

$$\tilde{G}_c = 30 + 10/s.$$

The compensator model is calculated as follows according to Equation 5.4:

$$G_{com}(s) = \frac{-10s^2 - 270s + 10}{10s^2 + 300s}$$

The serial link compensator is implemented to maintain the faulty PD controller as shown in Figure 5.4. Figure 5.5 indicates that the degraded performances are recovered after the compensator is added in the process at the instant 200; the outputs with the compensator are able to track the setpoint satisfactorily.

5.2.4.2 MIMO Case Study

Consider a 2-input-2-ouptut control process, the plant model is:

$$\mathbf{G}_p = \begin{bmatrix} \frac{1}{s+3} & \frac{1}{s^2+4s+3} \\ \frac{2}{s^2+4s+3} & \frac{1}{s^2+4s+3} \end{bmatrix}$$

A feedback controller is designed to govern the process; the controller model is:

$$\mathbf{G}_c = \begin{bmatrix} \frac{4s^3+20s^2+32s+16}{2s^3+11s^2+23s+20} & \frac{4s^2+24s+32}{2s^3+17s^2+62s+80} \\ \frac{-4s^3-16s^2-20s-8}{2s^3+11s^2+23s+20} & \frac{8s^3+56s^2+112s+64}{2s^3+17s^2+62s+80} \end{bmatrix}$$

A set of white noises with variance 0.5 as disturbances are added into the reference control signals. At the instant 315, a fault occurred in the controller, which is a constant gain matrix:

$$\mathbf{G}_f = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

It is attached on the controller output. As the result, control signals are reduced by 0.1 times afterwards, which lead to the process output deviating away from the desired range. Under the gain matrix added in the control process, the impact of the white noises mixed in the control signals are also reduced to disturb the process output, and makes the variation range of the process output be flat as shown in Figure 5.7.

To maintain the faulty controller, we identify the normal open loop model of the process:

$$\mathbf{M}_1^{-1} = \begin{bmatrix} \dfrac{0.7268}{s+0.2167} & 0 \\ \dfrac{0.141}{s+0.2167} & 0 \end{bmatrix}$$

The faulty open loop model matrix is estimated as:

$$\mathbf{M}_f = \begin{bmatrix} \dfrac{0.5036}{4.919\,s+15.92} & 0 \\ \dfrac{0.763}{4.919\,s+15.92} & 0 \end{bmatrix}$$

$\tilde{\mathbf{G}}_f$ can be calculated according to Equation 5.8:

$$\tilde{\mathbf{G}}_f = \begin{bmatrix} \dfrac{0.366}{4.919\ s^2+16.99\ s+3.45} & 0 \\ \dfrac{0.55}{4.919\ s^2+16.99\ s+3.45} & 0 \end{bmatrix}$$

The step responses of $\tilde{\mathbf{G}}_f\mathbf{G}_p$ and $\mathbf{G}_p\mathbf{G}_f$ are shown in Figure 5. 6, which illustrates that the two models are approximately identical.

Applying $\tilde{\mathbf{G}}_f$ into Equation 5.9, the equivalent fault process model $\mathbf{G}_{new}$ is obtained:

$$\mathbf{G}_{new} = \begin{bmatrix} \dfrac{0.5655}{4.9\,s^3+31.7\,s^2+54.4\,s+10.3} & \dfrac{0.5655}{4.9\,s^4+31.7\,s^3+74.1\,s^2+78.3\,s+13.8} \\ \dfrac{1.052}{4.9\,s^3+31.7\,s^2+54.4\,s+10.3} & \dfrac{1.052}{4.9\,s^4+31.7\,s^3+74.1\,s^2+78.3\,s+13.8} \end{bmatrix}$$

Based on $\mathbf{G}_{new}$, a new LQG controller $\tilde{\mathbf{G}}_c$ is designed, and the controller model is given:

$$\tilde{\mathbf{G}}_c = \frac{1}{s^5+10.33\ s^4+46.32\ s^3+113.9\ s^2+153.1\ s+94.53} \times \begin{bmatrix} 52.7\ s^3+177.7\ s^2+292.4\ s+199.7 & 36.8\ s^3+114.9\ s^2+173.9\ s+102.4 \\ 18.83\ s^3+59.3\ s^2+89.7\ s+52.19 & 18.2\ s^3+57.8\ s^2+88.8\ s+53.2 \end{bmatrix}$$

According to Equation 5.10, the model of the compensator is calculated and simplified as:

$$G_{com} = \begin{bmatrix} \dfrac{0.6s^6 + 1.39s^5 + 3s^4 + 2.2s^3 + 0.9s^2 + 0.2s + 0.016}{s^6 + 7s^5 + 15.1s^4 + 10.75s^3 + 3.4s^2 + 0.5s + 0.03} & 0 \\ \dfrac{0.46s^4 + 0.28s^3 + 0.32s^2 + 0.1s + 0.006}{s^8 + 8.96s^7 + 30s^6 + 47.9s^5 + 40.03s^4 + 18.11s^3 + 4.47s^2 + 0.57s + 0.03} & 4 \end{bmatrix}$$

Implementing the compensator to the faulty control process, results in Figure 5.7 show that the degraded performance is satisfactorily recovered.

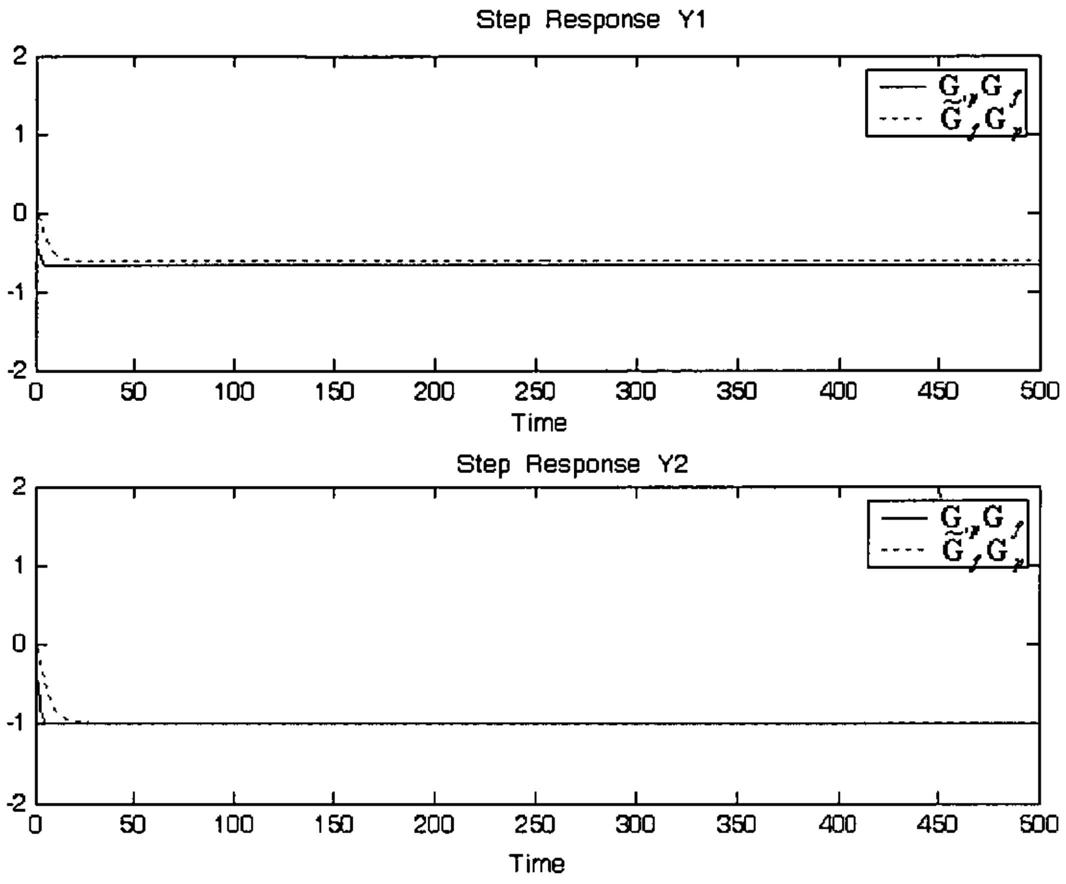


Figure 5.6. Step response

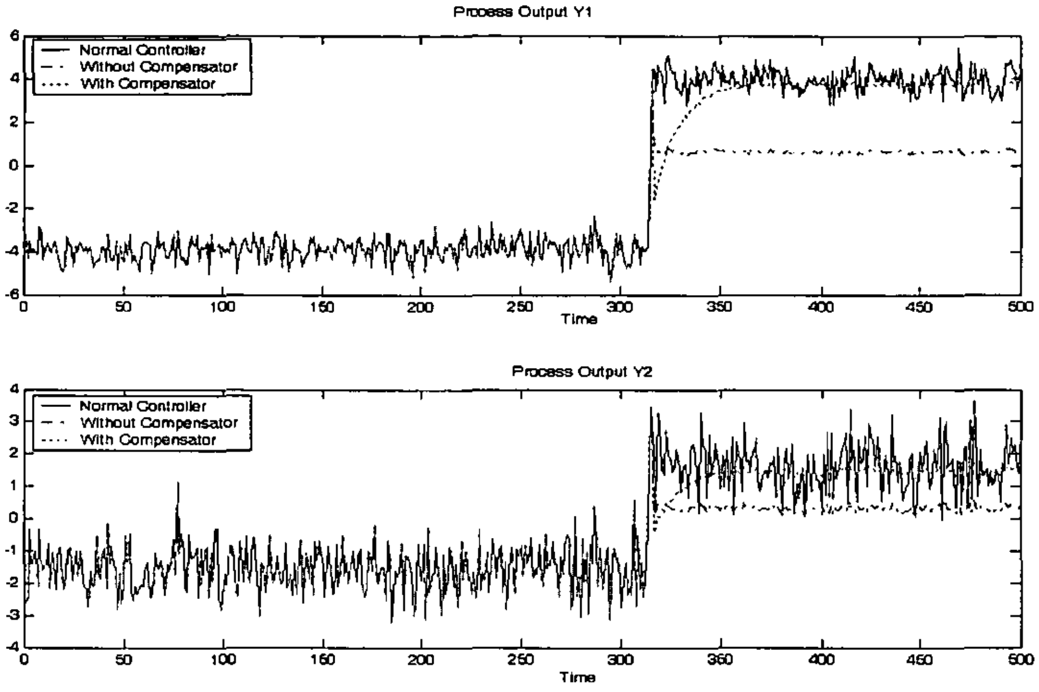


Figure 5.7. Faulty control process with and without compensation

Remark: The fault shift-based compensation is based on the fault equivalent transfer method, which shifts any fault occurring in the controller to the plant. A compensator is designed to maintain the control performance for the equivalent faulty plant and is then applied to the case where the controller failure occurs. The merit of the proposed approach is that it extends the range of traditional FTCs and focuses on the controller failure maintenance. However, there are several limitations for the proposed compensation approach in the actual applications, including:

1) It involves heavy computations. To design a compensator, we have to compute the fault model, $\mathbf{G}_f$, and then calculate the compensator model $\mathbf{G}_{com}$; The calculation procedure is complicated;
2) The assumption about the model, $\tilde{\mathbf{G}}_f$, is limited; in some situations, there may not exist such $\tilde{\mathbf{G}}_f$ to make $\tilde{\mathbf{G}}_f\mathbf{G}_p = \mathbf{G}_p\mathbf{G}_f$;
3) The compensator model calculation is required the model of the faulty controller, $\tilde{\mathbf{G}}_c$; as a result, the proposed compensation approach can be not applicable for the advanced controllers, such as model predictable controllers, which may not have a fixed controller model.

5.3 Design of a Model Reference-Based Compensator

5.3.1 Compensator Design

The compensation in the case of the controller failure can be also achieved by the application of model following technique to design a compensator. Suppose a faulty open loop of a system, which can be described by the following representation as:

$$\begin{aligned} \dot{x} &= \mathbf{F}x + \mathbf{G}e + q^{x} \\ y &= \mathbf{H}x + \mathbf{D}e \end{aligned} \tag{5.11}$$

where $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{H}$ and $\boldsymbol{D}$ are proper matrices. x is the system state, e the difference between the setpoint and process outputs, y the output, q^{x} is a zero mean white Gaussian sequence with covariance Q^{x}.

Consider that a desired reference model is represented by:

$$\begin{cases} \dot{x}^{m} = \mathbf{F}^{m}x^{m} + \mathbf{G}^{m}u^{m} \\ y^{m} = \mathbf{H}^{m}x^{m} + \mathbf{D}^{m}u^{m} \end{cases} \tag{5.12}$$

where $\mathbf{F}^{m}$, $\mathbf{G}^{m}$, $\mathbf{H}^{m}$ and $\mathbf{D}^{m}$ are proper matrices, x^{m} is the reference model state, y^{m} is the process output of the reference model and u^{m} is the control signal of the reference model.

Assume that a compensator is inserted into the faulty control loop as shown in Figure 5.8 (a), and then the faulty open loop model in Equation 5.11 is rewritten as:

$$\begin{cases} \dot{x}^{com} = \mathbf{F}x^{com} + \mathbf{G}u^{com} + q^{x} \\ y = \mathbf{H}x^{com} + \mathbf{D}u^{com} \end{cases} \tag{5.13}$$

where u^{com} is the compensator signal and x^{com} is the system state under the compensator control.

The problem of determining the perfect tracking solution is formulated as follows: design a compensator and make the compensator control signals u^{com} to drive the tracking error ε to zero asymptotically. The error ε is defined as follows:

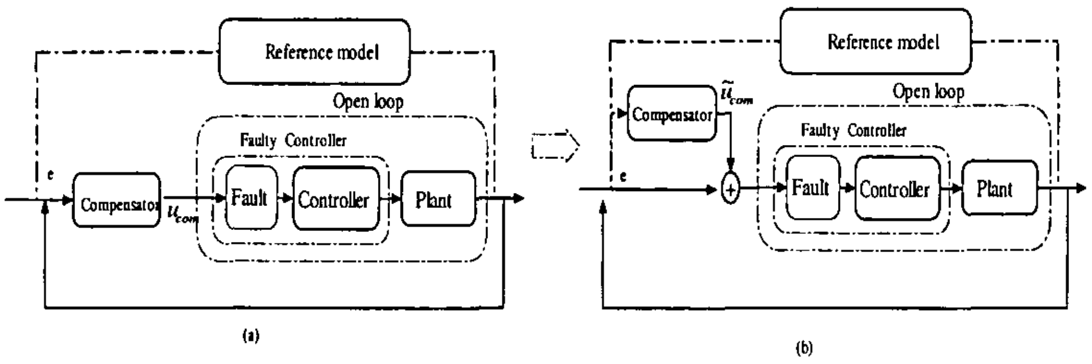


Figure 5.8. Compensator in the faulty control loop

$$\varepsilon = y - y^m = [\mathbf{H} \quad \mathbf{D}]\begin{bmatrix} x^{com} \\ u^{com} \end{bmatrix} - [\mathbf{H}^m \quad \mathbf{D}^m]\begin{bmatrix} x^m \\ u^m \end{bmatrix}$$

When perfect tracking occurs, the resulting process output and system state are denoted as:

$$\begin{cases} y^{com} = [\mathbf{H} \quad \mathbf{D}]\begin{bmatrix} x^{com} \\ u^{com} \end{bmatrix} = [\mathbf{H}^m \quad \mathbf{D}^m]\begin{bmatrix} x^m \\ u^m \end{bmatrix} \\ \dot{x}^{com} = \mathbf{F}x^{com} + \mathbf{G}u^{com} + q^x \end{cases} \tag{5.14}$$

The ideal control signal u^{com} generating perfect output tracking and the ideal state trajectories x^{com} are assumed to be a linear combination of the model states and model input (Ozcelik and Kaufman, 1999):

$$\begin{bmatrix} x^{com} \\ u^{com} \end{bmatrix} = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix}\begin{bmatrix} x^m \\ u^m \end{bmatrix} \tag{5.15}$$

where $\mathbf{S}_{i,j}$ are proper matrices

Matrices $\mathbf{S}_{i,j}$ satisfy an algebraic matrix equation by determining expression for $\begin{bmatrix} \dot{x}^{com} \\ y^{com} \end{bmatrix}$ (Broussard and O'Brien, 1980). Differencing x^{com} in Equation 5.15, and using $\dot{x}^m$ in Equation 5.11, y^{com} in Equation 5.14, we have:

$$\begin{bmatrix} \dot{x}^{com} \\ y^{com} \end{bmatrix} = \begin{bmatrix} \mathbf{S}_{11}F^m & \mathbf{S}_{11}G^m \\ \mathbf{H}^m & \mathbf{D}^m \end{bmatrix}\begin{bmatrix} x^m \\ u^m \end{bmatrix} + \begin{bmatrix} \mathbf{S}_{12}\dot{u}^m \\ 0 \end{bmatrix} \tag{5.16}$$

Using $\dot{x}^{com}$ and y^{com} in Equation 5.14, and x^{com} and u^{com} in Equation 5.15 we have:

$$\begin{bmatrix} \dot{x}^{com} \\ y^{com} \end{bmatrix} = \begin{bmatrix} \mathbf{F} & \mathbf{G} \\ \mathbf{H} & \mathbf{D} \end{bmatrix}\begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix}\begin{bmatrix} x^m \\ u^m \end{bmatrix} + \begin{bmatrix} q^x \\ 0 \end{bmatrix} \tag{5.17}$$

Therefore, we have:

$$\begin{bmatrix} \mathbf{F} & \mathbf{G} \\ \mathbf{H} & \mathbf{D} \end{bmatrix}\begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix}\begin{bmatrix} x^m \\ u^m \end{bmatrix}+\begin{bmatrix} q^x \\ 0 \end{bmatrix}=\begin{bmatrix} \mathbf{S}_{11}\mathbf{F}^m & \mathbf{S}_{11}\mathbf{G}^m \\ \mathbf{H}_m & \mathbf{D}_m \end{bmatrix}\begin{bmatrix} x^m \\ u^m \end{bmatrix}+\begin{bmatrix} \mathbf{S}_{12}\dot{u}^m \\ 0 \end{bmatrix} \tag{5.18}$$

One of possible solution for Equation 5.18 is to make (Broussard and O'Brien, 1980):

$$\begin{bmatrix} \mathbf{F} & \mathbf{G} \\ \mathbf{H} & \mathbf{D} \end{bmatrix}\begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix}=\begin{bmatrix} \mathbf{S}_{11}\mathbf{F}^m & \mathbf{S}_{11}\mathbf{G}^m \\ \mathbf{H}^m & \mathbf{D}^m \end{bmatrix} \tag{5.19}$$

Set

$$\begin{bmatrix} \boldsymbol{\tau}_{11} & \boldsymbol{\tau}_{12} \\ \boldsymbol{\tau}_{21} & \boldsymbol{\tau}_{22} \end{bmatrix}=\begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix}\begin{bmatrix} \mathbf{S}_{11}\mathbf{F}^m & \mathbf{S}_{11}\mathbf{G}^m \\ \mathbf{H}^m & \mathbf{D}^m \end{bmatrix}^{-1} \tag{5.20}$$

Matrices $\boldsymbol{\tau}_{ij}$ can be obtained by:

$$\begin{bmatrix} \boldsymbol{\tau}_{11} & \boldsymbol{\tau}_{12} \\ \boldsymbol{\tau}_{21} & \boldsymbol{\tau}_{22} \end{bmatrix}=\begin{bmatrix} \mathbf{F} & \mathbf{G} \\ \mathbf{H} & \mathbf{D} \end{bmatrix}^{-1} \tag{5.21}$$

The model reference-based compensator design is subject to the inverses of matrices $\begin{bmatrix} \mathbf{S}_{11}\mathbf{F}^m & \mathbf{S}_{11}\mathbf{G}^m \\ \mathbf{H}^m & \mathbf{D}^m \end{bmatrix}$ and $\begin{bmatrix} \mathbf{F} & \mathbf{G} \\ \mathbf{H} & \mathbf{D} \end{bmatrix}$, which exist when the two matrices are square nonsingular matrices and the determinant of the both matrices, $\left\|\begin{bmatrix} \mathbf{S}_{11}\mathbf{F}^m & \mathbf{S}_{11}\mathbf{G}^m \\ \mathbf{H}^m & \mathbf{D}^m \end{bmatrix}\right\| \neq 0$ and $\left\|\begin{bmatrix} \mathbf{F} & \mathbf{G} \\ \mathbf{H} & \mathbf{D} \end{bmatrix}\right\| \neq 0$.

Equation 5.20 is re-written as:

$$\begin{bmatrix} \boldsymbol{\tau}_{11} & \boldsymbol{\tau}_{12} \\ \boldsymbol{\tau}_{21} & \boldsymbol{\tau}_{22} \end{bmatrix}\begin{bmatrix} \mathbf{S}_{11}\mathbf{F}^m & \mathbf{S}_{11}\mathbf{G}^m \\ \mathbf{H}^m & \mathbf{D}^m \end{bmatrix}=\begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix} \tag{5.22}$$

Therefore, matrices $\mathbf{S}_{ij}$ can be solved according to Equation 5.22:

$$\mathbf{S}_{11} = \boldsymbol{\tau}_{11}\mathbf{S}_{11}\mathbf{F}^m + \boldsymbol{\tau}_{12}\mathbf{H}^m \tag{5.23}$$

$$\begin{cases} \mathbf{S}_{12} = \boldsymbol{\tau}_{11}\mathbf{S}_{11}\mathbf{G}^m + \boldsymbol{\tau}_{12}\mathbf{D}^m \\ \mathbf{S}_{21} = \boldsymbol{\tau}_{21}\mathbf{S}_{11}\mathbf{F}^m + \boldsymbol{\tau}_{22}\mathbf{H}^m \\ \mathbf{S}_{22} = \boldsymbol{\tau}_{21}\mathbf{S}_{11}\mathbf{G}^m + \boldsymbol{\tau}_{22}\mathbf{D}^m \end{cases} \tag{5.24}$$

Changing the compensator link from Figure 5.8 (a) to Figure 5.8 (b), the compensator control signal $\tilde{u}^{com}$ is given as:

$$\tilde{u}^{com} = \mathbf{S}_{21}x^m + \mathbf{S}_{22}u^m - e \tag{5.25}$$

5.3.2 Calculation of $\mathbf{S}_{11}$

It is clear that once $\mathbf{S}_{11}$ is determined all the other $\mathbf{S}_{ij}$ can be obtained. Actually, the solution for $\mathbf{S}_{11}$ in Equation 5.23 is a famous matrix problem. Equation 5.23 can be rewritten as:

$$\mathbf{S}_{11}\mathbf{F}^{m^{-1}} = \boldsymbol{\tau}_{11}\mathbf{S}_{11} + \boldsymbol{\tau}_{12}\mathbf{H}^m\mathbf{F}^{m^{-1}}$$

$$\boldsymbol{\tau}_{11}\mathbf{S}_{11} - \mathbf{S}_{11}\mathbf{F}^{m^{-1}} = \boldsymbol{\tau}_{12}\mathbf{H}^m\mathbf{F}^{m^{-1}} \tag{5.26}$$

where the solution of $\mathbf{S}_{11}$ is subject to the matrix $\mathbf{F}^{m^{-1}}$, which exists when $\mathbf{F}^m$ is square nonsingular and $|\mathbf{F}^m| \neq 0$.

Equation 5.26 can be solved according to (Bartels and Stewart, 1972).

$$\text{Set} \quad \tilde{\mathbf{A}} = \boldsymbol{\tau}_{11} \qquad \tilde{\mathbf{B}} = -\mathbf{F}^{m^{-1}}$$

$$\tilde{\mathbf{C}} = \boldsymbol{\tau}_{12}\mathbf{H}^m\mathbf{F}^{m^{-1}} \quad \mathbf{X} = \mathbf{S}_{11}$$

Therefore Equation 5.26 can be re-written as:

$$\tilde{\mathbf{A}}\mathbf{X} + \mathbf{X}\tilde{\mathbf{B}} = \tilde{\mathbf{C}} \tag{5.27}$$

Equation 5.26 has a unique solution if and only if the eigenvalues a_1, $a_2 \dots a_m$ of $\tilde{\mathbf{A}}$ and b_1, $b_2, \dots, b_n$ of $\tilde{\mathbf{B}}$ satisfy

$$a_i + b_i \neq 0, (\, i=1,2,..m; j=1,2,..n)$$

Equation 5.27 is solved as follows. The matrix $\tilde{\mathbf{A}}$ is reduced to a lower real Schur form $\tilde{\mathbf{A}}'$ by an orthogonal similarity transformation *U*; $\tilde{\mathbf{A}}$ is reduced to the real, block lower triangular form:

$$\tilde{\mathbf{A}}' = \mathbf{U}^T\tilde{\mathbf{A}}\mathbf{U} = \begin{bmatrix} \tilde{\mathbf{A}}'_{11} & 0 & \dots & 0 \\ \tilde{\mathbf{A}}'_{21} & \tilde{\mathbf{A}}'_{21} & 0 & \dots \\ \dots & \dots & \dots & 0 \\ \tilde{\mathbf{A}}'_{p1} & \tilde{\mathbf{A}}'_{p2} & \dots & \tilde{\mathbf{A}}'_{pp} \end{bmatrix}$$

where each matrix $\tilde{\mathbf{A}}'_{ii}$ is of order at most two. Similarly $\tilde{\mathbf{B}}$ is reduced to upper real Schur form by the orthogonal matrix *V*:

$$\tilde{\mathbf{B}}' = V^T \tilde{\mathbf{B}} V = \begin{bmatrix} \tilde{\mathbf{B}}'_{11} & \tilde{\mathbf{B}}'_{12} & \dots & \tilde{\mathbf{B}}'_{1q} \\ 0 & \tilde{\mathbf{B}}'_{21} & \dots & \tilde{\mathbf{B}}'_{2q} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \tilde{\mathbf{B}}'_{qq} \end{bmatrix}$$

where again each $\tilde{\mathbf{B}}'_{ii}$ is of order at most two. If

$$\tilde{\mathbf{C}}' = \mathbf{U}^T \tilde{\mathbf{C}} \mathbf{V} = \begin{bmatrix} \tilde{\mathbf{C}}'_{11} & \dots & \dots & \tilde{\mathbf{C}}'_{1q} \\ & & & \\ \dots & \dots & \dots & \dots \\ \tilde{\mathbf{C}}'_{P1} & \dots & \dots & \tilde{\mathbf{C}}'_{Pq} \end{bmatrix}$$

and

$$\mathbf{X}' = \mathbf{U}^T \mathbf{X} \mathbf{V} = \begin{bmatrix} \mathbf{X}'_{11} & \dots & \dots & \mathbf{X}'_{1q} \\ & & & \\ \dots & \dots & \dots & \dots \\ \mathbf{X}'_{P1} & \dots & \dots & \mathbf{X}'_{Pq} \end{bmatrix}$$

then Equation 5.27 is equivalent to

$$\tilde{\mathbf{A}}'\mathbf{X}' + \mathbf{X}'\tilde{\mathbf{B}}' = \tilde{\mathbf{C}}'$$

If the partitions of $\tilde{\mathbf{A}}'$, $\tilde{\mathbf{B}}'$, $\tilde{\mathbf{C}}'$, and X' are conformal, then

$$\tilde{\mathbf{A}}'_{kk}\,\mathbf{X}'_{kl} + \mathbf{X}'_{kl}\,\tilde{\mathbf{B}}'_{ll} = \tilde{\mathbf{C}}'_{kl} - \sum_{j=1}^{k-1} \tilde{\mathbf{A}}'_{kj}\,\mathbf{X}'_{jl} - \sum_{j=1}^{l-1} X'_{jki}\,\tilde{\mathbf{B}}'_{il}$$
$$(k=1, 2\dots p;\ l=1, 2\dots q)$$

These equations may be solved successively for $\mathbf{X}'_{11}, \mathbf{X}'_{21}, \dots \mathbf{X}'_{p1}, \mathbf{X}'_{12}, \mathbf{X}'_{22}, \dots\dots$

The solution of Equation 5.27 is then given by $\mathbf{X} = \mathbf{U}\mathbf{X}'\mathbf{V}^T$.

5.4 Controller Failure Detection and Maintenance Scheme

The proposed techniques of the control performance assessment, controller fault detection and maintenance can be incorporated into a scheme which assesses the controller performance, monitors the control signals and maintains the faulty controller to recover the degraded control performance to an acceptable range.

The main objective for the scheme is to maintain the control performance in an acceptable range, which should include the dynamic and the steady-state performance both under the normal and faulty situations. The performance observer on-line assesses the control performance via a performance index, which compares the actual achieved objective functions and the LQG benchmark values. When the performance is less than the predefined values, it is an unacceptable performance; the fault detector is triggered to check whether the cause is from the controller or not. The detector uses a moving window to observe the target control signals in real time. The control signals of the LQG control are treated as desired signals of the target controller. The detection of the controller failure is through a General Likelihood Ratio (GLR) based detector which observes the actual controller behavior and identifies a jump point of the discrepancies. To accommodate the controller failure, a compensator is inserted into the post-fault control loop in order to recover the dynamics of the closed-loop system.

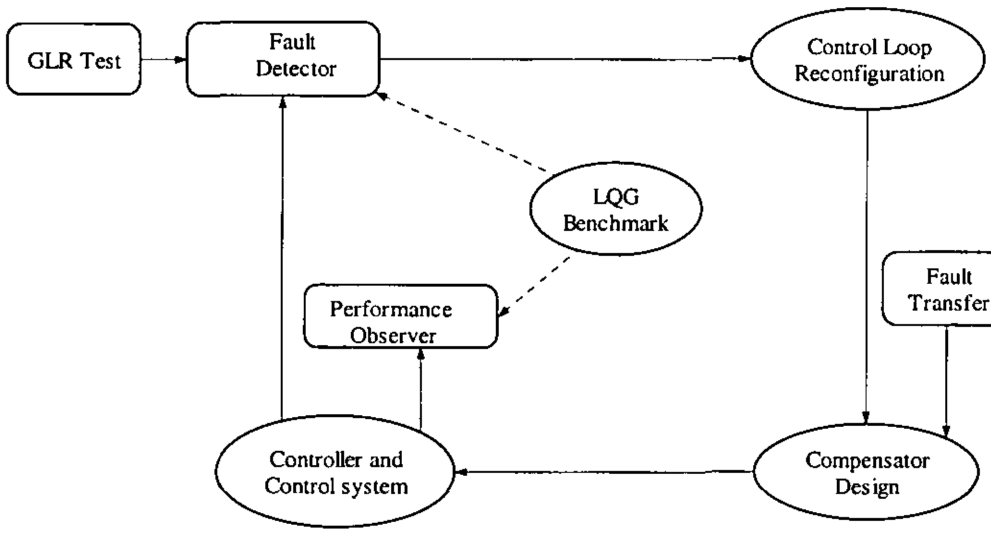


Figure 5.9. Controller failure detection and maintenance scheme

5.5 Case Study

In order to demonstrate the proposed scheme, a Process Control Unit has been chosen as a test-bed. The details of implementation are discussed in this section.

5.5.1 Experiment Description

The Process Control Unit (PCU) as shown in Figure 5.10 consists of the Process Rig, Computer Control Module, Power Supply and a PC internal interface card, together with comprehensive interactive control software. The Process Rig comprises two tanks - a process tank and a sump tank - as well as connecting pipes, a heater, and an alternative flow path through a cooler. Flow rate through the pipes, fluid temperature in the process tank and pipes, and fluid level in the process tank are all measured. This configuration is used to model the flow rate in pipes, the water temperature and the fluid level in the tank. The sump contains the store of fluid, which may be pumped at a controllable flow rate. From the pump, fluid may flow directly to the process tank or may be diverted via the cooler to cool the fluid temperature and flow back to the process tank. The process tank contains a heating element, which heats the water in the tank. The heated fluid in this tank can either overflow back into the sump or be drained by using the manual drain tap or the computer controlled drain solenoid, thus completing the fluid cycle.

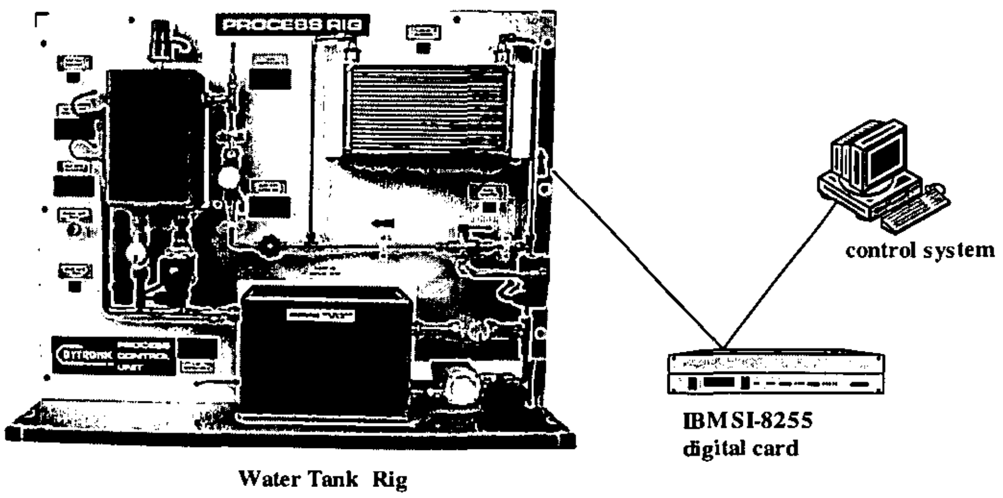


Figure 5.10. PCU rig

5.5.2 SISO Case

The water level is controlled by a PID controller. The I/O interface manages the data acquisition and signal conversion from analogue to digital and from digital to analogue. The PID controller measures the liquid level of the process tank and regulates the flow rate of the pump to maintain the liquid level of the process tank at a desired value.

5.5.2.1 Experimental Results

Figure 5.11 shows the water level process governed by the PID controller. In the normal operation, the outlet flow rate of the water tank is constant. The PID controller regulates the flow rate of the pump to maintain the desired water level. A faulty environment is established in this experiment, which increases the outlet flow rate. As a result, the original balance between the inlet controlled by the PID controller and the liquid level is upset as more water flows out; the PID controller fails to govern the new water tank process, and the water level is significantly below the reference as shown in Figure 5.11.

Table 5.1 System and reference models in SISO case

	F	G	H	D
Reference Model	$\begin{bmatrix} 0.5134 & 0 \\ 0 & 0.6065 \end{bmatrix}$	$\begin{bmatrix} 0.3299 \\ 0.1740 \end{bmatrix}$	$\begin{bmatrix} 0.6667 & 1.2500 \end{bmatrix}$	0
Faulty Open loop	$\begin{bmatrix} 1 & 0.02778 \\ 4 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.0375 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -0.2346 \end{bmatrix}$	1.111

A LQG control is designed via MPC solution in this experiment. Besides being used as a benchmark to monitor the PID controllers, the LQG control can be also used to assess the control process performance by comparing the actual objective function values with the LQG benchmark values (Huang and Shah, 1999). The performance index is given:

$$\eta = \frac{J_{LQG}}{J_{act}} \tag{5.28}$$

where J_{LQG} is the cost function of the LQG controller and J_{act} is the cost function of the actual controller.

Figure 5.12 shows the water level process performance calculated according to Equation 5.28. In the normal operation, the performance is healthy and the index is around 0.86. When the process situation is changed at the instant 300, the performance is deteriorated and the index is down to 0.65 shown in Figure 5.12. In the normal work state, 600 observed residuals are collected to fit the Gaussian distribution parameters and the identified mean μ_0 and variance σ_0 are 0.25821 and 0.5232 respectively. Figure 5.13 shows the probability density function (PDF) of the sampled residuals and implies that the sampling observed residuals are approximated to follow a Gaussian distribution. The fault detector uses a moving observation window to monitor the residuals; the threshold, λ_1, is set as 1.5. The GLR test detects the varied value of the mean of the observed residuals. Figure 5.14 shows the GLR test results for the discrepancies between the LQG and the PID control signals. After the instant 300, the likelihood ratio for the PID control signals is crossed the thread, which implies that the PID controller is faulty.

The reference model and the faulty open models in both normal and faulty operations are given as shown in Table 5.1 and the disturbance is a white noise with covariance 0.3.

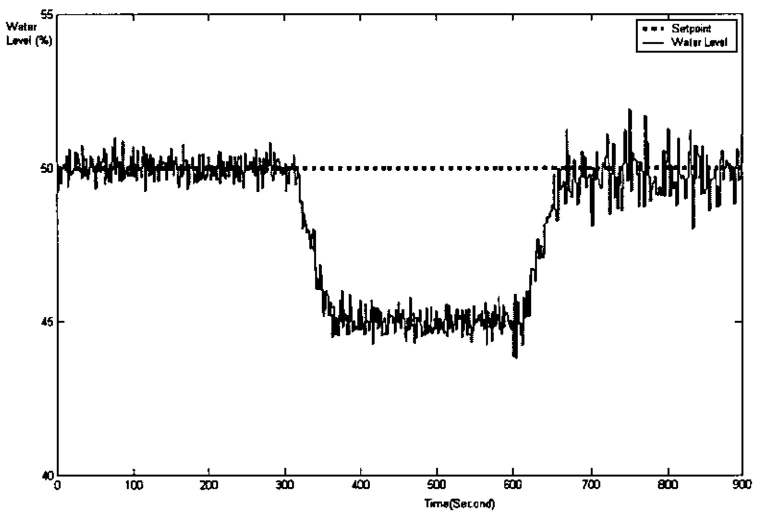


Figure 5.11. Water Level

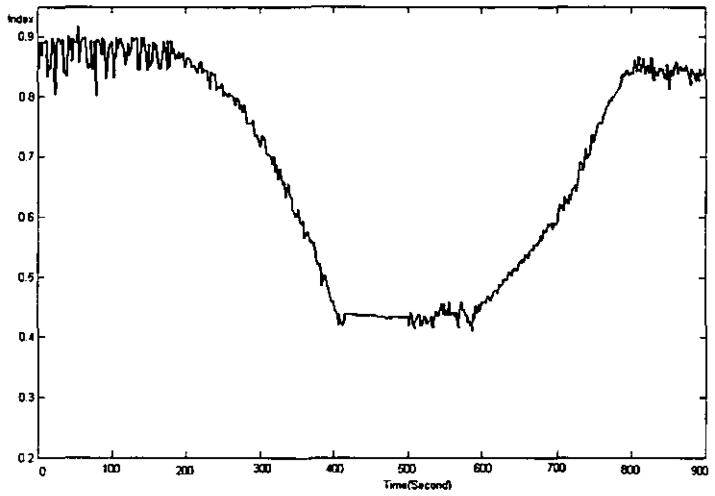


Figure 5.12. Performance index

5.5.2.2 Calculation of Compensator Model

According to Equation 5.21 and the reference model shown in Table 5.1, $\tau_{11}, \tau_{12}, \tau_{21}, \tau_{22}$ are obtained as:

$$\tau_{11} = \begin{bmatrix} 0 & 0.2500 \\ 50.0835 & -12.5209 \end{bmatrix}, \tau_{12} = \begin{bmatrix} 0 \\ 1.6678 \end{bmatrix}$$

$$\tau_{21} = \begin{bmatrix} -10.5732 & 2.6433 \end{bmatrix}, \tau_{22} = -1.2521$$

and $S_{11}, S_{12}, S_{21}, S_{22}$ are calculated as follows :

$$S_{11} = \begin{bmatrix} 0.0346 & 0.0793 \\ 0.2694 & 0.5227 \end{bmatrix}, S_{12} = \begin{bmatrix} 0.0450 \\ -0.9895 \end{bmatrix}$$

$$S_{21} = \begin{bmatrix} -0.6569 & -1.2354 \end{bmatrix}, S_{22} = 0.2089$$

Implementing the compensator into the controller failure scenario where the compensator control signals are obtained according to Equation 5.25, the results under the compensation in the Figure 5.11 clearly indicate the outputs of original faulty control loop with the added compensator are able to track the setpoint satisfactorily and the performance index is back to 0.82 shown in Figure 5.12.

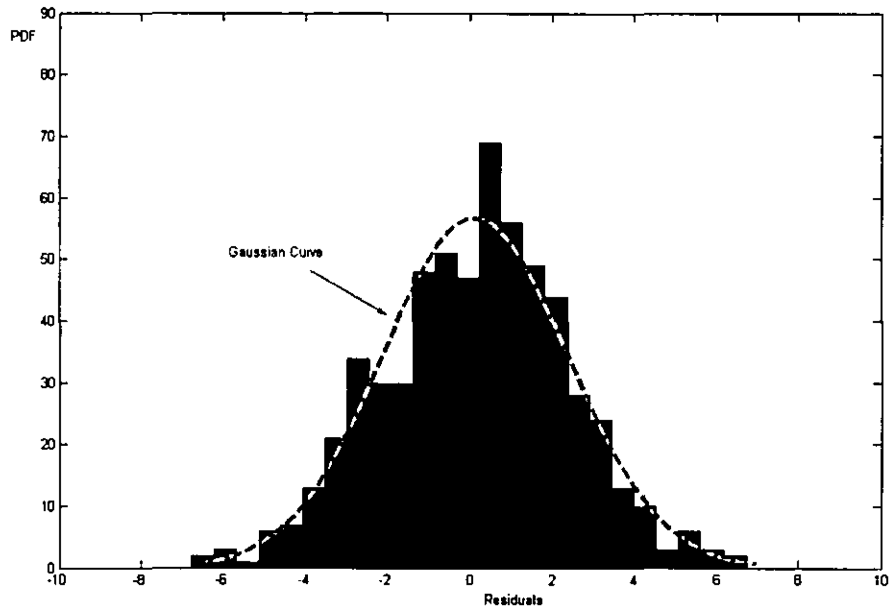


Figure 5.13. Possibilities density function (PDF) of the collected residuals

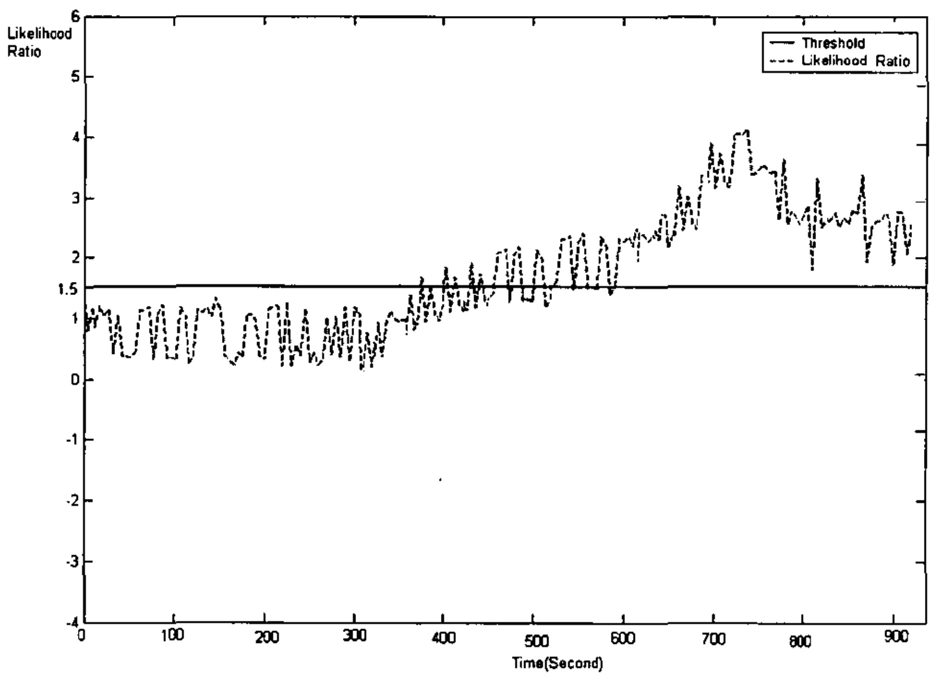


Figure 5.14. GLR test results

5.5.3 MIMO Case

Designing a Model Predictive Controller (MPC) to control the water level and temperature, the MPC measures the liquid level and temperature of the tank water and regulates the flow rate of the pump and the power supply of the heater to maintain the liquid level and water temperature at desired values. The water tank process model is identified as:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \dfrac{0.1594}{1-0.9442z^{-1}} & 0.012 \\ \dfrac{0.02144}{1-0.0939z^{-1}} & \dfrac{-0.25}{1-0.0029z^{-1}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

where y_1 and y_2 are water level and temperature respectively, u_1 and u_2 are control signals for water level and temperature respectively and z is the transfer symbol.

The control gain of the MPC is calculated as:

$$Gain_{MPC} = \begin{bmatrix} 0.17 & 0.0075 & 0.13 & -0.0014 & 0.1 & 0 & 0.1 & -0.0004 & 0.06 & 0 & 0.04 & 0.004 \\ 0.0009 & -0.22 & 0.0011 & -0.1633 & 0.0013 & -0.1050 & 0.0014 & -0.05 & 0.001 & 0 & 0.0017 & 0.03 \end{bmatrix}$$

The faulty environment is built by changing the control gain of the MPC at the instant 300 into:

$$\tilde{G}ain_{MPC} = \begin{bmatrix} 0.017 & 0.0025 & 0.013 & -0.0014 & 0.01 & 0 & 0 & 0 & 0.001 & 0 & 0.011 & 0.0011 \\ 0.0009 & -0.22 & 0.0011 & -0.1633 & 0.0013 & -0.1050 & 0.0014 & -0.05 & 0.001 & 0 & 0.0017 & 0.03 \end{bmatrix}$$

Based on the new control gain, the MPC will generate unsuitable control signals and fail to control the water tank process.

5.5.3.1 Experimental Results

Changing control gain, $Gain_{MPC}$ into $\tilde{G}ain_{MPC}$, it has more serious impact on the signal for the water level. As shown in Figures 5.15 and 5.16, the water level is significantly down from the setpoint and the water temperature is slightly increased. Both the water level and the temperature deviating from the setpoint lead to degrading the control performance as shown in Figure 5.17. The MPC control signals are detected by the GLR detector. Figure 5.18 shows the likelihood values for detecting the varied mean of residuals of the water level control signal. Figure 5.19 indicates the GLR test results for the residual variance. Both show that the likelihood values are cross the threshold, which implies that the MPC is faulty after the instant 300. Figures 5.20 and 5.21 show the test results for the control signal of the water temperature. The test results are slightly changed after the instant 300, and are still under the threshold, which implies that the changing control gain has minor impact on the control signal of the water temperature.

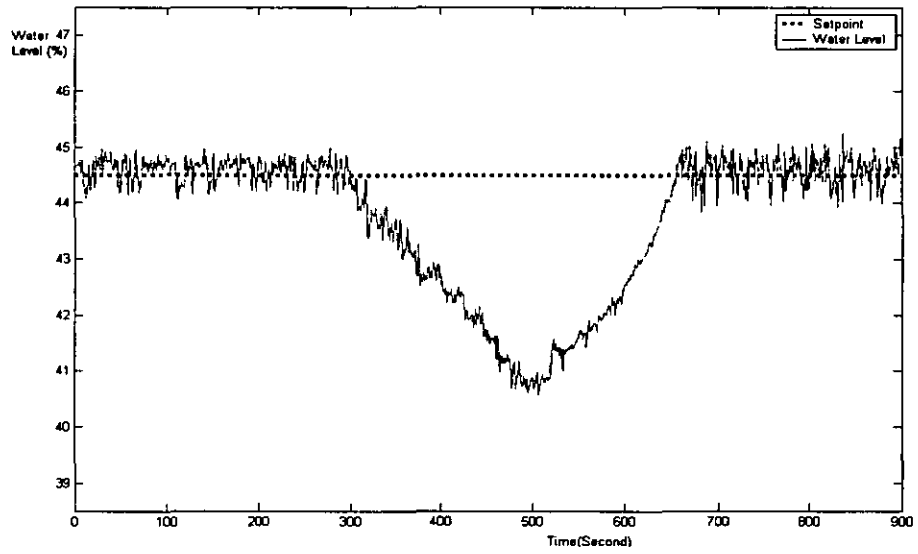


Figure 5.15. Water level

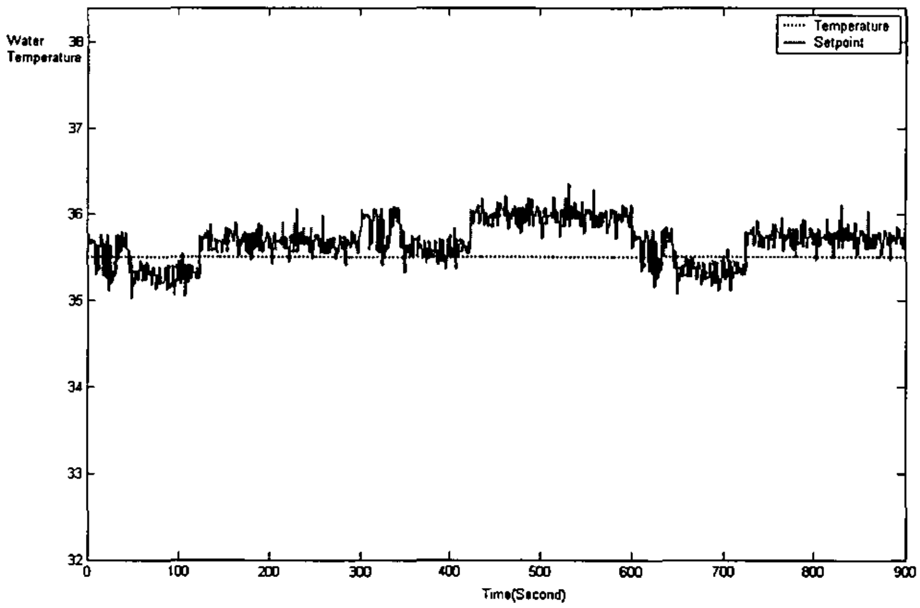


Figure 5.16. Water temperature

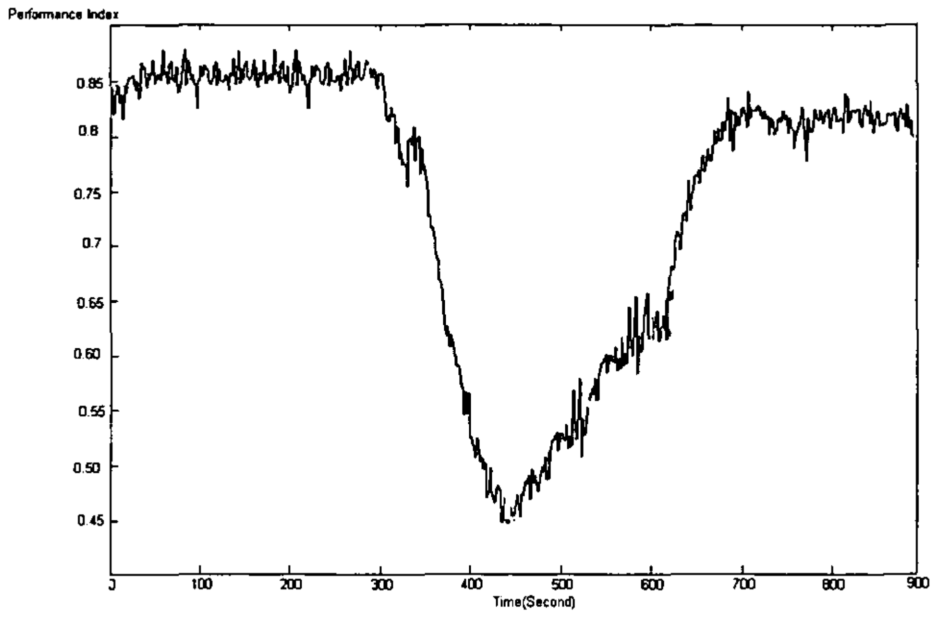


Figure 5.17. The monitored control performance

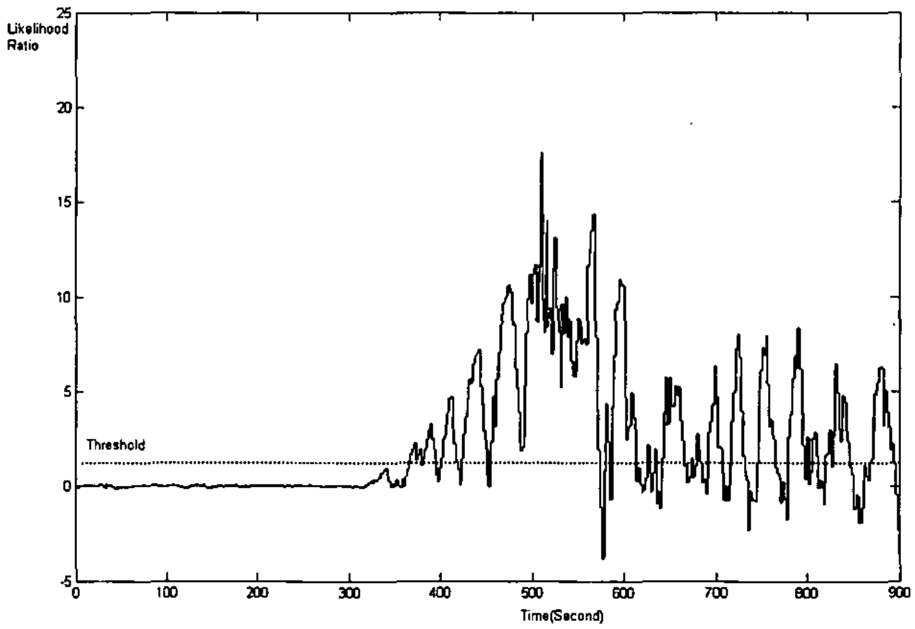


Figure 5.18. The mean value of the likelihood ratio test (for the control signal for water level)

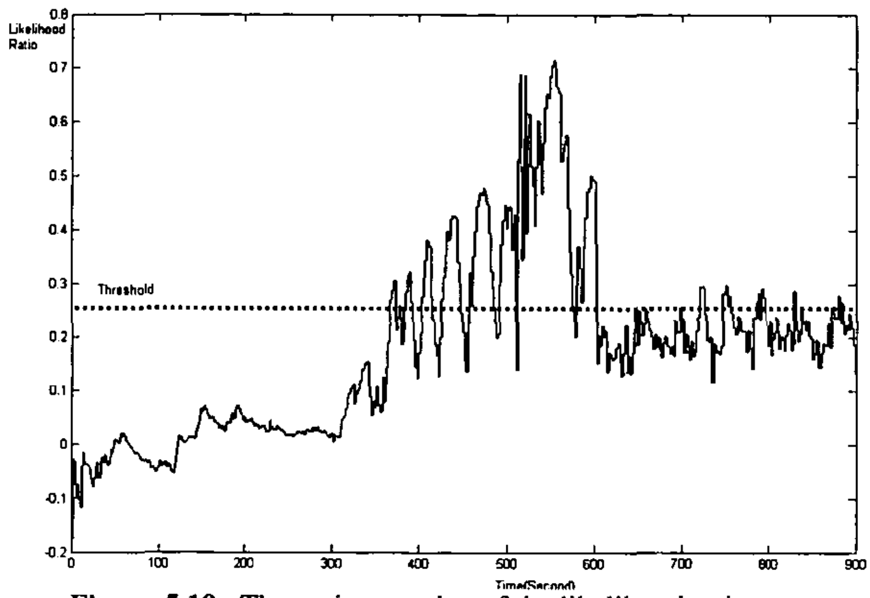


Figure 5.19. The variance value of the likelihood ratio test

(for the control signal for water level)

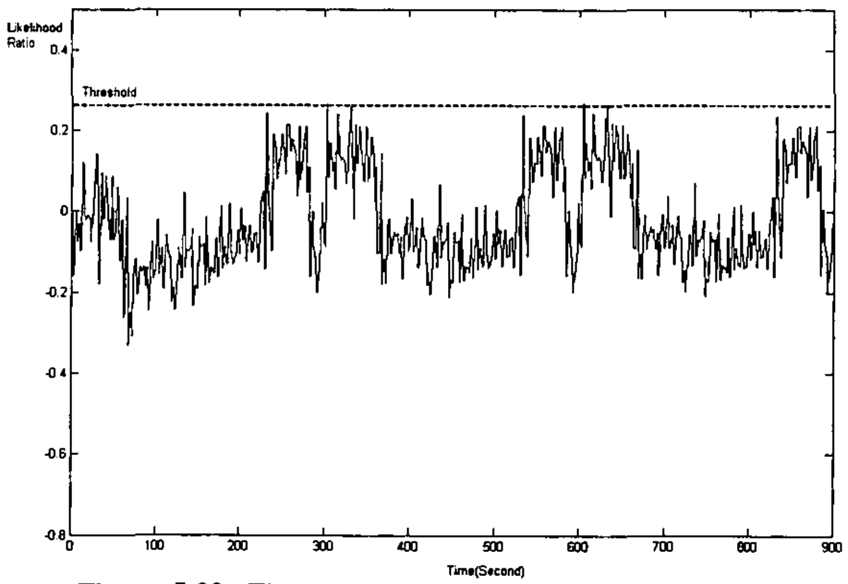


Figure 5.20. The mean value of the likelihood ratio test

(for the control signal for water temperature)

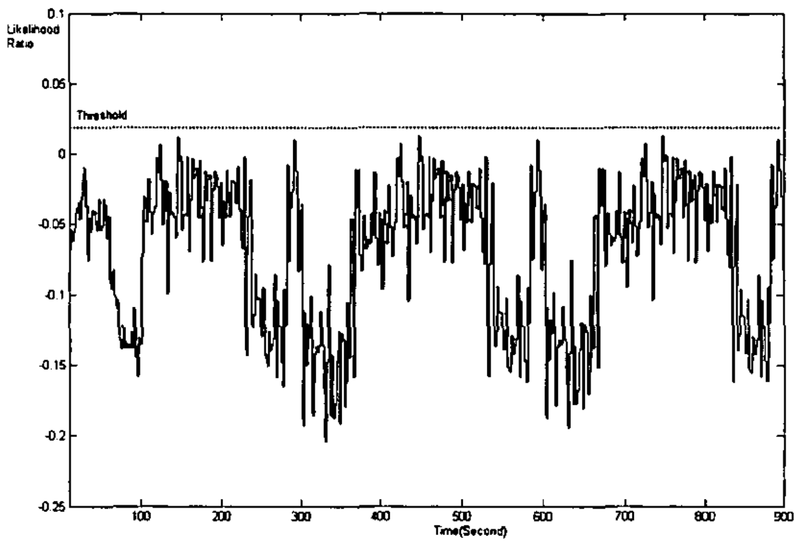


Figure 5.21. The mean value of the likelihood ratio test (for the control signal for water temperature)

Table 5.2 System and reference models in MIMO case

	F	G	H	D
Reference Model	$\begin{bmatrix} -0.6667 & 0 \\ 0 & -0.5001 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0.6667 & 0 \\ 0 & 1.25 \end{bmatrix}$	$\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$
Faulty Model	$\begin{bmatrix} 2.667 & 12.4 \\ 1.95 & 1.5201 \end{bmatrix}$	$\begin{bmatrix} 1.778 & -12.4 \\ -0.45 & 1.55 \end{bmatrix}$	$\begin{bmatrix} 1.267 & -1.45 \\ -0.45 & 1.25 \end{bmatrix}$	$\begin{bmatrix} 0.3 & 0.2 \\ -0.4 & -0.6 \end{bmatrix}$

Table 5.3 S_{ij} Matrices

S_{11}	S_{12}	S_{21}	S_{22}
$\begin{bmatrix} 0.4346 & 0.4480 \\ -0.3360 & -0.54 \end{bmatrix}$	$\begin{bmatrix} -0.1676 & -0.1075 \\ -0.2861 & -0.1872 \end{bmatrix}$	$\begin{bmatrix} -0.996 & -3.85 \\ -0.3619 & -0.9775 \end{bmatrix}$	$\begin{bmatrix} -0.0505 & -0.2402 \\ -0.4366 & -0.3159 \end{bmatrix}$

5.5.3.2 Compensation

To restore the degraded performance, a compensator is designed and inserted into the faulty control loop. The reference model is given as shown in Table 5.2. The identified open-loop faulty model is given in Table 5.2. According to the compensator model calculation equation, matrices S_{ij} are calculated as shown in Table 5.3. Implemented the compensator into the controller failure scenario, results as shown in Figures 5.16 and 5.17 clearly have indicated the outputs of original faulty control loop with the added compensator are able to track the setpoint satisfactorily and the performance index is back to 0.82 as shown in Figure 5.18.

5.6 Conclusions

The subject of process control loop performance monitoring and maintenance is attracting attention from companies wishing to gain a competitive edge. Traditional methods addressing this field often focus on sensor/actuator failure. In this chapter, we develop an online maintenance scheme for controller failure. The distinguishing feature of the scheme is that it extends the range of traditional FTCs and is able to monitor and maintain the controller failures. To maintain the faulty controller, two approaches are proposed to design a compensator, which is inserted in to the post-fault control loop to restore the degraded control performance. One is based on a fault transfer method, which transfers any fault happening in the control loop into the plant; a compensator is designed according to the assumed faulty plant model, and then is transferred into the controller failure scenario. The other is based on the model following technique, where the compensator is designed to make the new post fault control loop to exactly follow the desired reference model. Simulation and experimental results demonstrate their validity and effectiveness.

Chapter 6. Data Process and Transfer over the Internet for Real Time Applications

6.1 Introduction

The rest of thesis will extend the techniques of controller software performance monitoring, controller software failure detection and maintenance proposed in above chapters to the remote environment. The emerging Internet technologies offer unprecedented interconnection capabilities and ways of distributing collaborative work, and have a number of benefits to the high-level remote maintenance operations, including:

(1) Operators or managers can remotely monitor and adjust the plants and controllers in order to adapt to the quick change of running situations;

(2) Skilled plant managers situated in geographically diverse locations can collaborate to perform control tasks;

(3) Control system design and maintenance centres can be created which operate on the web and provide central supports to all authorised industrial processes. The controller software providers can remotely design, tune and maintain their geographical distributed software products. Control systems can be remotely designed, tested, and then installed in the plant floor;

(4) Business can relocate the physical location of plant management staff easily in response to business needs; plant engineers can demand better and faster ways of retrieving data and reacting to plant fluctuations anywhere around the world at any time.

The design methodologies used for the local computer-based maintenance systems are not appropriate for remote maintenance systems, some issues are arisen for remote maintenance systems and have to be considered when they are implemented into actual applications such as the heavy data process and transfer over the Internet. In the open Internet environment, the bulky data transfer is significantly complicated by heterogeneity and limited traffic resources of the Internet: a number of clients are simultaneously trying to connect with a same server; multiple data sources are exchanged and interacted over the Internet via various platforms that require the data transfer format has to be acceptable by the heterogeneous platforms; heavy, even gigabit amount of data such as graphics, desktop videos, and images upload into the Internet while communication bandwidth are limited due to the publicity of the Internet.

Little work has so far been done in the Internet environment. Methodologies used for the local high computing applications are not appropriate for Internet-based applications, as they do not consider the Internet environment features, such as diverse users and platforms. This chapter aims to propose an efficient data process and transfer mechanism over the Internet for Internet-based applications. A novel data model based on Extensible Markup Language (XML) and Hierarchical Data Format (HDF) is presented, which provides a generic solution to efficiently handle the heavy scientific data. A data priority policy is adopted in Java Remote Method Invocation (RMI) to deliver the data objects in sequences. A remote monitoring system for an industrial catalytic reactor is used as a case study to illustrate the proposed data transfer mechanism. The rest of the chapter is organized as follows. The enabling technologies for the heavy data process are described in Section 6.2. The real time data process is given in Section 6.3 followed by the discussion of HDF wrapped by XML in Section 6.4. A data object transfer mechansim is presented in Section 6.5. A case study to illustrate the proposed data transfer scheme is given in Section 6.6. Section 6.7 gives the conclusions.

6.2 Enabling Technologies

JDOM

Java Document Object Model (JDOM) is an open source library for Java-optimized XML data manipulations. As the name indicates, JDOM is Java optimized. It behaves like Java, using Java collections, and providing a low-cost entry point for using XML. JDOM users don't need to have tremendous expertise in XML to be productive and get their jobs done.

JDOM attempts to incorporate the best of Document Object Model (DOM) and Simple API for XML (SAX). It's a lightweight API designed to perform quickly in a small-memory footprint. JDOM also provides a full document view with random access and does not require the entire document to be in memory. The API allows for future flyweight implementations that load information only when needed. Additionally, JDOM fills the gaps where SAX and DOM leave off. All three- JDOM, SAX and DOM - are capable of manipulating XML documents; however, JDOM eliminates some of the weaknesses of the other two APIs. JDOM interoperates effectively with existing APIs, such as SAX and DOM; however, JDOM is not an abstraction layer or enhancement of those standards, JDOM provides a solid yet lightweight means of reading and writing XML data.

JNI

Java Native Interface (JNI) is a standard programming interface for writing Java native methods and embedding the Java virtual machine into native applications. The primary goal is binary compatibility of native method libraries across all Java virtual machine implementations on a given platform.

The JNI allows Java codes that run within a Java Virtual Machine (VM) to cooperate with applications and libraries written in other languages, such as C, C++, and assembly. In addition, the Invocation API allows embedding the Java Virtual Machine (JVM) into the native applications. Programmers use the JNI to write native methods to handle those situations when an application cannot be written entirely in the Java programming language.

6.3 Real Time Data Process

6.3.1 Real Time Web-Based Applications

Real-time web-based systems span a broad spectrum of complexity from web sites to highly sophisticated, complex, and distributed systems, such as e-Business and e-Automation. Basically, real-time web-based applications can be categorized into interactive applications and streamed applications. In interactive applications, system components interact and exchange over the Internet. Typical applications include Internet-based control systems, on-line games and shared virtual worlds or Internet-based distributed simulations. Streamed applications are essentially one way flows of information pushed by the server. Typical applications include traffic and stock prices information website and remote monitoring systems.

Real-time web-based applications are subject to the time and reliability performance requirements. Basically, discrete interactions have more stringent reliability requirements than continuous interactions; techniques supporting reliability play a more critical part in the implementation of discrete interactions than these of continuous interactions. A discrete interaction may consist of only a single packet; operations of the discrete based application are based on the retrieved data guiding the application to make decisions for the next tasks. The loss of that packet can have a major impact on the application's behaviour, which will make the system misjudge the running state and take inappropriate actions. In continuous interactions, however, packets are effectively and continuously passed through to the device; the loss information in a packet can be roughly estimated and compensated by the related packets.

6.3.2 Real Time Scientific Data in Web-based Applications

Real time scientific data applied in the web-based systems are communicated among the system components over the Internet. In the actual applications, the data often present the following features:

- Timeliness: Real time data are time sensitive and have strict time restrictions. The correctness of the system depends not only on the logical correctness of the computations performed but also on time factors; i.e. a continuous stream of media data bits is with strict time dependencies among those bits; late data in a stream will result in a media information interruption while very early data can cause buffer overflow as well.
- Heterogeneity and complexity: Scientific data measure physical phenomena and extend a large range of data types. The sampling scientific data can be a single binary number or a series of numbers describing physical phenomena like atomic weights and voltage, or the text and image description of the physical devices. The storage record is not one value at a time, but blocks of many thousands - corresponding to different times, different positions, different measuring points

and different variables.

- Server push: The data exchange in both applications is pushed by the server, which is responsible for setting up the communication channel, initiating transmission of data and providing various data access services to the remote clients.

Managing and using scientific data in the Internet environment present various challenges, including:

- Handling huge volumes of data: Considering the size of data amount, it is important to use efficient algorithms and mechanisms to store and discover the data as early as possible, such as data mining and other automated learning techniques. These techniques are increasingly important for the discovery of relevant data out of large volumes of data, which presents significant challenges for data storage and access.

- High data retrieval performance: Web-based applications need to offer high quality, cost-effective data entry services suited for high volume data computing such as data extraction from database, accessing and mining the data from distributed data sources with minimum operation time. Computer I/O is always slow and fast storage is always at a premium, so it is important to use efficient algorithms and mechanisms for data retrieval and storage. There is no one optimal storage or transfer method, different uses and access patterns place substantially different requirements on I/O systems. For this reason, it is important to provide options to control the details of data layout and storage.

- Across heterogeneous systems: Operation systems are not identical, and users are heterogeneous. The storage data context may be incompatible with the user system, and need reformatting, translation, or other modifications in order to create or recreate the intended meaning for the user's understanding (McGrath, 2003). While some data are stored and retrieved within the same context, other data are shared or disbursed to many contexts. This may require reformatting, translation, or other modifications of the data in order to create or recreate the intended meaning at the time of use.

6.3.3 XML and HDF

XML is a mark-up language for documents containing structured information. XML provides a facility to define tags and the structural relationships between them. Each "tag" in XML represents a data field, with the data held within this tag. XML forms a standard for storing structured content in text files and for data exchange via the Internet. HDF is an emerging data library and file format developed by the National Centre for Supercomputing Applications (NCSA) to meet the requirements of scientists and engineers in actual data storage and management. HDF is self-describing and extendible, scientific data wrapped by HDF can be manipulated in a discipline-independent fashion and various types of data can be freely transferred among different machines. XML and HDF can be used for the essential storage of scientific data. Both can store objects, describe the data and specify the layout of bits. There are several important features and differences in the capabilities of XML and HDF formats.

- XML is a universal standard with widely available software and enjoys massive commercial uses and supports. The universality of XML is overwhelming, trumping all concerns of efficiency. For example, XML is clearly well suited for data catalogues, various kinds of data and process description and heterogeneous systems with multiple users and multiple purposes; the portability of XML is essential for sharing data across space (geographic distribution), time (archiving), and conceptual domains (different users, communities, and uses). For HDF, the application range is limited while it gets full supports from the National Center for Supercomputing Applications (NCSA) and begins to put into some actual research applications (McGrath, 2003).
- XML is closely integrated with other software. XML is the default data process standard for the web applications and is compatible with many standard data tools and databases. The availability of generic software makes it easier to share the complex details of scientific data. An XML document can be used to deliver a description of the data without the data itself, as in a data catalogue or a "virtual

dataset". This enables users and communities to share data more easily, and to create customized views of data from many sources without replicating or reformatting. HDF, on the other hand, can tightly couple with some software; but the access is more complicated and the number of support software is limited. Additionally, HDF has specific requirements on the operation system and computer computing abilities.

- HDF is more suitable for heavy data storage than XML. The thousands of values could be stored one at a time in an XML document, but this is not practical – the heavy data wrapped by XML will complicate the XML data description structures and many large tags are needed, it would be equivalent to storing an image with each pixel as a separate XML document element (Leal et al., 2002). HDF, however, is created to address the data management needs of scientists and engineers. HDF employs various data extensibilities and chunking strategies to facilitate the heavy data access, management and storage (HDF, 2002); interfaces contained in HDF can be used to store and retrieve compressed or uncompressed N-dimensional scientific datasets together with the data attributes, such as labels, units, formats, and scales for all dimensions.

- The binary format in HDF is close to native computer memory. Binary formats are particularly suited to numeric computations, especially using dense matrices of numbers. As a result, data stored by HDF data models can be efficiently memorized. XML, on the contrast, can not be stored as native numbers; data processed by XML must be converted to/from Unicode and/or XML tags into a memory image (e.g., an array of numbers); it will cost a lot of time and computing resources.

- HDF is designed to have powerful algorithms to compress and write/read data. HDF provides highly customized data filtering and scatter-gather algorithms, which support raw data transformation during the I/O operations. Additionally, HDF data models and formats can be freely exchanged over the different operation systems and stored with most word types and depths. These features are essential for complex data storages. XML, however, can only be written/read as a continuous stream of objects encoded in Unicode; under the stream I/O, it is very difficult to manage very large objects or to navigate a large file (McGrath, 2003).

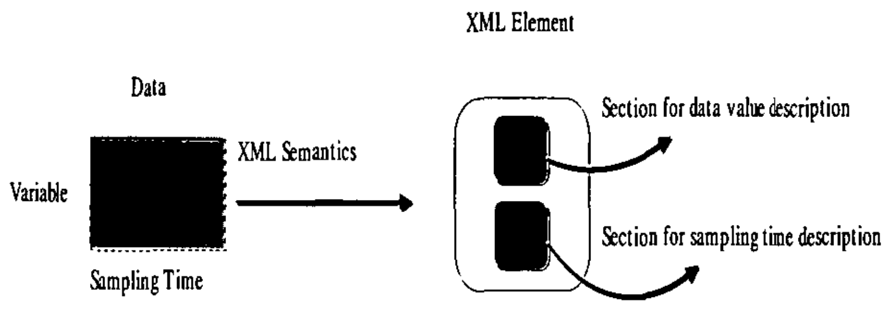

XML Element
Data
Variable
Sampling Time
XML Semantics
Section for data value description
Section for sampling time description

Figure 6.1. XML data structure

6.3.4 Heterogeneous Scientific Data Processed by XML and HDF

To simplify the heterogeneous scientific data process, the heterogeneous data are classified into light data and heavy data in this work based on the amount of the data and their physical meanings. The light data are considered to be small quantities of scientific data. The heavy data are considered to be large amounts of data. As a rule of thumb, if the data have "more than that 300" values, they are heavy data.

Light data are wrapped by XML in our study. XML is a natural candidate for light data description and binding. A XML document is composed of a hierarchy of elements and can be nested to any level depth. Elements are the logical units of information in an XML document. A real time data unit is 2-demensional, which consists of the sampling time and the variable values. Description of the real time data with XML is achieved by XML elements. Each data point corresponds to an element. The element has two sections, which define the value and the sampling time. Information between the two tags indicates the data content and sampling time respectively. The structure of a real time data stored in the XML is shown in Figure 6.1.

The light data wrapped by the XML semantics can be manipulated in Java. There are three popular methods to manipulate the XML document with Java, JDOM, DOM and SAX. The tree DOM/JDOM model is more suitable for the data storage and manipulating the XML document. The reason is that the data with multi-level attributes are hardly arranged in a single XML element under the SAX based method. The generated classes under the DOM/JDOM based method are lightweight and data binding applications use a minimum amount of memory and run efficiently.

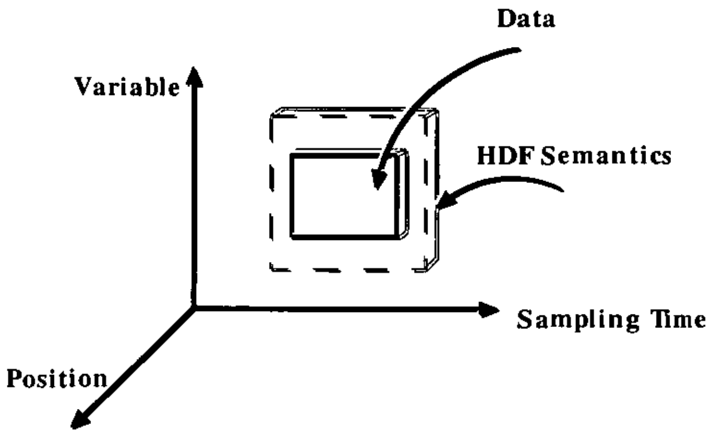


Figure 6.2. HDF data storage structure (ScadaOnWeb, 2000)

Heavy data are stored in the HDF format in our study through the HDF datasets and groups. Data stored in HDF are organized in a hierarchical structure with two primary structures: groups and datasets. A grouping structure contains instances of zero or more groups or datasets. The metadata contains group name and a list of group attributes. A dataset in HDF, such as a multidimensional array of numbers, has additional metadata logically associated with it, which describes attributes of the dataset such as the rank of the array, number of elements in each dimension, etc. Several datasets can form a group. HDF groups and the links between the groups can be designed to reflect the nature and/or intended usage of the stored data. A real time data in a HDF dataset shown in Figure 6.2 is a 3-dimension unit, which consists of the sampling time, the position of the variable in the dataset and the variable value. The description of the variables includes the properties of the data point, and the associated explanation of the properties.

6.4 Wrapping HDF by XML

6.4.1 HDF Objects Access from Java Applications

The HDF library packages developed by National Centre for Supercomputing

Applications (NCSA) are not a pure Java implementation; they include some C libraries; therefore, access to the HDF library from a Java application needs an appropriate wrapper to load the data and use the native HDF library locally. Figure 6.3 shows the involved layers of Java accessing the HDF library: the Java program, native methods (which have a Java stub and a C stub), a Java Native Interface (JNI) and the HDF libraries. The key part of the structure is JNI, which defines a standard for invoking C and C++ subroutines as Java methods. The Java programs are on top of the native method and JNI in order to implement native access to the HDF libraries. The HDF libraries have hundreds of entry points, with various types of arguments, including all numeric types, strings, and pointers to arrays. When the Java programs make a request by initialising a Java object, the object will be passed as an argument to a native method and the HDF libraries provide the means to pass objects from Java to C.

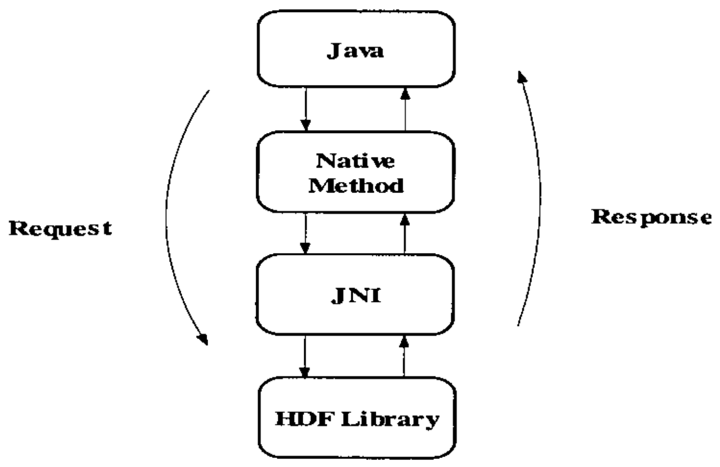


Figure 6.3. Accessing HDF in the Java environment

The HDF JNI layer is basically a series of calls that:

1. translate each input parameter from Java objects to C data structures;
2. call HDF;
3. translate the return values from C data structures into Java objects.

The HDF libraries define a large number of constants to be used as parameters to the API. The HDF JNI layer exports a set of Java constants that are mapped to the required C values. The Java program calls the native method to get the correct value for the C constants.

6.4.2 Wrapping HDF with XML

XML provides a standard way to store and structure the specific application data and is pervasive on the Web. Wrapping HDF with XML can fully incorporate the XML features and extend HDF from the local environment to the web environment, and therefore make the HDF format being acceptable to web-based applications.

Wrapping HDF data with XML consists of two stages: structure mapping and data mapping. The structure mapping is to make that the relationship among data in HDF can be fully expressed in XML. Data mapping is to transfer the HDF data content, data attributes and data type into the XML format.

6.4.2.1 Structure Mapping

XML descriptions are in a tree structure with one root and the objects nested in their parents. JDOM provides Java specific XML functionality. With the support of JDOM, the HDF group can be mapped to a JDOM document object, and the dataset is mapped to a JDOM element object.

Considering two cases for representing HDF with XML:

Case 1: HDF with a tree structure

For a simple HDF file, it has a tree structure and is implemented as a directed graph; the HDF file can be fully mapped to XML in this case and the HDF group is treated as a single JDOM Document object. All the objects within the HDF group are treated as elements of the structure. Attributes of groups can be represented naturally as the JDOM Document attributes.

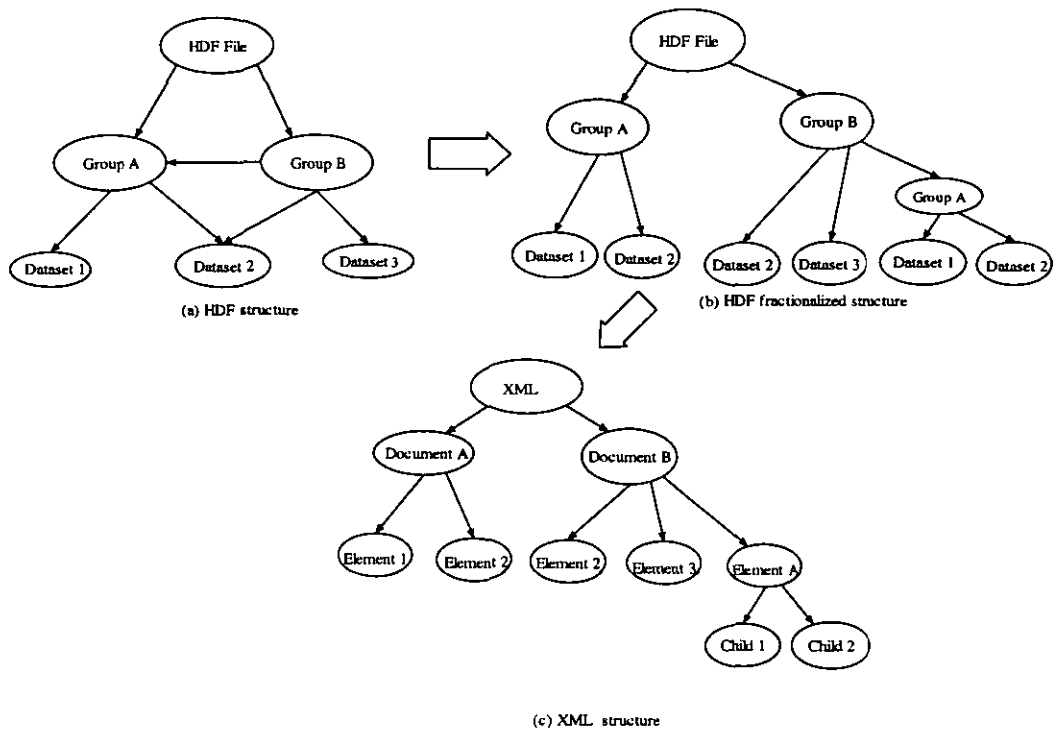


Figure 6.4. Converting HDF structure into XML structure

Case 2: HDF with a complicated structure

For some complicated HDF files, they have elaborate grouping structures where some datasets are shared by other groups and have more than one parent. The relationship between the groups is not parallel but parental. Therefore, their structure doesn't match the XML tree structure because there might be more than one parent for a dataset. To match the structure of XML, the structure of HDF with shared datasets needs to be fractionalized into certain small pieces in order to simplify into a tree structure. An example is given in Figure 6.4. Dataset 2 is shared by Group A and Group B, and Group A is a member of Group B. To simplify this complicated structure, the structure in Figure 6.4 (a) is decoupled into Figure 6.4 (b). Mapping the decoupled structure into a XML structure will lead to a XML tree structure shown in Figure 6.4 (c). Group A and Group B are mapped as JDOM Document A and Document B, which are at the first level in the hierarchical structure. The datasets are mapped as JDOM elements. Due to the child-parent relationship between Group A and Group B, Group A is mapped as Element A, a child of Document B. Dataset 1 and Dataset 2 are mapped as Child 1 and Child 2. The contents of Element 1 and

Element 2 in Document A are identical with Child 1 and Child 2 in Element A respectively in JDOM.

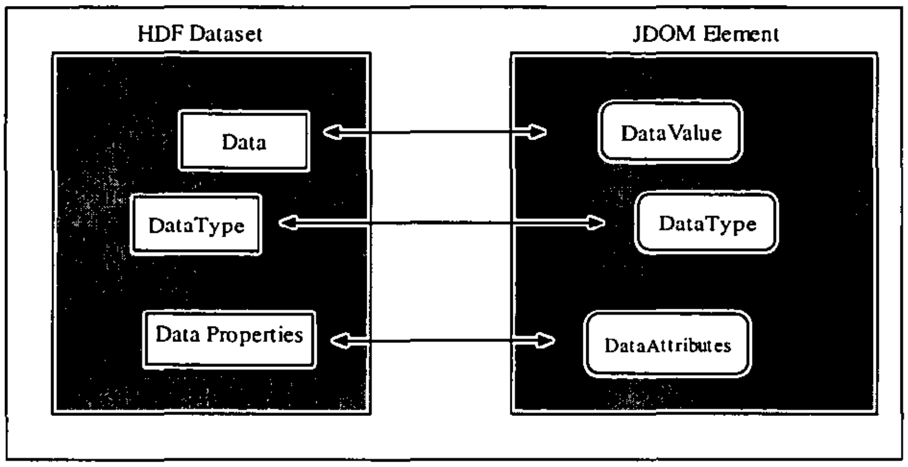


Figure 6.5. HDF data mapping into JDOM

6.4.2.2 Data Mapping

HDF dataset is a basic element in a HDF data file; several datasets form a group and one or more groups form a HDF file. A dataset is a multidimensional array of data elements, together with supporting metadata. In the implementation of a dataset, an application should specify a dataset name, a data type, a data space of the array and the dataset creation information. The data space describes the dimensionality of the data array. The data type is a collection of data type properties.

Data mapping between HDF and XML is achieved through JDOM elements, which have three sections: "DataValue", "DataType" and "DataAttributes". All data in "DataValue" are expressed as a string, which will be converted into the original data type as shown in Figure 6.5. Both JDOM and HDF can be accessed and communicated each other in the Java environment; therefore, the data in HDF can be transported into XML by Java programs to read data from a HDF dataset object and write the data into a JDOM element. "DataType" is used to save the data type. The dataset name, data space and other data attributes of the dataset are categorized as "Data Properties" and stored in "DataAttributes" of the JDOM element. The data in the HDF dataset are multidimensional. In the JDOM element, the multidimensional data of the HDF dataset are treated as a group of arrays, and "DataValue" is designed to describe these multi-dimensional data.

JDOM provides Java programmers with a simple way to write Java code for reading from, writing to and manipulating XML documents. These JDOM Document/Element objects are used in the same way as other Java objects, which can be transferred over the Internet in the Java environment through the RMI infrastructure described in the following section.

6.5 Data Object Transfer Mechansim

6.5.1 RMI based Data Transfer Structure

Figure 6.6 depicts the data transfer mechanism structure proposed in this study, which is built on the RMI infrastructure. The structure is composed of four basic elements: a RMI server, a RMI Object Registry, a data processor and remote clients. The RMI server provides the back-end communication services. The Object Registry serves the role of object management and naming service for the remote objects. The RMI transfer system can pass data objects as arguments and return values, and use the full power of object-oriented technology in the distributed computing. The data processor is to collect and process the data from the real sites. According to the heterogeneous feature, the transfer data are firstly categorized into the light and heavy data according to their physical meanings and actual requirements. The heavy data are organized in the HDF form. Both categories of the data are wrapped with XML and processed as JDOM Document/Element objects in the Java environment.

It needs to generate a client stub and a server skeleton for the RMI communication. The skeleton is a server-side entity that contains a method to dispatch calls to an actual remote object implementation. The stub is a proxy for a remote object in the client site to forward method invocations on the remote objects. The stub/skeleton will be bound onto the client/server through a binding action before the objects begin to transfer in the RMI infrastructure. In the RMI infrastructure, the Object Registry works together with the RMI server to achieve the object transfer. The RMI server creates remote objects and binds each instance of the remote objects with a name in the Object Registry. When a client invokes a method of a remote object, it first looks up the remote object within the Registry by the remote object name. If the remote object exists, the Registry will return a reference of the remote object to the client, which is used to make method invocations on the remote object. As long as there is a reference to the remote object, the object will not be shut down and keep being reachable from the remote clients.

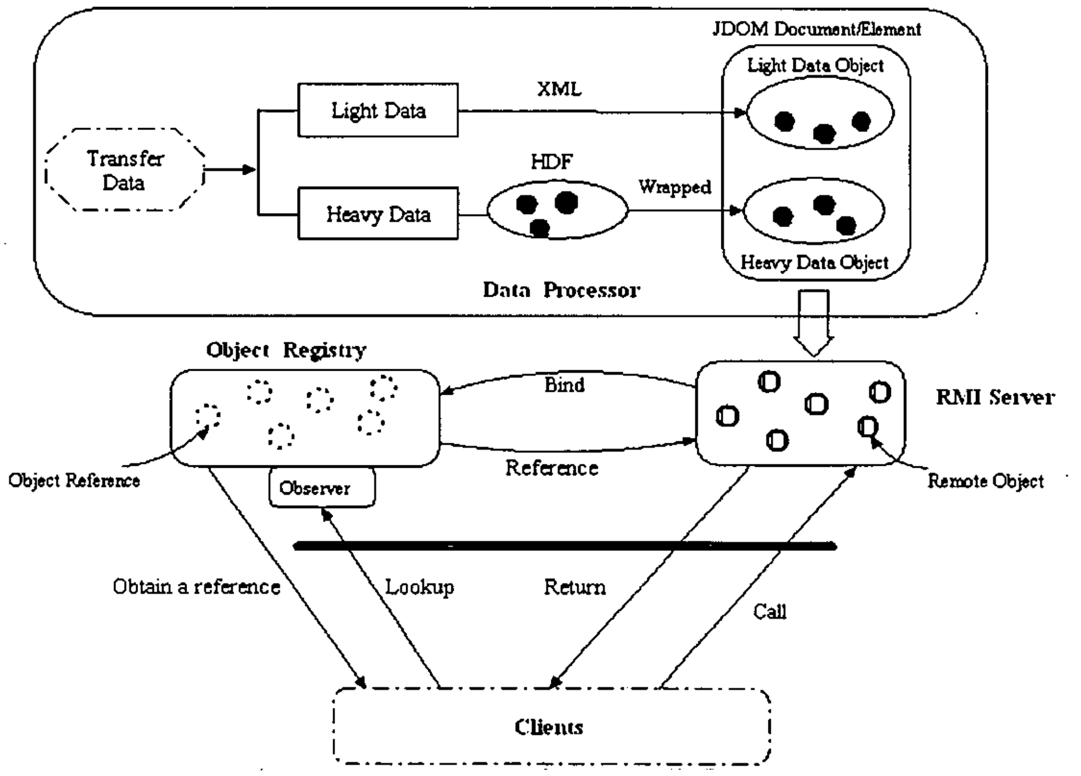


Figure 6.6. Data transfer structure based on RMI

The sequence of data transfer in the RMI infrastructure is summarized as the following five stages:

Stage 1: Categorizing the heterogeneous data into the light and heavy data according to its feature.

Stage 2: Organizing the heavy data in the HDF format and wrapping both types of data with XML

Stage 3: Generating the stub and skeleton of remote objects.

Stage 4: Starting up the Object Registry and binding the remote objects in the Object Registry.

Stage 5: Transferring data by involving the RMI method every sampling interval with the data object as an argument.

6.5.2 Data Object Priority

All the light and heavy data are stored in the XML documents, which are treated as data objects in the Java enviromement. The data objects describe both the data values and their structures. Assigning different priorities to the data objects can transfer some data objects faster than others in order to meet different requirements from the clients. The vital data are assigned with the top priority and the ordinary data with a lower priority. Data with a higher priority are sent out before the ones with lower priority. The priority levels are remained in the server and maintained once any new data transfer request arrives or a transmission is completed. The clients can terminate the data transfer of some data objects if necessary.

The data priority submission is achieved through an interface, PrioritySubmission, in the RMI infrastructure. By extending the interface *java.rmi.Remote*, the PrioritySubmission interface makes itself being available from any virtual machine. The signature of the interface PrioritySubmission is as follows:

```
public interface PrioritySubmission  extends Remote {
   public void DataPriority (String [] sequence) throws RemoteException;
   public void EmergencyStop (boolean [] stop) throws RemoteException;
............
}
```

The DataPriority () method and the EmergencyStop () method are supported by both the server and the client applications. The DataPriority() method is for the clients to submit the defined data priority to the server. The EmergencyStop () method is to notify the server to suspend the transfer some data objects. These methods are defined as being capable of throwing a Java.rmi.RemoteException. The data object name with a higher priority is placed in the beginning of the priority sequence, and the less important ones at the end of the sequence. The stop sequence is a Boolean array and defines a logic value True/False for the corresponding data. "False" indicates to notify the server to stop the data transfer, "True" to continue the data transfer.

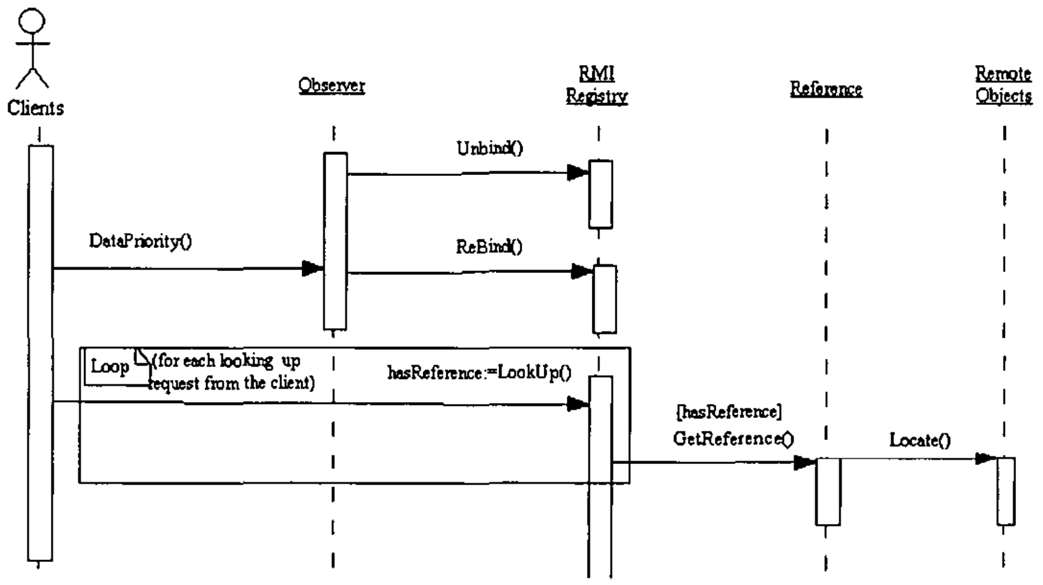
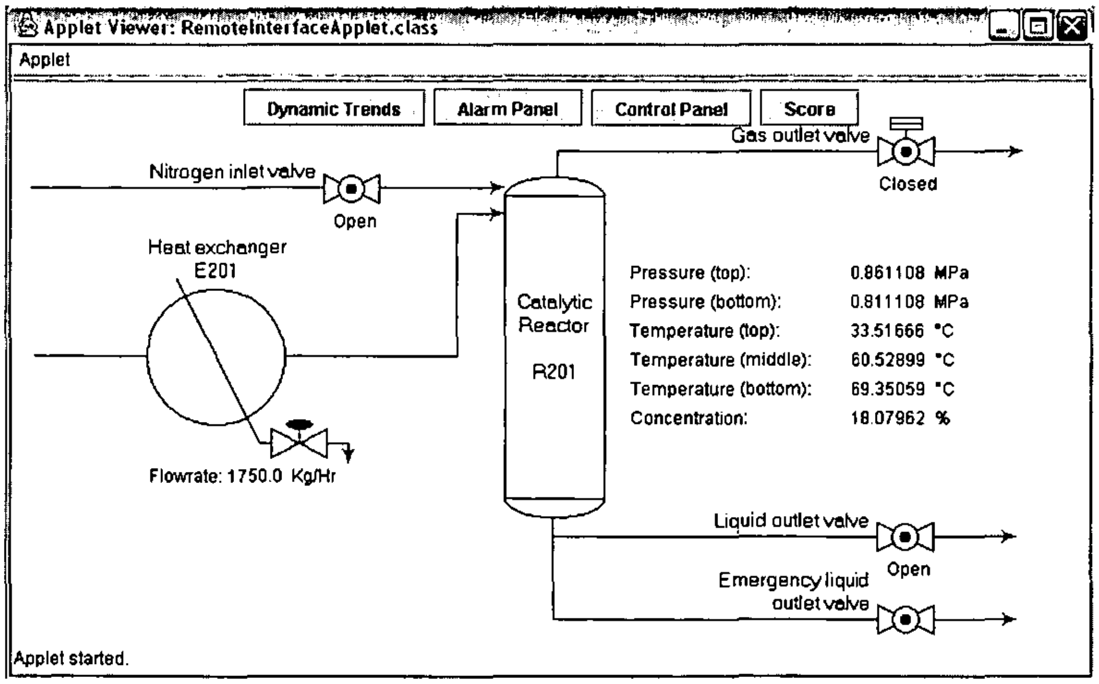


Figure 6.7. Procedure of looking up the remote objects

The data transfer with various priorities is realized through the proposed observer in the server side shown in Figure 6.6. The proposed observer acts as an object manger. When the clients submit an invocation to the remote objects, in the server side there will be an order to enter the Registry to look up the remote objects and get the references. The observer is designed to make the order being identical with the data priority sequence submitted by the clients. In details, the observer unbinds all the remote objects in the Registry before the remote objects are looked up. When the invocation is requested from the clients, the observer will rebind the remote objects step by step according to the data priority sequence. The higher priority object is first rebound and the lower priority object is rebound afterwards. Only the rebound remote objects can get the reference from the Registry and be located, and then be accessed by the clients; the other objects, which fail to get the reference, will keep submitting a request to the Registry until obtaining a reference. Therefore, by controlling the access of the reference, the remote object transfer is governed under the data priority levels. In the case of the suspension requests being received from the clients, the observer will be notified to refuse rebinding the remote objects, which are requested to terminate the data transfer so that the references of the remote objects cannot be obtained and the remote objects cannot be accessible by the clients.

The procedure of looking up the remote objects is shown in Figure 6.7. The looking up requirements from the client form a loop; each request repeatedly makes the looking up appeal until getting the reference. The LookUp () method is responsible for looking up the remote object's reference. The value of the Boolean variable "hasReference" is set by the return value of the LookUp() method. When "hasReference" is true, it indicates that the LookUp () method finds the reference and the method, GetReference(), will be invocated to catch the reference and locate the remote object via the Locate() method.



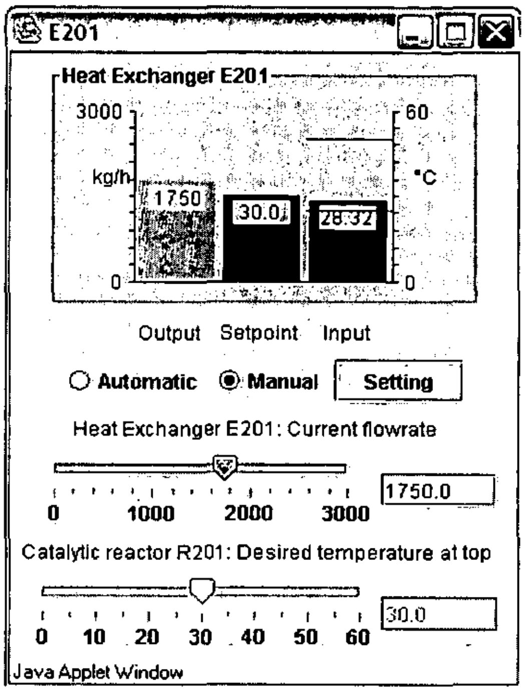
Pressure (top): 0.861108 MPa
Pressure (bottom): 0.811108 MPa
Temperature (top): 33.51666 °C
Temperature (middle): 60.52899 °C
Temperature (bottom): 69.35059 °C
Concentration: 18.07962 %

Applet started.

Figure 6.8. A local reactor process (Yang and Alty, 2002b)

6.6 Case Study

In order to demonstrate the proposed data transfer mechanism, a remote monitoring system for a local reactor process simulator has been chosen as a test-bed. The details of implementation are discussed in this section.

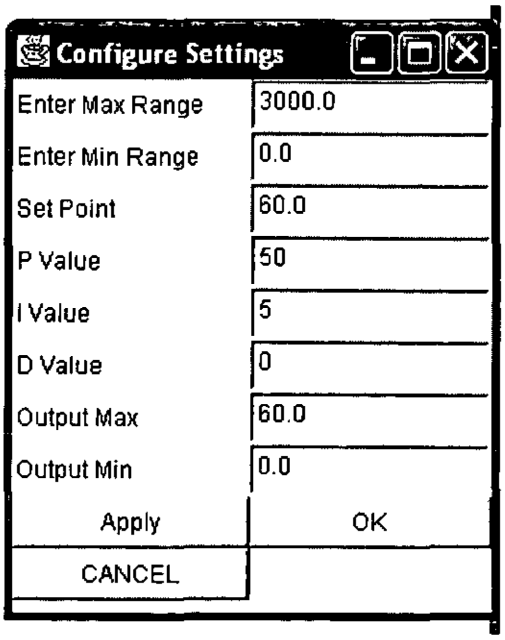


○ Automatic [✓ ◉ Manual]

Heat Exchanger E201: Current flowrate: 1750.0

Catalytic reactor R201: Desired temperature at top: 30.0

Figure 6.9. PID controller (Yang and Alty, 2002b)



Enter Max Range	3000.0
Enter Min Range	0.0
Set Point	60.0
P Value	50
I Value	5
D Value	0
Output Max	60.0
Output Min	0.0

Figure 6.10. PID controller configure setting (Yang and Alty, 2002b)

6.6.1 System Description

The local reactor process simulator (Yang and Alty, 2002b) shown in Figure.6.8 consists of a heat exchanger E201, a catalytic reactor R201, and four-hand valves for Nitrogen inlet, liquid outlet, gas outlet and emergency liquid outlet. The inlet temperature of the reactor is controlled by the PID controller shown in Figure 6.9 through manipulating the hot stream flow rate of the heat exchanger E201. The bar charts at the top show the current input, output and setpoint values of the controller. The controller panel is also used to adjust the hot stream flow rate of E201 and the desired top temperature of the catalytic reactor R201. The PID controller panel has two operating modes: manual and automatic modes. The parameters of the PID controller can be tuned in the manual mode shown in Figure 6.10, which allows the users to tune the P, I and D parameters and change the output range for the controller and the setpoint. Figure 6.11 shows the structure of the remote monitoring system, which consists of three parts: a reactor control process simulator, a RMI data transfer system, and several remote clients.

The local control system is connected with the data processor to collect the on-line information of the reactor process. The process information is analysed and processed in the XML and HDF data formats in the Java environment. The RMI system provides communication services to establish connections between the remote clients and the local control system. The remote clients can simultaneously monitor the local reactor process, define the data transfer priority level, and submit a request to the server. The acquisition data are processed as JDOM Document/Element objects. Once the RMI data transfer system accepts any request from the clients, the data objects will be transferred every sampling interval according to the defined data priority sequence.

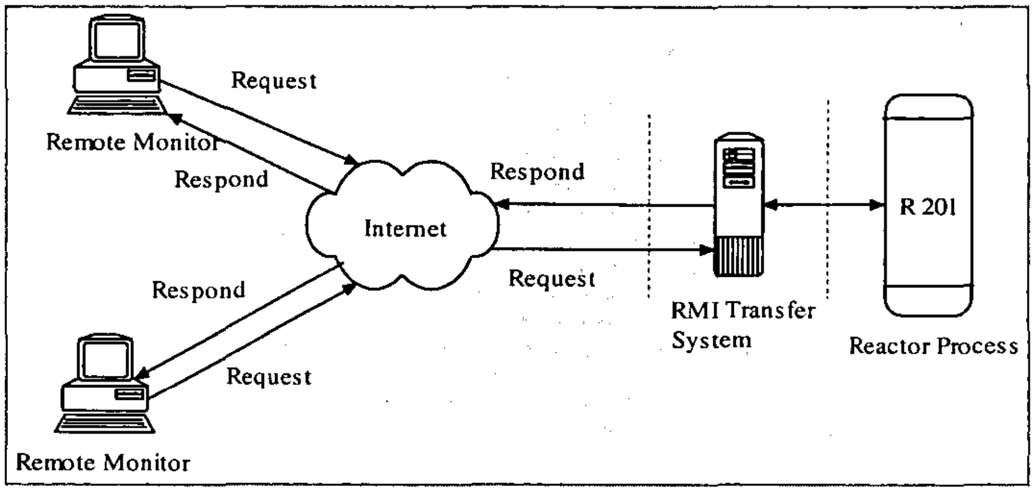


Figure 6.11. Architecture of remote monitoring system

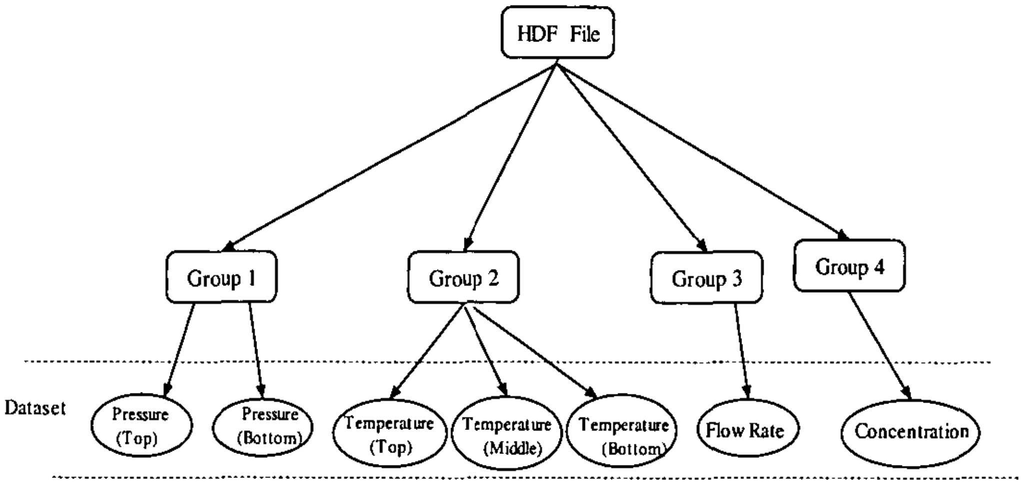
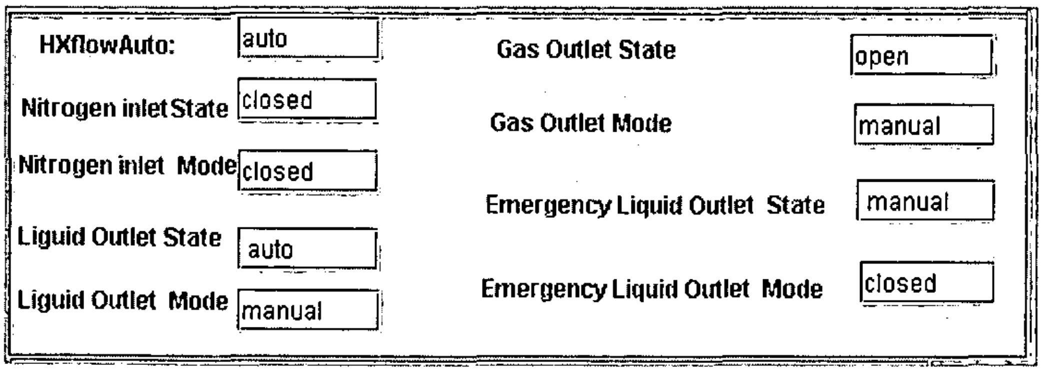
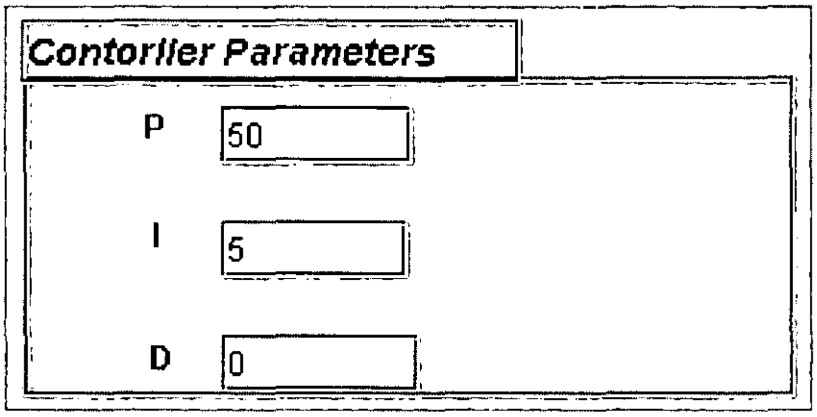


Figure 6.12. HDF structure



HXflowAuto:	auto	Gas Outlet State	open
Nitrogen inletState	closed	Gas Outlet Mode	manual
Nitrogen inlet Mode	closed	Emergency Liquid Outlet State	manual
Liquid Outlet State	auto	Emergency Liquid Outlet Mode	closed
Liquid Outlet Mode	manual		

Figure 6.13. Valve work status



Contorller Parameters

P 50

I 5

D 0

Figure 6.14. PID parameters

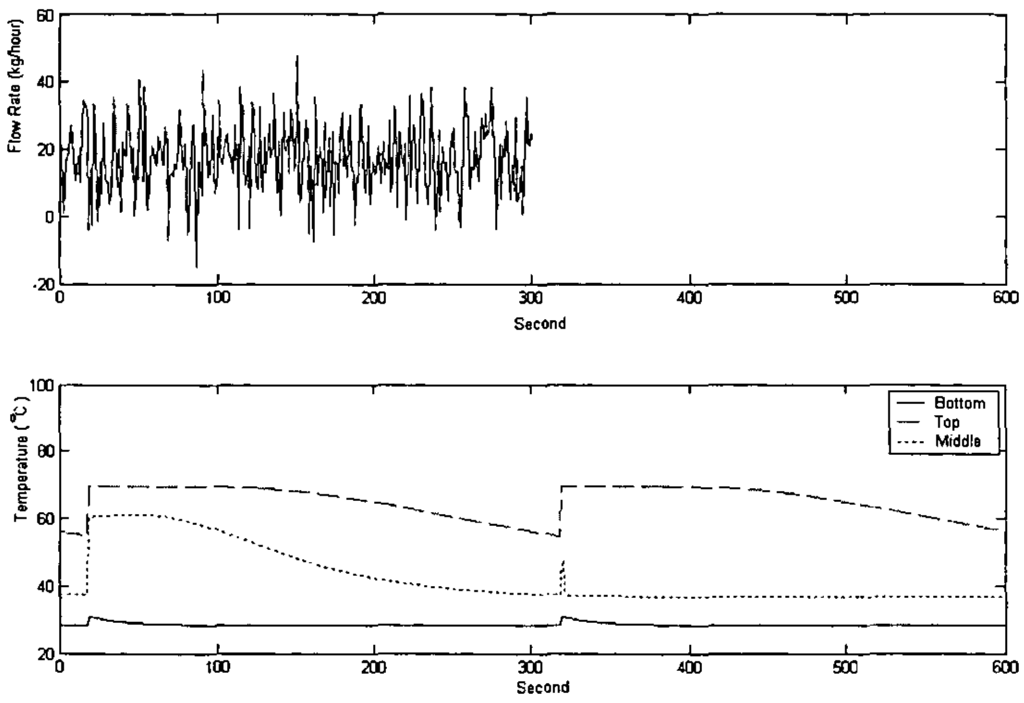


Figure 6.15. Output flow rate and temperature

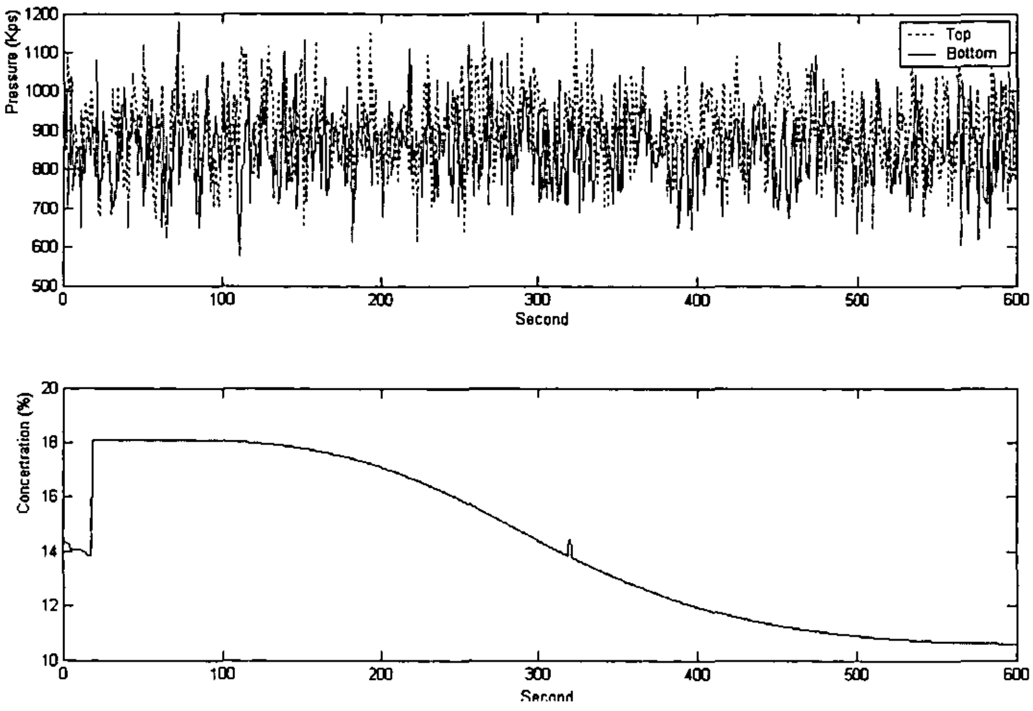


Figure 6.16. Output pressure and concentration

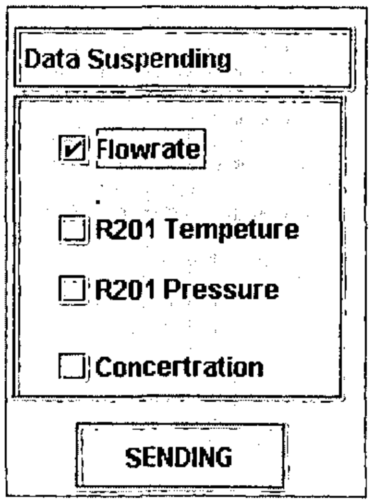


Figure 6.17. Data priority setting panel

6.6.2 Data Priority Establishment

All process variables are usually categorized into static data and dynamic data. The static data include the parameters of the PID controller and the state of the four values: nitrogen inlet, liquid outlet, gas outlet and emergency liquid outlet valves. Each valve has two states: the work state (close/open) and the work mode (manual/automatic). The static data are categorized as the light data. They share the following features: (1) limited data amount and (2) slow change. The dynamic data include all the process variables including the reactor temperature, pressure, concentration, and flow rate. The dynamic data are categorized as heavy data and collected every sampling interval in the wrapped HDF format. The code illustrates the light data and heavy data wrapped by XML and HDF respectively:

```
public Element LightDataWrapper() {// wrap the light data
 ..... .....
Element LightData = new Element("LightData ");
Document Data= new Document(LightData);
Element sectorElement= new Element("Sector");
LightData.addContent(sectorElement);
Element Parameter =new Element("Parameter");
sectorElement.addContent(Parameter);        // construct the XML file  structure

/// store PID controller parameters
Parameter.addContent(new Element("PID_P").setText(Integer.toString(get_PID_P()));
Parameter..addContent(new Element("Time").setText(this.getCurrentDateTime()));
   ......
//   write into a XML file
 XMLOutputter outputter = new XMLOutputter("PID.xml ", true);
   outputter.output(Data, System.out); }
```

```
public void  HeavyDataWrapper() {       // wrap the heavy data
............// construct HDF structure
Group root =
   (Group)((javax.swing.tree.DefaultMutableTreeNode)testFile.getRootNode()).getUserObject();
Group g1 = testFile.createGroup("Group1", root);
Group g2 = testFile.createGroup("Group2", root);
Group g3 = testFile.createGroup("Group3", root);
Group g4 = testFile.createGroup("Group4", root);/// design  groups
.....   ....
   Dataset PressureTop = new DataSet();// create a new dataset
   PressureTop= testFile.createScalarDS
```

```
        (" 32-bit double ", g1, Datatype, dims2D, null, null, 0, data1); //set the PressureTop dataset
   Dataset PressureBottom = new DataSet();
   PressureBottom = testFile.createScalarDS
        ("32-bit double ", g1, Datatype, dims2D, null, null, 0, data2); //set the PressureBottom dataset
...... ...... }
```

The main objective of the monitoring system is to monitor the control performance on-line; for example how well the controller works together with the process to track the setpoint of the process output. The output variables, such as the temperature and the pressure in the reactor, the work state of the control devices, and the PID parameters are essential for the monitoring purpose. The data priority levels are set as shown in Table 6.1 in terms of the monitoring system specification. All the light data are set with the highest priority. The data amount of the light data in fact utilizes a limited data communication bandwidth and introduces a minor extra load to the communication channel. The transfer priority of the flow rate is set with the lowest priority in this study.

Table 6.1. Data priority

		Transfer Priority Rank
Light Data	State of Nitrogen Inlet Valve State of Gas outlet Valve State of Liquid outlet Valve State of emergency liquid outlet Valve PID parameters	1
Heavy Data	Concentration	1
	Pressure (top) Pressure (bottom)	2
	Temperature (top) Temperature (middle) Temperature (bottom)	3
	Flow Rate	4

6.6.3 Implementation

XML is chosen to collect and process the light data. There are two XML files designed to store and transfer the light data. One is for the PID parameters. The other is for the work state and the work mode of the four manual valves. The sampling interval is 1 second. Each data unit stored in XML is a real time point, which includes the value of the variables and the corresponding sampling time. The defined heavy data are collected in a HDF file; the file has four groups, which collect the pressure, temperature, concentration and flow rate respectively. The structure of the HDF file is shown in Figure 6.12. It is in a tree structure and can fully match the XML structure; each data variable is 3-dimensional; it includes the sampling time, the variable position in the dataset and the variable value. The data type of the datasets is double and the data length is 300.

Table 6.2. Heavy Data XML file

```
<HDF>
 <Sector>
 <RootGroup>
   <Group>
  <GroupName>Group4</GroupName>
  <ParentGroup>root</ParentGroup>
    <DataSet>
          <DataSetName>Concentration</DataSetName>
     <DimensionSize>2D</DimensionSize>
     <DatafromHDF5File>"HeavyData.h5" </DatafromHDF5File>
          <Section>
        <Attributes>2D 32-bit 2x300</Attributes>
        <DataType>Double</DataType>
        <ConcentrationData>69.35008843960156 69.35019580872302 69.35031905320368
                          69.35045492026426 69.35059988341087 69.35075012824488
                          69.35090158467695 69.35104999315783 69.3511909944403
                           ... ...  ...  ....
                          69.35165201569092 69.35163345134774 69.35158186165908
                          69.35149661075924 69.35137784497036 69.35122651829651
                          69.3510444037771 69.35083409127101 69.35059897244024....
                           </ConcentrationData>
        < SamplingTime>/19:38:57/ 19:38:57/ 19:38:58/ 19:38:59/ 19:39:00
                          / 19:39:01/ 19:39:02/ 19:39:03/ 19:39:04/ 19:39:05/ 19:39:06
                          / 19:39:07  / 19:39:08/ 19:39:09/
                           ... ... ...  ....
                          / 19:40:08/ 19: 40:09/ 19: 40:10/ 19: 40:11/ 19: 40:12/ 19: 40:13/
                          19:40:14/ 19: 40:16/ 19: 40:17/ 19: 40:18/ 19:39:19/ 19:39:20/19:40:21/
                          19:40:22/ 19:40:23/ 19:40:24/ 19:40:25/ 19:40:26/ 19:40:27/ ....
                        </SamplingTime>
     </Section>
   </DataSet>
  </Group>
 </RootGroup>
</Sector>
</HDF>
```

The HDF file is wrapped by XML through the *HDF Wrapper.class*. Table 6.2 shows the XML file, the structure of the Java class is shown in Figure 6.18. This class implements the HDF java libraries (*ncsa.hdf.object.* *) and JDOM API (*org.jdom.**); it accesses the HDF file and then converts the HDF based data into a XML file. The object XMLOutputter creates and outputs a XML file with the specified format. The four groups in the HDF file are converted into four JDOM documents; the datasets are mapped into the elements; the property information is stored in the element attributes. The wrapped HDF objects and the XML objects are processed as JDOM Document/Element objects in the Java environment. The remote clients access the JDOM objects by invoking the "look-up" method. The RMI security manager has been installed in both the local and remote sites, which allows the legal Java virtual machine to download the stub and skeleton class files. The following code is part of the server application.

```
public class HelloImpl extends UnicastRemoteObject implements PrioritySubmission {
                                                      // Implement a remote interface
 public HelloImpl() throws RemoteException {
  super();
  ... ...
  public static void main(String args[]) {
  if (System.getSecurityManager() == null)
  System.setSecurityManager(new RMISecurityManager());// install  the RMI security manager
    ...
   HelloImpl obj = new HelloImpl();// initalize a new instance
   Naming.rebind("///131.231.126.211/Server", obj); //bind this object instance to the name "Server"
    ............}
   }
```

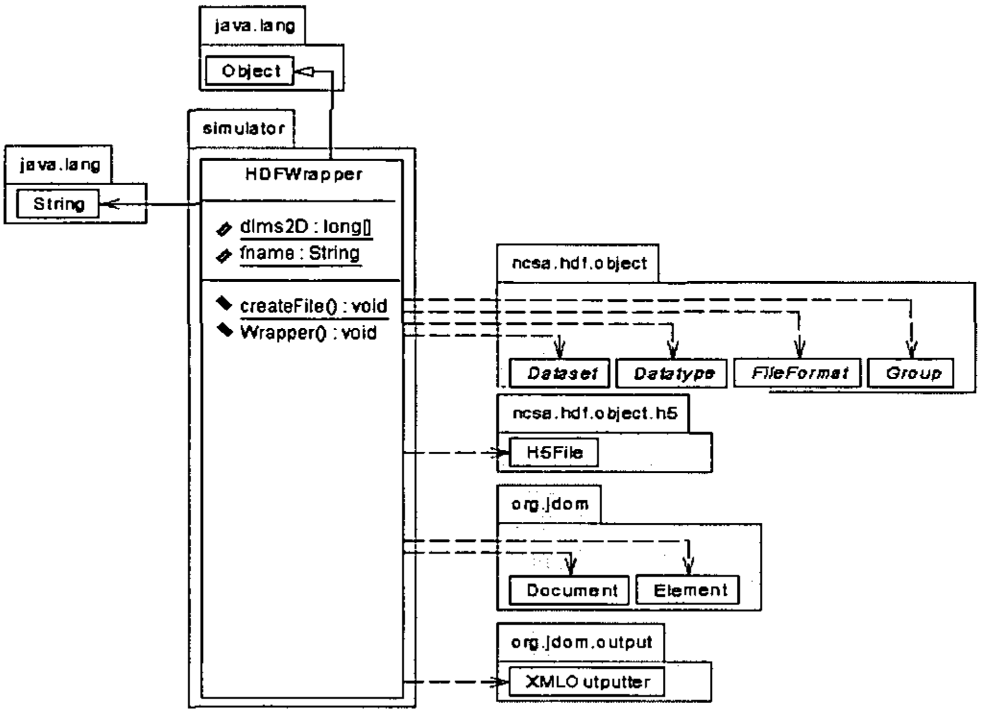


Figure 6.18. Structure of HDF Wrapper class

Table 6.3. PID parameters XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<Simulation>
 PID.xml <Sector>
                    <Parameters>
                              <PID_P>50</ PID_P >
                              <Sampling Time>11:34:47 12/08/2004</ Sampling Time >
                              <PID_I>5</ PID_I >
                              <Sampling Time>11:34:47 12/08/2004</ Sampling Time >
                             <PID_D>0</ PID_D >
                              <Sampling Time>11:34:47 12/08/2004</ Sampling Time >
          </Valve>
PID.xml </Sector>
</Simulation>
```

6.6.4 Simulation Results and Analysis

Using the RMI infrastructure, the manual valve work status and the PID controller parameters are sent via XML to the remote monitoring site. Figures 6.13 and 6.14 show the real time valves status/mode and the PID controller parameters. The defined light data are wrapped by XML. Table 6.3 gives a XML file for the PID controller parameters. Each data object such as PID_P in Table 6.3 is treated as an element object in JDOM, which has two sections, the parameter value and the sampling time. The defined heavy data are collected in the HDF format. The HDF groups and datasets are wrapped as the JDOM Document/Elements. The JDOM Document/Elements store both the light data and heavy data and are incorporated into the RMI transfer system. The server binds the data objects in the Object Registry so that any data object can be looked up from a remote site. The data priority policy is applied in the RMI based data transfer mechanism. Figure 6.15 shows the E201 Flow rate and R201 Top/Middle/Bottom temperatures. The flow rate of the hot stream is the output of the PID controller, and the input variable of the controller is the top temperature of R201 in Figure 6.15. Figure 6.16 shows the concentration of the product and the R201 Top/Bottom Pressures.

One of the distinguished features of the proposed data transfer mechanism is flexible. It is a two-way transmission channel. The server can transfer data to the clients and the clients can refuse accepting any non-wanted data by submitting their choices to the server. This flexible RMI based data transfer infrastructure enables the Internet channel being effectively used. Figure 6.17 shows the data suspending panel, in which the remote clients submit a request to suspend the data transfer of "Flowrate" at the instant 300. Once the RMI receives the request, the observer is notified to refuse rebinding the data object; therefore the reference to the flow rate data object cannot be obtained. Therefore, the transfer of this data object is terminated by the RMI server. Figure 6.15 shows that the flow rate data after the instant 300 is blank.

6.7 Conclusions

Web-based applications need an effective data transfer mechanism to process and transfer real time heterogeneous data. This chapter proposes a novel data object transfer mechanism. The mechanism is based on the RMI infrastructure and incorporates the XML and HDF data formats. The heterogeneous transfer data are categorised into two basic groups: light data and heavy data according to their physical meanings and the data amount. The light data are wrapped by XML. The heavy data are wrapped by HDF first and then by XML. Both the light and heavy data are processed as data objects in the Java environment with the support of JDOM. The data objects are transferred in the RMI based transfer structure. The structure is flexible; it not only allows the server to transfer the data objects, but also allows the clients to define and submit the data priority sequence to the server to guide the data transfer.

Chapter 7. Real Time Remote Maintenance of Controller Software over the Internet

7.1 Introduction

Remote maintenance of controller software is to maintain the control performance in an acceptable range over the Internet, which includes the dynamic and the steady-state performance both under the normal and faulty situations. It provides a platform to enable experts on the control/maintenance centre to remotely monitor, diagnose and maintain the geographically distributed controller software.

Traditional design methodologies used for the traditional maintenance management systems are not appropriate for designing the remote maintenance system for distributed controller software, as they do not consider the Internet related issues in remote maintenance applications, such as:

- Variable working load: process variables are collected from local control systems and are transferred to the remote maintenance system over the Internet, which leads to a variable data load caused by irregular Internet traffic, while the data load is determined from conception for a typical local maintenance system;
- Various users: the remote maintenance system will be open to authorized users and has uncertainty about who the users are, how many users there will be, and where they are;
- Heavy computing: the remote maintenance system needs to on-line monitor control performance, identify the running state of controller software and calculate compensator model. These operations involve heavy computing which may deteriorate the remote maintenance system's performance;
- Security: communication medium of the remote maintenance system is the Internet rather than any other private medium. The public Internet possesses security risks by nature of its open environment. Remote operation extending the scope of the Internet-based system can be falsified by outsiders through the Internet.

In this chapter, we seek to integrate the proposed techniques in the above chapters and apply them into the Internet environment to remotely maintain the controller software.

In detail, this chapter aims to develop a methodology for the design of an Internet-based controller software maintenance system. The design issues include remote control performance assessment, remote controller software state identification, degraded control performance recovery, and data process and transfer over the Internet. The rest of this chapter is organized as follows. The structure of the Internet-based controller software maintenance system is presented in Section 7.2. Section 7.3 describes the communcation infrastructure. The implementation of the monitoring level and the detection level of the maintenance system is given in Section 7.4. Section 7.5 discusses the implementation of the compensation level. A case study to illustrate the proposed remote maintenance system is given in Section 7.6. Section 7.7 is the conclusions.

7.2 Architecture of Remote Maintenance System

7.2.1 Remote Maintenance Platform

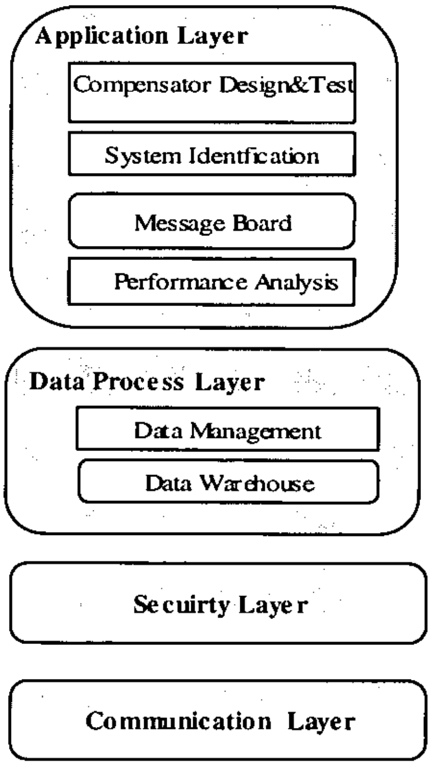


Figure 7.1. Platform for controller software maintenance

Platform Layers

A platform for the remote maintenance of controller software is shown in Figure 7.1. The platform meets the following objectives:

- Remotely monitoring the local control systems and assessing the running control performance;
- Enabling experts to remotely analyze the running control process, remotely interact the process and take appropriate actions to improve the control performance;
- Remotely maintaining the degraded control performance via dispatching a compensator to restore the degraded control performance.

The platform consists of four layers: a communication layer, a security layer, an application layer and a data process layer. Details of the four layers are discussed as follows:

1. Communication Layer

The communication protocol implemented in this platform is Java RMI (Remote Method Invocation). RMI assumes the homogeneous environment of the Java Virtual Machine (JVM), this protocol can therefore take advantage of the Java platform's object model whenever possible. RMI facilitates object function calls among Java Virtual Machines (JVMs), which can be located on separate computers. The communication layer establishes the back-end communication services for the remote maintenance application and the local control processes. Process variables will be sampled and collected, and then transmitted over the Internet to the maintenance center. Maintenance actions proposed by experts in the maintenance center will be sent to the local process sites to improve the control performance.

2. Security Layer

It needs to keep in mind that the remote application has to be secured against attacks from outside. Traditional maintenance management has been constructed as a closed system running in the local environment. The open feature of the Internet causes new security problems and important information can be falsified by hackers. Security strategies will be implemented in the maintenance platform to protect the remote applications from hazardous attacks.

3. Data Process Layer

Raw process variables transmitted from local sites are heterogeneous; they will be properly processed and managed in the maintenance application before they are used to analyze the control performance. The raw data are decoded and categorized into different groups to facilitate potential data mining. Whole transmitted data are stored in the data warehouse guided by the defined groups. The data warehouse provides data query and retrieval services for system components.

4. Application Layer

Control performance assessment measures are employed to analyze the control performance and generate quantitative values for the control performance assessment. These measures access the data warehouse to retrieve the required data. The assessment results are stored in a message board, which will broadcast these results to related components running in the platform. The system identification tool in this layer is to on-line identify the system model in both normal and faulty operations. A variety of models and identification techniques are implemented in this tool, such as the state space model and the autoregressive (AR) model, and the least square method and the recursive least squares method. The estimated models are used to calculate the compensator model and establish a simulation environment for the compensator test. Considering the calculation error and measurement error, the compensator will be repeatedly tested and tuned to obtain a satisfactory performance. Once the compensator is approved, it then sent to the local site via the communication layer to restore the degraded control performance.

Software Design

The maintenance platform is based on a multithreaded software design. Components are designed to run with threads to acquire process variables with multiple channels, distribute and analyze the variables simultaneously. The thread based design can improve the responsiveness, structure, and efficiency of the application. The application containing concurrent threads can run significantly faster on multiprocessor computers under multiprocessor operating systems like Windows NT, XP since each thread could get full use of its own respective CPU. In addition, in a multithreaded application, different threads are assigned different tasks which run independently and concurrently; the total elapsed time until a set of tasks is accomplished is the maximum of the individual task time rather than their sum, which can result in significant performance improvements.

Network Infrastructure

Latency (speed) and bandwidth (capacity) are two basic and separate elements to describe the features of a network. Latency refers generally to delays in processing network data. Bandwidth refers to the data rate supported by a network connection or interface. The combination of latency and bandwidth gives users a perception of how quickly data transfer over the Internet. In the present case, the discussion concerns bandwidth and latency over TCP/IP channels. LAN speeds are usually 100 Mbps and are usually symmetric (uplink throughput is equal to the down-link rate). A study (Masri et al., 2004) has shown that the LAN communication is more than enough to transfer the heavy data over the Internet to simultaneous five clients or more with a less than $1\mu s$ latency.

Data Flow

Components in the maintenance platform are driven by process variables transmitted from the local process site. The data flow is shown in Figure 7.2. The raw data are firstly categorized into different groups to store as Java objects; as a result, components in the platform can easily access and retrieve the related data via calling or visiting these objects. Whole process variables are administered by the data warehouse. System algorithms can call the data mining service of the warehouse to catch the required data to perform their tasks. Also, the warehouse is open to legal applications and works independently with other components; it facilitates to implement new system algorithms in the maintenance application.

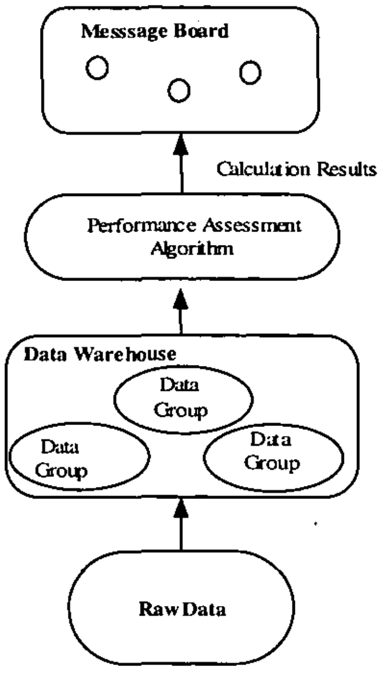


Figure 7.2. Data Flow

The generated performance results calculated by the system algorithms are stored in the message broad which is accessible by both the local and remote components. The calculation results are visualized in the charts of the user interface, which will give a dynamic image about the essential information of the current system status and performance. The dynamic images will be automatically refreshed after a certain period of time. This provides experts on the maintenance center with real-time information about the local control system.

Service Management

Experts can access the maintenance centre anywhere and anytime to remotely monitor and maintain local control processes, however, it is necessary to avoid more than one service/user simultaneously control a device of the local control system, which will lead to the concurrent access problem. Considering that, the services of the platform will be managed. The management registers all available maintenance services and arranges a priority sequence for the services and service users. Basically, local users have higher priorities than remote users. Priorities of the maintenance services are over the ones of the monitoring services. The implementation of services will follow the priority policy. The higher priority services/users will take the operation first when operating a same device. The service management can optimize the operation of services, avoid potential conflicts, reduce the number of actual links with the Internet and improve the capability of networking; it is also consistent with congestion control in the Internet (Warnier et al., 2003).

7.2.2 Framework of Remote Maintenance System

Figure 7.3 shows the perspective view of the remote maintenance framework. It consists of a Java Remote Method Invocation (RMI) server, a maintenance center and a local control process. The maintenance application provides various remote maintenance services over the Internet to local control systems. The server provides communication services; it uses the full power of object-oriented technology in distributed computations to transfer data objects between the remote maintenance system and the local control system. A Java RMI security manager is implemented in this work for security checking. The manger will stop illegal users to operate the maintenance center by blocking their IP addresses. The application layer of the platform decomposes into three functional levels: control performance monitoring level, detection level and compensation level.

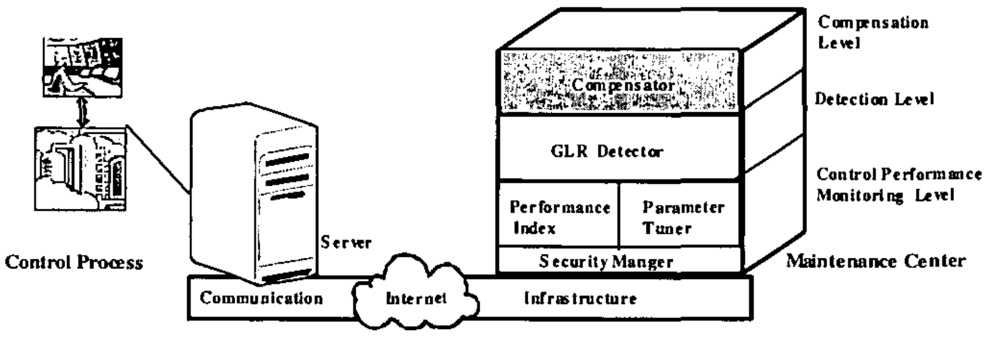


Figure 7.3. Structure of the remote maintenance system

The monitoring level measures the control performance and adjusts the control parameters. The control performance index component calculates the comparison of the cost function values of the actual control process and the LQG benchmark control process proposed in Chapter 3. The parameter tuner remotely monitors and tunes the parameters of the local controller software. A GLR based detector proposed in Chapter 4 is employed in the detection level to locate the cause of the degraded control performance. The detector uses a moving window to observe the controller software behaviors and identify the controller software state. Discrepancies between the benchmark LQG control signals and actual control signals are proven to be in an approximate Gaussian distribution. A baseline of the controller in the normal operation is treated as the asymptotic distribution of the discrepancies. The detector implements a GLR test to on-line monitor the discrepancies. The compensation level is triggered immediately after a faulty state of the controller software is detected. The compensator is to fix rather than replace the faulty controller software. On the basis of the controller behaviors in both normal and faulty states the compensator is designed to stabilize the control process and restore the degraded performance.

7.3 Communcation Infrastructure

Java RMI is a popular middleware platform for Internet-based applications and enables remote communication between programs written in Java. RMI allows Java developers to invoke object methods, and execute them on remote Java Virtual Machines (JVMs).

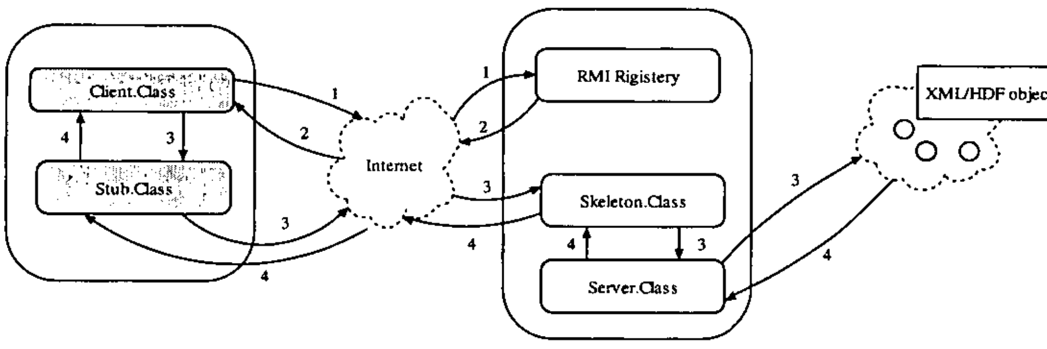


Figure 7.4. XML/HDF object transfer via RMI

The data transfer and process scheme proposed in Chapter 6 is incorporated in the controller software maintenance system. The collected process variables are categorized into light and heavy data based on the amount of the data and their physical meanings. The collected light and heavy data are wrapped by XML and HDF respectively and processed as XML/HDF objects in Java environment.

It needs to generate a client stub and a server skeleton for the RMI communication. The skeleton is a server-side entity that contains a method to dispatch calls to the actual remote object implementation. The stub is a proxy for a remote object in the client side to forward method invocations on remote objects. The stub/ skeleton will be bound onto the client/server through a binding action before the objects begin to transfer in the RMI infrastructure. The procedure of transferring a XML/HDF object in RMI involves the following four steps as shown in Figure 7.4:

1. The client looks up the server object in the RMI Registry to query a reference of the server object;
2. The RMI Registry returns a remote reference;
3. Via the stub, the client class interacts with the skeleton class to access the object transfer methods in the server object and invocate the methods; the XML/HDF object is defined as a return value of the methods;
4. The server catches the required XML/HDF object and returns the object transfer methods via the skeleton and RMI communication infrastructure to the client side.

7.4 Implementation of Monitoring and Detection Level

7.4.1 *Interaction among System Components*

The monitoring and detection levels in the structure shown in Figure 7.3 are further decomposed into ones shown in Figure 7.5. The data warehouse collects and stores the transferred process variables from the Internet. The warehouse provides data query and retrieval services for system components. These system components are motivated by the process variables retrieved from the data warehouse. Firstly, the process model is identified based on the process variables stored in the data warehouse. A LQG controller is designed to control the identified process model. The LQG controller and the identified process model form a LQG benchmark control loop. The generated LQG control signals and LQG process outputs are utilized to compute the LQG cost function. Secondly, the process variables retrieved from the data warehouse are used to compute the actual cost function. The values of the actual cost function and LQG control function are compared to get a performance index value according to Equation 3.14 in Chapter 3. A poor performance index value will trigger the GLR detector to work. Finally, the transmitted actual control signals are sent to the GLR detector. The GLR detector traces the discrepancies between the transmitted actual control signals retrieved from the data warehouse and the LQG control signals generated by the LQG controller, which are served as a benchmark of the monitored controller software. The likelihood ratio s_k to detect the mean of the discrepancies is computed according to Equation 4.5 in Chapter 4 and compared with the predefined threshold λ. The comparison result determines whether the compensation is required or not. In the normal operation, the compensator is inactive. Once s_k is greater than the threshold λ, which indicates that the controller software is in a faulty state, the compensator is activated.

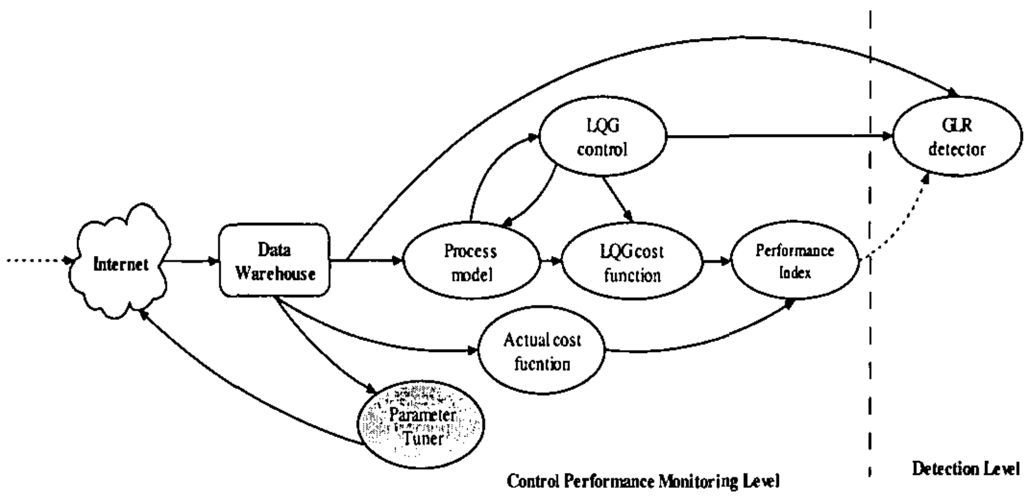


Figure 7.5. Interaction between the system components

7.4.2 Parallel Computing

Components in the remote maintenance system are required to co-operate with each other and perform their tasks concurrently. The multi-thread technologies are employed in the remote client applications. Each component is allocated with a Java thread, which allows the component programs to efficiently perform their tasks independently and share the latest process variables simultaneously. The running threads in the components are set with different priorities in order to effectively utilize the computer resources. The running components are managed by a Java thread scheduler. The scheduler can suspend a component by making its thread sleeping or kill a component by terminating its thread, and also monitor all running threads to decide which threads should be running and which should be waiting according to their priorities. Those with higher priorities are executed before lower priority threads. In this work, the LQG control component is set with the highest priority because of its twofold roles in the remote maintenance system: the benchmark for the controller software behavior and the control performance assessment. The process model component has the same level of the priority as the LQG control component because of the requirement of synchronously running the LQG control loop. The actual cost function component, the LQG cost function component and the performance index component are set with a higher priority. The up-to-date performance index value is used to decide the GLR component's priority setting. If the performance index is greater than 0.5, which indicates the control performance is satisfactory, the GLR component is set with a lower priority. Once the calculated performance index is less than 0.5, the GLR component is set as a higher priority to locate the cause of the poor performance as early as possible.

7.4.3 Parameter Tuner

The parameter tuner allows experts on the maintenance centre to remotely monitor and tune parameters of the local control software as shown in Figure 7.6. The server collects parameters of controller software and sends them out to the remote maintenance site over the Internet. The parameter tuner visits the data warehouse and edits the parameters. In actual applications, most of poor performances are caused by unsuitable parameters. The parameter tuner establishes a platform for control experts to diagnose the degraded control performances. By observing the running state of control processes, control experts propose a set of optimal parameters for the running control software to improve the degraded control performance. The server receives the new parameters from the parameter tuner and passes the new parameters to the local control software.

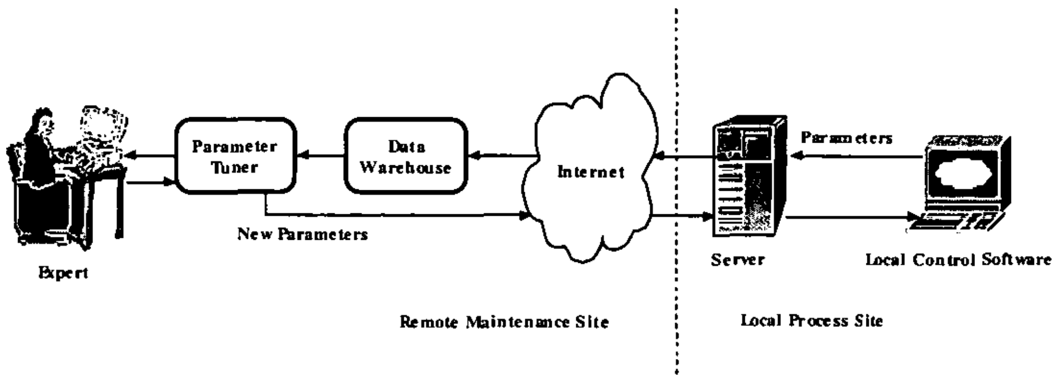


Figure 7.6. Remote tuning parameters of the local controller software

One issue arising from the implementation of the parameter tuner is the concurrent user access of the local controller software. It is possible that remote experts and local operators or distributed experts simultaneously change the controller software parameters. To deal with the multi-user concurrent access, in this work, we allocate users with different priorities. The users with a higher priority can immediately overwrite the commands issued by the users with a lower priority. Generally, the local operators have higher priorities than the remote experts considering the possibility of emergency in the local control systems. Any parameter change made in the local controller software must be authorized by the local operators.

One of the main advantages of implementing the parameter tuner is that the experts can remotely diagnose the possible causes of the poor performance by analysing the parameters of the controller software and remotely tune these parameters in order to possibly recover the performance. It has been reported that 80% of loops have performance problems, 30% of loops actually increase variability in the short term over manual control; most of causes of the poor performance are due to unsuitable parameters (Ingimundarson, 2003). Therefore, remotely monitoring and tuning the parameters of the local controller software may be a practical solution for these performance problems.

7.5 Implementation of the Maintenance Level

7.5.1 Compensator Test

A compensator is designed in the compensation level to maintain the degraded control performance. The compensation level consists of a computing engine and a test bed as shown in Figure 7.7. The computing engine mines the required data from the data warehouse and computes a compensator model. The compensator is designed via the use of the fault shift-based method, and the compensator model is calculated according to Equation 5.4 in Chapter 5. The obtained compensator model is sent to the test bed, where a simulation associated with the faulty open loop model is employed to refine and verify the compensator model. The compensator is tested and tuned in the test bed, which is similar to the virtual control laboratory (VCL, 2000). The test bed simulates the real implementation of the designed compensator. The structure of the test bed consists of a designer, the designed compensator, the identified faulty open loop model and an interactive interface. The compensator and the faulty open loop model form a virtual control loop. The designer monitors the performance of the virtual control loop and tunes the compensator parameters via the interactive interface. The test is to assess how well the compensator works with the faulty open loop to trace the setpoint. The interactive interface shows the deviation between the setpoint and the virtual process output. The deviation is used to assess the compensator's performance. A small deviation range indicates a healthy performance of the compensator. An unstable and large range of deviation indicates the demand of adjusting the compensator parameters. The parameters are adjusted until a satisfactory control performance of the virtual control loop is obtained.

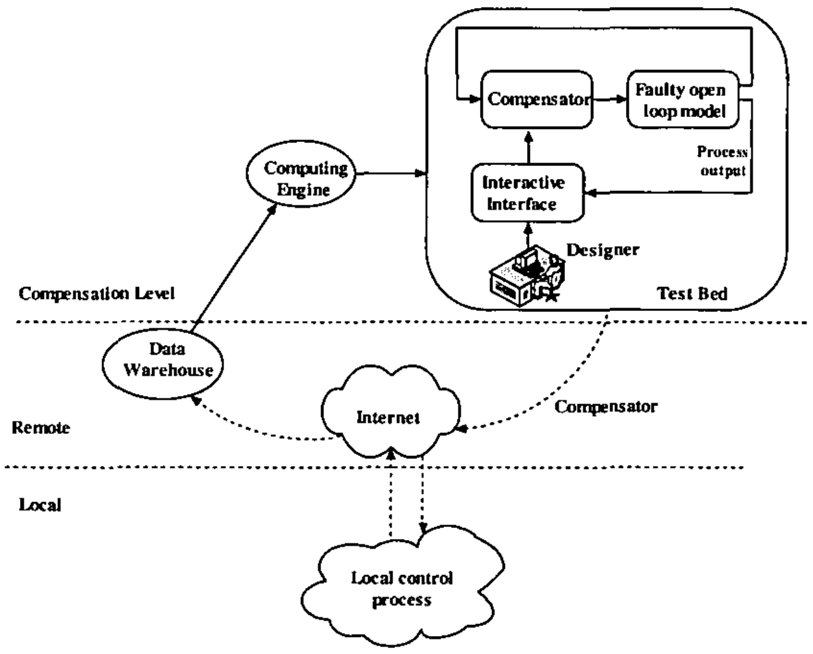


Figure 7.7. Components in the compensation level

7.5.2 Compensator Implementation

The compensator is designed and tested in the remote site within the remote maintenance system. Because of uncertain Internet time delay, it is not appropriate to locate the compensator in the remote site within the maintenance system. There are two ways to download the compensator from the remote site and implement it in the local site with the process plant: weak migration and strong migration, as shown in Figure 7.8. The weak migration transfers only parameters of the compensator from the design site to the local site, while the strong migration transfers both the parameters and the structure of the compensator to the real plant site. The strong migration is employed only if there is no any compensator installed before or the structure of the compensator has been changed since the last implementation. The weak migration is employed to update the parameters of the compensator only within the same structure of the compensator.

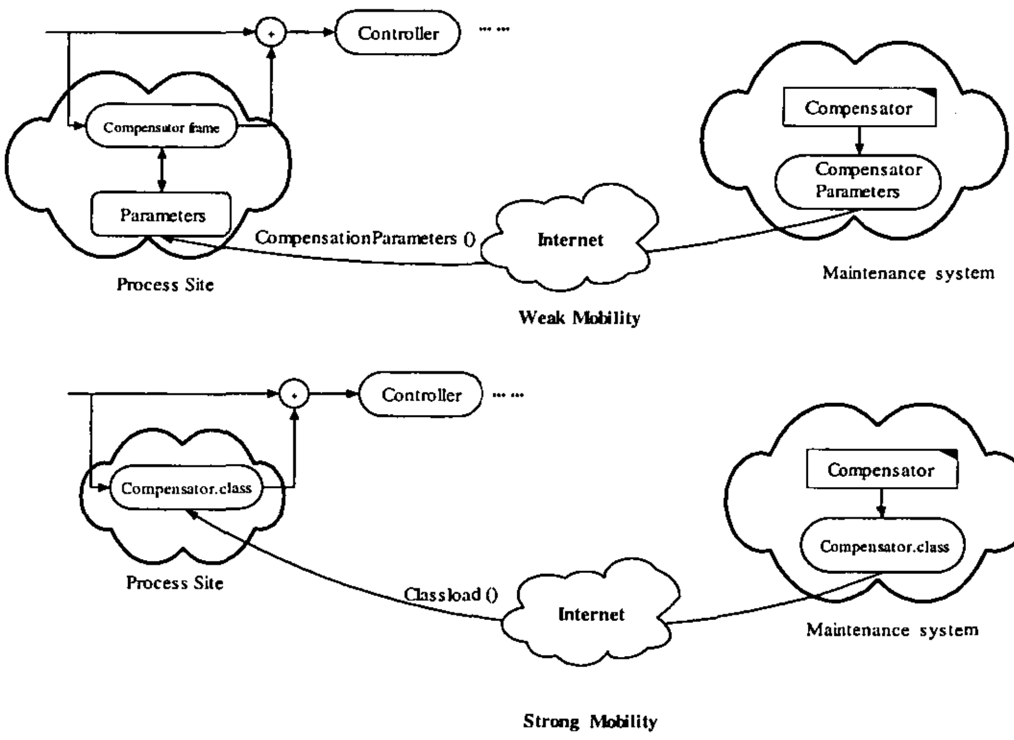


Figure 7.8. Compensator mobility

Both migrations are established on the basis of the Java RMI client/server *Remote* interface *java.rmi.Remote*. The weak migration transfers the parameters of the compensator from the maintenance system to the local site by passing them as the parameters of a remote method invoked. In detail, a client in the local process site creates and then visits a remote object after locating the remote server object via the RMI naming service, and calls a method *CompensationParameter()* from the remote object. This method brings the filled parameters in the remote server and travels from the remote maintenance system to the local client site. The strong migration, in contrast, transfers the code of the compensator from the remote maintenance system to the local client site through the Java RMI interface. A remote *ClassLoad()* method is used to search a suitable compensator from the maintenance system and bring the code to the local site.

The model of the designed compensator shown in Equation 5.4 can be re-written as the following general linear difference equation:

$$a_0 y(t) + a_1 y(t-1) + a_2 y(t-2) + .. + a_n y(t-n) = b_0 x(t) + b_1 x(t-1) + .. + b_n x(t-n) \quad (7.1)$$

where $y(t)$ and $x(t)$ are the output and input of the compensator at the instant t respectively. The parameters a_0, a_1, ..., a_n, and b_0, b_1, ..., b_n are the model coefficients obtained from the design process and parameter tuning in the test bed. The weak migration will only transfer the parameters a_0, a_1, ..., a_n, and b_0, b_1, ..., b_n from the remote maintenance system to the compensator frame located in the local site. The strong migration will transfer the complete code of the compensator designed in the remote maintenance system to the local site.

7.6 Case Study

To illustrate the proposed remote maintenance system, a Process Control Unit (PCU) in the Networks & Control Laboratory at Loughborough University has been chosen for the demonstration and evaluation.

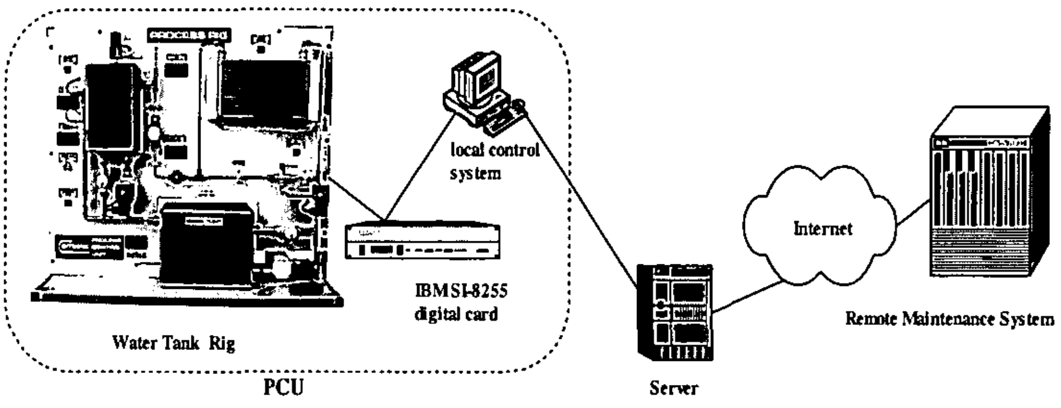
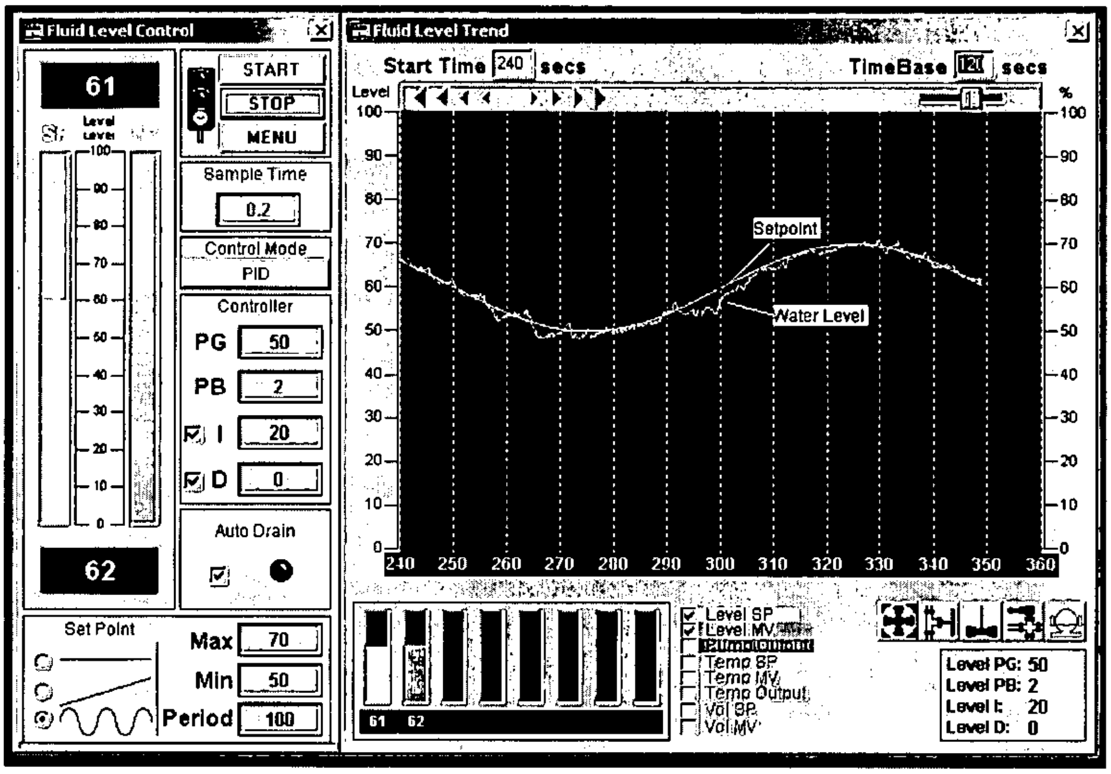


Figure 7.9. Experimental system layout

7.6.1 System Description

The layout of the experimental system is illustrated in Figure 7.9, which includes the PCU, a server, and the proposed remote maintenance system. The PCU consists of a water tank rig, an I/O interface, and a local Proportional-Integral-Derivative (PID) control system. Figure 7.10 gives the operation interface of the local PID control system. The PID parameters, setpoint, and sampling interval can be chosen by the local operators in the operation interface. The RMI server is used in the experimental system, and connected with the local control system and the remote maintenance system via a 100 Mb/second Ethernet network. The local control system and the remote maintenance system are physically located in the same laboratory.

Figure 7.11 shows the interface of the remote maintenance system. The three-level structure shown in Figure 7.3 is implemented here. Other two alternative control performance assessment methods such as IAE (Integral of Absolute Error) and MSE (Mean Squared Error) are also implemented with the performance index at the performance monitoring level. The user interface of the remote maintenance system also provides a platform for the remote engineer to design, test and remotely implement the compensator in the real site through the weak and/or strong migration.



61

62

Sample Time: 0.2

Control Mode: PID

Controller

PG: 50

PB: 2

I: 20

D: 0

Max: 70

Min: 50

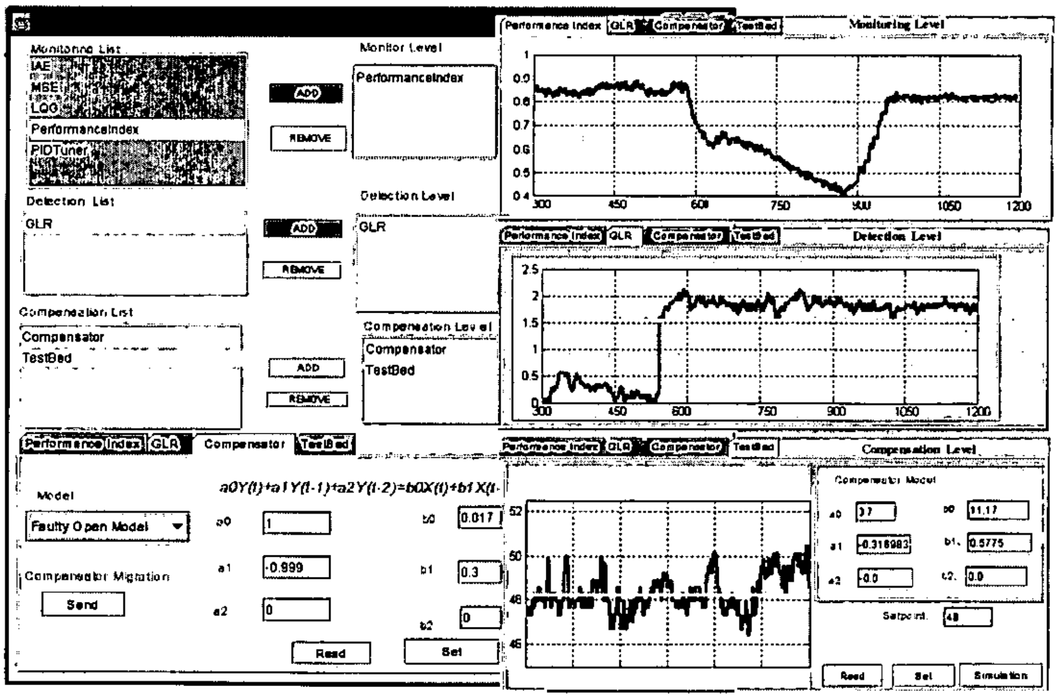
Period: 100

Start Time 240 secs

TimeBase 120 secs

Level PG: 50
Level PB: 2
Level I: 20
Level D: 0

Figure 7.10. Local control system



Model: Faulty Open Model

$a0Y(t)+a1Y(t-1)+a2Y(t-2)=b0X(t)+b1X(t-$

a0: 1
a1: -0.999
a2: 0
b0: 0.017
b1: 0.3
b2: 0

Compensator Model

a0: 1.7
a1: -0.316983
a2: 0.0
b0: 11.17
b1: 0.5775
b2: 0.0

Setpoint: 48

Figure 7.11. The interface of the remote maintenance system

7.6.2 Remote Data Process and Transfer

The collected data collected from the water tank process include PID parameters and process variables. Parameters of the PID controller are defined as light data. Java Document Object Model (JDOM) is a Java representation of an XML document and provides a way to represent that document for easy and efficient reading, manipulating, and writing XML from Java. The light data are processed into a JDOM Document. Table 7.1 is a XML file recording the PID parameters. The sampling interval is 1 second. Each of the PID parameters stored in XML is a real time data point, which includes the value of the variable and the corresponding sampling time. The process data include the setpoint, the liquid level and the controller output. The process data are categorized as heavy data collected and stored a HDF file; the file consists of three datasets, for the setpoint, liquid level and controller output respectively. The data length of the datasets is 300. Both light and heavy data are processed in the Java environment and outputted as JDOM Document objects and HDF objects. The JDOM objects and HDF objects are defined as return values of the data transfer methods. When the maintenance system invokes a data transfer method on a remote object in the server. The method is passed to the server over the Internet; the server catches the required data object and sends the data object back to the maintenance system as a return value of the data transfer method.

Table 7.1. PID XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<Simulation>
 PID.xml <Sector>
                 <Parameters>
                         <PID_P>3</ PID_P >
                         <Sampling Time>11:34:47 25/03/2005</ Sampling Time >
                         <PID_I>20</ PID_I >
                         <Sampling Time>11:34:47 25/03/2005</ Sampling Time >
                        <PID_D>0</ PID_D >
                         <Sampling Time>11:34:47 25/03/2005</ Sampling Time >
                   </ Parameters >
PID.xml </Sector>
</Simulation>
```

7.6.3 Performance Monitoring and Fault Detection Levels

The remote controller parameters tuning is implemented in the parameter tuner component of the maintenance system. A tuning environment is given for the comparison of the control performances with the original and new set of parameters as shown in Figure 7.12. All the process variables and the PID parameters are transferred to the data warehouse located in the remote maintenance system over the Internet. The process tank model is identified as $\frac{0.308z + 0.359}{z^2 - 0.4991z + 0.1}$, where z denotes the z-transform.

A LQG benchmark controller is designed in the remote maintenance system for the control performance monitoring and fault detection purposes. The LQG controller is based on the Model Predictive Control (MPC) (Huang and Shah, 1999). Both the prediction and control horizons in the MPC are set as 15 sampling intervals, i.e. 15 seconds in this case study. The ideal LQG benchmark controller works within the remote maintenance system at a minimum control cost. This cost will be compared with the actual control cost made by the PID controller; therefore the performance index is obtained. The differences between the ideal LQG benchmark controller and the actual PID controller are also used in the GLR fault detection.

In order to illustrate how well the remote maintenance system can identify the potential faults occurring in the local control system of the water tank process, a gain, K=0.1, is introduced in the output of the local PID controller at the instant 550 second. As the result, the output of the PID controller is reduced by 0.1 times afterwards. Figure 7.13 shows the water level deviating away from the setpoint immediately after the fault is introduced. Figure 7.14 illustrates the performance index significantly dropping from 0.85 at the normal operation to 0.42. The likelihood ratio generated by the GLR test jumps from 0.2 at the normal operation to 1.8 as shown in Figure 7.15, which is greater than 1.55, the predefined threshold for the GLR test. The faulty state has been detected.

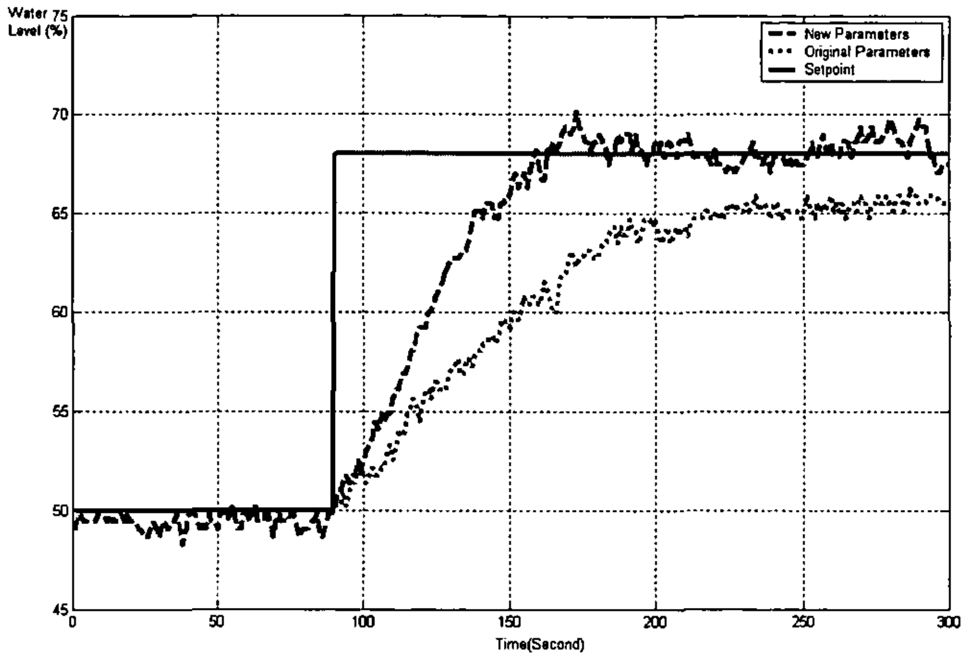


Figure 7.12. Parameter Change

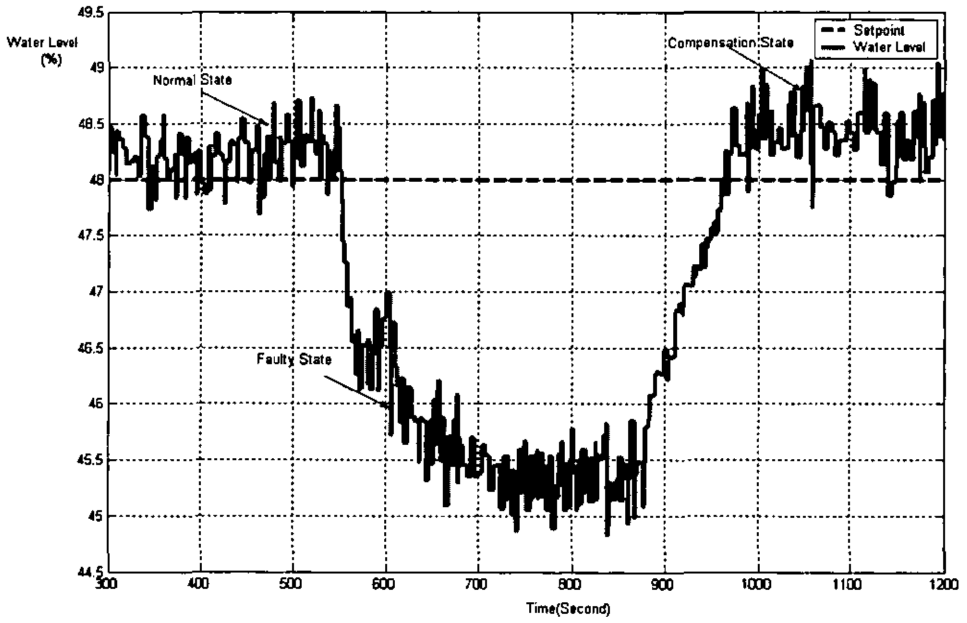


Figure 7.13. Water Level

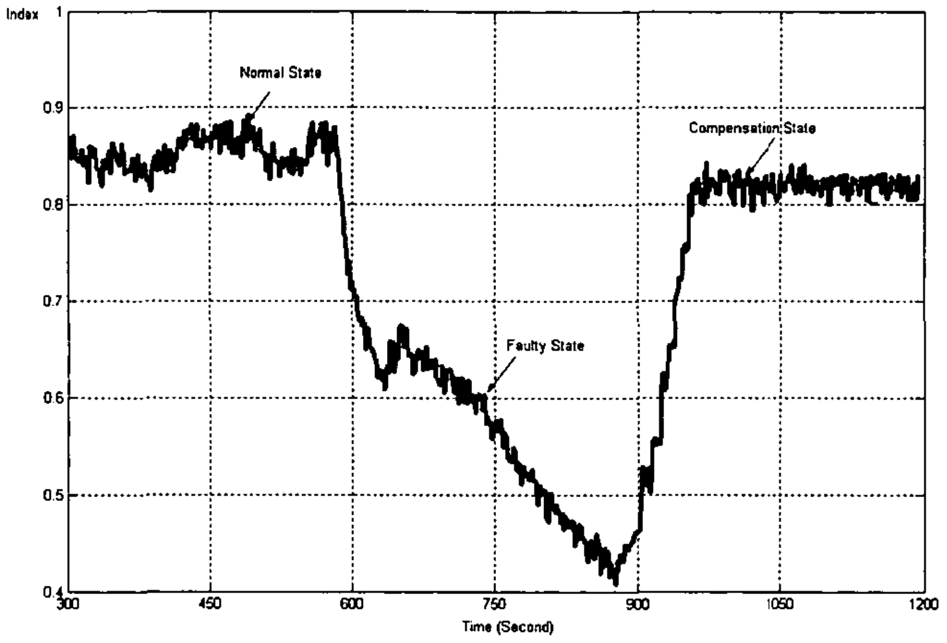


Figure 7.14. Performance Index

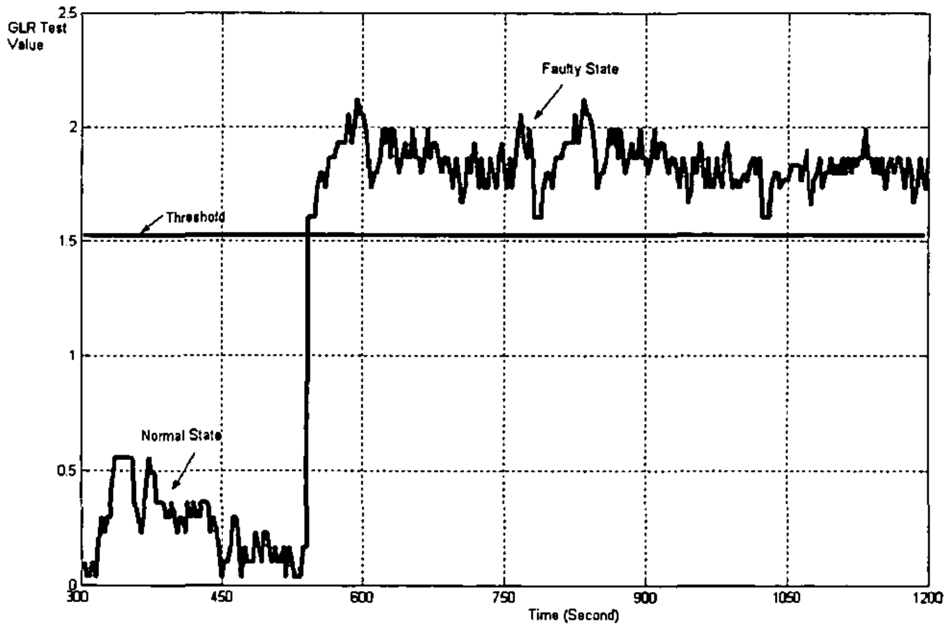


Figure 7.15. GLR Test

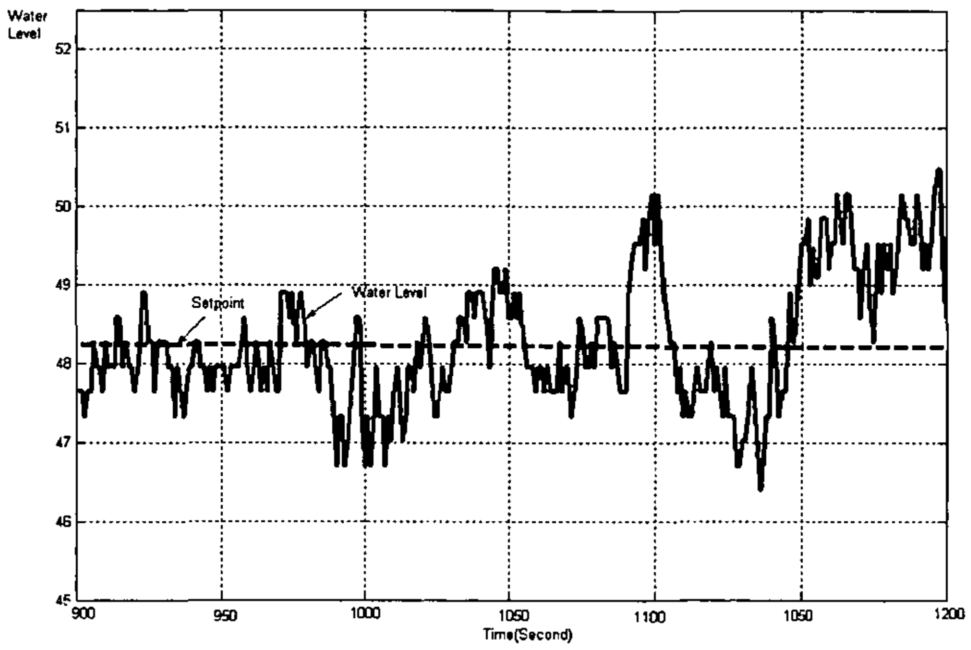


Figure 7.16. Test result of the compensator

7.6.4 Compensation Taken Place

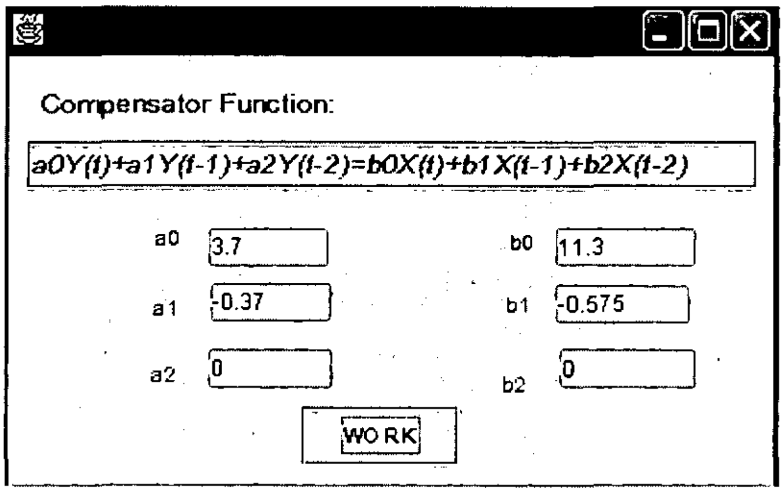
Once the likelihood ratio generated by the GLR test goes over the pre-defined threshold the compensation level is activated. A compensator is designed based on the open loop models of the process identified for the normal operation and the particular abnormal operation. A compensator model is obtained as $\frac{11.3z-0.5775}{3.7z-0.37}$ in terms of Equation 5.4. The compensator is tested in the test bed before implemented in the real application. Figure 7.16 gives the simulation result of the compensator and illustrates that the compensated system will have a satisfactory performance. The approved compensator is expressed as a linear difference equation as follows:

$$3.7y(t)-0.37y(t-1)=11.3x(t)-0.5775x(t-1) \tag{7.2}$$

where $y(t)$ and $x(t)$ are the output and input of the compensator at the instant t.

A general compensator structure as shown in Equation 7.1 has been previously implemented in the local control system as shown in Figure 7.17. After the compensator designed for the particular fault has been approved in the remote maintenance system the weak migration is employed to remotely install the compensator shown in Equation 7.2 in the real process site. The parameters {(3.7, -

0.37, 0), (11.3, -0.5775, 0)} are sent to the pre-defined compensator structure. The local operators put the compensator in action if they have been convinced that the compensation is necessary and safe. In our case study the compensator is put in action at the instant 880 seconds. Figures 7.13 and 7.14 show that the water level is back to the desired setpoint and the performance index back to the normal value 0.81 immediately after the compensator in place.


Compensator Function:

aOY(t)+a1Y(t-1)+a2Y(t-2)=bOX(t)+b1X(t-1)+b2X(t-2)

a0	3.7	b0	11.3
a1	-0.37	b1	-0.575
a2	0	b2	0

Figure 7.17. Compensator frame

7.7 Conclusions

There is a great deal of benefits for remotely monitoring, diagnosing, maintaining the controller software over the Internet. In this chapter, we present an Internet-based remote maintenance system for controller software, which provides a novel way for remotely supporting and maintaining the controller software in the process industries. This system is based on a highly efficient multithreaded software design that allows the system to acquire data from a large number of channels, distribute and analyze this data, in real time, over the Internet to multiple remote locations. As the consequence, controller software design and maintenance centres can be created which are operated on the web and provide central support to all the authorised industrial control software. Time and money can both be saved because the control software providers can remotely design, tune and maintain their geographical distributed software products. A water tank rig with a PID control is chosen as case study. The experimental results demonstrate the feasibility of the proposed Internet-based maintenance system.

Chapter 8. Mobile Agent-Based Remote Maintenance of Controller Software

8.1 Introduction

A mobile agent is a computer program, consisting of both code and data that is able to migrate autonomously from node to node to perform some computations on behalf of the user (Cabri et al., 2000). Traditional distributed systems are built out of stationary programs that pass data back and forth across a network. Mobile agents, by contrast, are programs that are computation moves; task processing is performed locally by the agents where data are stored. This approach substantially reduces the amount of network traffic generated and can bring a number of benefits for distributed software applications in the Internet environment including the ability to reduce network use, increasing asynchrony between clients and servers, adding client-specified functionality to servers and introducing concurrency (Wong et al., 1999).

Inspired by these distinguishing abilities and features of mobile agents in the Internet environment, in this chapter, the feasibility of mobile agent technologies implemented in the remote maintenance of controller software is investigated. A mobile agent-based system is proposed, which provides remote control performance assessment, controller software state identification and control performance maintenance services for the local controller software. Also, the travelling mobile agents can be remotely updated over the Internet. The running agent structure can be on-line reconfigured, and original services can be upgraded; new services can be on-line added into and implemented the travelling agents. A water tank process with a model predictive control (MPC) is used as a case study to illustrate the proposed remote agent-based maintenance system.

The rest of this chapter is organized as follows. Benefits of mobile agent implemented into controller software remote maintenance are discussed in Section 8.2. The architecture of the agent-based maintenance system is presented in Section 8.3. The dynamic updating of mobile agents is discussed in Section 8.4. A case study to illustrate the proposed agent-based maintenance system is given in Section 8.5. Section 8.6 gives conclusions.

8.2 Mobile Agent and Remote Maintenance of Controller Software

There are several challenges in the remote maintenance of controller software under traditional methods:

1. Bulk data transfer over the Internet. The process data collected from the local process sites need to be sent to the remote site. In practice, the collected data are often heterogeneous and heavy. The heterogeneous and heavy data transfer over the Internet will lead to complicated data encoding and decoding in the local and remote sites, and also irregular data transmission and data loss.
2. Unbalanced computing distribution. Control performance maintenance involves complicated heavy computations and will cost a lot of computing resources. Under the traditional server-client structure, the server application is responsible for data collection and transmission, and all the computing tasks are carried out in the client application, which may be overloaded under the limited computation abilities of the client application and result in a long and unacceptable response time.
3. Uncertain Internet time delay. The designed compensator located in the remote site needs to work with the faulty control process in real time to recover the degraded control performance. A study (Yang et al, 2003b) has demonstrated that the Internet contains the serious and uncertain time delays. Therefore, the implementation of the compensator at a remote site is required to be insensitive to the Internet time delay.

Mobile agents are a suitable candidate for the remote maintenance operations over the Internet. Mobile agents are beginning to be implemented into the remote monitoring and maintenance operations and have resulted in small-scale demonstrations and applications. Yu et al. (2003) established a Problem-Oriented Multi-Agent-Based E-Service System to provide rapidly responsive services for industrial processes. Jezic et al. (2004) proposed a Remote Maintenance Shell (RMS) for software management in large distributed environments. The method is based on the remote operations performed by cooperative mobile agents. Lovrek et al. (2003) developed an agent-based software maintenance application. It allows software installation, modification and verification on the remote target system without suspending its regular operation.

So far, little research has investigated the implementation of mobile agents into remote maintenance of controller software, however, this implementation can bring various benefits, including:

1. The compensator can work locally in the faulty control loop to restore the degraded performance without any influence from the Internet time delay;
2. Only analysis results of the target controller software are sent back to the remote site; the amount of data transfer transmitted over the network can be minimized and the usage of the network can be dramatically reduced;
3. The computation tasks can be distributed among the remote and local sites. Parts of work are performed in the client site and the other are operated in the server host as the agents are sent to the server site to perform their tasks. As a result, the heavy computing tasks can be dispensed into both the client and the server applications, which will be beneficial to improve the whole computing performance;
4. Mobile agent-based transactions are robust and flexible. They can react autonomously to changes in their execution environment, and are therefore more flexible in their operation.

8.3 Architecture of the Agent-Based Maintenance System

As shown in Figure 8.1, the agent-based maintenance system is separated into three basic elements wired by the communication bus: a mobile agent server, an agent host and an agent client. The agent client dispatches mobile agents to the agent host to operate their tasks, and receives the return values of the travelling agents. The agent server establishes a communication channel between the agent client and the agent host. The agent host receives and offers residence, necessary resources and runtime support to the agents. The agent host also links with the control process to collect process variables for the agent execution in real time. The following sections will give the details of the agent host and the agent client shown in Figure 8.1.

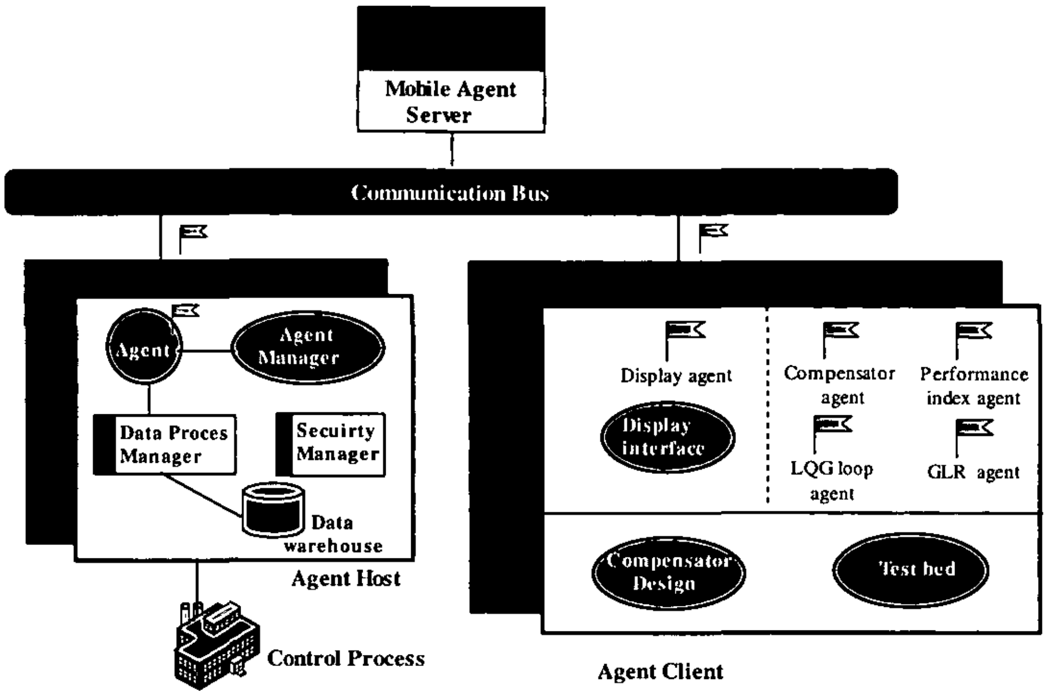


Figure 8.1. Structure of the remote maintenance system

8.3.1 Agent Host

The agent host comprises an agent manager, a data process manager, a security manager and a data warehouse. The agent manager provides a platform for the incoming agent execution, retrieving the downloading agent class and scheduling the agent's operation. The structure of the agent manager is shown in Figure 8.2.

The agent manager implements an interface called AgentHostInterface, which is not only available locally to the agent host, but also to the remote applications by extending to *Remote* class in the java.rmi.remote package. The AgentHostInterface provides one method, acceptAgent().

```
public interface AgentHostRemoteInterface extends Remote {
  public void acceptAgent(AgentInterface ai) throws RemoteException;
  ......}
```

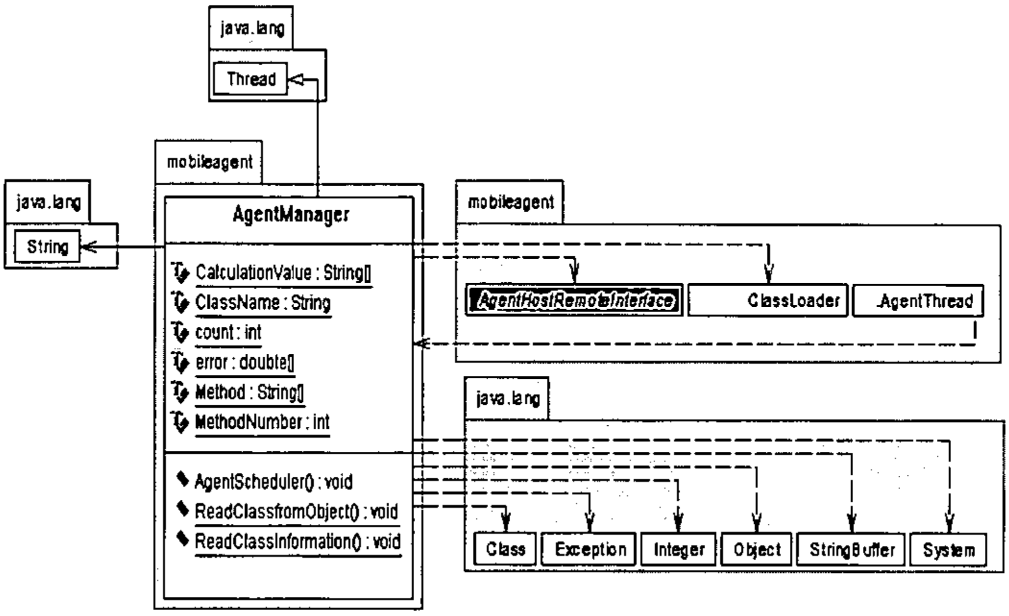


Figure 8.2 Structure of the agent manager

The agents call this method to submit their requests and travel to the agent host via invoking the *Classloader*. Once the agent request is accepted, the agent manager binds an incoming agent to a new instance of an inner class *AgentThread*, which extends a Java thread class. As a result, the travelling agent is running with a thread. The travelling agent in the host has three states: active, asleep and dead. Starting a thread indicates that the host accepts the agent travelling request and the agent begins to tour the host. The initial state of the accepted agent is active. In the active state, the manager arranges the necessary system resources to run the thread, schedules the agent to run and perform its task. The agent is asleep when the thread finishes a running cycle. The sleeping agent will be waked up by the agent manager to run in the next working cycle. The agent could die naturally as a result of finishing the journey or killed by the host through terminating the thread. The following code illustrates the management of the agents on the host:

```
public void acceptAgent(AgentInterface ai) throws RemoteException {
    AgentThread at = new AgentThread(ai);
      at.start();           // MobileAgentHost binds an incoming agent to an AgentThread:
      at.run();
    }
```

```
  public void TerminateAgent(AgentInterface ai ) throws RemoteException {
      AgentThread at = new AgentThread(ai);// terminate the mobile agent
   }

   // Inner class that extends Thread and runs a given mobile agent */
   private class AgentThread extends Thread {
     private AgentInterface myAgent = null;
      AgentThread(AgentInterface ai) {
        myAgent = ai;
      } }
```

The traveling agent class is downloaded via the *Classloader* and stored in the host by the agent manager. The traveling agent is registered in a registration table at the agent host. The registration information includes the agent name, the storage location, and the agent properties, such as the required data or the state of the agent.

The data process manager provides data querying and retrieval services for the travelling agents. The data process manager collects data from the process sites in real time. According to the amount of the data and their physical meanings, the collected data are categorized into light and the heavy data by the data process manager. The light data are considered to be small quantities of scientific data. The heavy data are considered to be large amounts of data. The light data and heavy data are wrapped by eXtensible Markup Language (XML) and Hierarchical Data Format (HDF) respectively. Both the light data and heavy data are processed into XML and HDF objects and stored into Java data objects. Under the tree based XML/HDF hierarchy structure, the data in the warehouse can be easily looked up and retrieved. When the data query request is passed to the data process manager, the data process manager first identifies the data feature, light or heavy data, according to the query specification, and directs to the XML or HDF objects. The data process manager then looks up the location of the data in the tree based XML/HDF structure according to the query specification and sends the retrieved data back to the agents.

The security manager is implemented for security checking and authenticates the agent before it is allowed to execute. The security manger protects the host and the mobile agent against unauthorized access. The mobile agent system automatically invokes the security manger to check the access of resources. The following code illustrates the installation of the security manager:

```
public void Register() {
....
if (System.getSecurityManager() == null)//
System.setSecurityManager(new RMISecurityManager()); // install a RMI security manager
MobileAgentHost host = new   MobileAgentHost();// create an agent host instance
 Naming.rebind("//131.231.126.211/Server", host); // bind the instance to the server
   ....
   }
```

8.3.2 Agent Client

The agent client consists of various mobile agents, a compensator design component, a test bed component and a display interface.

Mobile Agents

The agents can be categorized into two main types in terms of their functions:
User agent: The user agent is responsible for holding a user request, bringing it to the host, interacting with the host and returning back to the user site with the results of analysis. The user agent is a delegate of human users, coordinating with others in cyberspace.

In this work, a LQG loop agent, a GLR agent, a performance index agent, a compensator agent, and a data process agent are designed as user agents. The functions and tasks of these agents are defined as follows: the LQG loop agent includes a LQG component and a process model component. These two components form the LQG control loop as shown in 8.3. The *Porcessmodel()* method simulates the process model. The *LQG_ControlSingal()* generates the LQG control signals the process model. The *Simulation()* method simulates the LQG control loop via implementing the two above methods. The LQG loop agent serves as a benchmark for the control performance assessment and the controller fault detection.

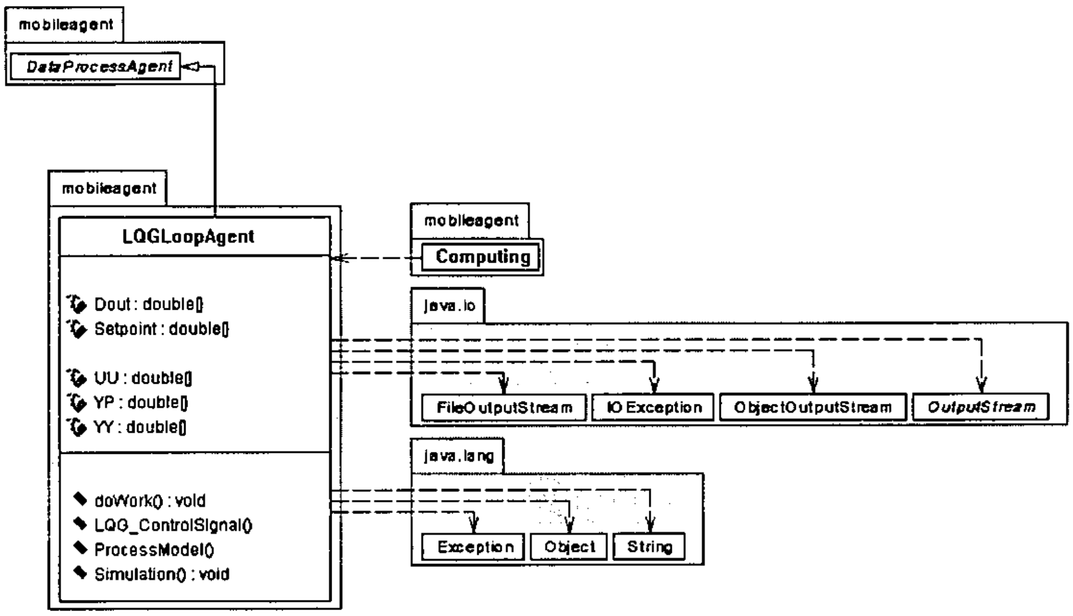


Figure 8.3. Structure of the LQGLoop agent

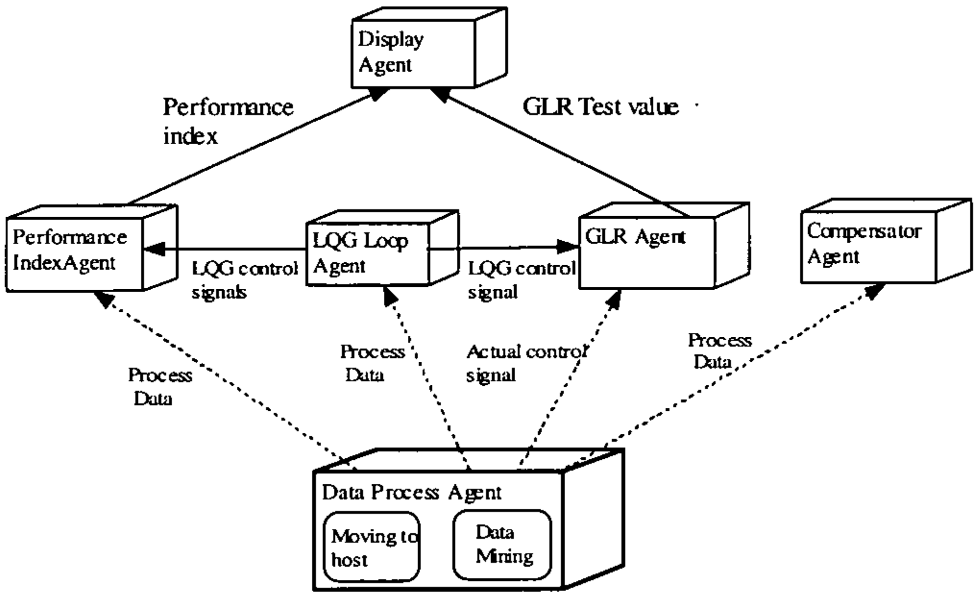


Figure 8.4. Interaction among agents

The GLR agent employs a GLR test to produce the discrepancies between the actual control signals and the LQG control signals generated by the LQG loop agent to identify the controller software state. The performance index agent computes cost functions for both the LQG control generated by the LQG loop agent and the actual control loop, and then obtains a performance index. The compensator agent is to recover the degraded control performance when the controller software is identified as being in a faulty state.

The data process agent provides two main services: host travelling and data mining:

1. Host travelling: it involves three steps: (1) looking up all the currently available mobile agent hosts; (2) selecting a host from the list; (3) moving to a new host by calling the acceptAgent () method. If the call on the selected host fails, repeatedly submits the request until it is accepted.
2. Data mining: It cooperates with the data process manager to query and retrieve the required data from the data warehouse.

The following code illustrates the two functions of the data process agent:

```
//// Host traveling:
public void AgentBytes ( byte[] AgentClass,String ClassName )
{
    FileOutputStream out = new FileOutputStream(ClassName);
    ObjectOutputStream f = new ObjectOutputStream(out);
    f.writeObject(AgentClass);
    f.flush();
    myAgentName=ClassName ;
}

/// agent host remotely call this method to transfer the agent class to the remote host site.
public void MovetoHost(){
....
 hosts= Naming.list();              // record all current mobile agent hosts
                                    // look up the available agent host to travel
AgentHostRemoteInterface agent = (AgentHostRemoteInterface ) Naming.lookup("//131.231.126.211/Server");
hosts.add(agent);                  // adding  the available host into the lists
agent.AgentAccept();              // send a travel request;
agent. AgentBytes (findClassBytes( ClassName+ ".class"),"mobileagent."+ClassName);
                                 // move the agent class to the agent host
    }...
```

```
  //// Data mining:
public void ReadXMLData(){
....
Element XMLData=null;
AgentHostRemoteInterface Host = (AgentHostRemoteInterface) Naming.lookup("//131.231.126.211/Server");
                                                                    // access the agent host
 XMLData=(Element)AgentHost.ReadXML();                              // retrieve  the XML based data
  .....  }

 public void ReadHDFData(){
....
AgentHostRemoteInterface Host = (AgentHostRemoteInterface) Naming.lookup("//131.231.126.211/Server");
                                                                  // access the agent host
PIDOut=AgentHost.ReadPIDOut();
 Setpoint=AgentHost.ReadLevelSP();
 ProcessOut=AgentHost.ReadProcessOut();                          // retrieve  the heavy data
.....
    }
```

Display agent: this agent is stationary, which is located in the agent client. The agent enables the visualization of the results of the user agents and shows the results on the display interface.

Compensator Design and Test

Different from other user agents, the compensator agent is on-line designed and tested by the compensator design and test bed components. Figure 8.5 depicts the life cycle of the compensator. The first stage is to establish a reference model, where the boundaries for the control process, the desired dynamic and steady-state performances are incorporated into the reference model selection. The next step in the life cycle is to identify the faulty open loop model. System identification technologies are employed to identify parameters of the mathematical model. The obtained model should be verified by means of comparing to the response of the real process. Based on the obtained model, a compensator can be deliberately designed. The compensator design component calculates the compensator model; where the compensator model is assumed to insert into the post-fault control loop and make the new post-fault control loop follow the pre-defined reference model. When the compensator design is completed, the test bed simulates the real implementation of the designed compensator, observes how well the compensator performs and repeatedly tunes the compensator parameters until a satisfactory control performance is obtained. When the compensator is approved, it is then dispatched and implemented in the local process site to recover the degraded control performance. The deviation between the expected and actual responses of the real process will be feedback to the remote site at the design stages of the model and the compensator to update the model and modify the compensator.

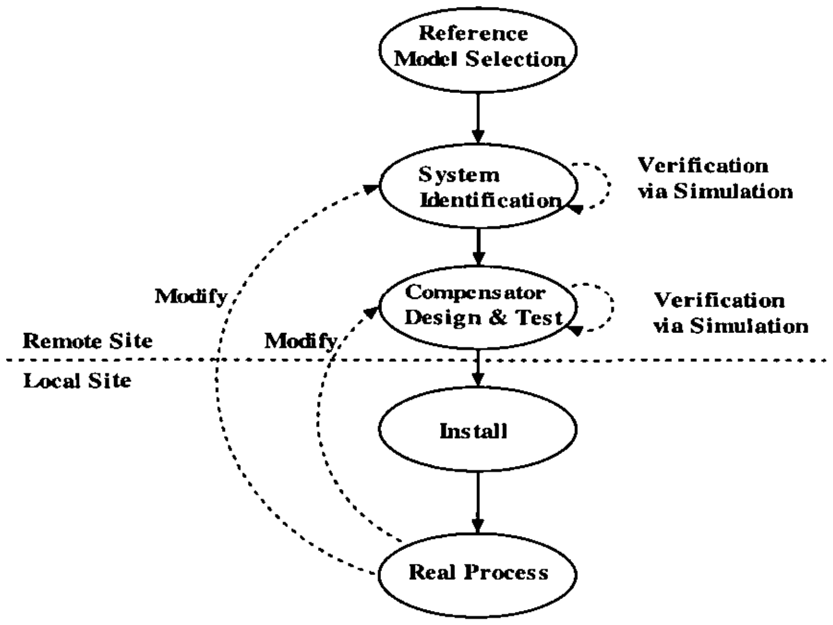

Reference Model Selection
System Identification
Verification via Simulation
Compensator Design & Test
Verification via Simulation
Modify
Modify
Remote Site
Local Site
Install
Real Process

Figure 8.5. Compensator life cycle

8.3.3 Agent Implementation on the Agent Host

The agent manager catches the required data to schedule and run the agents when a particular work situation is met. To catch the required data, the agent manger works closely with the data warehouse. The designed user agents will query and retrieve the data from the data warehouse. These mobile agents have different roles to play in the controller software maintenance and result in different work situation requirements in the agent implementation. A poor performance index value will trigger the GLR agent to work. The GLR agent on-line tests the discrepancies between the transmitted actual control signals and the LQG control signals. In the normal state, the compensation agent is in the remote site. Once the controller software is detected as being in a faulty state, the compensator agent moves to the local process site to recover the degraded control performance. The agent manager has a specification about the agent implementation sequence, which guides the agent manager to schedule the agents to work. The agent begins to work by starting the associated threads. These threads are set with priorities to meet the requirements in the agent implementation sequence. Those with higher priorities are executed before lower priority threads. The agent manager can suspend an agent by making its thread sleeping or kill a component through terminating its thread.

8.3.4 Communication among Agents through Agent Server

There are three forms of information exchange among mobile agents in terms of the information content:

- Message: Message is a static data, such as the state of the running controller software; it is public and broadcasted in the network and shared by all agents running in the host.
- Real time data: The real time data has strict time restrictions; the correctness of the data communication depends not only on the logical correctness of the computation performed but also on their timing.
- Data object: The data object is not single value but blocks of many thousands data. The data object communication is mainly in the display agent and the user agents. These user agents bring the analysis results to the agent site. The analysis results are processed as data objects. The display agent catches the data object, retrieves the data from the object and then displays the data on the user interface.

The agent server uses the shared communication bus to deliver the communication information to a messaging board. The messaging board is a persistent storage where the communication information is kept. All agents are implemented through an interface called *AgentInterface*, which can be accessed by both local and remote agents. The messaging board publishes all the information on the *AgentInterface*. Therefore, the agents can locally or remotely access the messaging board, query and retrieve the required data or message or object from the messaging board.

```
public interface AgentInterface  extends Remote  {
 public void DataExchange(Object Agent , Object  Data)throws RemoteException;
                                                    //store and exchanges data among agents
       ......  ......        ......
}
```

8.4 Dynamic Updating of Mobile Agents

8.4.1 Requirements of Dynamic Updating

Dynamic updating enables that the services provided by mobile agents are upgraded, deleted or added during their executions. New services can be delivered from time to time, and downloaded when needed. Assume an agent has a service A; with the dynamic updating, service A can be deleted or upgraded to a new service A'. The new service can be added into the agent. Dynamic updating enables the life time and application range of the mobile agents be significantly extended and the new technology or design strategy can be incorporated into the original design agents to meet up-to-date requirements; the faulty source code in the agent can be replaced or corrected before any damage happens. Requirements of implementing the dynamic updating of the mobile agents are summarized by Emako et al. (2003), including:

- Service interruption due to the agent upgrading should be minimal.
- Time lag of the upgrading procedure should be minimal.
- The solution should be simple to implement.
- Updating should not impact the behaviour of the existing service.
- The solution should be independent of the mobile agent platform.

The main challenge is how to minimize the services interruption and the impact of behaviours of the existing services. Generally, the updated agent and related agents are potential interrupted elements when a running agent is upgraded. The latest historic data in the original running agent may be lost and the service performance may be deteriorated during the transition of the updating. On the other hand, in a complicated mobile agent-based application, mobile agents are often not independent but interactive. Mobile agents need to cooperate with each other to perform their tasks. Any change in the service provided by a running mobile agent can require the reconfiguration of the mobile agents and cause service interruption.

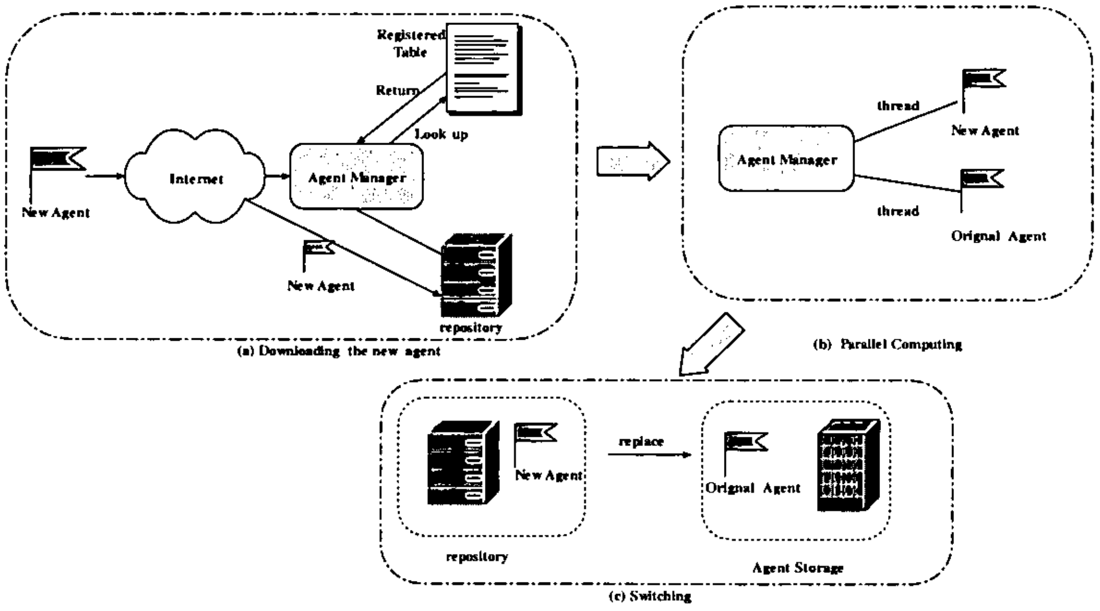


Figure 8.6. Agent replacement

8.4.2 Agent Replacement

The agent replacement makes the original agent be replaced by a new agent, which provides a new required service. The agent replacement consists of the following three stages as shown in Figure 8.6:

1. Downloading the new agent: The agent manager has a listener. When the new agent is downloaded into the host, the listener announces an event; the agent manager then looks up all travelling agents on the registered table according to the new agent name. If an agent is found with the same name in the registered

table, the new agent will be arranged to store in a temporary repository, which is different from the storage place of the original agent.

2. Parallel Computing: The new agent is invoked from the temporary repository by the agent manager. Like any other travelling agent, the agent manager allocates the new agent with a new thread. Both new and original agents are parallel running, where the new agent is trained to adapt to the new working situation to stabilize its performance. The computing results of both the original and new agents are sent back to the remote agent client site and compared by the experts.
3. Switching: Once the new agent is determined to be implemented, the new agent will be replaced with the original agent. As mentioned above, the agent is running under the thread control. To minimize the interruption during the replacement transition, the best replacement time is during the period of the agent sleeping, where the agent is suspended and the running agent class resource is released. The thread state can be monitored by the method, GetState(), which returns the state of the thread. When the thread enters into the asleep state, the running agent class resource will be released and the original agent class is overwritten by the new agent. When the replacement is finished, the new agent in the temporary repository will be automatically deleted in order to leave space for the next replacement.

8.4.3 Reflection

Reflection is a technique that first emerged in the programming language community to support the design of more open and extensible languages. The key to the approach is to offer a meta-interface supporting the inspection and adaptation of the underlying virtual machine. The reflection technique is nicely summarised by the following quote from Smith (1984):

"In as much as a computational process can be constructed to reason about an external world in virtue of comprising an ingredient process (interpreter) formally manipulating representations of that world, so too a computational process could be

made to reason about itself in virtue of comprising an ingredient process (interpreter) formally manipulating representations of its own operations and structures."

Reflection is now widely adopted in the language design, as witnessed for example by the Java Core Reflection API. Reflection is also increasingly being applied to a variety of other areas including operating system design, concurrent languages, and increasingly distributed systems. By exposing the underlying implementation, reflection becomes straightforward to monitor the implementation, e.g., performance monitors, quality of service monitors, or accounting systems (Capra et al, 2001). Reflection can also be used to alter the internal behaviour of the middleware (adaptation). Examples include replacing or changing the implementation of the underlying transport protocol to operate more optimally over a wireless link, introducing an additional level of distribution transparency in a running computation (such as migration transparency), or inserting a filter component to reduce the bandwidth requirements of a communication stream.

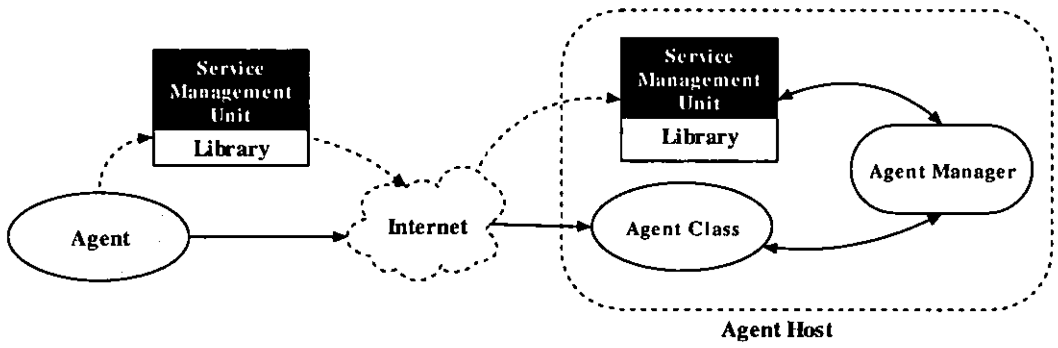


Figure 8.7. Service management unit

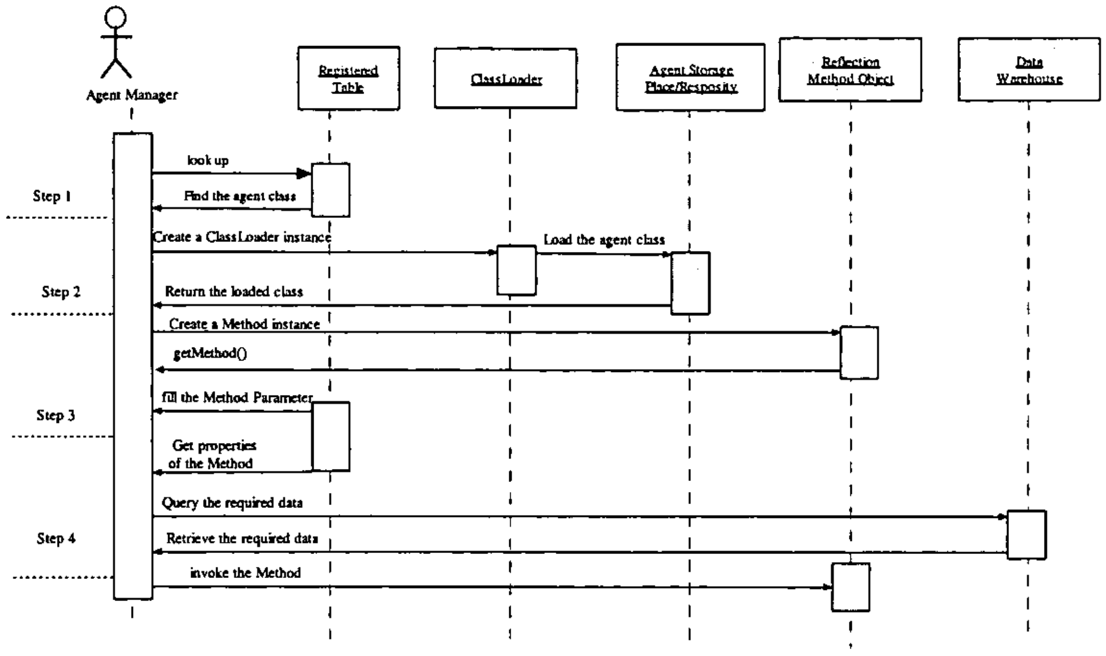


Figure 8.8. Procedures of implementing the agent services

8.4.4 Reflection in Mobile Agents

When the new agent is downloaded to the host, there is a challenge to get a general solution to run the service provided by the new agent, which is unknown to the agent host. The traditional solution is rather difficult and tedious, which involves with re-implementing the interfaces, and re-designing the sources code to modify method calls (Capra et al, 2001). The *Java.lang.reflect* package is a Java Reflection API, which provides classes, interfaces and platforms for obtaining and implementing reflective classes and objects. This Reflection API is the mechanism that allows for creating an instance of a class whose name is not known until runtime and invoking methods in the class even if the method names are not known as well until runtime. This feature is very helpful in implementing the new agent with new services, since both the new agent class and new services in the class may be totally different from the originals ones.

The integration between reflection technology and mobile agent updating is another new feature of our approach and is based on service management units as shown in Figure 8.7. The service management unit gives the host awareness about the new agent services and guides the host to implement the new agent services. Each mobile agent has a management unit with a unique name. The management unit is responsible for binding with the corresponding agent and recording the structure information of the mobile agent. The management unit travels with the agent to the host. The management unit has a library to store the mobile agent class information, including the class name, methods, and the parameters and attributes of methods. The library can be edited; the method list can be added and deleted. The library is stored as an object. In the host site, the travelling service management unit closely works with the host agent manager to implement the mobile agent's services. All management units will be registered in the host agent manager. The agent manager uses certain methods and objects of *Java.lang.reflect* to implement the services provided by the new agent class. The implementation procedure is illustrated in Figure 8.8, which consists of four steps as follows:

(1) Finding the agent class. According to the registered table, the agent manger visits the agent storage and the repository place to locate the stored agent class. If it can't find the class, the agent manager asks the agent client to send the agent class again.

(2) Loading the agent class from the class storage place and repository. An instance of Java *ClassLoader* class is initialled. Given a name of the class, the class loader will load the required class.

(3) Creating a method object of *Java.lang.reflect* by invoking the method getMethod (String Name,...), where the parameter of name is the method name of the found agent class; the method is retrieved from the management unit library.

(4) Invocating the method. The agent manager visits the management unit to obtain the properties of the method. The properties guide the agent manager to find the required data to invoke the method. When the required data of the method are retrieved from the host data warehouse, the retrieved data will be filled into the method parameter and then the method is invoked.

To implement all services of the agent class, steps (3) and (4) will be repeated until all the methods in the management unit library are invoked. The following code illustrates these steps:

```
Public static  void ImplementAgentClass (String ObjectName,String MethodName){

File f= new File(ObjectName);// find the agent class

if ( file==true) // if the agent class is available  then loading the agent class

  {

   FileInputStream in = new FileInputStream(ClassName);

   ObjectInputStream s = new ObjectInputStream(in);

   byte[] Data=  ( byte[])s.readObject();

    }

      .... ...

 ClassLoader  loader= new ClassLoader();                                   // create  a class loader

 Class c= loader.findClass(ClassName,Data);                                // create a Class object

 Method mm = c.getMethod(Method[i],  new Class[] {data.getClass() });  // create a Method object

 CalculationResults= (String)mm.invoke(null, new Object[] {data});        // invoke the method

.....}
```

The management unit, the agent replacement, and the agent manager work together to achieve the dynamic updating of the mobile agents. The management unit records the reconfiguration information of the agent class structure. The information of adding, deleting or changing services are stored in the management unit library. The management unit is bound to the agent. When the new agent replaces the original agent, the new management unit also overwrites the original management unit during the thread sleeping time. Objects and methods of the Java Reflection API are called by the agent manager to implement the services of the new agent.

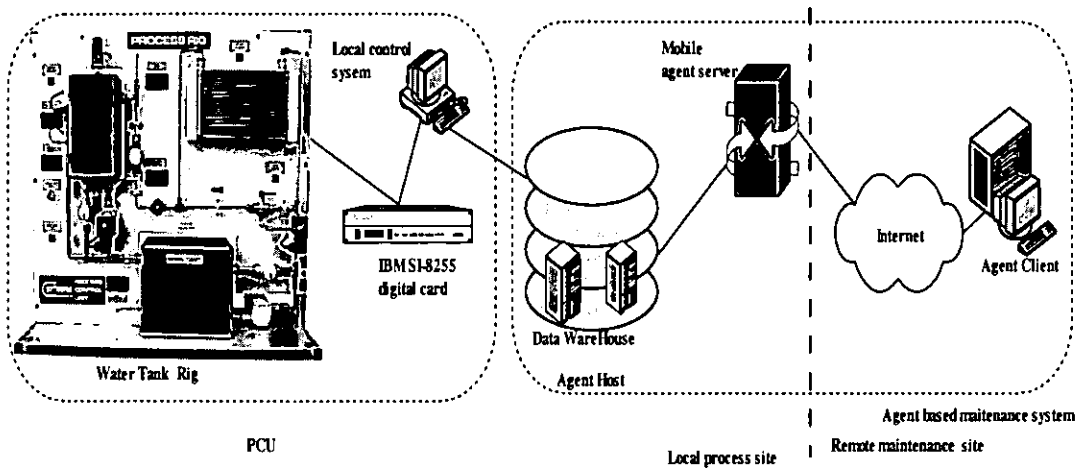
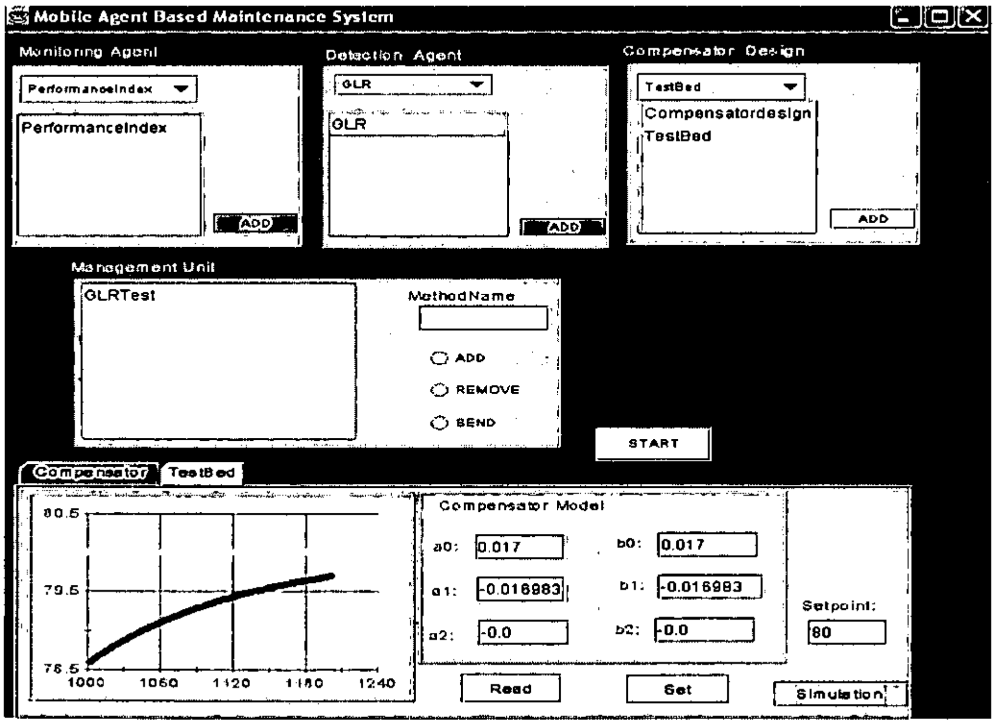


Figure 8.9. Experitmental system layout

8.5 Case Study

To illustrate the proposed agent-based remote maintenance system, a Process Control Unit (PCU) in the Networks & Control Laboratory at Loughborough University has been chosen for the demonstration and evaluation.



PerformanceIndex
PerformanceIndex
GLR
GLR
TestBed
Compensatordesign
TestBed
GLRTest

a0: 0.017 b0: 0.017
a1: -0.016983 b1: -0.016983
a2: -0.0 b2: -0.0
Setpoint: 80

Figure 8.10. User interface of the remote agent-based system

8.5.1 System Description

As shown in Figure 8.9, the whole system consists of two parts: the PCU and the proposed mobile agent-based maintenance system. The PCU consists of a water tank rig, an IBM 8255 digital Data Acquisition (DAQ) card, and a local MPC control system. The DAQ instrument is in charge of A/D and D/A conversion, which converts the analogue signals of the liquid sensor into a digital value, and converts the digital value of the pump output into an analogue value to drive the pump. The local MPC control system measures the liquid level of the process tank and regulates the flow rate of the pump to maintain the liquid level of the process tank at a desired value. The mobile agent server is used in the experimental system and connected with the agent host and the remote agent client via a 100Mb/second Ethernet network. The local control system and the remote maintenance system are physically located in the same laboratory.

Orignal Agent management unit library

Class Name	Method	Required data
Performance IndexAgent	PerformanceIndexCalculation()	MPC cost,LQGcost

(a)

↓

New Agent management unit library

Class Name	Method	Required data
Performance IndexAgent	PerformanceIndexCalculation()	MPC cost,LQGcost
Performance IndexAgent	ComputeIAE()	Setpoint, ProcessOutput

(b)

Figure 8.11. Agent management unit replacements

Figure 8.10 shows the interface of the remote agent-based maintenance system. Agents are selected from the monitoring and detection agent lists respectively. The selected agents are configured in the agent management unit panel. Methods of the agents will be filled into the management unit library. New methods in the agent class will be added and the original methods can be removed from the management unit library. The user interface also provides a platform for the remote engineer to design the compensator agent; the compensator agent is on-line designed and tested in the test bed.

8.5.2 Experiment Design and Result Analysis

Two experiments are carried out in this case study. One is to add a new service in the running performance index agent; the other is to replace the original running compensator with a new compensator in order to maintain the faulty controller software.

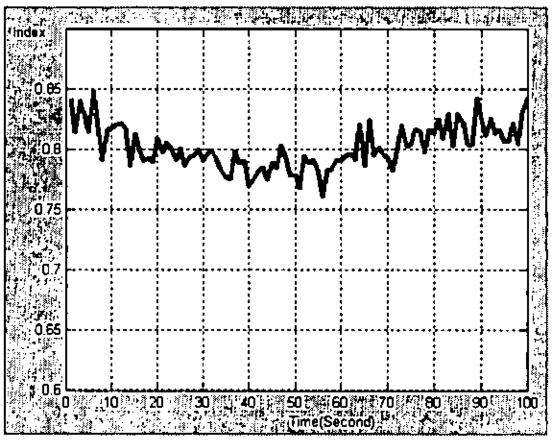
Experiment 1: adding a new service in the running performance index agent

The performance index agent implements a LQG benchmark to assess the control performance of the actual running MPC controller; it compares the cost value of the MPC control with the one of the LQG control. This comparison is to get a general measurement about the MPC control performance relative to the benchmark LQG control. To get a more comprehensive assessment of the running MPC control performance, an Integral of Absolute Error (IAE) measurement service is adopted and added into the performance index agent class.

The IAE measurement is employed to address the control performance during setpoint change. The implementation of updating the performance index agent consists of five steps:

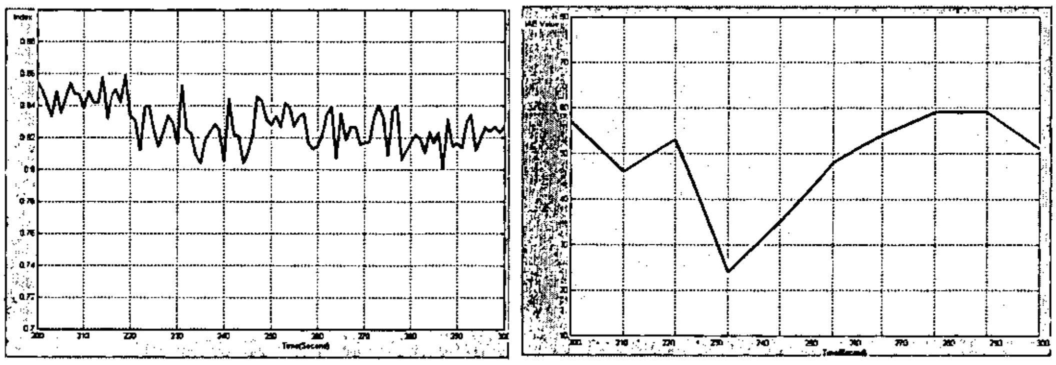
1. The agent management unit library located in the agent client site is edited. The original management unit library is shown in Figure 8.11(a). The method ComputeIAE() and properties of the method are added into the management unit library to reconfigure the index agent structure as shown in Figure 8.11(b). The method ComputeIAE () implements the IAE measurement for the control process.
2. Sending a message to the agent host to notify that a new index agent class and a new management unit will be downloaded into the host.
3. When the host receives the new agent class and management unit, both are stored in a temporary repository and registered in the registrable table.

4. The agent manager arranges both the new and original agents to parallel run.
5. The original index agent and management unit are replaced by the new agent class and management unit during the thread sleeping time. Figure 8.12 (a) shows the measurement results of the original index agent, which only implements the performance index measurement for the water tank process. Figure 8.12(b) shows that the measurement results of the new index agent, which start from the instant 200.



Performance Index measurement

(a)



Performance Index measurement (b) IAE measurement

Figure 8.12. Measurement results

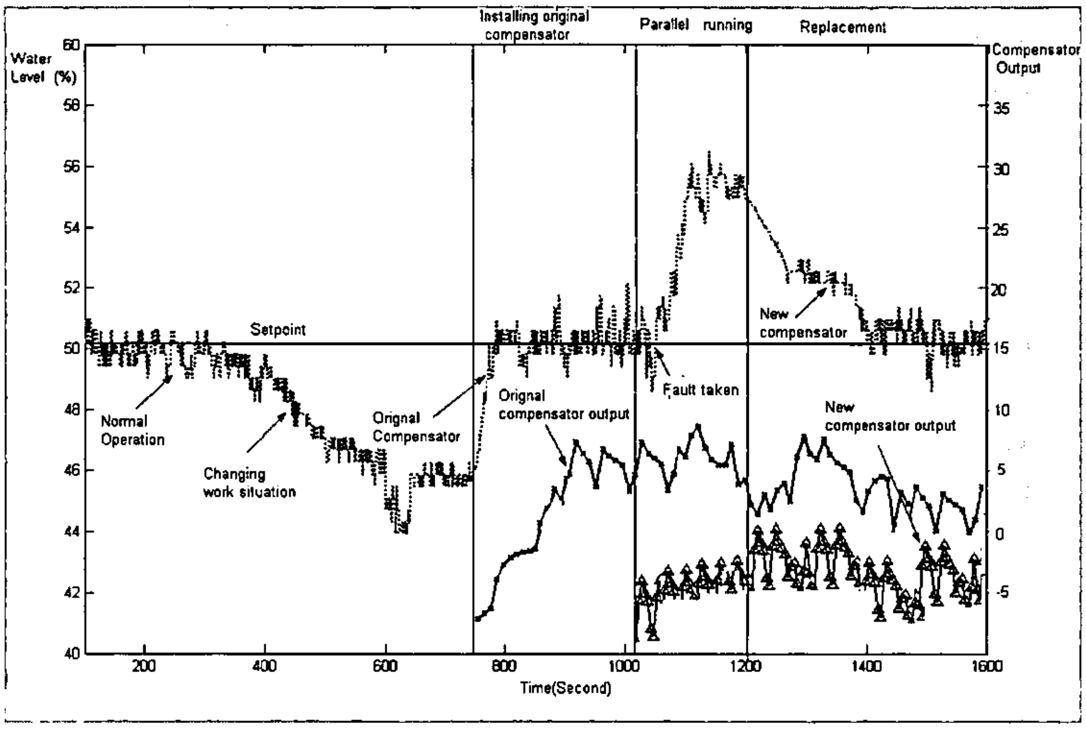


Figure 8.13. Record during updating operation.

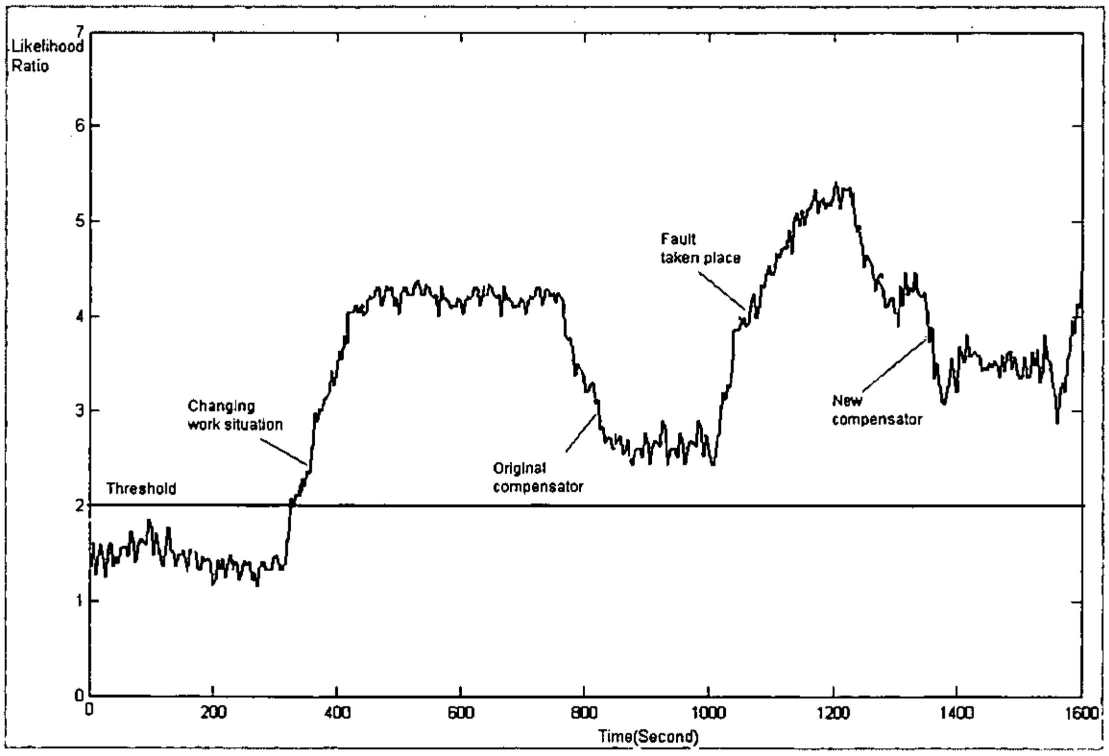


Figure 8.14. GLR test results

Experiment 2: Updating the running compensator

Experiment design

In this experiment, two faulty situations are designed and implemented. One is to change the water tank process model by altering the outlet flow rate at the instant 300. In the normal operation, the outlet flow rate of the water tank is constant. The MPC regulates the flow rate of the pump to maintain the desired water level. When the outlet flow rate is increased, the water tank model is changed and the original balance between the inlet flow rate controlled by the MPC and the outlet flow rate of the water tank is upset and more water will flow out; the MPC based on the original water tank model will misjudge the new running situation and generate inappropriate control signals to govern the new water tank process, which is considered as being "faulty". The water level is down from the setpoint as shown in Figure 8.13 at the instant before 750.

The GLR agent employs the GLR test to identify the discrepancies between the LQG control signals collected from the LQG loop agent and the MPC control signals collected from the data warehouse. The GLR test value shown in Figure 8.14 indicates that the GLR test value is across the predefined threshold after the instant 300, which implies that the running MPC controller is faulty. Once the likelihood ratio generated by the GLR test goes over the pre-defined threshold, the compensator design is activated.

The compensator is designed via model reference technique and is in an explicit model-following framework so that the dynamics of the post-fault closed-loop system, G_{loop} follows that of a given reference model, G_r, and the serial link compensator model can be obtained as follows, details see (Zhang and Jiang, 2003, Broussard and O'Brien,1980):

$$G_{com}(s) = S_{21}[sI - F_m]^{-1} G_m + S_{22} - I \qquad (8.1)$$

where I is a unit matrix, s is the Laplace transform symbol, F_m and G_m are system matrices of G_r, S_{21} and S_{22} are constant gain matrices calculated on the basis of G_r and G_{loop}.

The reference model G_r is given as $\frac{1.109}{s+0.6931}$ and a compensator model is obtained as $\frac{0.21s+3.313}{s+0.6931}$ in terms of Equation 8.1 and dispatched into the local process site to install at the instant 750. Figure 8.13 shows that the water level is back to the desired set point after the compensator is in place.

At the instant 1050, another fault is introduced into the PCU local control system, which is a gain, K= 8, attached in the output of the local MPC controller. Both faults are totally different and the running compensator can't cover the new fault as shown in Figure 8.13, the water level rises and deviates significantly from the setpoint. The GLR test value deviates away from the pre-defined threshold. To restore the degraded performance, a new compensator is designed and the compensator model is $\frac{0.2s\ -0.8}{s+0.7038}$, the new compensator is to replace the running the compensator to fix the faulty MPC controller.

Compensator Updating

Being different from the performance index agent updating, where the performance index agent class needs to reconfigure when a new service is added, the step of updating the management unit can be ignored in the compensator updating because both the compensators only have a single service, a compensation function. The procedure of the compensator class replacement is shown in Figure 8.15. Figure 8.15(a) shows the original compensator is implemented. When the new compensator is ready, the agent manager enables the new compensator to participate the control system, but the output of the new compensator does not drive the process as shown in Figure 8.15(b). Figure 8.15(c) shows that the new compensator replaces the original compensator, and the original compensator is disconnected with the water tank process control system.

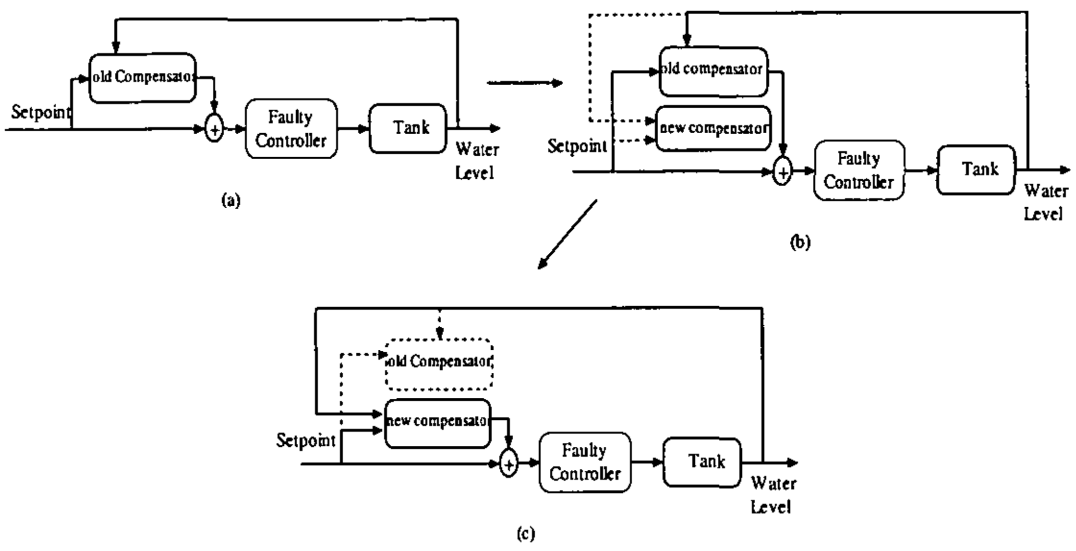


Figure 8.15. Compensator class replacement

Figure 8.13 illustrates the dynamic trends of a set of control system variables during the replacement. The original compensator is put in action at the instant 750. At the instant 1200 seconds, the new compensator substitutes the running original one to fix the new fault in the MPC. The result shows that the whole installation and updating is smoothly carried out and there is nearly no impact to the ordinary process operation and the degraded control performance can be recovered when the compensators are installed.

8.6 Conclusions

This chapter presents an approach to implement the mobile agent in the controller software remote maintenance, a mobile agent-based architecture is proposed, which enables distributed access, parallel computing, remote control performance monitoring, software state identification and maintenance. The distinguished feature of the proposed mobile agent application is that the mobile agent structure is flexible, and can be on-line remotely updated. As a consequence, maintenance centers can be created, which are operated on the web and provide central support to all the authorized industrial control systems; the faulty or outdated agent class can be remotely replaced or upgraded; therefore, the maintenance of the running agent is more easy and efficient. The case study illustrates what the mobile agent application environment looks like and how it can be implemented. The experimental results demonstrate the feasibility of the proposed mobile agent-based maintenance system.

Comparing to the traditional implementations, the main benefits of the proposed mobile agent-based application are three-fold. Firstly, the remote compensator is downloaded and installed in the local process site. Considering that the compensator must meet the real-time requirements and there is a potential large time delay over the Internet, remotely downloading and implementing the compensator minimize the impact of the Internet time delay on the controller software maintenance. This benefit is clear and has demonstrated by (Yang et al, 2003b). Secondly, the agent-based application can minimize the network usage. The traditional application needs to transfer the whole process data to the remote site in order to analyze the control software behaviours while the agent-based application only needs to send the analysis results to the remote site, which significantly reduces the data transfer load over the Internet. Finally, the agent-based application provides a general solution to the on-line updating of the travelling agent, new services can be on-line added into the running agents and does not affect the agent working as illustrated in Section 8.7.2 while the traditional application often needs to suspend the working application and then re-implements the interface or modifies the source code to implement the new service; the whole procedure in the traditional application will be complicated and off-line.

Chapter 9. Conclusions and Perspectives

There is a great deal of benefits for process industries to take up the Internet for their process plants and control systems and get them "Web-enabled". The technology to allow connection of process plants and control systems to the Internet is easily available now, but there is no real evidence that adequate consideration has been given to the maintenance of these systems, particularly to the controller software from remote specialized staff in a central location (technical head office, or system suppliers). In this thesis we have an insight into the remote maintenance of controller software over the Internet. Several techniques are proposed for specified fields; both theoretical work and case studies are conducted.

9.1 Conclusions of This Thesis

This thesis made contributions to controller software remote maintenance summarized as below:

(a). LQG Based Controller Performance Assessment

We have a deep investigation in the controller performance assessment, and develop an effective LQG based controller performance assessment approach. The optimal LQG cost function values are obtained directly from the solution of the Lyapunov equation and subspace matrices, which are identified based on system input/output data. A performance measure is proposed to assess the controller performance based on a comparison between the actual achieved objective functions and the LQG benchmark values.

(b). GLR Based Controller Fault Detection

Traditional fault detection techniques are not applicable to controller fault detection. In this work, the LQG controller is adopted to be used as the benchmark for controller fault detection. A baseline of controller normal operation is treated as the asymptotic distribution of the discrepancies. Fault detection characterized by sudden jumps in the mean/variance of the discrepancy is formulated as an online change point identification problem. A detector based on the GLR test is employed to on-line monitor the change point.

(c). Controller Failure Maintenance

To accommodate the controller failure, two such methods designing a compensator to maintain the controller failure are proposed: one is the use of the fault shift-based method to design a compensator; the other is the use of the model following technique to design a compensator. The compensator is designed and inserted into the post-fault control loop so that the dynamic performance of the post-fault control loop is recovered into an acceptable range.

(d). Heavy Data Process and Transfer over the Internet

Features and challenges of real time data transfer over the Internet are investigated and a data transfer mechanism over the Internet for real time web-based applications is proposed. The mechanism incorporates the eXtensible Markup Language (XML) and Hierarchical Data Format (HDF) to provide a flexible and efficient data format. The heterogeneous transfer data are classified into light data and heavy data, which are stored by XML and HDF respectively; the HDF data format is then wrapped by XML in the Java environment. The wrapped HDF objects and XML objects are sent across computer networks with the support of the Java Remote Method Invocation (RMI) data transfer infrastructure.

(e). Platform Prototype for Controller Software Remote Maintenance

A methodology for the design of the Internet-based controller software maintenance system is developed. The design issues include remote control performance assessment and recovery, remote controller software state identification, and data process and transfer over the Internet. Based on this methodology, an Internet-based controller software maintenance system is established and implemented into a process control unit as a case study.

(f). Implementing Mobile Agents into Controller Software Remote Maintenance

The integration of mobile agent technique and the controller software remote maintenance is explored. A mobile agent-based system for controller software remote maintenance is established. The traveling mobile agents can be remotely updated over the Internet. The running agent structure can be on-line reconfigured; original services can be upgraded and new services can be on-line added into and implemented the traveling agents.

9.2 Perspectives

Based on the work presented in this thesis, we realize that further research on the following aspects of controller software remote maintenance should or could be carried out regarding both theoretical work and applications.

(a). Robust Fault Detection

One of important properties for applicability of change detection algorithms is the robustness. An effective fault detection algorithm should be robust to the impact of disturbance and faults of other components in the control system, such as actuator and sensor faults. A further investigation regarding this issue may need to be carried although a brief discussion in Chapter 4 is given. On the other hand, the proposed detection algorithm in Chapter 4 leaves an issue to the system designer to choose appropriate values for these thresholds of the detection algorithm. A systematic method to choose suitable thresholds needs to be considered.

(b). Compensator Design Considering Uncertainties

Further research work can be conducted to study how to those compensator design algorithms proposed in Chapter 5 to handle uncertainties and/or under disturbances. Particularly, the robust compensator design should take account of the allowable system performance in the event of controller failures and the uncertainties. Also, it needs to investigate the suitable time to insert the compensator into the faulty control loop in respect to stability of the control loop.

(c). Security in the Remote Maintenance of Controller Software

It is necessary to keep in mind that it has to be secured against attacks from outside for an Internet-based maintenance system. The Internet technologies make it possible to connect these systems with control and information networks; however, using a connection to an open network and the use of universal technology causes new problems. The most serious of those new problems concern data security. Hackers constantly invent new methods to get access of services to cause a lot of damage. Unauthorized access, wiretapping, system failures caused by viruses and denial of services due to network and server overload are most used to cause a lot of damage.

(d). Real Time Remote Monitoring and Control

One of possible prerequisites for implementing the Internet-based maintenance application is to make it enable to real time monitor and maintain the remote control systems. Internet-based applications have a variable working load under limited traffic resources of the Internet, especially in real time operations. Although several techniques are proposed and result in some successful applications, little work is found to propose an effective and systematic solution to tackle this issue, which has to be considered in Internet-based applications.

At last, we hope that the work presented in this thesis will stimulate further research in the field of controller software remote maintenance and its applications.

References

1. Abran, A., and Nguyenkim, H., "Measurement of the Maintenance Process from a Demand-based Perspective," Journal of Software Maintenance: Research and Practice, Vol. 5, No. 2, pp. 63-90, (1993).
2. Ahmed-Zaid, F., loannou,P., and Gousman,K., "Accommodation of Failures in F-16 Aircraft Using Adaptive Control", IEEE control systems Magazine, Vol. 11, No. 1, pp. 73-78, (1991).
3. Ahn, S. H., and Chu, W. S., "Internet-based Aircraft Composite Repair", Korea-Japan Joint Symposium on Composite Materials, October 10, Yamaguchi City, Japan, (2002).
4. Aktan, B., Bohus, C.A., Crowl, L.A., and Shor, M.H., "Distance Learning Aapplied to Control Engineering Llaboratories", IEEE Transactions on Education, Vol. 39, No. 3, pp. 320-326, (1996).
5. "All about HDF 5", Available at: http://hdf.ncsa.uiuc.edu, (2000).
6. Almeida, D., "A Look at HDF/HDF-EOS Data Formats from a Remote Sensing Perspective", Available at: http://www.geography.hunter.cuny.edu/~dalmeida/, (2003).
7. Anderson, R. J., and Spong, M. W., "Bilateral Control of Teleoperators with Time Delay", IEEE Transactions on Automatic Control, Vol. 34, No. 5, pp. 494–501, (1989).
8. Angeli, C., and Chatzinikolaou, A., "On-Line Fault Detection Techniques for Technical Systems: A Survey", International Journal of Computer Science & Applications Vol. 1, No. 1, pp. 12-30, (2004).
9. "ARPA History", Available at: http://www.computermuseum.li/, (1999)
10. Arioui, H., Kheddar, A., and Mammar, S., "A Predictive Wave-based Approach for Time Delayed Virtual Environments Hepatics Systems", Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication, Berlin, Germany, pp. 134-139, (2002).
11. Atherton, R., "Java Object Technology can be Next Process Control Wave", Control Engineering, Vol. 45, No. 13, pp. 81-85, (1998).
12. Bangolae, S., Jayasumana, A.P., and Chandrasekar, V., "Gigabit Networking: Digitized Radar Data Transfer and Beyond", Proceedings IEEE International Conference on Communications (ICC'03), pp. 684-688, (2003).
13. Bao, J., Zhang, W. Z. and Lee, P.L., "Decentralized Fault-tolerant Control System Design for Unstable Processes", Chem. Eng. Sci. Vol. 58, No. 22, pp. 5045-5054, (2003).
14. Barbara, S., "PID Controller Performance Assessment Based on Closed-Loop Response Data", Ph. D thesis Chemical Engineering, University of California, USA, (1999).

15. Barkstrom, B. R., "Ada 95 Bindings for the NCSA Hierarchical Data Format", Proceedings ACM SIGAda Annual International Conference (SIGAda 2001), pp. 27-30, (2001).

16. Bartels, R.H., and Stewart, G.W., "Solution of the Matrix Equation AX + XB = C", Comm. of the ACM, Vol. 15, No. 9, pp. 820-826, (1972).

17. Basseville, M., and Nikiforov, I., "Detection of Abrupt Changes - Theory and Application", Prentice-Hall, (1998).

18. Basila, M., Jr., Stefanek, G., & Cinar, A., "A Model-object Based Supervisory Expert System for Fault Tolerant Chemical Reactor Control", Computers and Chemical Engineering, Vol. 14, No. 45, pp. 551-560, (1990).

19. Basu, S., Mukherjee, A., and Klivansky, S., "Time Series Models for Internet Traffic", Proceedings of IEEE Infocom Conference, San Francisco, pp. 164–171, (1996).

20. Becraft, W., and Lee, P., "An Integrated Neural Network/Expert System Approach for Fault Diagnosis", Computers and Chemical Engineering, Vol. 17, No. 10, pp. 1001-1014, (1993).

21. Bellamine, M., Abe, N., Tanaka,K., Chen,P., Taki, H., "A Virtual Reality Based System for Remote Maintenance of Rotating Machinery" IEICE-Transactions on Info and Systems, Vol. E88-D, No. 5, pp. 894-903, (2004).

22. Bennett, S., "Real-Time Computer Control: An Introduction", First Edition, Prentice Hall, (1988).

23. Bertino, E., and Sandhu, R., "Database security - Concepts, Approaches, and Challenges", IEEE Transactions on Dependable and Secure Computing, Vol. 2, No. 1, pp. 2-19, (2005).

24. Bertrand, E., Roch H., and Glitho, S. P., "A Mobile Agent-Based Advanced Service Architecture for Wireless Internet Telephony: Design, Implementation, and Evaluation", IEEE Transactions on Computers, Vol. 52, No. 6, pp. 690-704, (2003).

25. Bhattacharyya, S., Kurose, J. F., Towsley, D., and Nagarajan, R., "Efficient Rate-Controlled Bulk Data Transfer Using Multiple Multicast Groups", Proc. IEEE INFOCOM'98, San Francisco, pp. 895-907, (1998).

26. Bishop,S., Fairbairn, W., Norrish, M., Sewell, P., Smith, M., and Wansbrough, K., "The TCP Specification: A Quick Introduction", University of Cambridge Computer Laboratory, Technical Report , (2004).

27. Bishop, S., Fairbairn, W., Norrish, M., Sewell, P., Smith, M., and Wansbrough, K., "TCP, UDP, and Sockets: Rigorous and Experimentally-validated Behavioral Specification.

Volume 1: Overview", University of Cambridge Computer Laboratory, Technical Report, (2005a).

28. Bishop,S., Fairbairn, W., Norrish, M., Sewell, P., Smith, M., and Wansbrough, K., "Rigorous Specification and Conformance Testing Techniques for Network Protocols, as Aapplied to TCP, UDP, and Sockets", SIGCOMM, pp. 265-276, (2005b).

29. Blanke, M., Staroswiecki, M., and Wu, N. E., "Concepts and Methods in Fault-tolerant Control", Tutorial at American Control Conference, pp. 2606–2620, (2001).

30. Bloch,G, Theillol,D., and Thomas, P., "Simultaneous Detection, Location and Identification of Faults for Dynamic Systems", In IFAC Fault Detection, Supervision and Safety for Technical Processes, Espoo, Finland , pp. 47-51,(1994).

31. Branchaud, M., "A Survey of Public Key Infrastructures", Master's Thesis, Department of Computer Science, McGill University, Montreal, (1997).

32. Broussard, A. R., and O'Brien, M. J., "Feedforward Control to Track the Output of A Forced Model". IEEE Transactions on Automatic Control, Vol. 25, No. 4, pp. 851–853, (1980).

33. Bugliesi, M., Colazzo, D., Crafa, S., "Type Based Discretionary Access Control", CONCUR 2004 - Concurrency Theory, 15th International Conference. Proceedings (Lecture Notes in Computer. Science. Vol.3170). Berlin, Germany: Springer-Verlag, pp. 225-39, (2004).

34. Burl, J., "Linear Optimal Control: H_2 and H_∞ Methods", Addison-Wesley, (1999).

35. Buse, D. P., and Wu, Q. H., "Mobile Agents for Remote Control of Distributed Systems", IEEE Transactions on Industrial Electronics, Special Issue on Distributed Network-Based Control Systems and Applications, Vol. 51, No. 6, pp. 1142-1149, (2004).

36. Bykat, A., "Intelligent Monitoring and Diagnosis Systems: A Survey", Applied Artificial Intelligence, Vol. 5, No. 4, pp. 339-352, (1991).

37. Cabri, G., Leonardi, L., and Zambonelli, F., "Agents for Information Retrieval: Issues of Mobility and Coordination", Journal of Systems Architecture, Vol. 46, No. 15, pp. 1419-1433, (2000).

38. Camarinha-Matos, L. M., Vieira, W. J., and Castolo, O., "Mobile Agents Approach to Virtual Laboratories and Remote Supervision", Journal of Intelligent and Robotic Systems, Vol. 35, pp. 1-22, (2002).

39. Capra. L., Mascolo, C., Zachariadis, S., and Emmerich, W., "Towards A Mobile Computing Middleware: A Synergy of Reflection and Mobile Code Techniques", 8th

IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'01), Bologna, Italy, pp. 148-154, (2001).

40. Celi, R. Huang, C.Y and Shih, I.C., "Reconfigurable Flight Control for A Tandem Rotor Helicopter", 52nd Annual National Forum of the American Helicopter Society, USA, pp. 1569-1588, (1996).

41. "CFD", http://www.cgns.org/WhatIsCGNS.html, (2002).

42. Chen, G., McAvoy T.J. and Piovoso, M.J., "Multivariate Statistical Controller for on-line Quality Improvement", Journal of Process Control, Vol. 8, No. 2, pp. 139-149, (1998).

43. Chen, L., and Modarres, M., "Hierarchical Decision Process for Fault Administration", Computers and Chemical Engineering, Vol. 16, No. 5, pp. 425- 448, (1992).

44. Cheung, J. T., and Stephanopoulos, G., "Representation of Process Trends part I. A Formal Representation Framework", Computers and Chemical Engineering, Vol. 14, No. 45, pp. 495-510, (1990).

45. Chiang, L.H., Russell, E.L., and Braatz, R.D., "Fault Detection and Diagnosis in Industrial Systems", Springer, (2001).

46. Clarke, J. A., Hare, J. J., and Brown, J. D., "Implementation of a Distributed Data Model and Format for High Performance Computing Applications", Available at: http://www.hpcmo.hpc.mil/Htdocs/UGC/UGC00/paper/jerry_clarke_paper.pdf, (2000).

47. Copinga, G., Verhaegen, M., and van de Ven,M., "Toward a Web-based Study Support Environment for Teaching Automatic Control," IEEE Control System Magazine. Vol. 20, August, pp. 8-19, (2000).

48. Collings, I.B., Krishnamurthy, V., and Moore, J.B., "On-line Identification of Hidden Markov Models via Recursive Prediction Error Techniques", IEEE Transactions on Signal Processing, Vol. 42, No. 12, pp. 3535-3539, (1993).

49. Cybenko, G., Gray, Robert R., Wu, Y., and Khrabov, A., "Information Architecture and Agents", Available at: http://citeseer.ist.psu.edu/cybenko94information.html. (1994)

50. Dai, C., and Yang, S., "An Approach for Control Software Supervision'', Proceedings of the 9th Chinese Automation & Computing Society Conference in the UK, pp. 41-46, (2003).

51. Dai, C., Yang, S.H., and Tan, L., "An Approach for Controller Fault Detection", Proceedings of the 5th World Congress on Intelligent Control and Automation, IEEE, Hangzhou, China, pp. 1637-1641, (2004a).

52. Dai, C., and Yang, S.H., "Maintaining Control Performance in Faulty Control Systems, IEEE International Conference on Systems", Man & Cybernetics, W. Thissen, W.,

Wieringa, P., Pantic, M. and M. Ludema, M. (eds), IEEE, IEEE International Conference on SMC , The Hague, The Netherlands, pp. 5074-5078, (2004b).

53. Dayal, B.S., and MacGregor, J.F., "Identification of Finite Impulse Response Models: Methods and Robustness issues", Ind. & Eng. Chem. Res., Vol. 35, No. 11, pp. 4078-4090, (1996).

54. Dix, A., "Design of User Interfaces for the Web", IEEE Computer Society, pp. 2-11, (1999).

55. Dong, D., and McAvoy, T.J., "Nonlinear Principal Component Analysis Based on Principal Curves and Neural Networks", Proc.of the American Control Conf, pp. 1284-1288, (1994).

56. Dreyer, T., et al., "ScadaOnWeb-Web Based Supervisory Control and Data Acquisition", 17th International Conference on Electricity Distribution, pp. 1-5, (2003).

57. Favoreel, W., and Moor,D., "SPC: Subspace Predictive Control", Katholieke Universiteit, Leuven. SE-581 83 Link ping, Sweden, Technical report, pp. 98-49, (1998a).

58. Favoreel, W., De Moor, B., Gevers, M., and van Overschee, P., "Model-free Subspace Based LQG-Design", Katholieke Universiteit, Leuven, Technical report, pp. 98-34, (1998b).

59. Furuya, M., Kato, H., and Sekozawa, T., "Secure Web-based Monitoring and Control System, Industrial Electronics Society", IECON 2000, 26th Annual Conference of the IEEE, pp. 2443-2448, (2000).

60. Ganjefar, S., Momeni, H., and Janabi-Sharifi, F., "Teleoperation Systems Design Using Augmented Wave-variables and Smith Predictor Method for Reducing Rime-delay Effects", Proceedings of the International Symposium on Intelligent Control, Vancouver, BC, Canada, pp. 333-338, (2002).

61. Garetto, M., and Towsley, D., "Modelling, Simulation and Measurements of Queuing Delay under Long-Tail Internet Traffic", ACM Sigmetrics 2003, San Diego, CA, pp. 47-57, (2003).

62. "Gaussian distribution", Available at: http://farside.ph.utexas.edu/teaching/sm1/lectures/node20.html, (2005).

63. Geladi, P., and Kowalski, B.R., "PLS Tutorial", Anal. Chim. Acta. Vol. 185, No. 1, pp. 1-17, (1986).

64. Glitho, R. H., Olougouna, E., and Pierre, S., "Mobile Agents and Their Use for Information Retrieval: A Brief Overview and an Elaborate Case Study", IEEE Network Magazine, Vol. 16, No. 1, pp. 34-41, (2002).

65. "Gopher", Available at: http://www.hvcn.org/info/gswc/primer/gopher.htm, (1998).

66. Graham, C., Goodwin, S., Graebe, F., and Salgado, E., "Control system Design", Prentice Hall, (2000).

67. Gray, B., "Introduction to Mobile Agents: Performance, Security and Programming Examples", Available at: http://www.vs.inf.ethz.ch/ASA-MA/tutorials.html#mobile, (2000).

68. Gray,R.S., Cybenko, G, Kotz, D., Peterson, R.A., and Rus, D., "Applications and Performance of a Mobile-Agent System." Software--Practice and Experience, Vol. 32, No. 6, pp. 543-573, May, (2002).

69. "HDF5", Available at: hdf.ncsa.uiuc.edu/HDF5/RD100-2002/All_About_HDF5.pdf, (2002).

70. "HDF5 Compound Data: Technical Issues for XML, Java, and Tools", Available at: http://hdf.ncsa.uiuc.edu/HDF5/XML/tools/compound-data.html, (2002).

71. Hajji, H., Far., B., and Cheng, H., "Detection of Network Faults and Performance Problems", Available at http://www.internetconference.org/ic2001/papers/,(2001) .

72. Halley, A., and Gauld, P., "Integration Where Java Meets Process Control", Control and Instrumentation, Vol.31, No.4, pp. 57-58, (1999).

73. Han, K., Kim, H., Kim, S., and Kim, J. H., "Internet Control Architecture for Internet-based Personal Robot", Autonomous robots, Vol.10, No.2, pp. 135-147, (2001).

74. Hannaford, B., "A Design Framework for Teleoperators with Kinaesthetic Feedback", IEEE Transactions on Robotics and Automation, Vol.5, No.4, pp. 426-434, (1989).

75. Harris T., "Assessment of Control Loop Performance." The Canadian Journal of Chemical Engineering, Vol.67, pp. 856-861, (1989).

76. Harris, T.J., Boudreau, T., and Macgregor, J.F., "Performance Assessment of Multivariable Feedback Controllers", Automatica, Vol. 32, pp. 1503-1517, (1996).

77. Harris, T.J., Seppala, C.T., and Desborough, L.D., "A Review of Performance Monitoring and Assessment Techniques for Univariate and Multivariate Control Systems", Journal of Process Control, Vol. 9, pp. 1–17, (1999).

78. Hayton,P., Sch¨olkopf,B., Tarassenko, L., and Anuzis, P., "Support vector novelty detection applied to jet engine vibration spectra," in NIPS'2000, pp. 2479-2487 , (2000).

79. Hencko, H., Juric, M., Zivkovic, A., Rozman, I., Domajnko, T., and Krisper, M., "Java and Distributed Object Models: An Analysis", ACM SIGPLAN, Notices, Vol. 33, No. 12, pp. 57-65, (1998).

80. Henriksson,D., Lu,Y., and Abdelzaher,T., "Improved Prediction for Web Server Delay Control", 16th Euromicro Conference on Real-Time Systems, Catania, Italy, pp. 61-68, (2004).

81. Henry, S., and Kafura, D., "The Evaluation of Software Systems Structure Using Quantitative Software Metrics", Software practice and Experience, Vol. 14, No. 6, pp. 561-573, (1984).

82. Hiromitsu, K., Furuya,M., and Tamano,M., "Risk Analysis and Secure Protocol Design for WWW-Based Remote Control with Operation-Privilege Management", SMC 2001 IEEE e-Systems and e-Man for Cybernetics in Cyberspace, Vol. 2 , No. 7, pp. 1107-1112 , (2001).

83. Huang, B., S.L. Shah, E.Z., and Zurcher, J., "Performance Assessment of Multivariate Control Loops on a Paper-Machine Headbox", Canadian Journal of Chemical Engineering, Vol.75, February, pp. 134-142, (1997a).

84. Huang, B., Shah, S.L., and Kwok, E.Z., "Good, Bad or Optimal? Performance Assessment of Multivariable Process", Automatica, Vol. 33, No. 6, pp. 1175-1183, (1997 b).

85. Huang, B., and Shah, S.L., "Performance Assessment of Control Loops: Theory and Applications", Springer Verlag, New York, NY, (1999).

86. Hori, S., and Shimizu, Y., "Designing Methods of Human Interface for Supervisory Control Systems", Control Engineering Practice, Vol.7, pp. 1413-1419, (2001).

87. Hugo, A., "Performance Assessment of Process Controllers", Available at: www.controlartsinc.com/patent3.htm, (2000).

88. Ingimundarson, A., "Dead Time Compensation and Performance Monitoring in Process Control", Lund Institute of Technology, Sweden, PhD Thesis, (2003).

89. "Internet", Available at: en.wikipedia.org/wiki/Interne,(2006).

90. "Internet's Tomorrow", Available at: http://www.insead.fr/IVC/Guide/Tomorrow/ (2003).

91. "Internet History", Available at: http://www.walthowe.com/navnet/history.html, (2005).

92. "Internet Pioneers", Available at: http://www.ibiblio.org/pioneers/, (2000).

93. "Intuitive Technology Corp", Available at: http://www.aglance.com, (1999).

94. Isaksson A.J., "PID Controller Performance Assessment", Proceedings of the 1996 Control Systems Conference, Halifax Can., pp. 163-169, (1996).

95. Isermann,.R " Fault Diagnosis of Machines via Parameter Estimation and Knowledge processing tutorial paper", Automatica, Vol. 29, No. 4, pp. 815-835, (1993).

96. Iung, B., "From Remote Maintenance to MAS-based E-maintenance of An Industrial Process",
Available at: http://www.ingentaconnect.com/content/klu/jims/2003/00000014/00000001, (2003).

97. Iung, B., and Morel, G., "Towards Intelligent Maintenance Integrated within the Enterprise", MCPL'2000, 2nd IFAC-IFIP-IEEE Conference on Management and Control of Production and Logistics. Invited Lecture. 5–8 July, Grenoble, France, pp. 414–420, (2000).

98. Iung, B., Muhl, E., Leger, J.B., Morel, G., "MAS Modelling Paradigm Application for the Development of A Proactive Maintenance Intelligent System", 1st IFAC-MAS'99 Workshop on Multi-Agent-Systems in Production, December 2–4, Vienna, Austria, pp. 89–96, (1999).

99. Joshi, J., Aref, W., Ghafoor, A., Spafford, E., "Security Models for Web-based Applications", Communications of the ACM, Vol. 44, Issue 2, pp. 38-44, (2001).

100. Janusz, M., and Venkatasubramanian, V., "Automatic Generation of Qualitative Description of Process Trends for Fault Detection and Diagnosis", Engineering Applications of Artificial Intelligence, Vol. 4, No. 5, pp. 329-339, (1991).

101. Jezic, G, Kusek, M., Marenic, T., Ljubi, T., Lovrek, I., Desic, S., and Dellas, B., "Mobile Agent-Based Software Management in Grid", 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'04), pp. 345-346, (2004).

102. Jiang, J., "Design of Reconfigurable Control System Using Eigenstructure Assignment," International Journal of Control, Vol. 59, No. 2, pp. 395-410, (1994).

103. Johansen, D., "Mobile Agent Applicability", Proceedings of the Second International Workshop on Mobile Agents, Lecture Notes in Computer Science, pp. 80-98, (1998).

104. Juricek, B., Seborg, D.E., and Larimore, W.E., "Predictive Monitoring for Abnormal Situation Management", Journal of Process Control, Vol. 11, pp.111-128, (2001).

105. Juric, B., Matjaz, R. I., Hericko, M., "Performance Comparison of CORBA and RMI", Information and Software Technology Journal, Elsevier Science, Vol. 42, No. 13, pp. 915-933, (2000).

106. Karsai, G., Biswas, G., Abdelwahed, S., Narasimhan,S., and Pasternak,T. T., "Towards Fault-Adaptive Control of Complex Dynamic Systems", Software Enabled Control, Book, (2002).

107. Kasimir, K., Pieck, M., and Dalesio, L., "Integrating LabVIEW into a Distributed Computing Environment," ICALEPCS'01, San Jose, CA, USA, Nov, pp. 27-30, (2001).

108. Kawai, K., "An Intelligent Multimedia Human Interface for Highly Automated Combined-cycle Plants", Control Engineering Practice, Vol. 5, No. 3, pp. 401-406, (1997).

109. Kemmerer, R., and Vigna, G., "Intrusion Detection: A Brief History and Overview", Security & Privacy, Supplement to IEEE Computer Magazine, pp. 27-30, (2002).

110. Kerrigan, E. C., and Maciejowski, J. M., "Fault-Tolerant Control of a Ship Propulsion System using Model Predictive Control", Proceedings of the European Control Conference, Karlsuhe, Germany, Available at: http://www-control.eng.cam.ac.uk/Homepage/papers/cued_control_73.pdf,(1999).

111. Kikuchi, J., Takeo, K., and Kosuge, K., "Teleoperation System via Computer Network for Dynamic Environment", Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Leuven, Belgium, pp. 3534–3539, (1998).

112. Koç, M. and Lee, J., (2001). "A System Framework for Next-Generation E-Maintenance Systems", http://www.uwm.edu/CEAS/ims/pdffiles/E-Maintenance.PDF.

113. Konno, T., and Tateoka, M., "Accuracy Improvement of Anomaly-based Intrusion Detection System Using Taguchi method," Proceedings. The 2005 Symposium on Applications and the Internet Workshops. Los Alamitos, CA, USA: IEEE Computer Society, pp. 90-3, (2005).

114. Konstantinov, K. B., and Yoshida, T., "Real-time Qualitative Analysis of the Temporal Shapes of the Process Variables". American Institute of Chemical Engineers Journal, Vol. 38, No. 11, pp. 1703- 1715, (1992).

115. Kozub, D. and Garcia C., "Monitoring and Diagnosis of Automated Controllers in the Chemical Process Industries." In Proc. AIChE. St.Louis, (1993).

116. Kramer, M.A., "Auto associative neural networks", Comp. Chem. Eng., Vol. 16, No. 4, pp. 313-328, (1992).

117. Kresta, J.V., MacGregor, J.F., and Marlin, T.E., "Multivariate Statistical Monitoring of Process Operating Performance", Can. J. Chem. Eng., Vol. 69, pp. 35-47, (1991).

118. Kumamaru, K., and Soderstron, T., "Fault Detection and Model Validation Using of Kull-back Discrimination Information", Trans. Of Society Instrument and Control Engineers, Vol. 22, No. 10, pp. 1135-1140, (1986).

119. Kusiak, A., "International Journal of Intelligent Manufacturing. Special issue on Internet-Based Distributed Intelligent Manufacturing System" (Banaszak, E. (Eds.), Vol. 14, No. 1, January, (2003).

120. Kussel, R., Liestmann, V., Spiess, M., and Stich, V., "Teleservice a Customer Oriented and Efficient Service", Journal of Material Processing Technology, vol. 107, pp. 363–371, (2000).

121. Lau, C., Churchill, S., Kim, J., and Y. Kim, "Web-based Home Telemedicine System for Orthopaedics", Proceedings of SPIE Medical Imaging 2001, vol. 4319, pp. 693-698, (2001).

122. Laurent, H., and Doncarli, C., "Stationary Index for Abrupt Changes Detection in the Time-frequency Plane," IEEE Signal Processing Letters, Vol. 5, No. 2, pp. 43 – 45, (1998).

123. Ledoux, T., and Bouraqadi, N., "Adaptability in Mobile Agent Systems using Reflection", RM'2000, Workshop on Reflective Middleware, Available at http://www.comp.lancs.ac.uk/computing/rm2000/, (2000).

124. Lee, J., "Strategy and Challenges on Remote Diagnostics and Maintenance for Manufacturing Equipment", Proceedings of the Annual Symposium on Reliability and Maintainability, pp. 368–370, (1997).

125. Lehti, P. and Fankhauser P., "XML Data Integration with OWL: Experiences and Challenges", SAINT 2004, pp.160-170, (2004).

126. Leal, D.,Schröder, A., and Schwan, M., "ScadaOnWeb - Web Based Supervisory Control and Data Acquisition",
Available at : http://www.scadaonweb.com/publication.html, (2002).

127. Leonard, F., "An IAE Optimum Control Tuned by One Parameter", ECC 97, Brussels, Belgium, Available at: www.cds.caltech.edu/conferences/ related/ECC97, (1997).

128. Ljung. L., "System Identification. Theory for the User". N.J, Prentice Hall, Englewood Cliffs, (1987).

129. Lovrek, I., Jezic, G, Kusek,M., Ljubi,I.,Caric, A., Huljenic,D., Desic, S., and Labor,O., "Improving Software Maintenance by using Agent-based Remote Maintenance Shell", Available at: http://doi.ieeecomputersociety.org/10.1109/ICSM.2003.12354. (2003).

130. Lozano, R., Chopra, N., and Spong, M. W., "Passivation of Force Reflecting Bilateral Teleoperators with Time Varying Delay", Proceedings of Mechatronics Conference, Entschede, the Netherlands, pp. 24-26, (2002).

131. Lunt, T. F., "A Survey of Intrusion Detection Techniques", Computers and Security Vol. 12, No. 4, pp. 405-418, (1993).

132. Luo, R. C., and Chen, T.M., "Development of a Multibehaviour-based Mobile Robot for Remote Supervisory Control through the Internet", IEEE transactions on mechatronics, Vol. 5, No. 4, pp. 376-385, (2000).

133. Lynch,C.B., and Dumont ,G.A., "Control Loop Performance Monitoring", IEEE transactions on control systems technology ,Vol. 4 ,No. 2, pp. 185-192, (1996).

134. MathML, Available at: http://www.w3.org/Math/, (2000).

135. Mahund, M., Jinag, J., and Zhang,Y., " Active Fault tolerant Control Systems", Springer, (2003).

136. Masri, S. F., Sheng, L-H., Caffrey , J, P., Nigbor ,R. L., Wahbeh ,M., and Abdel-Ghaffar, A. M., "Application of a Web-enabled Real-time Structural Health Monitoring System for Civil Infrastructure Systems", Smart Materials and Structures, Vol. 13, Issue 6, pp. 1269-1283, (2004).

137. MATLAB Support". Available at:
http://csd.newcastle.edu.au/control/appendices/appendixe-node1.html, (2000).

138. McGrath, R., "XML and Scientific File Formats", Available at:
http://www.ncsa.uiuc.edu/NARA/XML_and_Binary.pdf, (2003).

139. Michelson, et al., "HDF5 information model and implementation specification for weather radar data", Available at:
http://www.smhi.se/cost717/doc/WDF_02_200204_1.pdf, (2003).

140. Mirfakhrai, T., and Payandeh,S., "A Delay Prediction Approach for. Teleoperation over the Internet," in Proc. IEEE Int. Conf. Robotics and Automation, pp.2178-2183, (2002).

141. Morató, D., Aracil, J., Díez, L.A., and Izal, M., "On Linear Prediction of Internet Traffic for Packet and Burst Switching Networks", Proceedings International Conference on Computer Communications and Networks (ICCCN 2001), Scottsdale, Arizona, USA, pp. 138-143, (2001).

142. Naeem, W., Sutton, R., and Ahmad, S. M., "LQG/LTR Control of an Autonomous Underwater Vehicle Using a Hybrid Guidance Law", Proceedings of GCUV'03 conference, pp. 35-40, (2003).

143. Nam, B., and Sussman, A., "Improving Access to Multi-dimensional Self-describing Scientific Datasets", Available at: http://csdl.computer.org/comp/ proceedings, (2003).

144. Niemeyer, G., and Slotine, J., "Stable Adaptive Teleoperation", IEEE Journal of Oceanographic Engineering, Vol. 16, No. 1, pp. 152-162, (1991).

145. Niemeyer, G., "Using Wave Variables in Time Delayed Force Reflecting Teleoperation", PhD thesis, MIT, Cambridge, MA, (1996).

146. Niemann, H., and Stoustrup J., "Controller Reconfiguration based on LTR Design", 42nd IEEE Conference on Decision and Control, Hawaii, USA, pp. 2453-2458, (2003).

147. "NIFTI", Available at: http://www.bic.mni.mcgill.ca/nifti/, (2003).

148. Nomikos, P., and MacGregor, J.F., "Multivariate SPC Charts for Monitoring Batch Processes", Technometrics, Vol. 37, No. 1, pp. 44-59, (1995).

149. Nyberg, M., "Model Based Fault Diagnosis Methods, Theory, and Automotive Engine Applications", Department of Electrical Engineering Linkoping University, Sweden, (1999).

150. OWL, Available at: http://www.w3.org/TR/owl-features/, (2001).

151. "OPC Foundation", "OLE for Process Control, OPC Overview", Available at: http://www.opcfoundation.org, (1998).

152. Ougrinovski., U., and Petersen, I., "Robust Control System Design via Minimax LQG Control Theory", Available at: http://www.ee.adfa.edu.au/School/research/LQG, (2003).

153. Ozcelik, S., and Kaufman ,H., "Design of Robust Direct Adaptive Controllers for SISO Systems: Time and Frequency Domain Design Conditions", International Journal of Control, Vol. 72, No. 6, pp. 517-530, (1999).

154. Patwardhan, R.S., and Shah, S.L., "Performance Analysis of Model Predictive Controllers: An Industrial Case Study", AIChE Annual Meeting, Miami, FL, Available at: http://www.ualberta.ca/dept/chemeng/control/fulllist.html, (1998).

155. Patwardhan, R. S., "Studies in the Synthesis and Analysis of Model Predictive Controllers", PhD Thesis, Department of Chemical and Materials Engineering, University of Alberta, (1999).

156. Patwardhan, R., Shah, S. L., and Qi, K., "Assessing the Performance of Model Predictive Controllers", Canadian Journal of Chemical Engineering, Available at: pubs.nrc-cnrc.gc.ca/cjche/ch80954-5.html, (2002).

157. Pekilis. B.R., and Seviora, R.E., "Detection of Response Time Failures of Real-Time Software," Eighth International Symposium on Software Reliability Engineering (ISSRE '97) Albuquerque, NM, November 02 - 05, pp. 38-47, (1997).

158. Poindexter, S. E., and Heck, B. S., "Using the Web in Your Courses: What can you do? What should you do?" IEEE Control System, Vol. 19, No. 1, pp. 83–92, (1999).

159. Pradosh, K.M., "Public Key Cryptography", Available at: http://portal.acm.org/, (2001).

160. Qin, J. S., "Control Performance Monitoring-A Review and Assessment", Computers and Chemical Engineering, Vol. 23, pp. 173-186, (1998).

161. Qiu, B., and Gooi,H.B, "Web Based SCADA Display Systems (WSDWS) for access via internet," IEEE Trans. Power System., Vol. 15, pp. 681-686, (2000).

162. RDF, Available at: http://www.w3.org/RDF/, (2001).

163. Rizzoni, G., and Min P.S., "Detection of Sensor Failures in Automotive Engines", IEEE Trans. on Vehicular Technology, Vol. 40, No. 2, (1999).

164. Rovetta, A., Sala,R., Wen, X., and Togno. A., "Remote Control in Telerobotic Surgery", IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, Vol. 26, No. 4, pp. 438-443, (1996).

165. Rothermel, K. Hohl, F., "Mobile Agents", Second International Workshop, MA'98, Stuttgart, Germany, September 1998, Proceedings Springer, (1999)

166. Roychoudhuri L., and Al-Shaer,E., "On Loss Prediction for Real-time Packet Audio", IEEE , INFOCOM , USA, (2005).
Available at: http://dawn.cs.umbc.edu/INFOCOM2005/roychoudhuri-abs.pdf

167. Sacile, R., "Telemedicine Systems for Collaborative Diagnosis over the Internet: towards Virtual Co-laboratories", Proceedings of Research Challenges 2000, pp. 191-196, (2000).

168. Sanchez, J., Morilla,F., and Ruiperez, P., "Virtual and Remote Control Labs Using Java: A Qualitative Approach," IEEE Contr. System. Magazine, Vol. 22, No. 2, pp. 8-10, (2002).

169. Sandhu, R., and Samarati, P., "Authentication, Access Control and Intrusion Detection. In A. Tucker, editor, Database Security VII: Status and Prospects", pp. 1929-1948, CRC Press Inc, (1997).

170. Sandhu, R and Samarati, P., "Authentication, Access Control and Audit, ACM Computing Surveys", 50th anniversary commemorative issue, Vol. 28, No. 1, (1996).

171. "ScadaOnWeb", Available at: www.scadaonweb.com/home.html, (2002).

172. Sematech, "International SEMATECH e-Diagnostics Guidebook", Available at http://www.semiconductorportal.com/Content/Y2001/M09/D10/21708/Guidebook.pdf, (2001).

173. Serjantov, A., Sewell,P., and Wansbrough, K., "The UDP Calculus: Rigorous Semantics for Real Networking", In Theoretical Aspects of Computer Science (TACS2001),

Available at: citeseer.ist.psu.edu/serjantov01udp.html, (2001).

174. Shah, S., Patwardhan, R., and Huang, B., "Multivariate Controller Performance Analysis: Methods, Applications and Challenges", 2001 CPC-6 conference, pp. 187-219, (2001).

175. Singh, S.K., "Computer aided process control", Prentice Hall, (1988).

176. Slijepcevic, S., Potkonjak, M., Tsiatsis, V., Zimbeck, S., and Srivastava, M.B., "On Communication Security in Wireless ad-hoc Sensor Networks", Proceedings Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. WET ICE 2002. Los Alamitos, CA, USA: IEEE Comput. Soc, pp. 139-44, (2002).

177. Smith, B., "Reflection and Semantics in Lisp". pp. 23-35. POPL84', (1984).

178. Smith, O. J. M., "Closed Control of Loops with Dead Time", Chemical Engineering Progress Transactions, Vol. 53, No. 5, pp. 217-223, (1957).

179. Stramigioli, S., van der Schaft, A., Maschke, B., and Melchiorri, C., "Geometric Scattering in Robotic Telemanipulation", IEEE Transactions on Robotics and Automation, Vol. 18, No. 4, pp.588-596, (2002).

180. Stanfelj N., Marlin, T. E. , and MacGregor, J. F., "Monitoring and Diagnosing Process Control Performance -the Single Loop Case", Industrial & Engineering Chemistry Research, Vol. 32, No. 2, pp. 301-314, (1993).

181. Stefano, A., Bello, L., and Mirabella, O.. "Virtual Plant: A Distributed Environment for Distance Monitoring and Control of Industrial Plants", Proceedings of IEEE Intelligent Engineering Systems, pp. 445-449, (1997).

182. Surgenor, B., and Jofriet, P., "Thermal Fault Analysis and the Diagnostic Model Processor", In IFAC symposium, On-line fault detection and supervision in the chemical process industries, Eds. Dhurjati P. and G. Stephanopoulos. No.1, Pergamon Press, (1993).

183. Swanda, A., and Seborg, D., "Evaluating the Performance of PID Type Feedback Control Loops Using Normalized Settling Time", ADCHEM 97, Banff, Canada, (1997).

184. "The Real Benefit of XML", Available at: http://www.fcw.com/fcw/articles/2000/0522/tec-bragg-05-22-00.asp, (2000).

185. "The Bell-shaped, Normal, Gaussian Distribution", Available at: http://www.andrews.edu/~calkins/math/webtexts/stat06.htm, (2000).

186. "Teleworking", Available at: "http://www.ccsr.cse.dmu.ac.uk/resources/general/ethicol/Ecv7no2.pdf, (1997).

187. "The Virtual Control Laboratory", Available at: http://eweb.chemeng.ed.ac.uk/courses/control/course/map/, (2000).

188. "Tim Berners-Lee", Available at: http://www.w3.org,(2005).

189. Tsukui, R., Beaumount, P., Tanaka, T., and Sekiguchi, K., "Power System Protection and Control Using Intranet Technology", Power Engineering Journal, pp. 249-255, (2001).

190. Tzafestas S.G., "System Fault Diagnosis Using the Knowledge-Based Methodology", Fault diagnosis in dynamic systems, Theory and application, Prentice Hall, (1989).

191. Urbancsek, T., and Vajda, F., "Internet Telerobotics for Multi-Agent Mobile Microrobot Systems – A New Approach", INES 2003, IEEE proceedings, pp. 462-466, Assiut-Luxor, Egypt, (2003).

192. Valera, A., Diez, J.L., Valles, M., and Albertos, P., "Virtual and Remote Control Laboratory Development" IEEE Control Systems Magazine, Vol. 25, No. 1, pp. 35-39, (2005).

193. Vigna, G, Robertson, W., and Balzarotti, D., "Testing Network-based Intrusion Detection Signatures Using Mutant Exploits" Proceedings of the ACM Conference on Computer and Communication Security (ACM CCS), pp. 21-30, (2004).

194. "Virtual Control Lab", Available at: http://www.esr.ruhr-unibochum. De/VCLab/, (2002).

195. Von Lukas, U., "IT-security in E-Engineering - Challenges and Solutions", 12th European Concurrent Engineering Conference, pp. 63-67, (2005).

196. Wang, X., Reeves, D., and Wu, S. F., "Tracing-Based Active Intrusion Response" Available at: arqos.csc.ncsu.edu/papers/2001-09-sleepytracing-jiw.pdf, (2001).

197. Wang, N., and Schmidt, D., "Overview of the CORBA Component Model", Available at: http://portal.acm.org/citation.cfm?id=379581, (2001).

198. Wang, J.F., Tse, W., and He, L.S., "Remote Sensing, Diagnosis and Collaborative Maintenance with Web-enabled Virtual Instruments and Mini-servers" Int J Adv Manufacture Tech, No. 124, pp. 764–772, (2004).

199. Warnier, E., Yliniemi, L., and Joensuu, P., "Web Based Monitoring and Control of Industrial Processes", Available at:http://herkules.oulu.fi/isbn9514275152/, (2003).

200. Weise, J., "Public Key Infrastructure Overview", Available at: http://www.sun.com/blueprints/0801/publickey.pdf, (2003).

201. Wing, P., and O'Higgins, B., "Using Public-key Infrastructure for Security and Risk Management", IEEE Communications Magazine, Vol. 37, No. 9, pp. 71-73, (1999).

202. Widener, P., Eisenhauer, G, and Schwan, K., 'Open Metadata Formats: Efficient XML-Based Communication for High Performance Computing', HPDC, pp. 315-324, (2001).

203. Wittenmark, B., Åström, K. and Årzén. K., "Computer Control: An Overview", Available at: www.ifac-control.org/publications/ pbriefs/PB_Wittenmark_etal_final.pdf. (2002).

204. Wong, D., Paciorek, N., and Moore, D., "Java-based Mobile Agents", Communications of the ACM, Vol. 42, No. 3, pp. 92-102, (1999).

205. Wu, Q. H., and Buse, D. P., "An architecture for e-Automation", The IEE Computing and Control Engineering Journal, Vol. 14, No. 1, pp. 38-43, (2003).

206. XDF, Available at: "http://xml.gsfc.nasa.gov/XDF/XDF_home.html", (2002).

207. XDMF, Available at: "http://www.arl.hpc.mil/ice/XdmfUser.html", (2002).

208. Xia, C., and Howell. J., "Control Loop Status Monitoring", Available at: http://www.mech.gla.ac.uk/Dept/staff/vcard.html?PersonID=100, (2003).

209. Xue, W., and Huai, J., "Extended Role-based Access Control Model", Journal of Beijing University of Aeronautics and Astronautics, Vol. 31, No. 3, pp.298-302, (2005).

210. Yang, M., Ru, J.F., and Li, X. R., "Predicting Internet End-to-End Delay: Multiple-model approach", Technical report, University of New Orleans, Jan. (2004).

211. Yang, M., Ru, J.F., Chen, H., and Rao, N.S.V., "Predicting Internet End-to-End Delay: A Statistical Study", Annual Review of Communications, Vol. 58, (2005).

212. Yang, S., Tan, L.S. and Chen, X., "Specification and Architecture Design for Internet-Based Control Systems", Proceedings of the 26th Annual International Computer Software and Applications Conference, IEEE Computer Society, Oxford, UK, pp. 75-80, (2002a).

213. Yang, S.H., and Alty, J.L., "Development of a Distributed Simulator for Control Experiments through the Internet, Future Generation Computer Systems", Vol. 18, No. 5, pp. 595-611, (2002b).

214. Yang, S., Chen, X. and Alty, J.L., "Design Issues and Implementation of Internet Based Process Control", Control Engineering Practice, Vol. 11, No. 6, pp. 709-720. (2003a).

215. Yang, S.H., Chen, X. and Yang, L., "Integration of Control System Design and Implementation over the Internet Using the Jini Technology", Software Practice & Experience, Vol. 33, No. 12, pp. 1151-1175, (2003b).

216. Yang, L. and Yang, S., "Ensure the Safety and Security of Internet-based Control", Measurement + Control, Vol. 38, No. 1, pp. 22-26, (2005).

217. Yeung, K., and Huang, J., "Development of the Internet based control experiment. Decision and Control", Proceedings of the 40th IEEE Conference, Vol. 3, pp. 2809–2814, (2001).

218. Yokokohji, Y., Tsujioka, T., and Yoshikawa, T., "Bilateral Control with Time-varying Delay Including Communication Blackout", Proceedings of the IEEE Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Orlando, FL. pp. 285-292, (2002).

219. Yu, Q., Chen, B., and Cheng, H., "Web-Based Interactive Control System Design and Analysis", IEEE Control Systems, Vol. 24, No. 3, pp. 45-57, (2004).

220. Zaharia,H., Leon, F., and Gâlea, D., "A Framework for Distributed Computing Using Mobile Intelligent Agents", New Trends in Computer Science and Engineering, pp. 219-224, (2003).

221. Zhang, Y., Wen, C., and Soh, Y., "Indirect Closed-loop Identification by Optimal Instrumental Variable Method", Automatica, Vol. 31, No. 11, pp. 2269–2271, (1997).

222. Zhang, Y., and Jiang, J. "Integrated Design of Reconfigurable Fault Tolerant Control Systems". Journal of Guidance, Control, and Dynamics, Vol. 24, No. 1, pp.133–136, (2001).

223. Zhang, Y., and Jiang, J., "Fault-tolerant Control System Design with Explicit Consideration of Performance Degradation", IEEE Trans. on Aerospace and Electronics Systems, Vol. 39, No. 3, pp. 838-848, (2003).

224. Zulkernine, M., and Seviora, R.E., "Assume-Guarantee Supervisor for Concurrent Systems" 15th International Parallel and Distributed Processing Symposium (IPDPS'01) Workshops 04 23 - 04, San Francisco, California, USA, pp. 1552-1560, (2001).