

ABSTRACT

AL-MUTAIRI, NAIF BEJAD. Adaptive Fuzzy Modulation for Networked PI Control Systems. (Under the direction of Dr. Mo-Yuen Chow.)

Networks and their applications have been evolving substantially in the last two decades. Network-based control systems, also called Networked Control Systems (NCS), are one of the hottest topics in network-based applications. By networked control system we mean controlling the system through the network, i.e., using networks for remote control and automation.

One major concern of using networks in remote control and automation is the Quality of Service (QoS) deterioration effects. These QoS deterioration effects such as: network-induced time delays, bandwidth constraints, multiple-packet transmission, and dropping network packets are serious tribulations. Without considering the effect of the QoS deterioration the performance of the NCS closed-loop control system could be degraded and even destabilized.

In this thesis, an adaptive fuzzy logic modulation scheme is proposed to compensate for the network-induced time delays in the network-based PI control systems. This scheme is based on modulating the control signal provided by the PI controller and applied to the remote controller via the network. A network-based controlled DC motor is used in this thesis to illustrate the effectiveness of the proposed scheme.

Adaptive Fuzzy Modulation for Networked PI Control Systems

by

NAIF BEJAD AL-MUTAIRI

A dissertation submitted to the Graduate Faculty of

North Carolina State University

in partial fulfillment of the

requirements for the Degree of

Doctor of Philosophy

Department of Electrical and Computer Engineering

Raleigh

2002

APPROVED BY:

Dr. Mo-Yuen Chow
Chair of Advisory Committee

Dr. H. Joel Trussell

Dr. James J. Brickley

Dr. Fen Wu

Dedication

To my parents,
for their everlasting and unconditional love and support.

Biography

Naif Bejad Almutairi was born on January 1, 1974 in Kuwait. He received the Bachelor of Science in Electrical Engineering from Kuwait University, Kuwait in June 1996 and continued his graduate studies there in September 1996. He received his Master of Science degree from Kuwait University, Kuwait in January 1999.

In February 1999 he got a scholarship from Kuwait University to continue his graduate studies in the USA. In August 1999 he entered to North Carolina State University to pursue the Ph.D. degree in Electrical Engineering. He defended his dissertation in May 2002. He expects to join the Department of Electrical Engineering at Kuwait University in August 2002. His research interests are in the fields of control systems, networked control systems, adaptive control, and fuzzy logic control systems.

Acknowledgement

First and foremost, I would like to thank God, for without His strength and encouragement this degree would have never materialized.

I would like to thank my advisor, Professor Mo-Yuen Chow for the direction and support he has given to me for my entire Ph.D. career. In addition, I would like to acknowledge my advisory committee members, Dr. James J. Brickley, Dr. H. Joel Trussell, and Dr. Fen Wu, for their constructive comments.

Table of Contents

List of Figures.....	xi
List of Tables	xvii
Preface.....	xviii
1 Introduction.....	1
1.1 Network-based control.....	1
1.1.1 General Aspects	1
1.1.2 Why move to network-based control.....	2
1.2 Fundamental issues in network-based control systems	5
1.2.1 General Aspects	5
1.2.2 Network-induced delay.....	6
1.2.3 Single-packet versus multiple-packet transmission	9
1.2.4 Dropping network packets	10
1.3 Network-based control applications	10
1.3.1 Time-delay sensitive applications.....	10
1.3.1.1 Internet-Based Real-Time Control Engineering Laboratory	10
1.3.1.2 Unmanned vehicles.....	11
1.3.1.3 Voice and Video over networks.....	11
1.3.2 Time-delay insensitive applications.....	12
1.4 Network-based control: Mathematical formulation.....	12
1.4.1 Delays in network-based control systems.....	13
1.4.1.1 Sensor-to-controller delay, τ_k^{sc}	14

1.4.1.2	Computational delay, τ_k^c	15
1.4.1.3	Controller-to-actuator delay, τ_k^{ca}	15
1.4.2	Effects of delays on network-based control systems	16
1.4.2.1	Degrade System Performance	16
1.4.2.2	Decreased Stability Region.....	17
1.4.3	Formulating network-based control systems with the network-induced time delay	18
1.4.3.1	Remote System and Remote Controller.....	19
1.4.3.2	Communication network.....	20
1.4.3.3	Central Controller	21
2	Current Control Techniques.....	23
2.1	General aspects	23
2.1.1	State augmentation: Example	25
2.2	Augmented deterministic discrete-time model method	27
2.3	Queuing methods	30
2.3.1	Deterministic predictor-based delay compensation method.....	30
2.3.2	Probabilistic predictor-based delay compensation method.....	32
2.4	Optimal stochastic control method	34
2.5	Perturbation method.....	36
2.6	Sampling time scheduling method.....	40
2.7	End-user gain adaptation method.....	44
2.8	Conclusion	47

3	Network-Based Controlled DC Motor	48
3.1	DC motor as a network-based time-delay sensitive control application.....	48
3.2	DC motor system dynamics	49
3.3	Evaluation performance criteria for control systems	51
3.3.1	Rise time, t_r	51
3.3.2	Delay time, t_d	51
3.3.3	Settling time, t_s	52
3.3.4	Maximum (percent) overshoot, M_p	52
3.3.5	Summation of squared error, C	53
3.4	DC motor speed control design.....	53
3.4.1	Design requirements	53
3.4.2	PI control design	54
3.5	Network-based controlled DC motor: Configuration	58
3.6	Network-based controlled DC motor: Modeling using SIMULINK.....	60
3.7	Network-based controlled DC motor: Simulation	62
3.7.1	Effect of the network-induced time delays on the output of the network-based controlled DC motor	63
3.7.2	Effect of the network-induced time delay, τ_c , on the control signal provided by the central controller	64
4	Adaptive Fuzzy Modulation for Networked PI Control Systems: Partial Adaptation	66
4.1	Direct PI controller parameters updating.....	66

4.1.1	On-line tuning: Instantaneous cost function	67
4.1.2	On-line tuning: Moving window cost function.....	68
4.1.2.1	Heuristic PI control action	68
4.1.2.2	Drawbacks on using on-line direct PI updating.....	70
4.1.2.2.1	Counterexample: violating heuristics of PI controller action	71
4.1.3	Off-line PI controller parameters selection.....	73
4.1.3.1	Illustration example: Searching region for (K_p, K_i)	74
4.1.4	Conclusion	75
4.2	Fuzzy logic modulator: Design.....	77
4.2.1	Network-induced time-delay fuzzy compensation	77
4.3	Fuzzy logic modulation: Membership functions tuning	80
4.3.1	PI parameterization	81
4.3.2	Fuzzy logic modulation: Simulation.....	81
4.4	Partial adaptation: consequent parameters tuning.....	84
4.4.1	On-line tuning: Instantaneous cost function	84
4.4.2	On-line tuning: Moving window cost function.....	85
4.4.3	Off-line consequent parameters selection.....	85
4.4.3.1	Compensation technique	89
4.5	Simulation Results and Discussion.....	89
4.5.1	On-line tuning with instantaneous cost function	89
4.5.2	On-line tuning with moving window cost function	90
4.5.3	Off-line controller parameters selection	92

4.6	Conclusion	93
5	Adaptive Fuzzy Modulation for networked PI control systems: Full Adaptation..	94
5.1	Full adaptation	94
5.1.1	On-line tuning: Instantaneous cost function	96
5.1.2	On-line tuning: Moving window cost function.....	97
5.1.3	Off-line tuning: Using lookup table.....	98
5.1.3.1	Optimal (β_1, β_2) pair selection process	99
5.1.3.2	Optimal Membership function parameters	100
5.2	Simulation results and discussion	102
5.2.1	On-line tuning with instantaneous cost function	102
5.2.2	On-line tuning with moving window cost function	104
5.2.3	Off-line controller parameters selection	107
5.2.4	On-line AFM as a continuous gain scheduling.....	110
5.3	Conclusion	119
6	Stabilization of networked control system using fuzzy logic modulation.....	121
6.1	Stability analysis in NCS	121
6.2	Fuzzy modulation for networked PI control system	123
6.3	Hermite-Biehler Theorem applied to networked PI control system	124
6.4	Stabilization using fuzzy logic modulator parameter	132
6.4.1	Increasing in stability region.....	136
6.4.2	Delay-free closed loop system stability	137
6.5	Simulation results.....	137

6.5.1	Stable region	137
6.5.2	Unstable region	138
6.5.3	On border of stable/unstable regions	139
6.6	Conclusion	140
7	References:.....	141

List of Figures

Figure 1-1 On the left is a control panel for a point-to-point wired automation system. On the right is a system that is wired with DeviceNet. Notice the dramatic reduction of wires and the physical simplification of the system.	3
Figure 1-2 Networked systems can be reconfigured in software, rather than physically changed.	4
Figure 1-3 Timing diagram for two nodes on a random access network.....	7
Figure 1-4 Timing diagram for an arbitrary node on a scheduling network.....	8
Figure 1-5 Unmanned vehicle.....	11
Figure 1-6. Block diagram of a discrete-time model of a network-based control system	13
Figure 1-7 Timing diagram of delay generations.	14
Figure 1-8 A feedback loop for demonstrating the effect of poor system response.	16
Figure 1-9 Step response of the system in Fig 1-9, (a) without delay, (b) with delay.....	16
Figure 1-10 An example of a sampled-data closed-loop control system.....	17
Figure 1-11 Pole locations of the system in Fig. 1-11, (a) $T=0.13$, (b) $T=0.215$	18
Figure 1-12 An overall real-time network-based control system.	19
Figure 2-1 The relationship among $u(t)$, the delayed signal $u(t - \tau)$, and the sampling time h	26
Figure 2-2 Block diagram of deterministic predictor-based delay compensation method.	30
Figure 2-3 Block diagram of probabilistic predictor-based delay compensation method.	32
Figure 2-4 Block diagram of a system in the perturbation method.	37
Figure 2-5 Arbitration field in the frame formats of: (a). CAN 1.0. (b). CAN 2.0.....	38

Figure 2-6 Window of data transmissions in a sampling period T_1	42
Figure 2-7 Network-based control system with M control loops.	42
Figure 2-8 Block diagram of a peer-to-peer network-based DC motor control system.	46
Figure 3-1 DC motor.....	48
Figure 3-2 DC motor circuit diagram.	49
Figure 3-3 Typical step response of a second order linear time-invariant system.....	52
Figure 3-4 Model of the wheel control of an unmanned vehicle.....	54
Figure 3-5 Open-loop unit-step response.....	54
Figure 3-6 Uncompensated closed-loop system.	55
Figure 3-7 Step response of the uncompensated DC motor	55
Figure 3-8 DC motor with PI controller	56
Figure 3-9 Step response of DC motor with PI controller.....	58
Figure 3-10 Network-based PI control system.	59
Figure 3-11 General block diagram of the network-based controlled DC motor.	59
Figure 3-12 Timing diagram of delay generations.	60
Figure 3-13 SIMULINK blocks for the network-based controlled DC modeling.....	61
Figure 3-14 Network channels: (a) Server to Client model, (b) Client to Server model.....	61
Figure 3-15 Step response of the network-based controlled DC motor with different random time delays and a 1 (msec) sampling time.....	62
Figure 3-16 The relation between τ_c and control signal $u_c(t)$	64
Figure 3-17 Effect network-induced time delays on $e(t)$ and $u_c(t)$	65

Figure 4-1 (a) Error curve of step response, (b) Difference of the error signal, $\Delta e = e(k) - e(k-1)$. (1): Region 1, (2): Region 2, (3): Region 3, (4): Region 4.	71
Figure 4-2 (a) Plot of $\frac{\partial J(k)}{\partial K_p}$, (b) Plot of $\frac{\partial J(k)}{\partial K_i}$, (1): Region 1, (2): Region 2, (3): Region 3, (4): Region 4.	72
Figure 4-3. (a) Searching region for (K_p, K_i) , (b) Step response of the network-based controlled DC motor with off-line direct PI controller parameters tuning.	74
Figure 4-4. (a) Searching region for (K_p, K_i) , (b) Step response of the network-based controlled DC motor with off-line direct PI controller parameters tuning.	75
Figure 4-5 A modified network-based PI control system with fuzzy modulation.	77
Figure 4-6 Network-based controlled DC motor with fuzzy modulation.	78
Figure 4-7 Initial setting for the membership functions associated with input variable e and the output variable β	80
Figure 4-8 Membership functions associated with input variable e and output variable β	82
Figure 4-9 Step response of the network-based controlled DC motor with fuzzy modulation, (a) Smaller time delays, (b) Longer time delays.	83
Figure 4-10. Flowchart of the off-line consequent parameters selection process.	86
Figure 4-11. Searching region for the optimal (β_1, β_2) pair in partial AFM.	87
Figure 4-12. Step response of the network-based controlled DC motor with fuzzy modulation corresponding to different values of λ	88
Figure 4-13. Step response of the network-based controlled DC motor with on-line updating based on minimizing instantaneous cost function.	90

Figure 4-14. Step response of the network-based controlled DC motor with on-line updating based on minimizing moving window cost function: (a) $m=5$, (b) $m=10$, (c) $m=20$	91
Figure 4-15. Step response of the network-based controlled DC motor with off-line selection based on minimizing pre-computed cost function.....	92
Figure 5-1. Membership functions associated with the input variable e , and the output variable β	95
Figure 5-2. Flowchart of the off-line full AFM parameters selection process.	99
Figure 5-3. Flowchart of the off-line (β_1, β_2) selection process.....	100
Figure 5-4. (a) Flowchart of the $(\gamma_{11}, \gamma_{12})$ adaptation process, (b) Flowchart of the $(\gamma_{21}, \gamma_{22})$ adaptation process.....	101
Figure 5-5. Searching regions in the off-line AFM parameters tuning.....	101
Figure 5-6. Step response of the network-based controlled DC motor with on-line updating based on minimizing instantaneous cost function.	102
Figure 5-7. (a) Initial and final membership functions for full AFM, (b) Overlapping between membership functions during the adaptation process.....	104
Figure 5-8. Step response of the network-based controlled DC motor with on-line updating based on minimizing moving window cost function: (a) $m=5$, (b) $m=10$, (c) $m=20$	105
Figure 5-9. For $m=5$, (a) Initial and final membership functions for full AFM, (b) Overlapping between membership functions during the adaptation process.....	106
Figure 5-10. For $m=10$, (a) Initial and final membership functions for full AFM, (b) Overlapping between membership functions during the adaptation process.....	107

Figure 5-11. For $m=20$, (a) Initial and final membership functions for full AFM, (b) Overlapping between membership functions during the adaptation process.....	107
Figure 5-12. Step response of the network-based controlled DC motor with off-line selection based on minimizing pre-computed cost function.	108
Figure 5-13. (a) Membership functions for full AFM in Region. 1 and 2, (b) Overlapping between membership functions during the adaptation process.....	109
Figure 5-14. Step response of the network-based controlled Dc motor with off-line/on-line AFM under different network QoS. (a) Partial AFM, (b) Full AFM.	111
Figure 5-15 Two different QoS with their associated optimal (β_1, β_2) pair.....	112
Figure 5-16 (a) $u_C(t)$, (b) $\omega(t)$	112
Figure 5-17 (a) $u_C(t)$, (b) $\omega(t)$	113
Figure 5-18 (a) $u_C(t)$, (b) $\omega(t)$	114
Figure 5-19 (a) $u_C(t)$, (b) $\omega(t)$	114
Figure 5-20 (a) $u_C(t)$, (b) $\omega(t)$	115
Figure 5-21 (a) $u_C(t)$, (b) $\omega(t)$	116
Figure 5-22 Illustration case of the tread-off problem in the off-line AFM.	118
Figure 5-23 (a) $u_C(t)$, (b) $\omega(t)$	119
Figure 6-1. Network-based controlled DC motor with fuzzy modulation.....	124
Figure 6-2. Block diagram of networked PI control system.....	124
Figure 6-3. Simplified block diagram of networked PI control system.....	125
Figure 6-4. Plot of $\left(1 - \frac{z^2}{L^2} T_1 T_2\right) \cos z + k \beta k_p$ and $\frac{z}{L} (T_1 + T_2) \sin z$	129

Figure 6-5. Plot of $\delta_i(z)$ and $\delta_r(z)$.	132
Figure 6-6. Plot of the stabilizing fuzzy logic modulator β as a function of L .	136
Figure 6-7. (a) Typical step response of the network-based controlled DC motor, (b) Corresponding fuzzy logic modulator parameter β .	138
Figure 6-8. (a) Typical step response of the network-based controlled DC motor, (b) Corresponding fuzzy logic modulator parameter β .	139
Figure 6-9. (a) Typical step response of the network-based controlled DC motor, (b) Corresponding fuzzy logic modulator parameter β .	140

List of Tables

Table 3-1 DC motor parameters.	50
Table 4-1 List of the 4 regions corresponding to e and Δe	69
Table 4-2 Summarization of the 4 regions and the corresponding K_p and K_i updating.	70
Table 4-3. Different values of λ and the corresponding optimal (β_1, β_2) pair	88
Table 5-1 Comparison between full/partial AFM and direct PI tuning: On-line tuning with instantaneous cost function.	103
Table 5-2 Comparison between full/partial AFM and direct PI tuning: On-line tuning with moving window cost function.....	106
Table 5-3 Comparison between full/partial AFM and direct PI tuning: Off-line tuning.....	109

Preface

Networks and their applications have been evolving substantially in the last two decades. Network-based control systems, also called Networked Control Systems (NCS), are one of the hottest topics in network-based applications. By networked control system we mean controlling the system through the network, i.e., using networks for remote control and automation.

One major concern of using networks in remote control and automation is the Quality of Service (QoS) deterioration effects. These QoS deterioration effects such as: network-induced time delays, bandwidth constraints, multiple-packet transmission, and dropping network packets are serious tribulations. Without considering the effect of the QoS deterioration the performance of the NCS closed-loop control system could be degraded and even destabilized.

Disturbances and unanticipated events often occur in network-based applications. If the network cannot provide the required QoS, the network-based control application may have to lower its performance requirement; it may also use the assigned QoS to do its best to maintain the application availability and stability. The approach in this thesis is to treat the network QoS as a given condition and, to design control schemes that explicitly take the QoS deterioration issues into account and at the same time improve the system performance and stability of the networked control system.

From a control designer's viewpoint, the plant of the network-based control system includes not only the plant to be controlled but also the network that connects the system components. The network part of the NCS is difficult to handle, because the network system

traffic is stochastic in nature, and nonlinear. Time varying behaviors are inherently included. Currently, there is still no accurate mathematical model to describe the network-based control system behavior.

In control systems, despite the introduction of many sophisticated control theories and techniques, the majority of industrial processes nowadays are still regulated by Proportional-Integral (PI) controllers. Realistically, in order to switch to network-based control applications it is of interest to discuss, study, and analyze the effect of using networks on PI closed-loop control systems. At the same time, we need to design and develop a compensator for the network-induced effects on the network-based PI control systems. Most of the current available control techniques generally focus on conventional control techniques. In order to consider the uncertainty of communication delays, computational intelligent approaches such as: fuzzy logic, neural networks, and genetic algorithms may possibly be useful for network-based control in some cases. Fuzzy logic control has been used for control plants where the plant modeling is difficult and, when conventional control methods have shown limited success. Fuzzy logic controllers usually work well in an environment with a large uncertainty and unknown variation in plant parameters and structure. Fuzzy logic is used when an accurate mathematical model of the controlled system is not available, and when human experts are available to provide linguistic fuzzy control rules or linguistic fuzzy descriptions about the system.

In this thesis, an adaptive fuzzy logic modulation scheme is proposed to compensate for the network-induced time delays in the network-based PI control systems. This scheme is based on modulating the control signal provided by the PI controller and applied to the remote controller via the network. A network-based controlled DC motor is used in this

thesis to illustrate the effectiveness of the proposed scheme. The organization of the thesis is as follows:

Chapter 1 introduces the networked control systems. Two types of networked control systems applications are presented with examples: NCS time-delay sensitive applications, and NCS time-delay insensitive applications. An investigation on the effect of such QoS variations on the closed-loop system performance of the NCS, and on the stability region of the closed-loop NCS is presented. Mathematical formulation of the NCS problem is also derived.

Chapter 2 presents an extensive literature review on the current network-based control techniques.

Chapter 3 uses a network-based controlled DC motor for motion control purpose as a time-delay sensitive networked control system application. It illustrates the adverse effects of using a network in the closed-loop control system and its use in demonstrating and measuring the effectiveness of the proposed controller using fuzzy logic, so it may compensate for the network-induced time delay effects.

Chapter 4 proposes an intelligent controller using fuzzy logic on top of a PI gain to adaptively compensate for the network-induced time delay in a time-delay sensitive networked control system application. Nonetheless, with this proposed fuzzy logic compensator, we will not need to completely redesign the existing PI controller; we will simply modulate the PI controller action with the output of the intelligent fuzzy logic controller to compensate for the network-induced time delay effects in the network-based control systems. A partial adaptation scheme for this fuzzy logic modulator is also proposed to tune the consequent parameters.

Chapter 5 proposes a full Adaptive Fuzzy Modulation (AFM) and compare it with two other tuning methods proposed in chapter 4 to compensate for the network-induced time delays in networked PI control systems. In this full AFM, not only are the consequent parameters be tuned adaptively, but also the membership functions parameters in the antecedent part are tuned adaptively.

Chapter 6 considers the problem of stabilizing the networked control system using PI controller with fuzzy logic modulation. Using a version of the Hermite-Biehler Theorem applicable to quasipolynomials, a complete analytical characterization of all stabilizing fuzzy logic modulator parameter values to stabilize the networked PI control system are provided.

1 Introduction

Networks and their applications have been evolving substantially in the last two decades [1]. In this chapter, an introduction to the networked control systems is presented. Two types of networked control system applications are presented with examples: NCS time-delay sensitive applications, and NCS time-delay insensitive applications. In NCS time-delay sensitive applications, the time is critical, i.e., if the delay-time exceeds the critical time limit, the networked control plant can be either damaged, or degradation in performance is obtained. While in NCS time-delay insensitive applications, a task or a program runs based on time, but its deadline is not critical to the overall system performance.

One major concern of using networks in remote control and automation is the QoS variations effects such as network-induced time delays, bandwidth constraints, multiple-packet transmission, and dropping network packets. These QoS variations can degrade the performance of the closed-loop control systems designed without considering its effect and can even destabilize the closed-loop control system. An investigation of the effect of such QoS variations on the closed-loop system performance of the NCS, and the effect of the QoS variations on the stability region of the closed-loop NCS are presented. Mathematical formulation of the NCS problem is also derived.

1.1 Network-based control

1.1.1 General Aspects

Using a local area network-based control architecture has many advantages than a traditional point-to-point design including low cost of installation, ease of maintenance,

greater flexibility [2, 3]. For these reasons, the network-based control architecture is already used in many applications, for example in automobiles [4] and aircraft [5, 6]. Ethernet was not designed to deliver the determinism required to handle real-time network traffic. But high-speed Ethernet, especially with switching [7], can provide large throughput that is desirable for the real-time applications. Combine it with Internet protocol (IP); this architecture becomes a highly desirable candidate for real-time network-based control applications [7-9].

Switches can greatly reduce collision. A switch divides the network into multiple collision domains. Traffic arriving at the switch is sent only to one domain intended to receive the transmission. Thus the switch isolates traffic. The only contention is when multiple messages arrive at the switch intended for the same node; most switches support store-and-forward hardware to handle these cases. Because this is done in hardware, it takes only a few time slots.

There are compelling reasons to use the Internet Protocol network (IP network) for real-time control systems [10]. IP is truly the language of the information age, reaching into every area of modern life. Designs incorporating IP networking will be able to take advantage of this connectivity by monitoring and debugging systems remotely, collecting usage statistics, and notifying service personnel of impending failures. IP-based designs will allow more efficient deployment of resources, better service, and lower cost.

1.1.2 Why move to network-based control

The first and most obvious reason is to save wiring when interconnecting multiple devices. Large factory automation and process control applications use industrial networks

(fieldbuses) extensively. The elimination of large, unwieldy bundles of cables and the associated mess is an obvious advantage as shown in Fig. 1-1. Networking allows us to connect hundreds of devices to a single trunk line instead of using hundreds of individual wires. As systems grow past the threshold of 100 I/O points, the added expense of network hardware is offset by saving in wiring time. These benefits grow exponentially with the size of the system [11]. Network-based control systems have a higher component cost, but they save on wiring, labor, and wiring error.

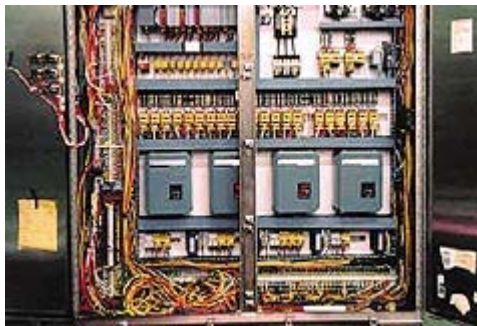


Figure 1-1 On the left is a control panel for a point-to-point wired automation system. On the right is a system that is wired with DeviceNet. Notice the dramatic reduction of wires and the physical simplification of the system.

The modularity of a networked system as shown in Fig. 1-2 is another major advantage. Connectivity is achieved through software, so taking a large system apart, putting it on a truck, and reassembling it somewhere else is much easier. The modularity brings interesting possibilities to machine design.

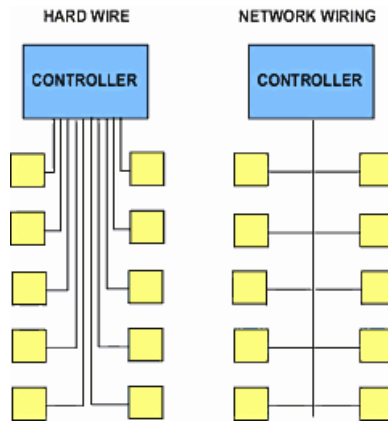


Figure 1-2 Networked systems can be reconfigured in software, rather than physically changed.

In addition to the wiring saving and modularity, there are four major reasons to move to network-based control:

- 1) **Diagnostics:** A networked device often can tell us if it is malfunctioning, or if something's about to go wrong. This information can be of great help. It is even more valuable if the information can be accessed remotely via the Internet.
- 2) **Self-Configuration:** Machines controllers can automatically detect which modular components are connected to the network and determine what software configuration to load. This feature reduces hours or even days off the delivery and setup time of a large system.
- 3) **IP network Infrastructure is already built:** The existing IP-based designs will allow more efficient deployment of resources, better service, and lower cost. Making IP networks excellent candidates for exchanging control signals and/or sensor signals from central controller to remote controller, and vice versa.
- 4) **Globalization of IP networks:** IP networks are reaching into every area of modern life. Designs incorporating IP networking will be able to take advantage of this

connectivity by monitoring and debugging systems remotely, collecting usage statistics, and notifying service personnel of impending failures.

1.2 Fundamental issues in network-based control systems

1.2.1 General Aspects

The intersection of the communication network in the feedback control loops makes the analysis and design of a network-based control system complex [12]. Conventional control theories with many ideal assumptions, such as synchronized control and nondelayed sensing and actuation, must be reevaluated before they can be applied to network-based control systems. Specially, the following issues need to be addressed: the first issue is the network-induced time delay (sensor to controller delay and controller to actuator delay) that occurs while exchanging data among devices connected to the shared medium such as IP network. This delay, either constant or time varying, can degrade the performance of the closed-loop control systems designed without considering the effect of delay and can even destabilize the closed-loop control system. Next, the network can be viewed as a web of unreliable transmission paths. Some packets not only suffer transmission delay but, even worse, can be lost during transmission. Thus, how such packet dropouts affect the performance of a network-based control system is an issue that must be considered. Another issue is that plant outputs may be transmitted using multiple network packets (so-called multiple-packet transmission), due to the bandwidth and packet size constraints of the network. Because of the arbitration of the network medium with other nodes on the network, chances are that all/part/none of the packets could arrive by the time of control calculation.

There are two main approaches for accommodating all of these issues in network-based control system design. One way is to design the control system without regard to the packet delay and loss but design a communication protocol that minimizes the likelihood of these events. For example, various congestion control and avoidance algorithms have been proposed [13, 14] to gain better performance when the network traffic is above the limit that the networks can handle. The other approach is to treat the network protocol and traffic as given conditions and design control strategies that explicitly take the above mentioned issues into account. For example, to handle time delay, one might formulate control strategies based on the study of delay-differential equations [15]. In this thesis, an adaptive fuzzy logic modulation scheme is proposed to compensate for the network-induced time delay in the network-based PI control systems. This scheme is based on modulating the control signal provided by the PI controller and applied to the remote controller via the network.

1.2.2 Network-induced delay

The network-induced delay in network-based control systems occurs when sensors, actuators, and controllers exchange data across the network. This delay can degrade the performance of closed-loop control systems designed without considering it and can even destabilize the closed-loop control system. Depending on the medium access control (MAC) protocol of the control network, network-induced delay can be constant, time varying, or even random. MAC protocols generally fall into two categories: random access and scheduling [16]. Carrier sense multiple access (CSMA) is most often used in random access networks, whereas token passing (TP) and time division multiple access (TDMA) are commonly employed in scheduling networks. Control networks using CSMA protocols

include DeviceNet [17] and Ethernet [18]. Fig. 1-3 illustrates various possible situations for this type of network.

Fig. 1-3 depicts two nodes continually transmitting messages (with respect to a fixed time line). A node on a CSMA network monitors the network before each transmission. When the network is idle, it begins transmissions immediately, as shown in Case 1 of Fig. 1-3. Otherwise it waits until the network is not busy. When two or more nodes try to transmit simultaneously, a collision occurs.

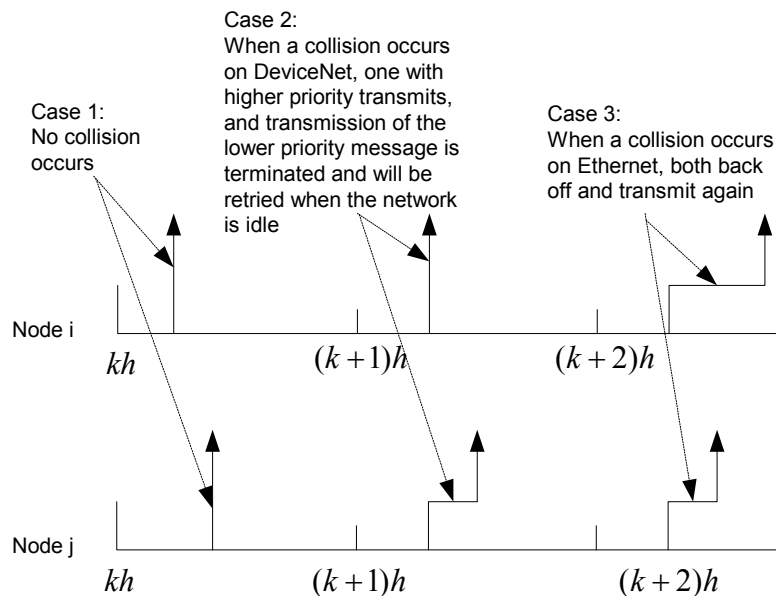


Figure 1-3 Timing diagram for two nodes on a random access network.

The way to resolve the collision is protocol dependent. DeviceNet, which is a controller area network (CAN), uses CSMA with a bitwise arbitration (CSMA/BA) protocol. Since CAN message are prioritized, the message with highest priority is transmitted without interruption when a collision occurs, and transmission of the lower priority message is terminated and will be retried when the network is idle, as shown in Case 2 of Fig. 1-3.

Ethernet employs a CSMA with collision detection (CSMA/CD) protocol. When there is a collision, all of the affected nodes will back off, wait a random time (usually decided by the binary exponential backoff algorithm [18]), and retransmit, as shown in Case 3 in Fig. 1-3. Packets in these types of networks are affected by random delays, and the worst-case transmission time of packets is unbounded. Therefore, CSMA networks are generally considered nondeterministic. However, if network messages are prioritized, higher priority messages have a better chance of timely transmission.

The TP protocol appears in token bus (IEEE Standard 802.4), token ring (IEEE Standard 802.5) [18], and the fiber distributed data interface (FDDI) MAC [19] architectures; TDMA is used in FireWire [20]. A timing diagram for this type of network is shown in Fig. 1-4. These protocols eliminate the contention for the shared network medium by allowing each node on the network to transmit according to a predetermined schedule. In a token bus, the token is passed around a logical ring, whereas in a token ring, it is passed around a physical ring.

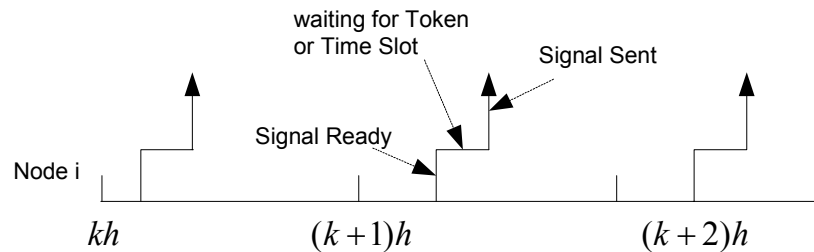


Figure 1-4 Timing diagram for an arbitrary node on a scheduling network.

In scheduling networks, it is possible to arrange for periodic transmission of messages. For example, FireWire has a transmission cycle ($125 \mu\text{sec}$) divided into small time slots, where each transaction is guaranteed a time slot to transmit in every cycle. Packet

transmission delays on scheduling networks occur while waiting for the token or time slot. They can be made both bounded and constant by transmitting packets periodically.

1.2.3 Single-packet versus multiple-packet transmission

Conventional sampled-data systems assume that plant outputs and control inputs are delivered at the same time, which may not be true for network-based control systems with multiple-packet transmissions. Due to network-induced delays, the controller may not be able to receive all of the plant output updates at the time of the control calculation.

Single-packet transmission means that sensor or actuator data are lumped together into one network-packet and transmitted at the same time, where as in multiple-packet transmission, sensor or actuator data are transmitted in separate network packets [12], and they may not arrive at the controller and plant simultaneously. One reason for multiple-packet transmission is that packet-switched networks can carry only limited information in a single packet due to packet size constraints. Thus, large amounts of data must be broken into multiple packets to be transmitted. The other reason is that sensors and actuators in a network-based control system are often distributed over a large physical area, and it is impossible to put the data into one network packet.

Different networks are suitable for different types of transmissions. Ethernet, originally designed for transmitting information such as data files, can hold a maximum of 1500 bytes of data in a single packet [18]. Hence, it is more efficient to lump the sensor data into one packet and transmit it together as single-packet transmission. On the other hand, DeviceNet, featuring frequent transmission of small-size control data, has a maximum 8-byte

data field in each packet; thus, sensor data often must be shuttled in different packets on DeviceNet.

1.2.4 Dropping network packets

Network packets drops occasionally happen on a network-based control system when there are node failures or message collisions. Although most network protocols are equipped with transmission-retry mechanisms, they can retransmit only for a limited time. After this time has expired, the packets are dropped. Furthermore, for real-time feedback control data such as sensor measurements and calculated control signals, it may be advantageous to discard the old, untransmitted message and transmit a new packet if it becomes available. In this way, the controller always receives fresh data for control calculation.

1.3 *Network-based control applications*

Network-based control applications can be classified into two categories: time-delay sensitive applications, and time-delay insensitive applications.

1.3.1 Time-delay sensitive applications

In this type of applications, the time is critical, i.e., if the delay-time exceeds the critical time limit, the plant or the device can be either damaged, or degradation in performance is obtained.

1.3.1.1 Internet-Based Real-Time Control Engineering Laboratory

The virtual laboratory concept [21, 22] allows students continuous access to their hypothetical experimental setups and is rapidly becoming feasible due to progress in Internet-

based networking technology. In a typical virtual laboratory educational paradigm, the experiments are emulated via a fast processor, and the students retrieve the experimental data over a computer network. In this paradigm, since the students need not be exposed to the actual experimental setups, the virtual laboratories can remain continuously accessible. Some of the advantages of using virtual laboratory paradigm are to minimize the cost to setup such a laboratory since it requires only a robust communications network, there is no need to purchase instruments typically used in the actual setups, and the maintenance is substantially lower.

1.3.1.2 Unmanned vehicles

Unmanned vehicles [23, 24] have generated much interest in recent years due to their great promise for performing repetitive, dangerous, and information-gathering tasks in hazardous or remote environments such as space, air, water, and ground.



Figure 1-5 Unmanned vehicle.

1.3.1.3 Voice and Video over networks

A major market trend today is the integration of voice and data communications into one single network. There are two main reasons for this trend: to eliminate the higher cost of

operating and maintaining two separate networks, and to support new multimedia applications, which typically allow users to talk and exchange data and images in one single session [25, 26]. Recent advances in computing technology, compression technology, high-bandwidth storage devices, and high-speed networks have made it feasible to provide real-time multimedia services over the Internet. Real-time multimedia, as the name implies, has timing constraints. For example, audio and video data must be played continuously. If the data do not arrive in time, the play process will pause, which is annoying to human ears and eyes.

1.3.2 Time-delay insensitive applications

In this type of applications, a task or a program runs based on time, but its deadline is not critical, e.g., e-mail, ftp, DNS, and http [27].

1.4 *Network-based control: Mathematical formulation.*

Although using networks in control system applications enhances the flexibility of integrated sensors, actuators, and controllers, network-induced delays could degrade the dynamic performance or even destabilize the real-time network-based control systems [28, 29]. In the following sections we will first define the types of delays in the network-based control system, then investigate the effect of such network-induced time delays on the closed-loop performance, and the effect of the network-induced time delays on the stability region of the closed-loop network-based control systems. Finally, we will formulate mathematically the network-based control problem.

1.4.1 Delays in network-based control systems

Due to data transmissions and controller processing, there would be communication delays in a discrete-time network-based control loop in addition to a delay from a sampling period T . Major types of these communication delays, along with a block diagram of a discrete-time model are depicted in Fig. 1-6.

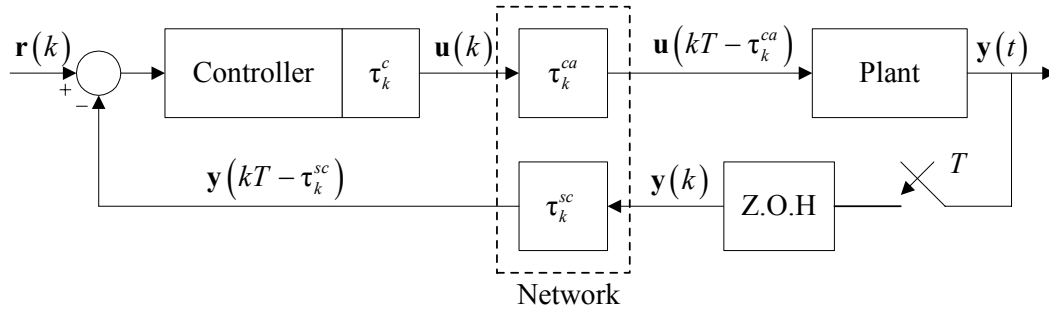


Figure 1-6. Block diagram of a discrete-time model of a network-based control system

where,

$y(t)$: Continuous system output

$r(k) \triangleq r(kT)$: Discrete reference signal

$u(k) \triangleq u(kT)$: Discrete control signal

$y(k) \triangleq y(kT)$: Discrete system output.

The timing diagram of delay generations is shown in Fig. 1-7.

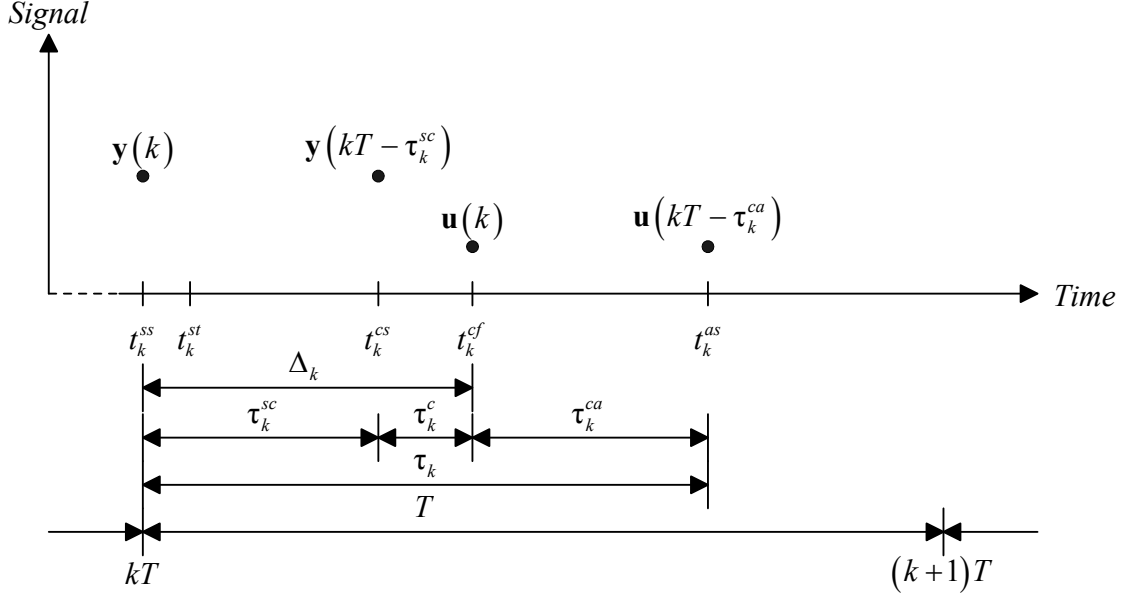


Figure 1-7 Timing diagram of delay generations.

Three types of delays in Fig. 1-6,7, τ_k^{sc} , τ_k^{ca} , and τ_k^c are described as follows.

1.4.1.1 Sensor-to-controller delay, τ_k^{sc}

This delay is generated when a sensor transmits a measurement to a controller. The sensor-to-controller delay at time index k is computed by:

$$\tau_k^{sc} = t_k^{cs} - t_k^{ss} \quad (1.1)$$

where t_k^{cs} and t_k^{ss} are the time instants at which the controller starts to compute the control signal based on the received measured signal, and the sensor starts to measure the system output, respectively. The sensor-to-controller delay also includes the transportation delay between sensor and controller described by:

$$\tau_k^{str} = t_k^{cs} - t_k^{st} \quad (1.2)$$

where t_k^{st} is the time instant when the sensor starts to transmit the measured output.

1.4.1.2 Computational delay, τ_k^c

Computational delay is the time referred for a controller to compute a control signal based on the received measurement. This delay is described by:

$$\tau_k^c = t_k^{cf} - t_k^{cs} \quad (1.3)$$

where t_k^{cf} is the time instant when the controller finishes computing a control signal. The computational delay can be considered as a kind of transmission delay.

1.4.1.3 Controller-to-actuator delay, τ_k^{ca}

This delay occurs when a controller sends a control signal to an actuator. The controller-to-actuator delay is defined as:

$$\tau_k^{ca} = t_k^{as} - t_k^{cf} \quad (1.4)$$

where t_k^{as} is the time instant that the actuator receives the control signal and starts to operate. Since an actuator normally starts to perform its task immediately after it receives the control signal, the controller-to-actuator delay can be thought of as a transportation delay in this case.

The sum of the delays in (1.1), (1.3), and (1.4), is referred to as the control delay τ_k [30]. The difference between the sampling time of sensor and the sampling time of controller is called the time skew Δ_k [1]. Some network-based control techniques [30-33] assume that the sampling time T is greater than τ_k . However, this is not always the case. The delay τ_k can exceed T in some techniques. Although different publications may use different definitions or notations of these three delays, the main concept remains the same.

1.4.2 Effects of delays on network-based control systems

The general effects of network-induced delays on a closed-loop control systems [34-36] could be listed as follows:

1.4.2.1 Degrade System Performance

Generally, delays can affect the magnitude of the transient behaviors in a control system. This degrades the overall system performance. This effect can result in the increasing of both overshoot and settling time. An example of this case is given as a delayed feedback loop in Fig. 1-8. Comparing the step response of the system without delay (i.e., $\tau = 0$) in Fig. 1-9(a), and the step response of the system with the delay $\tau = 0.2$ s in Fig. 1-9(b), the system with the delay obviously has the higher overshoot and the longer settling time.

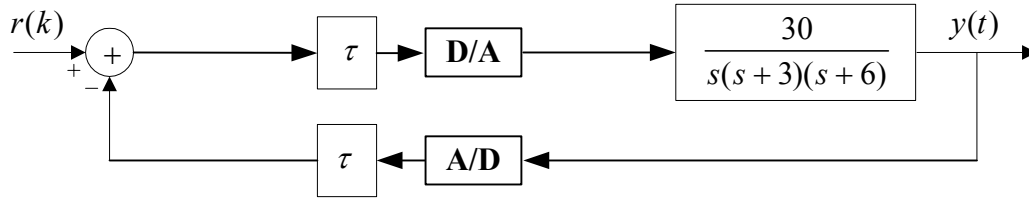


Figure 1-8 A feedback loop for demonstrating the effect of poor system response.

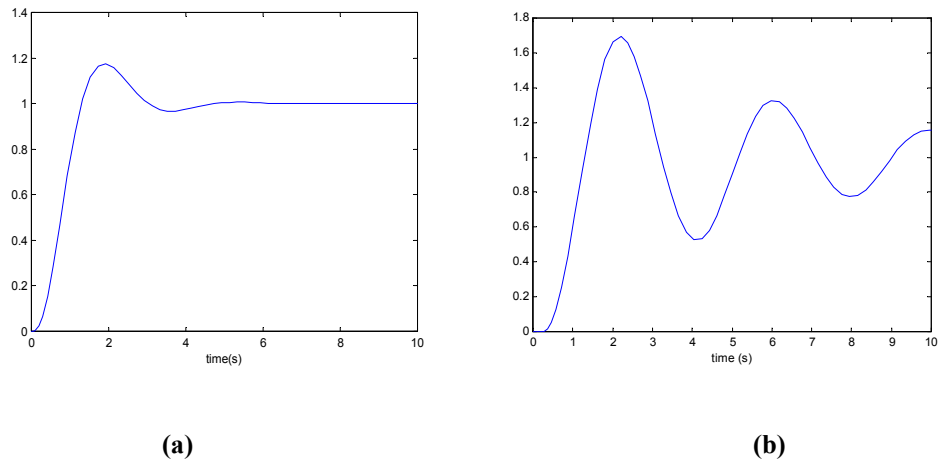


Figure 1-9 Step response of the system in Fig 1-9, (a) without delay, (b) with delay.

1.4.2.2 Decreased Stability Region

Delays can reduce the stability margin of a system and cause the system to become unstable. A basic example is too long a sampling time T . Choosing too long a sampling time can move system poles out of the unit circle, and the system then becomes unstable. An example of this case can be demonstrated by using a sampled-data closed-loop system [37] in Fig. 1-10.

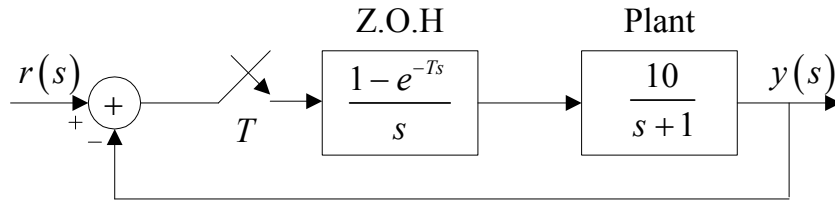


Figure 1-10 An example of a sampled-data closed-loop control system.

The closed-loop transfer function of this system obtained using z-transform is:

$$M(z) = \frac{10(1 - e^{-T})}{z - (11e^{-T} - 10)} \quad (1.5)$$

If the sampling time T is set to 0.13 (sec), the closed-loop system is stable because the pole is inside the unit circle as shown in Fig. 1-11 (a). On the contrary, the system is no longer stable if T is of 0.215 (sec) as depicted in Fig. 1-11 (b) since the pole is out of the unit circle.

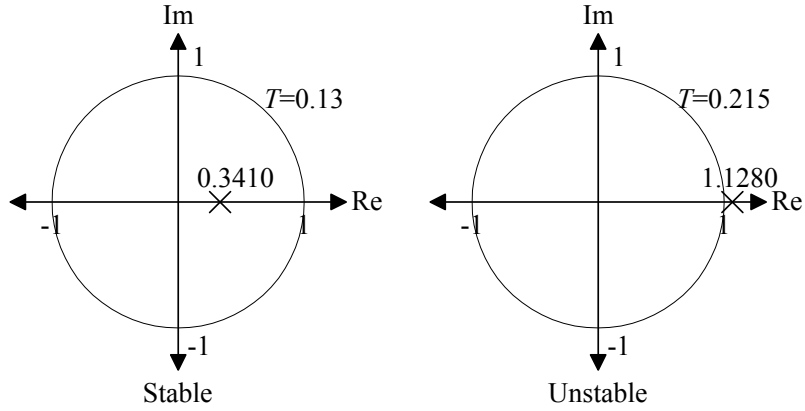


Figure 1-11 Pole locations of the system in Fig. 1-11, (a) $T=0.13$, (b) $T=0.215$.

Because network-based control systems usually involve delays, stability analysis is one of the most important concerns in the analysis and control of network-based systems. Most network-based control techniques have been developed along with an analysis of stability. Nevertheless, the analysis method is usually specific to the control technique used and may not be applicable to other control techniques due to different network setups and mathematical formulations.

1.4.3 Formulating network-based control systems with the network-induced time delay

There are many ways to design and implement a network-based control system. One may choose between different networks (e.g., Internet, FieldBus, CAN, etc.), protocols (e.g., IP, ATM, Token ring, etc.), computation power, intelligence, and control algorithms. A general block diagram of the network-based control system [38] is shown in Fig. 1-12.

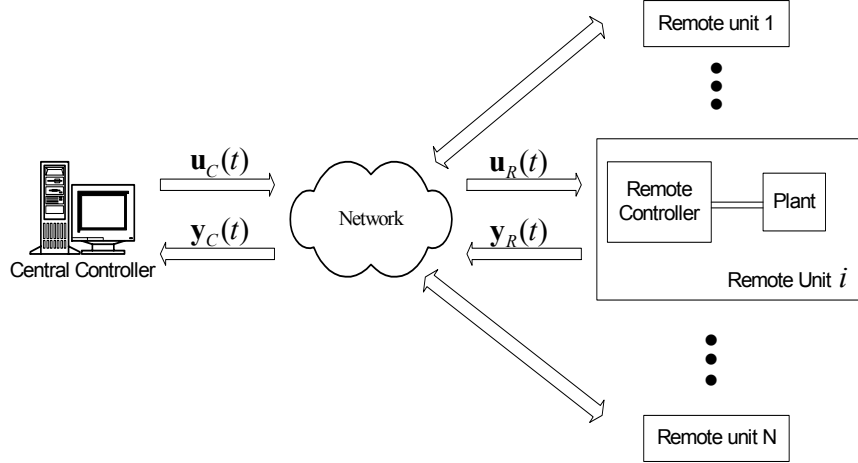


Figure 1-12 An overall real-time network-based control system.

The network-based control system can be categorized into three units: Distributed remote unit, communication network, and central Controller.

1.4.3.1 Remote System and Remote Controller

Each distributed remote controller can be assumed to have enough computing power to do relatively simple pre-programmed control – such as converting the control signal received from the central controller via network into a PWM signal to derive a DC motor (the remote process). Each remote controller can send local measurements, such as motor current, speed, temperature, and local environment information, back to the central controller via a network.

Each remote process has its own system dynamics that can be described by the state space description [39] shown in (1.6),

$$\dot{\mathbf{x}}_R = \mathbf{f}_R(\mathbf{x}_R, \mathbf{p}_R, \mathbf{u}_R, t). \quad (1.6)$$

where the state vector $\mathbf{x}_R = \begin{bmatrix} x_{R1} \\ \vdots \\ x_{Rn} \end{bmatrix} \in X^n$, the state space; $\mathbf{p}_R = \begin{bmatrix} p_{R1} \\ \vdots \\ p_{Rq} \end{bmatrix} \in \mathfrak{R}^q$ are the system

parameters; the input vector $\mathbf{u}_R = \begin{bmatrix} u_{R1} \\ \vdots \\ u_{Rr} \end{bmatrix} \in U^r$, the input space; $t \in \mathfrak{R}^+$ is the time parameter;

and $\mathbf{f}_R \in \mathfrak{R}^n$ is the state transfer function of the remote plant:

1.4.3.2 Communication network

The Quality of Service provided by the network as a function of time is denoted as $QoS(t)$. There are different ways to define Quality of Service for an end-to-end user profile, which is the central controller to the remote controller. Three of the most popular QoS measures are defined as follows:

- 1) QoS_1 : denotes the central controller to remote controller network throughput; it is used to indicate how fast the signal can be sampled and sent as a packet through the network.
- 2) QoS_2 : denotes the central controller to remote controller network average delay; it is used to indicate, on the average, how long of a delay is expected to deliver a packet from central controller-to-remote controller.
- 3) QoS_3 : denotes the central controller to remote controller network delay variance; it is used to indicate variance in delay time between packets.

1.4.3.3 Central Controller

The central controller can be a highly sophisticated controller that requires lots of computing power and memory that is not suitable at the remote site. The central controller is powerful and can provide advanced real-time control laws to all remote units, including fault diagnosis and accommodation control, network traffic condition monitoring and adaptation to the quality of service, $QoS(t)$, provided by the network. The central controller will provide the control signal $\mathbf{u}_c(t)$ to each of the remote systems.

The control signal is assumed to be the result of optimizing a cost function C , for example: The plant output error, $e(k) = r(k) - y(k)$, which is the difference between the reference signal, $r(k)$, and the remote unit plant output, $y(k)$, is used as cost function C ,

$$\mathbf{C} = \sum_{k=0}^L e(k)^2 \quad (1.7)$$

where L is the appropriate time index such that the tracking has arrived at steady state. Then the control signal provided by the central controller is given by:

$$\mathbf{u}_c(t) = \min C(\mathbf{x}_R, \cdot). \quad (1.8)$$

Let $z^{-\tau}$ be a time delay operator, and let $QoS(t)$ be provided by the network. We define:

$$\mathbf{u}_R(t) = \mathbf{u}_c(z^{-\tau_R}, QoS(t)), \quad (1.9)$$

$$\mathbf{y}_C(t) = \mathbf{y}_R(z^{-\tau_C}, QoS(t)), \quad (1.10)$$

where τ_R is the time delay in transmitting a signal from the central site to the remote site, and τ_C is the time delay in transmitting a signal from the remote site to the central site.

The time delay $z^{-\tau}$ and $QoS(t)$ are functions of network variables such as the type and number of signals to be transmitted, the network traffic congestion condition, the network throughput, the network protocol used, the network management/policy used, and the controller processing time.

2 Current Control Techniques

2.1 General aspects

Due to the communication delay concerns, the techniques used to control network-based systems have to maintain the performance and stability of the systems when the network conditions changed. Several methods have been formulated based on multiple types of network behaviors and configurations in conjunction with several ways to treat delay problems in network-based control systems [40, 41].

A widely used fundamental idea to formulate many network-based control methods is to use state augmentation. This approach has been used in discrete-time systems with constant delays for several years [42]. Several researchers use the state augmentation approach as the first step to developing their own techniques [1, 31, 32, 43, 44]. The main concept of state augmentation is to derive additional state-space equations to cooperate with the original state-space equations of a system as described in the following mathematical formulation.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t), \mathbf{x} \in \mathbb{R}^n, \mathbf{u} \in \mathbb{R}^m \quad (2.1)$$

$$\dot{\mathbf{v}} = \mathbf{g}(\mathbf{x}, \mathbf{u}, \mathbf{v}, t), \mathbf{v} \in \mathbb{R}^q \quad (2.2)$$

where $\mathbf{x}$ is the state vector of the original system, $\mathbf{u}$ is the system input vector, t is time, $\mathbf{v}$ is the augmented state vector, and $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$ are the functions representing the dynamics of the system and augmented states, respectively.

The combination of (2.1) and (2.2) forms the new state-space equation as:

$$\dot{\mathbf{s}} = \mathbf{f}_g(\mathbf{s}, \mathbf{u}, t), \mathbf{s} \in \mathbb{R}^{n+q} \quad (2.3)$$

where $\mathbf{s} = \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix}$, and $\mathbf{f}_g(\bullet)$ is the augmented dynamics equations of original and additional states. The augmented state vector $\mathbf{v}$ can be obtained in different ways depending on different aspects of the network-based control systems. The idea of state augmentation is quite simple. However, the number of additional states is typically proportional to the number of original states, inputs, and delay time. This method therefore can substantially increase the system complexity, especially for high-dimensional systems, even though the formulation is mathematically correct. An example of this approach on a linear system will be shown in section 2.1.1. In addition to the state augmentation approach, several other techniques have been developed to handle network delays [30-32, 45].

Several assumptions have been made in the previously mentioned approaches in order to develop generic network-based control methods. Some of these assumptions are:

- 1) Network transmissions are error-free, i.e., no message is lost.
- 2) Every data message always has the same constant length.
- 3) The time skew Δ_k is constant.
- 4) The computational delay τ_k^c is constant and is much less than the sampling period T .
- 5) Network traffic cannot be overloaded.
- 6) Single-packet transmission.

Some of these assumptions, such as 1), are obviously not practical. Hence, some control methods utilizing unrealistic assumptions may not be applicable for real-world applications. Nevertheless, several important phenomena can be observed, and many ideas can be obtained from simulations and experimental setups under these assumptions. These benefits are substantially useful for future developments.

Relaxation of some assumptions has been reported in some control techniques [46, 47]. On the other hand, some techniques may require additional assumptions [30-32, 48] such as the sum of sensor-to-controller delay and controller-to-actuator delay is less than the sampling period (i.e., $\tau_k^{sc} + \tau_k^{ca} < T$).

2.1.1 State augmentation: Example

Time delays are common in mathematical models of industrial processes. The theory of continuous-time systems with time delays is complicated because the systems are infinite-dimensional.

It is, however, easy to sample systems with time delays because the control signal is constant between sampling instants, which makes the sample-data system finite-dimensional. Let the system be described by

$$\dot{x}(t) = Ax(t) + Bu(t - \tau) \quad (2.4)$$

It is assumed initially that $\tau \leq h$, where h is the sampling time. The control signal $u(t)$ and the delayed control signal $u(t - \tau)$ are shown in Fig. 2-1.

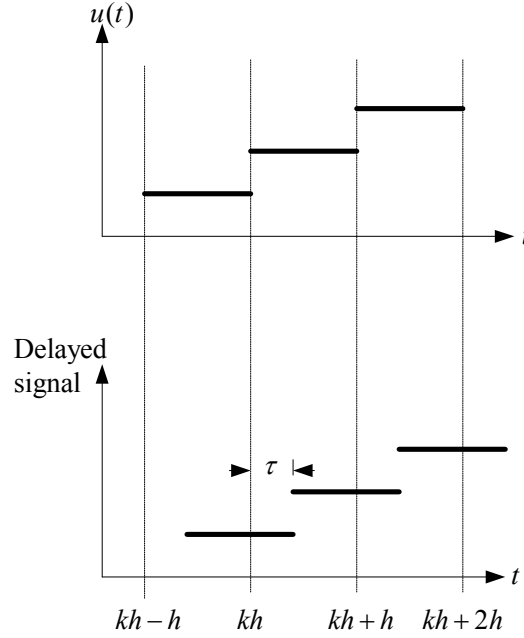


Figure 2-1 The relationship among $u(t)$, the delayed signal $u(t - \tau)$, and the sampling time h .

Integration of (2.4) over one sampling period gives

$$\begin{aligned} x(kh + h) &= e^{Ah} x(kh) + \int_{kh}^{kh+h} e^{A(kh+h-s')} Bu(s' - \tau) ds' \\ &= \Phi x(kh) + \Gamma_0 u(kh) + \Gamma_1 u(kh - h) \end{aligned} \quad (2.5)$$

where

$$\Phi = e^{Ah}, \Gamma_0 = \int_0^{h-\tau} e^{As} ds B, \Gamma_1 = e^{A(h-\tau)} \int_0^{\tau} e^{As} ds B \quad (2.6)$$

A state-space model of (2.5) is given by

$$\begin{bmatrix} x(kh + h) \\ u(kh) \end{bmatrix} = \begin{bmatrix} \Phi & \Gamma_1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(kh) \\ u(kh - h) \end{bmatrix} + \begin{bmatrix} \Gamma_0 \\ I \end{bmatrix} u(kh) \quad (2.7)$$

Notice that r extra state variables $u(kh - h)$, which represent the past values of the control signal, are augmented, where r is the number of inputs.

For longer time delay, $\tau > h$, if the time delay is represented by

$$\tau = (d - 1) + \tau' \quad (2.8)$$

where d is an integer, the following equation is obtained:

$$x(kh + h) = \Phi x(kh) + \Gamma_0 u(kh - (d - 1)h) + \Gamma_1 u(kh - dh) \quad (2.9)$$

where Γ_0 and Γ_1 are given by (2.6) with τ replaced by τ' . Notice that if $\tau > 0$, then $d \cdot r$ extra state variables are used to describe the delay. From this example we see that using state augmentation approach, the complexity of the system increases significantly and proportionally to the dimension of states and inputs.

2.2 Augmented deterministic discrete-time model method

Halevi and Ray proposed a method to control network-based systems on a cyclic service network in [1]. In this method, a controller and a sensor are the time-driven components, whereas an actuator is the event-driven device. By time-driven device, input receiving or output transmission is controlled by a sampling time, which could be represented by a clock signal, while an event-driven device starts to process immediately at the arrival time of a device input.

This approach can be applied on a linear plant described by the standard state-space equations as:

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} \quad (2.10)$$

$$\mathbf{y} = \mathbf{Cx} \quad (2.11)$$

where $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$ are the realization of the system.

For network-based control, this linear plant can be converted to the discrete-time equations [42] under the assumption that $\mathbf{u}$ is piecewise constant during each sampling time T as follows.

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}\mathbf{u}(k) \quad (2.12)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) \quad (2.13)$$

where $\mathbf{\Phi} = \exp(\mathbf{A}T)$, and $\mathbf{\Gamma} = \int_0^T \exp(\mathbf{A}s) ds \mathbf{B}$

A linear controller is used in this approach. In the case of a zero value set point, the controller dynamics can be described by:

$$\xi(k+1) = \mathbf{F}\xi(k) - \mathbf{G}\mathbf{z}(k), \xi \in \mathbb{R}^w \quad (2.14)$$

$$\mathbf{u}(k) = \mathbf{H}\xi(k) - \mathbf{J}\mathbf{z}(k) \quad (2.15)$$

where ξ is the controller state vector, $\mathbf{z}(k)$ is the most current available measurement at the instant when $\mathbf{u}(k)$ is processed by the controller (i.e., $\mathbf{z}(k) = \mathbf{y}(k-i)$, $i = \{1, \dots, j\}$), and $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$, and $\mathbf{J}$ are constant matrices describing the dynamics of the controller. The control input $\mathbf{u}$ for the plant in (2.12) is the output of this controller.

The main method used to cope with communication delay problems in this approach is to combine and to rearrange (2.12)-(2.15) into an augmented state-space equation in the following form for at most $(l+1)$ different values of $\mathbf{u}$ in each sampling period T .

$$\mathbf{X}(k+1) = \mathbf{\Omega}(k+1)\mathbf{X}(k) \quad (2.16)$$

where $\mathbf{X}(k) = [\mathbf{x}^T(k), \mathbf{y}^T(k-1), \dots, \mathbf{y}^T(k-j), \xi^T(k), \mathbf{u}^T(k-1), \mathbf{u}^T(k-l)]^T$ is the augmented state vector, and $\mathbf{\Omega}(k+1)$ is the augmented state transition matrix computed from $\mathbf{\Phi}$, $\mathbf{\Gamma}$, $\mathbf{C}$, $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$, and $\mathbf{J}$.

For periodic delays, there exists a positive integer M such that $\tau_{k+M}^{sc} = \tau_k^{sc}$. Using this property, the authors in [1] proved that the system in (2.16) is asymptotically stable if all

eigenvalues of $\Xi_k^M = \prod_{j=1}^M \Omega(k+M-j)$ are contained within the unit circle. $\Xi_k^M = \Xi_1^M$ was

also proved as a lemma.

Ray and Halevi also suggested a way to improve this network-based system by appropriately selecting Δ_k [46]. Choosing Δ_k according to the probability distributions of τ_k^{sc} and τ_k^{ca} can reduce the probabilities of message rejection P_{mr} and vacant sampling P_{vs} . Also, the appropriate Δ_k can reduce the average transportation delay τ_a , where $\tau_a = E[\tau_k^{sc} + \tau_k^{ca}]$, and can increase the stability margin. Thus, if the optimal time skew Δ_k is obtained, the performance of the system can be improved. The performance index used to obtain the optimal Δ_k in [46] is:

$$J = (1-\alpha)\phi(\tau_a) + \alpha TP_{vs}; \alpha \in [0,1] \quad (2.17)$$

where $P_{vs} = P_{mr}$ if the sampling period of the sensor and controller are both equal to T , and ϕ is a function of τ_a defined from various requirements (e.g., minimum average delay, minimum vacant sampling, etc.).

The optimal Δ_k for these goals can be obtained by using the sufficient conditions $\partial J / \partial \Delta_k = 0$, and $\partial^2 J / \partial \Delta_k^2 > 0$ for a minimization requirement or $\partial^2 J / \partial \Delta_k^2 < 0$ for a maximization requirement. In some cases, the optimal Δ_k can be considered directly from the probability distribution of delays, and the probabilities of message rejection P_{mr} , and vacant sampling P_{vs} . The scheme is also useful for random delay networks.

The structure of the augmented discrete-time approach is straightforward and easy to understand for a network-based control system with periodic communication delays.

Moreover, this method can be modified to support non-identical sampling periods of a sensor and a controller as mentioned in [49]. However, as a result of extensive state augmentation, the complexity of the system increases significantly and proportionally to the dimension of the states and inputs. More computational time is required as well.

2.3 Queuing methods

2.3.1 Deterministic predictor-based delay compensation method

Luck and Ray derived a delay compensation method for network-based control systems with random delays in [43, 50]. The main idea of delay compensation is to utilize an observer to estimate the plant states and a predictor to compute predictive control inputs based on past output measurements. A block diagram of the delay compensation algorithm is depicted in Fig. 2-2.

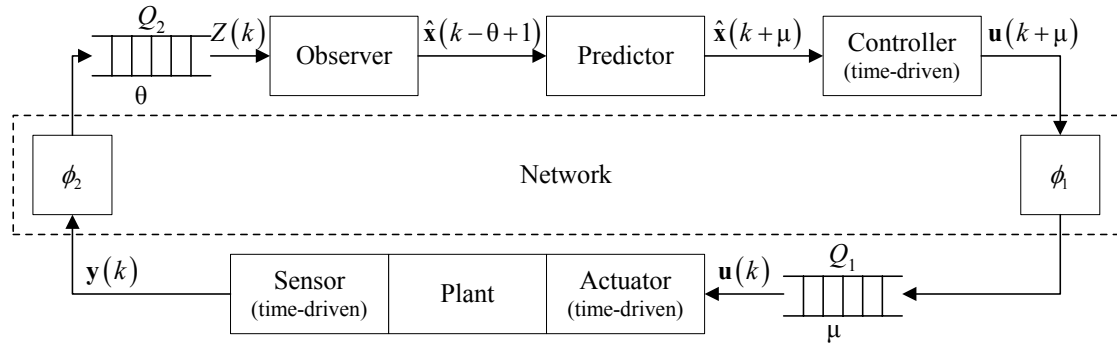


Figure 2-2 Block diagram of deterministic predictor-based delay compensation method.

In order to keep track of past measurements, these data have to be stored in a FIFO (First-In-First-Out) queue denoted as Q_2 at a controller. Another queue at an actuator denoted as Q_1 is also used to store predictive control inputs from the controller. Thus, both sensor-to-controller delay and controller-to-actuator delay are transformed to constant delays. Also,

the network-based control system with random delay becomes time-invariant and is much easier to control than random delay systems.

The delay compensation can be applied on a network-based control system, which has the following dynamics.

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}\mathbf{u}(k - \phi_1) \quad (2.18)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) \quad (2.19)$$

$$\mathbf{z}(k) = \mathbf{y}(k - \phi_2) \quad (2.20)$$

$$Z(k) = \{\mathbf{z}(k), \mathbf{z}(k-1), \dots\} \quad (2.21)$$

$$\mathbf{u}(k) = \Upsilon(Z(k)) \quad (2.22)$$

where $\mathbf{z}(k)$ is the delayed output vectors, $Z(k)$ is the set of past delayed outputs, and $\Upsilon(\bullet)$ is the control law. The non-negative integers ϕ_1 and ϕ_2 are the number of delayed samples at the system input and output vectors, respectively. The sizes of queues at the actuator and the controller are determined from the upper bound of ϕ_1 and ϕ_2 denoted as μ and θ , respectively. Therefore, the first measurement in Q_2 processed by the observer is $\mathbf{y}(k - \theta)$, and the control input at the beginning of Q_1 has to be $\mathbf{u}(k + \mu)$

The steps for applying the delay compensation algorithm are listed as follows:

1. Using the set of past measurements $Z(k)$ in Q_2 , the observer estimates the plant state

$$\hat{\mathbf{x}}(k - \theta + 1).$$

2. The predictor uses the estimated state to predict $\hat{\mathbf{x}}(k + \mu)$.

3. The controller computes the predictive control input $\mathbf{u}(k+\mu)$ from $\hat{\mathbf{x}}(k+\mu)$, and then stores this control input into Q_1 .

Since the performance of the observer and the predictor are highly dependent on the model certainty, the dynamic model of the plant has to be precise. A way to analyze the model uncertainty is suggested in [43]. In addition to model uncertainty, this approach results in unnecessarily long delays because of the queues used.

2.3.2 Probabilistic predictor-based delay compensation method

Chan and Özgüner developed another predictor-based method for network-based control systems with random delays by using a probabilistic approach along with the knowledge of the data lengths in a queue to improve the prediction [45]. A time-driven sensor, a time-driven controller, and an event-driven actuator are used in this approach. This method can be applied on network-based systems containing the discrete-time plant in (2.12)-(2.13). In order to utilize the queue lengths, the plant has to be connected as illustrated in Fig. 2-3.

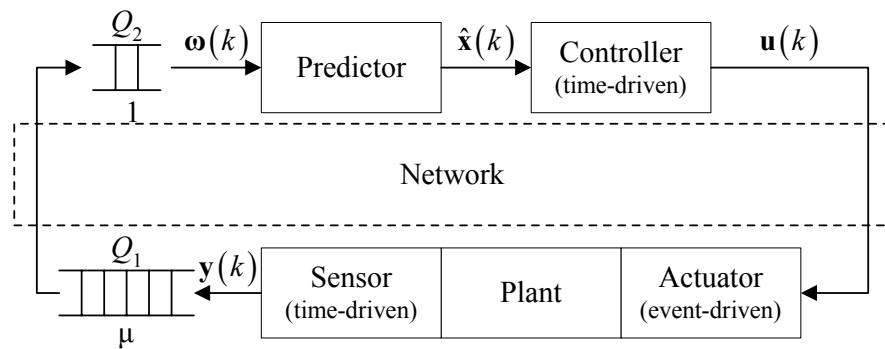


Figure 2-3 Block diagram of probabilistic predictor-based delay compensation method.

The FIFO queue Q_1 at the sensor has a capacity of μ , while the shift register Q_2 at the predictor has only a unit length. The output measurement $\mathbf{y}(k)$ is stored in Q_1 to be sent to Q_2 when the network traffic is available based on the priorities of system components. The number of data messages stored in Q_1 queue is defined as i . The output from Q_2 is denoted as $\omega(k)$.

At the k^{th} sampling period, if the sensor cannot send $\mathbf{y}(k)$ before Q_2 is read, $\omega(k)$ will be set to equal the previous measurement $\omega(k-1)$. In other cases, $\omega(k)$ can be identical to any values in the set $\{\mathbf{y}(k), \mathbf{y}(k-1), \dots, \mathbf{y}(k-\mu)\}$. However, the possible choices of $\omega(k)$ can be reduced to only two values if i is known. By using a lemma stated in [45], $\omega(k)$ is either $\mathbf{y}(k-i)$ or $\mathbf{y}(k-i+1)$. To satisfy this case, the value of i has to be attached to every data message of $\mathbf{y}(k)$ for predictor processing when the message is sent.

Using the information of Q_1 , the predictor computes the estimated current state $\hat{\mathbf{x}}(k)$ by:

$$\hat{\mathbf{x}}(k) = \mathbf{P}_0 (\Phi^{i-1} \omega(k) + \mathbf{W}_i) + \mathbf{P}_1 (\Phi^i \omega(k) + \mathbf{W}_{i+1}) \quad (2.23)$$

$$\text{where } \mathbf{W}_i = \begin{cases} 0, & i = 1 \\ \begin{bmatrix} \Gamma & \Phi\Gamma & \dots & \Phi^{i-2}\Gamma \end{bmatrix} \begin{bmatrix} \mathbf{u}^T(k-1) & \mathbf{u}^T(k-2) & \dots & \mathbf{u}^T(k-i+1) \end{bmatrix}^T, & i \neq 1 \end{cases}$$

The matrices $\mathbf{P}_0$ and $\mathbf{P}_1$ are weighting matrices. These two matrices are computed from the probabilities of the occurrences of $\mathbf{y}(k-i)$ and $\mathbf{y}(k-i+1)$. This formula is based on the assumption that the full state information is available (i.e., $\mathbf{y}(k) = \mathbf{x}(k)$). If the full state information cannot be obtained, an observer can be applied in this case without major

modification. With the predictive states, various control laws can be applied in conjunction with this scheme. In this work [45], PID control performance was evaluated based on the network delay assumptions. The PID controller in this illustration was applied on a vehicle speed control system. With the predictor, this feedback controller behaved closely to the no-delay case. However, since this approach is not a control algorithm by itself, the stability test has to be evaluated with a controller algorithm. Its vigorous proof was not shown in this work.

Even prediction can be improved by this method. Unnecessary delays from queues still remain as in the case in the previous approach. Also, this method can be applied only on sensor-to-controller transmissions since the effects of controller-to-actuator delay τ_k^{sc} are neglected.

2.4 Optimal stochastic control method

Nilsson and Wittenmark published an optimal stochastic control approach for control of a system plant over a random delay network in [30]. The effects of communication delays in network-based systems were treated as an LQG (Linear-Quadratic-Gaussian) problem. This approach uses an event-driven controller, an event-driven actuator, and a time-driven sensor. In addition to the general assumptions in section 2.1, the following assumptions are required.

- 1) The sum of sensor-to-controller delay and controller-to-actuator delay is less than the sampling period (i.e., $\tau_k^{sc} + \tau_k^{ca} < T$).
- 2) The controller delay τ_k^c is included in τ_k^{ca} .

3) The information of all past delays (i.e., $\{\tau_0^{sc}, \tau_1^{sc}, \dots, \tau_k^{sc}, \tau_0^{ca}, \tau_1^{ca}, \dots, \tau_{k-1}^{ca}\}$) is known.

A linear plant and a linear controller are used in this method. The dynamic of a linear plant derived from the continuous plant in (2.10)-(2.11) is described by:

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}_0(\boldsymbol{\tau}_k)\mathbf{u}(k) + \mathbf{\Gamma}_1(\boldsymbol{\tau}_k)\mathbf{u}(k-1) + \mathbf{v}(k) \quad (2.24)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{w}(k) \quad (2.25)$$

where

$$\boldsymbol{\tau}_k = [\tau_k^{sc}, \tau_k^{ca}]^T$$

$$\mathbf{\Phi} = \exp(\mathbf{A}T)$$

$$\mathbf{\Gamma}_0(\boldsymbol{\tau}_k) = \int_0^{T-\tau_k^{sc}-\tau_k^{ca}} \exp(\mathbf{A}\zeta) d\zeta \mathbf{B}$$

$$\mathbf{\Gamma}_1(\boldsymbol{\tau}_k) = \int_{T-\tau_k^{sc}-\tau_k^{ca}}^T \exp(\mathbf{A}\zeta) d\zeta \mathbf{B}$$

The stochastic processes $\mathbf{v}(k)$ and $\mathbf{w}(k)$ are uncorrelated Gaussian white noises with zero mean and covariance matrices. These equations are modified from the constant delay system in [42]. The goal of the optimal stochastic control approach is to minimize the following cost function in the case of available full state information.

$$J(k) = E[\mathbf{x}^T(N) \mathbf{Q}_N \mathbf{x}(N)] + E \sum_{k=0}^{N-1} \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{u}(k) \end{bmatrix}^T \mathbf{Q} \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{u}(k) \end{bmatrix} \quad (2.26)$$

Here, $\mathbf{Q}_N$ and $\mathbf{Q}$ are weighting matrices.

Based on this cost function, the authors in [30] derived a control law for optimal state feedback by using dynamic programming, which is conceptually described as:

$$\mathbf{u}(k) = -\mathbf{L}(k, \boldsymbol{\tau}_k) \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{u}(k-1) \end{bmatrix} \quad (2.27)$$

where $\mathbf{L}(k, \boldsymbol{\tau}_k)$ is a function of k and $\boldsymbol{\tau}_k$. The delays are assumed to be independent in this case. If the full state information is not available, optimal estimators such as the Kalman filter can be applied for (2.27). Nevertheless, this case requires the past information of plant output and input (i.e., $\{\mathbf{y}(0), \dots, \mathbf{y}(k), \mathbf{u}(0), \dots, \mathbf{u}(k-1)\}$) in conjunction with the past information of the delays.

In addition, the authors in [30] derived a modified version of this control law to use with the delays modeled by using the Markov Chains in [47]. In this case, $\mathbf{L}(k, \boldsymbol{\tau}_k)$ in (2.27) will be replaced by $\mathbf{L}(k, \boldsymbol{\tau}_k, r_k)$, which is a function of k , $\boldsymbol{\tau}_k$, and r_k , where r_k is a state of a Markov chain. The methods to cope with sampling period jittering, vacant sampling, and multiple delays in single plant output vectors are also briefly discussed. The stability of this system for both independent delays and delays modeled by Markov chains are also given along with this work by using stochastic stability analysis.

Based on the cost function in (2.26), this approach has been shown to have better performance than the deterministic predictor-based method since the delay problems are solved by not using queues. However, the drawback is a large requirement of controller memory to store a large amount of past information from the initial point. Also, since this approach is based mainly on assumption 1), this approach may not be effective for a system with fast response time.

2.5 Perturbation method

The perturbation method was introduced by Walsh, Beldiman, and Bushnell [44, 48, 51]. A time-driven sensor, an event-driven controller, and an event-driven actuator are

utilized as the basic system components in this method. Only data messages from a sensor can be transmitted through a network. A control loop in this method consists of a nonlinear controller and a nonlinear plant. However, the similar analysis and derivations used in this approach can be also applied on linear systems as described in [48]. Fig. 2-4 shows the block diagram of this system.

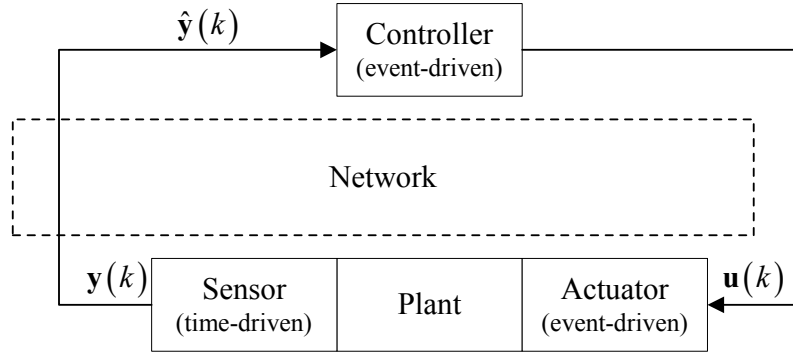


Figure 2-4 Block diagram of a system in the perturbation method.

This approach can be applied on both cyclic service and random access types of networks. However, these networks are restricted to be priority-based networks. This means the networks can assign different priorities to data transmissions of system components. These priorities are encoded in a data frame of each message. The number of encoded priorities depends on different protocols. For example, the data frames of CAN 1.0 and CAN 2.0 protocols use 11 and 29 bits, respectively, in their arbitration fields to encode the priorities of system components as shown in Fig. 2-5. Thus, there can be $2^{11}=2048$ and $2^{29}=536870912$ priorities in CAN 1.0 and CAN 2.0 networks, respectively. The higher number of bits used in a frame, the more priority levels can be encoded.

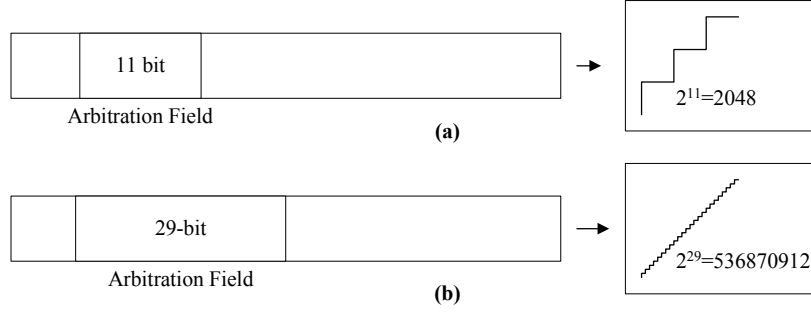


Figure 2-5 Arbitration field in the frame formats of: (a). CAN 1.0. (b). CAN 2.0.

These priorities can be either fixed or changeable. Fixed and changeable priority scheduling algorithms are denoted as static and dynamic scheduling, respectively. In addition, the authors proposed a novel protocol of dynamic scheduling called Try-Once-Discard (TOD), and compared this protocol with generic static and dynamic scheduling. In the TOD protocol, the priorities of sensor transmissions are determined from the error between the previous and the current data messages. When the sensor with the highest priority transmits a message, messages from other sensors are discarded.

Unlike the previous methods, the key concept of the perturbation method is to formulate communication delay effects in network-based systems as a perturbation of a continuous-time system under the assumption of no observation noise. This case requires a very small sampling period so that a continuous-time model can approximate a network-based system.

The dynamics of the network-based system are represented by:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{e}(t)) \quad (2.28)$$

where $\mathbf{x}(t) = [\mathbf{x}_p^T(t), \mathbf{x}_c^T(t)]^T$ is the partitioned vector containing the plant state vector $\mathbf{x}_p(t)$ and the controller state vector $\mathbf{x}_c(t)$. The function $\mathbf{f}(\bullet)$ is a nonlinear function.

The output of the plant is $\mathbf{y}(t)$ defined in (2.11). The most updated output received by the controller is denoted as:

$$\hat{\mathbf{y}}(t) \triangleq \mathbf{y}(t - \tau^{sc}) \quad (2.29)$$

where τ^{sc} indicates the sensor-to-controller delay of (2.29). The different between $\mathbf{y}(t)$ and $\hat{\mathbf{y}}(t)$ is formulated as an error of the network-based system described as:

$$\mathbf{e}(t) = \mathbf{y}(t) - \hat{\mathbf{y}}(t) \quad (2.30)$$

The dynamic of $\mathbf{e}(t)$ can also be represented by a nonlinear equation, which is a function of $t, \mathbf{x}(t)$, and $\mathbf{e}(t)$ as follows.

$$\dot{\mathbf{e}}(t) = \mathbf{g}(t, \mathbf{x}(t), \mathbf{e}(t)) \quad (2.31)$$

The main idea of all derivations in this work [44, 48, 51] is to model $\mathbf{e}(t)$ as a special type of vanishing perturbation [52]. In this case, if $\mathbf{e}(t) \equiv 0$, then $\mathbf{y}(t) = \hat{\mathbf{y}}(t)$. Also, from the control design, the Lyapunov function $V(t, \mathbf{x})$ of this case has the following properties.

$$c_1 \|\mathbf{x}\|^2 \leq V(t, \mathbf{x}) \leq c_2 \|\mathbf{x}\|^2 \quad (2.32)$$

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial \mathbf{x}} \mathbf{f}(t, \mathbf{x}, 0) \leq -c_3 \|\mathbf{x}\|^2 \quad (2.33)$$

$$\left| \frac{\partial V}{\partial \mathbf{x}} \right| \leq c_4 \|\mathbf{x}\| \quad (2.34)$$

$$\|\mathbf{x}\| < c_5 \leq \infty \quad (2.35)$$

where c_1, c_2, c_3, c_4 and c_5 are positive constants. If $c_5 \rightarrow \infty$, the system is globally exponentially stable.

This concept was used to derive a delay bound ρ such that the network-based system remains stable if sensor-to-controller delay is less than this bound (i.e., $\tau^{sc} < \rho$). The bound ρ is denoted as the Maximum Allowable Transfer Interval (MATI). Also, the authors proved that the bound of error $\mathbf{e}(t)$, $\|\mathbf{e}(t)\|$, could be also computed from ρ if static or dynamic scheduling algorithms including TOD are used in this network-based control scheme. The bound $\|\mathbf{e}(t)\|$ can be different depending on how many priorities are encoded. For other various encoding schemes such as logarithm encoding, the methods to compute $\|\mathbf{e}(t)\|$ derived by using the same concept were described in [53]. The exponential stability of the network-based systems with a static scheduling algorithm and with TOD was proved by considering ρ . This case assumes that many priorities can be encoded in data frames such that these priorities can be approximated as continuous priorities. As such, the frame format of CAN 2.0 can be used. Stability cannot be always guaranteed if priorities cannot be approximated as mentioned. Nevertheless, in this case, the plant states $\mathbf{x}_p(t)$ can be designed to locate in a bound for a specific condition.

A major advantage of the perturbation method is that it can be applied on a nonlinear system. However, this approach is not applicable to a system containing controller-to-actuator delay.

2.6 Sampling time scheduling method

Hong introduced the sampling time scheduling method in [31]. The fundamental concept of this approach is to appropriately select a long enough sampling period for a discrete-time network-based system such that communication delays do not affect the control

performance, and the system remains stable. In this case, the control delay τ_k in a discrete-time control loop must be assumed to be less than the sampling period T of the loop. The control loop consists of a time-driven sensor and controller, and an event-driven actuator. This method can be applied on cyclic service networks, in which all system component connections on the network are known in advance. This approach does not only solve the periodic delay problems, but can also increase network utilization in the case where there are multiple control loops on the same network.

Hong mentioned that for a discrete-time system in (2.12)-(2.13), the effect of communication varying-delay is exerted by the interval $[kT, (k+1)T)$ rather than by the time-varying delay occurring in the same interval. Thus, if the worst case of delays of network transmissions can be found, and this delay does not exceed the sampling period T , the network-based control system will operate in the same way as a generic discrete-time system. Nevertheless, the sampling time T cannot be greater than a maximal allowable bound φ . The bound φ is defined as the longest possible sampling period of the system, of which the system is still stable. Different control loops on the same network can have different bounds. This concept is similar to the delay bound ρ in the previous method, but it is used for different purposes. The method suggested by Hong can only be used in case the system in (2.12)-(2.13) is a one-dimensional system. General frequency-domain analysis can be used to find φ and to check system stability in this case. The algorithm of this method is based on the window concept illustrated in Fig. 2-6.

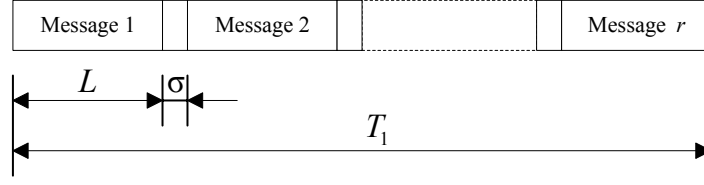


Figure 2-6 Window of data transmissions in a sampling period T_1 .

where L and σ are the transmission periods of a pure data message and message overhead, respectively, T_1 is the sampling period of the system, which has the smallest allowable delay bound compared to other control loops on the same network, and r is the number of data messages that can be served by the network on the medium during the worst case of network transmissions. The maximal delay bound of this system is denoted as φ_1 . The number of control loops on the network is defined as M , an example is shown in Fig. 2-7.

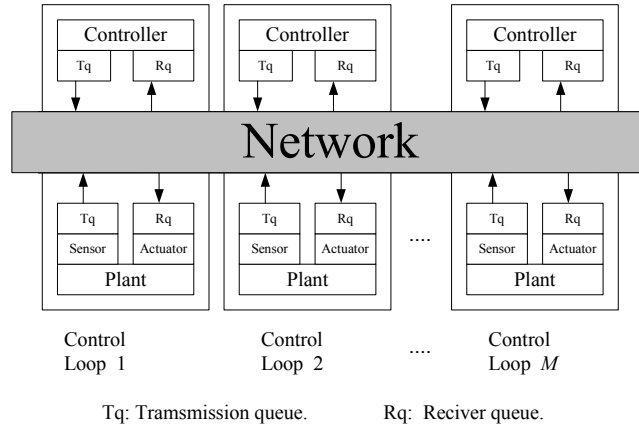


Figure 2-7 Network-based control system with M control loops.

The author proved that the sampling time T_1 could be computed by:

$$T_1 = \frac{\varphi_1 + L}{3} \quad (2.36)$$

In order to assign sampling periods to other control loops (i.e., the 2nd to M^{th} control systems) on the same network by the scheduling algorithm, these control loops have to be

indexed in an ascendant order from the sizes of their maximal allowable delay bounds. The maximum allowable delay bounds of the 2nd to M^{th} loops are denoted as φ_2 to φ_M , respectively. Using this index, the sampling time of the 2nd to M^{th} control loops was proved as expressed as:

$$T_i = k_i T_1, \quad \forall i \in \{2, 3, \dots, M\} \quad (2.37)$$

$$k_i = \Lambda \left(\frac{\varphi_i - (T_1 - L)}{2T_1} \right) \quad (2.38)$$

where $a = \Lambda(b)$ indicates that $a = 2^{\nu_i}$, $\nu_i \in \{0, 1, 2, \dots\}$, which is the “closest” to but does not exceed b .

In a special case, in which the number of all system components and other resources connected on the same network is less than r , the sampling time period of all control loops has to be determined by:

$$T_i = \frac{\varphi_i - (T_1 - L)}{2}, \quad \forall i \in \{1, 2, \dots, M\} \quad (2.39)$$

It was also proved that the network utilization is optimum if

$$2 \sum_{i=1}^M \frac{T_1}{T_i} = r \quad (2.40)$$

Even this algorithm can be applied only on a single-dimensional system, the concept of maximal allowable bound itself can also be used for a multi-dimensional system. Using this concept, Kim, Kwon, and Park developed another algorithm for the multi-dimensional cases in [32, 33]. In their approach, the delay bound of each system is obtained from different stability criteria. The dynamics of such a multi-dimensional system on the network is briefly expressed as follows.

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{A}_1\mathbf{x}(t - \tau^{sc}) + \mathbf{A}_2\mathbf{x}(t - \tau^{sc} - \tau^c) + \mathbf{A}_3\mathbf{x}(t - \tau^{ca} - \tau^c) \quad (2.41)$$

where $\mathbf{x}(t) = [\mathbf{x}_p^T(t) \quad \mathbf{x}_c^T(t)]^T$, $\mathbf{x}_p(t)$ is the plant state vector, $\mathbf{x}_c(t)$ is the controller state vector, τ^{sc} , τ^c , and τ^{ca} are sensor-to-controller, computational, and controller-to-actuator delays denoted for continuous-time models, respectively. The matrices $\mathbf{A}$, $\mathbf{A}_1$, $\mathbf{A}_2$, and $\mathbf{A}_3$ are calculated from the realizations of the plant and controller. Two existing asymptotically stable criteria based on the Lyapunov function can be used to find the maximal allowable delay of this system [54-56]. The first stability criterion is used to guarantee that a system can have a very large delay bound. If this criterion is not satisfied, another criterion will be used to find a limited delay bound. The sampling times of multi-dimensional systems on a network can be obtained by using a more complicated algorithm with these two stability criteria. However, the main idea of the algorithm is similar to the one-dimensional case.

With this approach, not only the delay problem can be solved, but also network utilization can be assigned appropriately for control loops and other resources connected to the same network. However, since the sampling times of control loops are multiples of T_1 , the sampling time delays may be longer than necessary similar to the queuing methods mentioned in Section 2.3.

2.7 End-user gain adaptation method

Chow and Tipsuwan have proposed a method of using gain adaptation to compensate for the Quality of Service *QoS* variations and deterioration in the network-based control systems [38, 57]. In this method, the controller can monitor the network's *QoS* via

appropriate middleware, and then the controller parameters are adapted to provide the best control performance with the given QoS .

The authors have used two QoS measures to define QoS for an end-to-end user profile. The first measure is QoS_1 , which is the point-to-point (central controller to the remote controller) network throughput; it is used to indicate how fast the signal can be sampled and sent as a packet through the network. The second measure is QoS_2 , which is the point-to-point maximal delay bound of the largest packet; it is used to indicate how long it takes to deliver a packet from the central controller to the remote controller. In this work they use a periodic sampling time, h , which satisfies the following criterion:

$$h > \tau_C + \tau_{PC} + \tau_R + \tau_{PR} \quad (2.42)$$

where τ_{PC} and τ_{PR} are the central controller processing time and remote controller processing time, respectively; τ_R is the delay time to transmit the control signal u_C from the central controller to the remote controller; τ_C is the delay time to transmit the measured signal y_R from the remote controller to the central controller. This criterion is used to guarantee that packet transfers between the central and remote controllers can be processed in a sampling period. Delay times τ_C and τ_R are estimated by computing delays such as transmission delay and propagation delay.

Both numerical and experimental simulations are presented to illustrate the effects of QoS variation on network-based control system. A peer-to-peer network-based DC motor control system is used to experimentally demonstrate the effects of QoS variation on a network-based control system. The actual system setup is shown in Fig. 2-8.

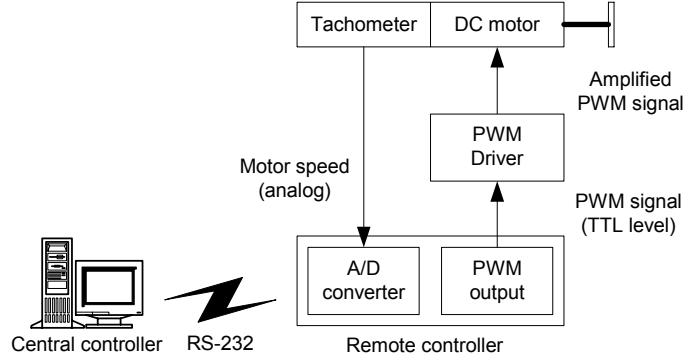


Figure 2-8 Block diagram of a peer-to-peer network-based DC motor control system.

The cost function used in adapting the controller parameter in order to compensate for the QoS variation is given by:

$$C = \frac{1}{N} \sum_{k=1}^N |r(k) - y(k)| \quad (2.43)$$

where N is the appropriate time index such that the tracking has arrived at steady state, $r(k)$ and $y(k)$ are the reference and output signals at time k . The cost function C considers how far the transient and steady state system response is from the reference signal.

In this method, the authors use a look-up table technique to perform the searching and to show the feasibility of the proposed proactive gain adaptation algorithm with respect to network QoS . The cost values are stored in a table with respect to network throughput, delay, variation, and controller parameters. Using the monitored network QoS , they search for the best K_p and K_i from the stored table that can provide the optimal cost. Linear interpolation is used when the exact entry cannot be found from the table with the given inputs.

2.8 Conclusion

From the presented network-based control techniques, it is noticeable that these techniques can have their own advantages and disadvantages. In order to apply one of these techniques, designers have to consider both the network and the control algorithms. Nevertheless, none of these techniques is perfect because many unrealistic assumptions are extensively used. Network-based control needs much research and improvement in order to be used for practical applications. Furthermore, as a result of fast growing network technologies, the improvement and development of network-based control would be substantially useful in the future.

Currently available control techniques generally focus on conventional control techniques. In order to consider the uncertainty of communication delays, computational intelligent approaches such as fuzzy logic, neural networks, and genetic algorithm could be useful for network-based control.

3 Network-Based Controlled DC Motor

In this chapter, we will use a network-based controlled DC motor for motion control purpose as a time-delay sensitive networked control system application. It illustrates the adverse effects of using an network in the closed-loop control system and its use in demonstrating and measuring the effectiveness of the proposed controller using fuzzy logic, so it may compensate for the network-induced time delay effects.

3.1 DC motor as a network-based time-delay sensitive control application

A common actuator found in many industrial control applications such as unmanned vehicles, robotics, etc, is the DC motor, which is a popular device for actuation and propulsion systems.



Figure 3-1 DC motor

As stated above, since many control applications use DC motors as a drive force, the DC motor is a good choice when we want to study and control network-based PI control systems. Also, the DC motor is a good example for time-delay sensitive applications [58]; the adverse effect of long time delay on the DC motor system performance and stability are

stated as follows: when the DC motor turns ON, the controller will rapidly increase the input voltage for the output speed to track the required reference speed in an acceptable rise time, settling time criteria. But as the output speed approaches the required reference speed, the controller will reduce the input voltage since the error signal between the reference speed and DC motor speed is reduced. When network is used in the network-based closed-loop DC motor control, the input voltage command from the controller to the DC motor will be delayed due to network-induced time-delay. If this delay is long enough, the DC motor will still run on the non-reduced input voltage, which causes high overshoot response and may lead to burning the DC motor.

3.2 DC motor system dynamics

The DC motor is a power actuator device that delivers to a load [59]. The DC motor circuit diagram is shown in Fig. 3-2.

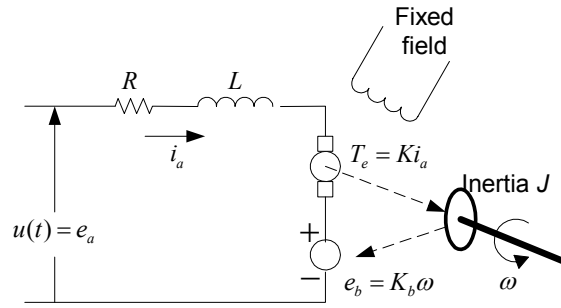


Figure 3-2 DC motor circuit diagram.

This section describes the dynamics of the DC motor used in this thesis as a network-based time-delay sensitive application. The loop equation for the electrical circuit of DC motor with a driven load is:

$$u(t) = e_a = L \frac{di_a}{dt} + R i_a + K_b \omega \quad (3.1)$$

The mechanical torque balance based on Newton's law is:

$$J \frac{d\omega}{dt} + B\omega + T_l = T_e = K i_a \quad (3.2)$$

where $u = e_a$ is the armature winding input voltage; L is the armature winding inductance; i_a is the armature winding current; R is the armature winding resistance; J is the system moment of inertia; B is the system damping coefficient; K and K_b are the torque constant and the back emf constant, respectively; T_l is the load torque; and ω is the rotor angular speed. The parameters of the DC motor used in this thesis are:

Table 3-1 DC motor parameters.

J	Inertia	3.53125e-6 Nm-sec
L	Inductance	2.88e-3 mH
R	Terminal Resistance	6.43 Ω
B	Damping Coefficient	0.1e-3 Nm
K	Torque Constant	2.55e-3 Nm/A
K_b	Back emf Constant	0.255e-3 V/(rad/sec)

By letting $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} i_a \\ \omega \end{bmatrix}$, the electro-mechanical dynamics of the DC motor can be

described by the following state-space description [60]:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \frac{di_a}{dt} \\ \frac{d\omega}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{K_b}{L} \\ \frac{K}{J} & -\frac{B}{J} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} u \quad (3.3)$$

$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{x} \quad (3.4)$$

assuming $T_l = 0$, which is equivalent to the general state space description in (2.10) and (2.11).

3.3 Evaluation performance criteria for control systems

Four different control schemes will be designed in this thesis for controlling the speed of the DC motor; the first one is a nominal PI control design without considering network-induced time-delay effects, the second is a network-based PI controller with direct PI parameters adaptation. In an attempt to compensate for the network-induced time delay effect, the third is a network-based PI controller with partially adapted fuzzy modulator to compensate for the network-induced time delay, and the fourth is a network-based PI controller with a fully adapted fuzzy modulator to compensate for the network-induced time-delay.

In order to compare these control schemes, we define the following five evaluation performance criteria [61]:

3.3.1 Rise time, t_r

The rise time is the time required for the DC motor speed to rise from 10% to 90% of its steady-state value.

3.3.2 Delay time, t_d

The delay time is the time required for the DC motor speed to reach half the steady-state value the very first time.

3.3.3 Settling time, t_s

The settling time is the time required for the DC motor speed curve to reach and stay within range about the final value of size specified by the absolute percentage of the steady-state value (usually 2% or 5%).

3.3.4 Maximum (percent) overshoot, M_p

The maximum overshoot is the maximum peak value of the DC motor speed curve measured from unity. If the final steady-state value of the DC motor speed differs from unity, then it is common to use the maximum percent overshoot. It is defined by

$$\text{Maximum percent overshoot} = \frac{\omega(t_p) - \omega(\infty)}{\omega(\infty)} \times 100\% \quad (3.5)$$

where $\omega(t)$ is the DC motor speed, t_p is the time required for the DC motor speed to reach the first peak of the overshoot. Rise time, delay time, settling time, and maximum (percent) overshoot are all shown in Fig. 3-3, which is a typical step response of a second order linear time-invariant system.

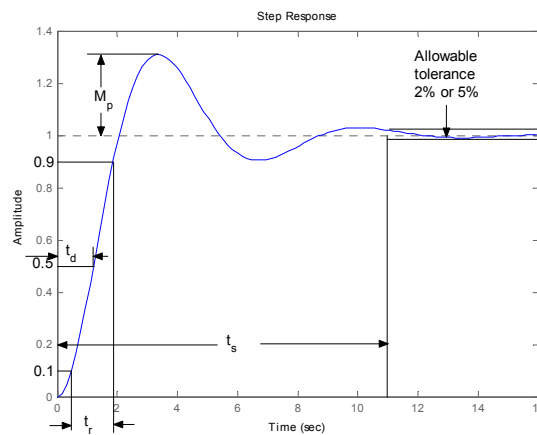


Figure 3-3 Typical step response of a second order linear time-invariant system.

3.3.5 Summation of squared error, C

The plant output error, $e(k) = r(k) - \omega(k)$, which is the difference between the reference speed signal, $r(k)$, and the output speed of the DC motor, $\omega(k)$, is used as an evaluation criterion C .

$$C = \sum_{k=0}^L e(k)^2 \quad (3.6)$$

where L is the appropriate time index such that the tracking has arrived at steady state.

3.4 DC motor speed control design

In order to start designing a controller for the DC motor, some specifications need to be firstly set on the industrial control application (e.g., unmanned vehicles) that uses the DC motor as an actuator. So by first setting desired specifications on that industrial control application. Then we can design our DC motor controller accordingly.

3.4.1 Design requirements

One of the important industrial control applications is the unmanned vehicle. For the unmanned vehicle, one of the main objectives is to accurately control the speed of the vehicle wheels. Each wheel control is assumed identical. The goal is to obtain a fast response with a rapid rise time and settling time to a step reference command while not exceeding an overshoot of 5% to avoid having an accident with another vehicle on the road, and at the same time smooth movement is achieved at the desired speed. A simple model of the wheel control for an unmanned vehicle [59] is shown in Fig. 3-4.

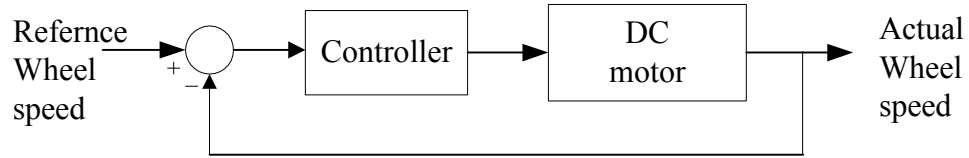


Figure 3-4 Model of the wheel control of an unmanned vehicle.

Our uncompensated DC motor can rotate at 3.96 rad/sec with an input voltage of 1 Volt as shown in Fig. 3-5. Since the most basic requirement of a DC motor is that it should rotate at the desired speed, the steady-state error of the DC motor speed should be smaller than 1% . The other performance requirement is that the motor will accelerate to its steady-state speed quickly after it is turned on. In this case we want it to have a 2% settling time of less than 30 msec. Since a speed faster than the requirement may damage the motor, we also want to have a rise time of less than 20 msec, and an overshoot smaller than 2% .

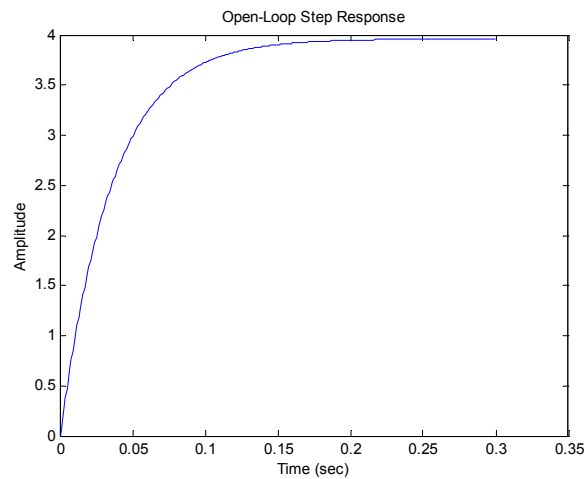


Figure 3-5 Open-loop unit-step response.

3.4.2 PI control design

It is well known that, despite the advent of many sophisticated control theories and techniques, the great majority of industrial processes nowadays are still regulated by

Proportional-Integral (PI) controllers [62]. In this section we will design a PI controller to control the speed of the DC motor. The open-loop transfer function of the DC motor $G_{DC}(s)$ can be derived from (3.3)-(3.4) and using the DC motor parameters in Table 3-1, it is given by:

$$\begin{aligned}
 G_{DC}(s) &= \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} \\
 &= \frac{2.507 \times 10^5}{s^2 + 2261s + 6.329 \times 10^4} \\
 &= \frac{2.507 \times 10^5}{(s + 2233)(s + 28.35)}
 \end{aligned} \tag{3.7}$$

The uncompensated closed-loop system is shown in Fig. 3-6.

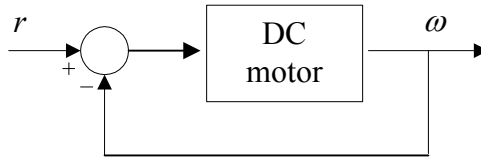


Figure 3-6 Uncompensated closed-loop system.

The step response of the uncompensated DC motor subject to tracking a reference speed signal (r) of 50 (rad/sec) is shown in Fig. 3-7.

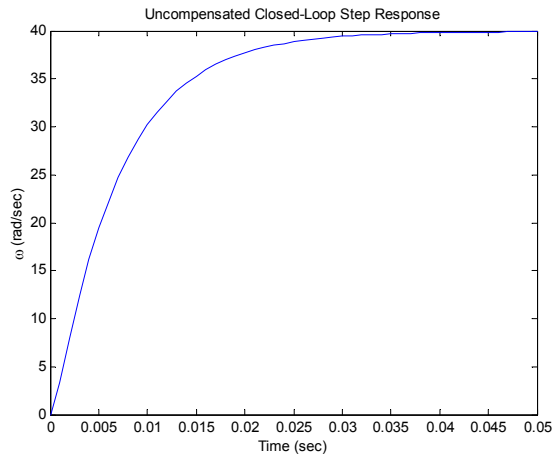


Figure 3-7 Step response of the uncompensated DC motor

By calculating the performance measures defined in section 3.3, we get:

$$\begin{aligned} t_r &= 14.8 \text{ msec}, t_d = 5.2 \text{ msec}, t_s = 26.8 \text{ msec} \\ M_p &= N/A, e_{ss} = 10.1024 \text{ rad/sec, and } C = 1.8262\text{e}+004 \end{aligned} \quad (3.8)$$

In order to eliminate the steady-state error, $e_{ss} = 10.1024 \text{ rad/sec}$, we add a PI controller in the forward path of the closed-loop system as shown in Fig. 3-8.

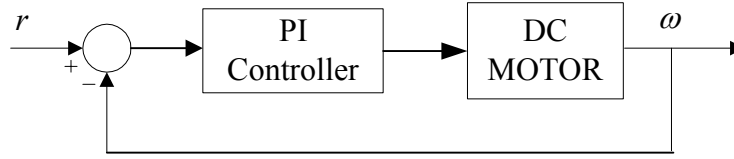


Figure 3-8 DC motor with PI controller

The transfer function of the PI controller is:

$$G_{PI}(s) = \left(K_p + \frac{K_i}{s} \right) = K_p \left(\frac{s + \frac{K_i}{K_p}}{s} \right) \quad (3.9)$$

where K_p and K_i are the proportional gain and integral gain of the PI controller, respectively.

We notice that using a PI controller we add an open-loop pole at the origin to increase the system type from zero to one, in order to eliminate the steady-state error due to step input. To improve the steady-state error without affecting the transient response, we need to place the zero of PI controller near to the pole at the origin, so that the angular contribution of the PI controller zero and PI controller pole cancel out [37]. Since the ratio of lengths from the PI controller pole and the PI controller zeros is approximately unity, thus we have improved the steady-state error without appreciably affecting the transient response.

Different (K_p, K_i) pairs can be found based on the above method such as $(1, 28.4)$, $(1.5, 42.5)$, $(2, 56.7)$, etc. The key idea is that the zero of the PI controller at $\frac{K_i}{K_p}$ is placed near the dominant pole of the open-loop transfer function of the DC motor such that PI controller zero can cancel the effect of dominant open-loop pole in the forward transfer function. Also, the zero at 28.35 is “close” to open-loop pole at zero compared to distance between the zero at 28.35 and the open-loop pole at 2233.

For illustration, $(K_p, K_i) = (1.5, 42.5)$ is used as a nominal PI controller parameters pair through this thesis. The forward transfer function $G_{PI}(s)G_{DC}(s)$ is:

$$G_{PI}(s)G_{DC}(s) = \frac{376106.1947(s + 28.33)}{s(s + 2233)(s + 28.35)} \quad (3.10)$$

Notice that the PI controller zero at 28.33 is close to the open-loop pole at 28.35 in order to cancel the effect of dominant open-loop pole in the forward transfer function. The transfer function from output speed $\Omega(s)$ to reference speed $R(s)$ i.e., the closed-loop DC motor system with PI controller transfer function is:

$$\begin{array}{ccc} R(s) & \xrightarrow{\quad} & \boxed{\frac{376106.1947(s + 28.33)}{(s + 2049)(s + 183.6)(s + 28.33)}} \xrightarrow{\quad} \Omega(s) \end{array}$$

The step response of the closed-loop DC motor system with PI controller subject to tracking a 50 rad/sec reference speed signal is shown in Fig. 3-9.

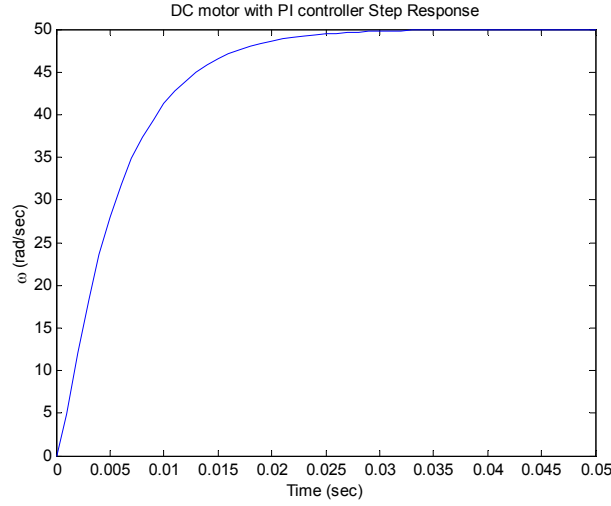


Figure 3-9 Step response of DC motor with PI controller.

From Fig. 3-9, we see how the PI controller set the steady-state error to zero, and at the same time, an accepted transient response [63] that met the desired specifications on the DC motor is obtained:

$$\begin{aligned} t_r &= 12.34 \text{ msec}, t_d = 4.29 \text{ msec}, t_s = 21.52 \text{ msec}, \\ M_p &= 0\%, e_{ss} = 0 \text{ rad/sec, and } C = 9.2361\text{e}+003 \end{aligned} \quad (3.11)$$

3.5 Network-based controlled DC motor: Configuration

In this section, we will form the network-based controlled DC motor by connecting the PI controller and the DC motor via a network, i.e., control the DC motor through the network.

As described in Chapter 1, the network-based control system consists of three units: distributed remote unit (Remote controller + DC motor), central controller (PI controller), and communication network as shown in Fig. 3-10.

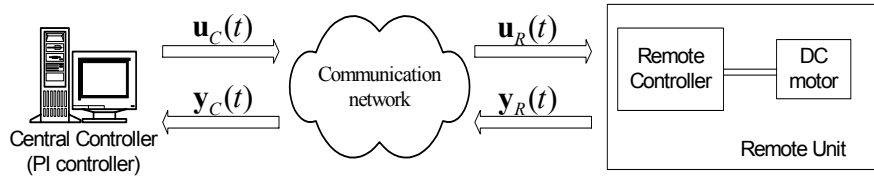


Figure 3-10 Network-based PI control system.

A general block diagram of the network-based controlled DC motor is shown in Fig. 3-11.

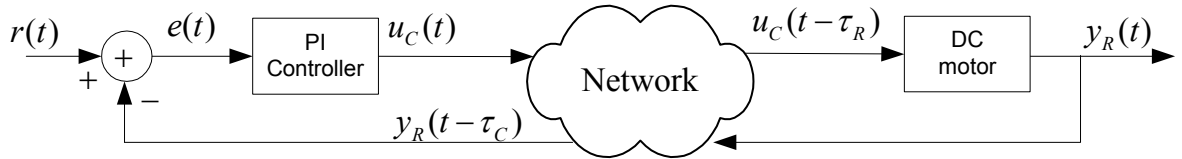


Figure 3-11 General block diagram of the network-based controlled DC motor.

The remote controller in the remote unit is assumed to simply convert the control voltage signal from the central controller $u_C(t)$ into a PWM signal to drive the DC motor. The remote controller value $u_R(t)$ can be mathematically expressed as:

$$u_R(t) = u_C(t - \tau_R) \quad (3.12)$$

where τ_R is the delay time to transmit the control signal u_C from the central controller to the remote controller. The remote controller also sends the monitored signal $y_R(t)$ of the remote system back to the central controller, $y_C(t)$, and these two signals are related as:

$$y_C(t) = y_R(t - \tau_C) \quad (3.13)$$

where τ_C is the delay time to transmit the measured signal y_R from the remote controller to the central controller.

The central controller uses a PI control algorithm to compute the control signal required for the remote unit to track a step reference signal based on monitoring the system signals sent from the remote system via the network. The PI controller used has the form:

$$u_c(t) = K_p e(t) + K_i \int e(t) dt \quad (3.14)$$

where K_p is the proportional gain; K_i is the integral gain; $e(t) = r(t) - y(t)$ is the error signal; $r(t)$ is the reference signal for the system to track, and $y(t)$ is the system output. In our case, $y(t) = \omega(t)$, the DC motor speed, and $u(t)$ is the input voltage to the DC motor.

Fig. 3-12 shows the timing diagram of delays τ_C and τ_R , assuming that the processing time in the central controller and remote controller is very small compared to the delays τ_C and τ_R , so it can be ignored.

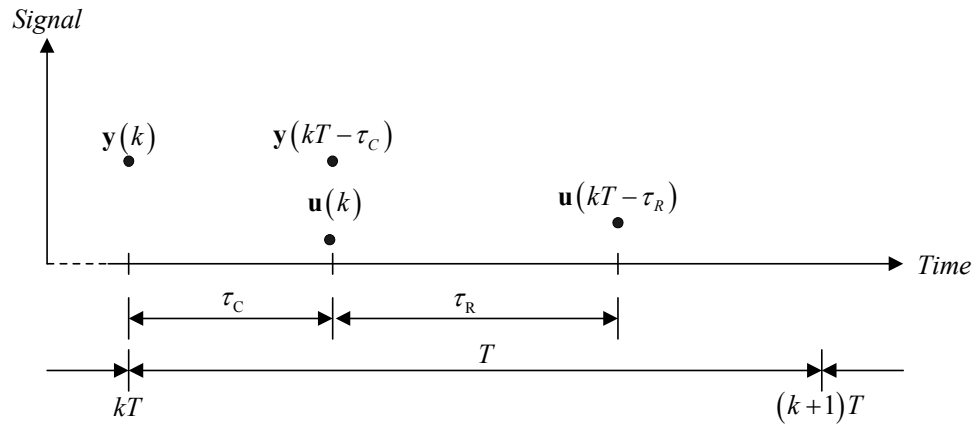


Figure 3-12 Timing diagram of delay generations.

3.6 Network-based controlled DC motor: Modeling using SIMULINK

The SIMULINK model used in modeling and simulating network-based controlled DC motor [38, 57] is shown in Fig. 3-13.

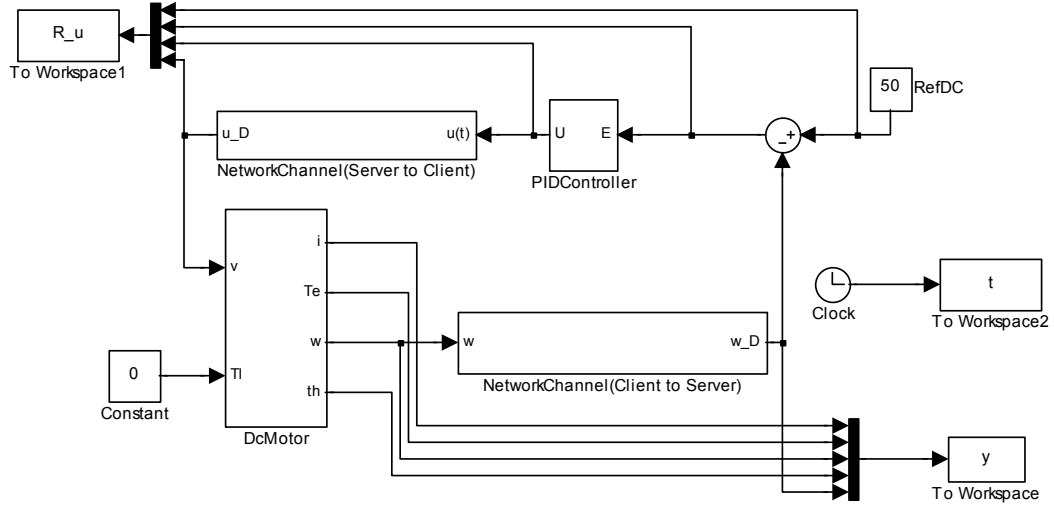


Figure 3-13 SIMULINK blocks for network-based controlled DC modeling.

Network Channel (Server to Client) model and (Client to Server) model are shown in more detail in Fig. 3-14.

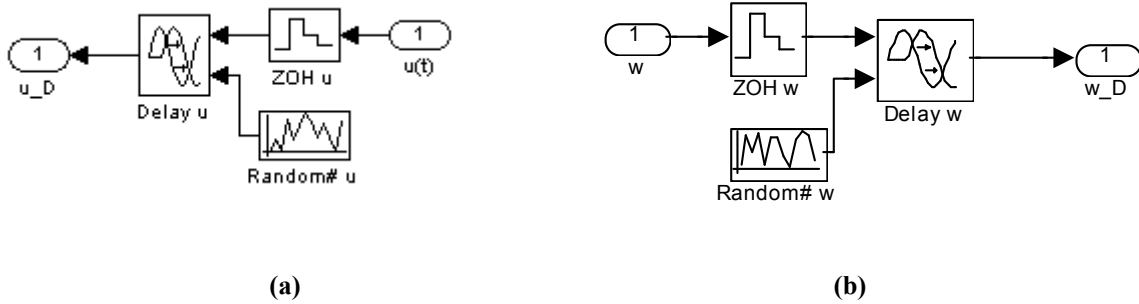


Figure 3-14 Network channels: (a) Server to Client model, (b) Client to Server model.

As shown in Fig. 3-14, a gaussian random number (μ, σ^2) is used to represent the network-induced time delays, where μ and σ^2 are the mean and variance of the network-induced random time delay.

3.7 Network-based controlled DC motor: Simulation

Using the SIMULINK blocks in Fig 3-13, and simulating for 0.3 seconds, we get the step response of the network-based controlled DC motor subject to tracking a reference speed of 50 (rad/sec) and using PI controller parameters $(K_p, K_i) = (1.5, 42.5)$. Fig. 3-16 depicts the Step response of the network-based controlled DC motor, with different random time delays and a 1 (msec) sampling time.

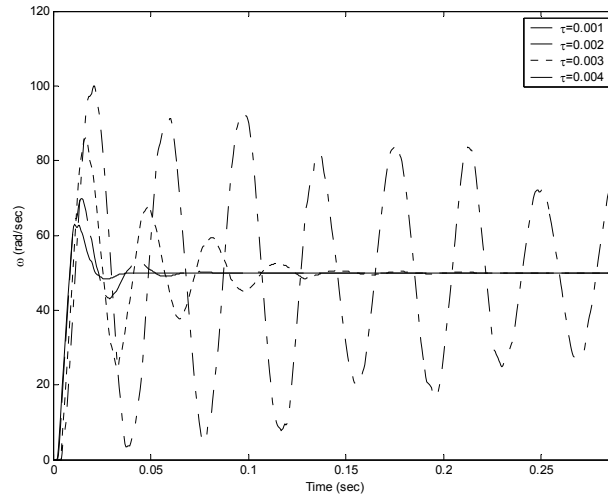


Figure 3-15 Step response of the network-based controlled DC motor with different random time delays and a 1 (msec) sampling time.

Fig. 3-15 shows that the network-induced time delay degrades the control-loop system performance by increasing both settling time and overshoot, while for longer time delay the network-based control system becomes unstable. In Fig. 3-15 and whenever τ is mentioned in this thesis, this implies $\tau_C = \tau_R = \tau$.

3.7.1 Effect of the network-induced time delays on the output of the network-based controlled DC motor

In this section, we will investigate the effect of network-induced time delay on the control-loop system performance mathematically. The differential equation of the remote process

$$\dot{\mathbf{x}}_R = \mathbf{f}_R(\mathbf{x}_R, \mathbf{p}_R, \mathbf{u}_R, t) \quad (3.15)$$

is used with the state-space description in (3.3)-(3.4) to give:

$$\begin{aligned} \dot{\mathbf{x}}_R &= \mathbf{A}\mathbf{x}_R + \mathbf{B}\mathbf{u}_C(t - \tau_R) \\ y_R &= \mathbf{C}\mathbf{x}_R \end{aligned} \quad (3.16)$$

where

$$\begin{aligned} e(t) &= r(t) - y_R(t - \tau_C) \\ u_C(t) &= K_p e(t) + K_i \int e(t) dt \end{aligned} \quad (3.17)$$

and solving for state trajectory, we get:

$$\mathbf{x}_R(t) = \int_0^t e^{\mathbf{A}(t-s)} \mathbf{B} u_C(s - \tau_R) ds \quad (3.18)$$

assuming that the initial condition $\mathbf{x}_R(0) = \mathbf{0}$. Equations (3.16)-(3.18) show mathematically how the network delays τ_C and τ_R affect the state trajectory and thus the output of the controlled DC motor.

3.7.2 Effect of the network-induced time delay, τ_c , on the control signal provided by the central controller

The control signal provided by the central controller and network-induced time delay τ_c , time to transmit the measured signal from the remote controller to the central controller, are related via the following equation:

$$u_c(t) = K_p (r(t) - y_R(t - \tau_c)) + K_i \int (r(t) - y_R(t - \tau_c)) dt \quad (3.19)$$

Fig. 3-16 shows the norm of the control signal $\|u_c(t)\|$ versus different values of τ_c , where $\|u_c(t)\|$ is given by [64]:

$$\|u_c(t)\| = \|u_c(t)\|_2 = \sqrt{\sum_{k=0}^L |u_c(k)|^2} \quad (3.20)$$

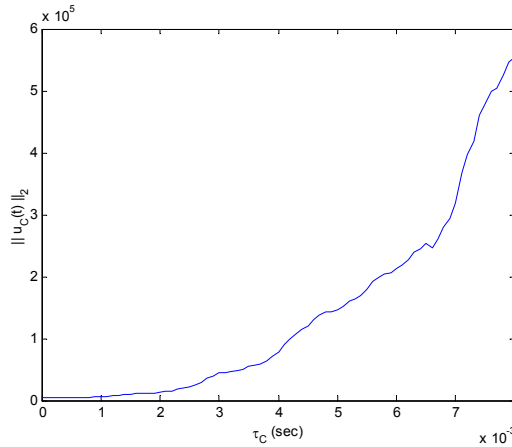


Figure 3-16 The relation between τ_c and control signal $u_c(t)$.

As shown in Fig. 3-16, the network delay τ_c affects the network-based controlled DC motor by increasing the value of the control signal $u_c(t)$ provided by the central controller, which may lead to unstable closed-loop control systems.

The explanation of having such a relation in Fig. 3-16 is as follows: The control signal $u_C(t)$ is dependent on the error signal $e(t)$, which is dependent on the measured output signal of the controlled DC motor $y_R(t)$ coming from the remote site via the network. So, at the beginning of the simulation, $e(t)$ will be a large value, which leads to a large control signal $u_C(t)$. However, as $y_R(t)$ approaches $r(t)$, i.e., $e(t) \approx 0$, there is a time delay τ_C for the central controller to realize that the control signal should be decreased. Within this delay time period, $u_C(t)$ will keep increasing, which will cause $y_R(t)$ to increase to a higher value. Thus, $e(t)$ will be higher than before which leads $u_C(t)$ to resume increasing. This time-delay effect may lead to an unstable closed-loop system response as shown in Fig. 3-15. The stated case is shown in Fig. 3-17, with $r(t) = 50$ (rad/sec).

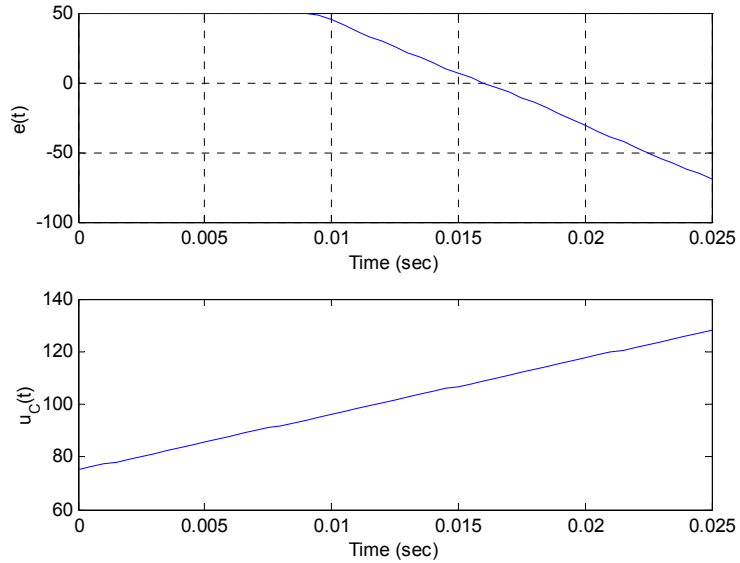


Figure 3-17 Effect network-induced time delays on $e(t)$ and $u_C(t)$.

4 Adaptive Fuzzy Modulation for Networked PI Control Systems: Partial Adaptation

When using networks in control systems, a network-induced time delay is generated, which has adverse effects on the network-based closed-loop control systems such as performance degradation and destabilization as shown in the previous chapter. In this chapter, we propose an intelligent controller using fuzzy logic on top of a PI gain to adaptively compensate for the network-induced time delay in time-delay sensitive networked control system applications. Actually with this proposed fuzzy logic modulator, the existing PI controller will not be completely redesigned, but modulate the PI controller action with the output of the intelligent fuzzy logic controller to compensate for the network-induced time delay effects in the network-based control system. A partial adaptation scheme for this fuzzy logic modulator is also proposed to tune the consequent parameters.

4.1 Direct PI controller parameters updating

In this section, three methods are presented for direct PI controller parameters tuning. The first two tuning methods are on-line, while the third one is off-line selection of the PI controller parameters. In the online tuning methods, PI controller parameters are tuned based on minimizing instantaneous cost function and moving window cost function, respectively. For the off-line selection process, a look-up table of PI controller parameters under different QoS variations is used to select the “optimal” gains having the minimum value of a pre-defined cost function.

4.1.1 On-line tuning: Instantaneous cost function

In the on-line direct PI controller updating, the parameters are updated based on minimizing the instantaneous performance measure $J(k)$,

$$J(k) = e(k)^2 \quad (4.1)$$

where $J(k)$ places an instantaneous penalty on both long response time and poor convergence. One method to minimize $J(k)$ is to use gradient descent [65] to adjust the PI controller parameters accordingly. The parameters are adjusted via steepest descent algorithm:

$$K_p(k+1) = K_p(k) - \eta \frac{\partial J(k)}{\partial K_p} \quad (4.2)$$

$$K_i(k+1) = K_i(k) - \eta \frac{\partial J(k)}{\partial K_i} \quad (4.3)$$

where η is the rate of descent.

Using (3.9), and bilinear approximation [66]:

$$s = \frac{2}{h} \frac{z-1}{z+1}, \quad (4.4)$$

the digital PI controller used is described by the difference equation [67]:

$$u(k) = u(k-1) + \left(K_p + \frac{K_i h}{2} \right) e(k) + \left(-K_p + \frac{K_i h}{2} \right) e(k-1) \quad (4.5)$$

where, $u(k)$ is the control signal provided by the PI controller; $e(k)$ is the plant error signal; h is the sampling time; K_p is the proportional gain of the PI controller, and K_i is the integral gain of the PI controller.

Using (4.1)-(4.5), the parameters are adjusted via steepest descent algorithm as follows:

$$K_p(k+1) = K_p(k) - 4\eta \frac{e(k)^2 - e(k)e(k-1)}{(2K_p(k) + K_i(k)h)} \quad (4.6)$$

$$K_i(k+1) = K_i(k) - 2h\eta \frac{e(k)^2 + e(k)e(k-1)}{(2K_p(k) + K_i(k)h)} \quad (4.7)$$

4.1.2 On-line tuning: Moving window cost function

In this section, another on-line direct PI controller parameters tuning method is presented. This method is based on minimizing a cost function with a moving window. The new cost function is given by:

$$J(k) = \sum_{i=k-m}^k e(i)^2 \quad (4.8)$$

where m is the window size. Applying the gradient descent minimization algorithm to tune the PI controller parameters, we get:

$$K_p(k+1) = K_p(k) - 4\eta \sum_{i=k-m}^k \frac{e(i)^2 - e(i)e(i-1)}{(2K_p(i) + K_i(i)h)} \quad (4.9)$$

$$K_i(k+1) = K_i(k) - 2h\eta \sum_{i=k-m}^k \frac{e(i)^2 + e(i)e(i-1)}{(2K_p(i) + K_i(i)h)} \quad (4.10)$$

With $m = 0$ we return to the instantaneous cost function presented in Section 4.1.1

4.1.2.1 Heuristic PI control action

In this section we will show that by using on-line direct updating for the PI controller parameters in the network-based control systems; we violate the heuristics of PI controller action, and for the off-line direct tuning, we will show that the major drawback of this approach is the weak definition of the searching region for (K_p, K_i) .

From the equation of the digital PI controller in (4.5), we see that two values are affecting the action of the PI controller besides the controller parameters and sampling time, namely $e(k)$ and $e(k-1)$. If we define:

$$\Delta e = e(k) - e(k-1) \quad (4.11)$$

as the difference error signal, we can investigate four regions corresponding to values of e and Δe , and derive the corresponding heuristic actions of the PI controller in these regions [68]. The regions are defined in Table 4-1.

Table 4-1 List of the 4 regions corresponding to e and Δe .

Δe	Negative	Positive
e		
Positive	Region 1	Region 4
Negative	Region 2	Region 3

In Region 1, where e is positive and Δe is negative, we need to increase the system output in order to speed up the system, which corresponds to increasing both K_p and K_i . In Region 2, both e and Δe are negative, and we need to decrease the system output to reduce the overshoot, which corresponds to decreasing both K_p and K_i . In Region 3 where e is negative and Δe is positive, we need to decrease the system output in order to reduce the error between the reference signal and the system output since the system output is greater than the reference signal, which corresponds to decreasing both K_p and K_i . Finally, in Region 4, where both e and Δe are positive, we need to increase the system output in order

to track the reference signal since the system output is less than the reference signal, which corresponds to increasing both K_p and K_i . Table 4-2 summarizes the four regions mentioned and the corresponding K_p and K_i updating.

Table 4-2 Summarization of the 4 regions and the corresponding K_p and K_i updating.

Δe e	Negative	Positive
Positive	Region 1: Increase both K_p and K_i .	Region 4: Increase both K_p and K_i .
Negative	Region 2: Decrease both K_p and K_i .	Region 3: Decrease both K_p and K_i .

4.1.2.2 Drawbacks on using on-line direct PI updating

At the minimum point, $\begin{bmatrix} \frac{\partial J(k)}{\partial K_p} \\ \frac{\partial J(k)}{\partial K_i} \end{bmatrix} = \mathbf{0}$, i.e.,

$$e(k)^2 = e(k)e(k-1) \quad (4.12)$$

and
$$\begin{aligned} e(k)^2 &= -e(k)e(k-1), \\ \Rightarrow e(k) &= -e(k-1) \end{aligned} \quad (4.13)$$

Of course, both (4.12)-(4.13) are satisfied when $e(k) = e(k-1) = 0$, i.e., at the steady state. But due to network effects, where nonlinearity and time varying are inherently included

in the closed-loop system, uncompensated network-based control systems usually have oscillatory or near oscillatory behavior. If we look at (4.13), where the oscillation is occurring, at this instant, no updating will occur for K_i since $\frac{\partial J(k)}{\partial K_i} = 0$, while from (4.2), (4.6), K_p will keep decreasing until we have a pure integrator, which cannot stabilize the system [62]. Another drawback of using on-line direct PI parameters adaptation is illustrated via the following counter example to show that using on-line direct PI adaptation violates the heuristic actions of PI controllers.

4.1.2.2.1 Counterexample: violating heuristics of PI controller action

A DC motor with nominal PI controller parameters of $(K_p, K_i) = (0.01, 30)$, and sampling time $h = 0.01$ msec. is used as a counterexample to show that using on-line direct PI adaptation violates the heuristic actions of PI controllers. From the error curve of step response and the difference signal of the error curve shown in Fig. 4-1,

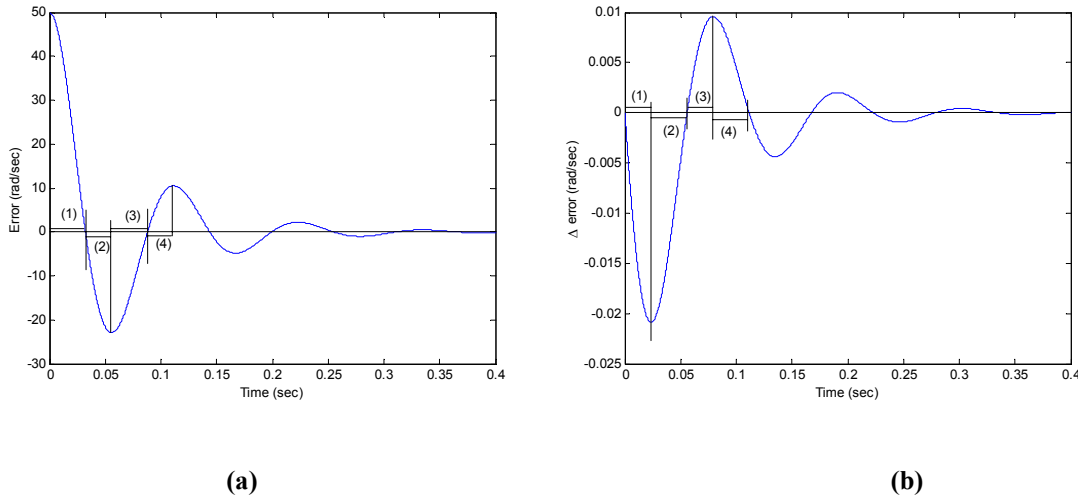


Figure 4-1 (a) Error curve of step response, (b) Difference of the error signal, $\Delta e = e(k) - e(k-1)$.
(1): Region 1, (2): Region 2, (3): Region 3, (4): Region 4.

we observe the following: If we check the adaptation laws for K_p and K_i , we see that (4.2) and (4.6) does not satisfy the requirements on K_p in Region 3 and Region 4, which violates the heuristic actions of the PI controller in Region 3 and Region 4. From (4.3) and (4.7), we see that K_i always decreases in all regions which violates the heuristic actions of the PI controller in Region 1 and Region 4. To show that, we plotted the quantities $\frac{\partial J(k)}{\partial K_p}$, and $\frac{\partial J(k)}{\partial K_i}$ in Fig. 4-2.

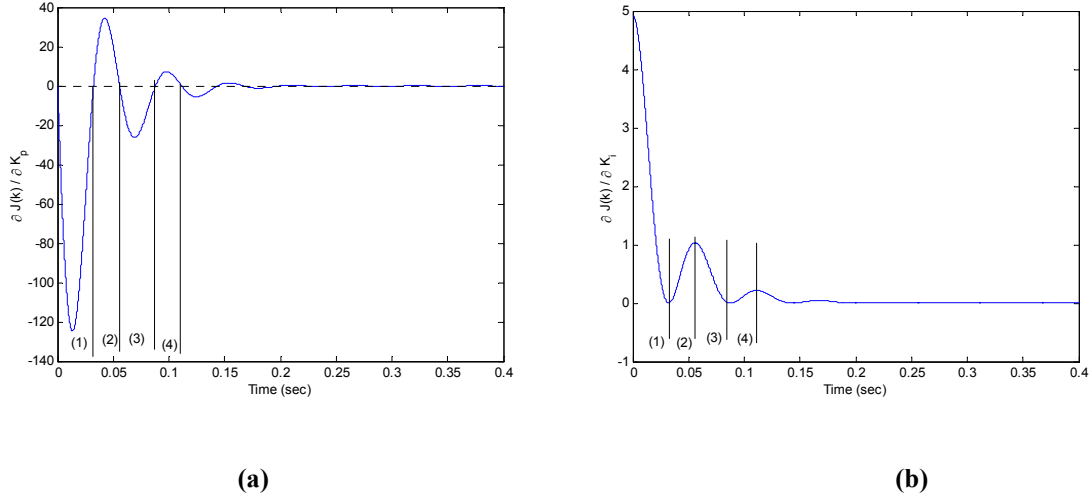


Figure 4-2 (a) Plot of $\frac{\partial J(k)}{\partial K_p}$, (b) Plot of $\frac{\partial J(k)}{\partial K_i}$, (1): Region 1, (2): Region 2, (3): Region 3, (4): Region 4.

It is clear from Fig. 4-2(a), that $\frac{\partial J(k)}{\partial K_p}$ does not change its sign accordingly such that the requirements on the updating of K_p listed in Table 4-2 are satisfied. The on-line direct updating on K_p violates the requirements on K_p in Region 3 and Region 4. On the other hand, we see from Fig. 4-2(b) that $\frac{\partial J(k)}{\partial K_i}$ is always positive, which corresponds to updating

K_i with a smaller value. This violates the requirements on K_i since we need to increase the K_i in Region 1 and Region 4. Thus this updating algorithm violates the purpose of using PI controller.

4.1.3 Off-line PI controller parameters selection

In [38, 57], an approach based on searching for the “optimal” (K_p, K_i) pair given the available network QoS is presented such that a pre-defined/pre-computed cost function is minimized. Here in this section, we will use the same idea but with a modified cost function given by:

$$\mathbf{J} = \lambda \mathbf{J}_1 + (1 - \lambda) \mathbf{J}_2 \quad (4.14)$$

where

$$\mathbf{J}_1(K_p, K_i) = \frac{\sum_{k=0}^N e(k)^2}{\|\mathbf{J}_1(K_p, K_i)\|_\infty}, \quad \mathbf{J}_2(K_p, K_i) = \frac{\sum_{i=1}^M \Delta e_b(i)^2}{\|\mathbf{J}_2(K_p, K_i)\|_\infty}, \quad (4.15)$$

and $\{\Delta e_b(i)\} = \{\Delta e(k) \mid e(k)\Delta e(k) > 0\}$.

$\mathbf{J}_1$ places a penalty on both response time and poor convergence; $\mathbf{J}_2$ places an extra penalty on overshoot, undershoot, and oscillatory behavior, and λ is the weighting parameter between the two sub-cost functions.

This cost function is used to pre-compute the gains and store them in a look-up table, and then this look-up table is used to perform searching for the “optimal” (K_p, K_i) pair corresponding to the minimum cost function value with respect to a given network QoS.

One concern of this approach is the search region for (K_p, K_i) . With a given search region for (K_p, K_i) , we may obtain a (K_p, K_i) pair corresponding to the local minimum of the cost function in this region, but the (K_p, K_i) pair is not necessary the global minimum we are looking for as will illustrated in the following example.

4.1.3.1 Illustration example: Searching region for (K_p, K_i)

With the searching region shown in Fig. 4-3(a), and incremental values of $\Delta K_p = \Delta K_i = 0.01$, the “optimal” (K_p, K_i) pair corresponding to the minimum cost function is $(K_p, K_i) = (0.5, 1)$. The step response of the network-based controlled DC motor subject to tracking a reference signal of 50 (rad/sec) is shown in Fig. 4-3(b). A sampling time of 1 (msec), a gaussian random time delay of mean 5 (msec) and variance 1.6×10^{-6} , and $\lambda = 0.5$ are used.

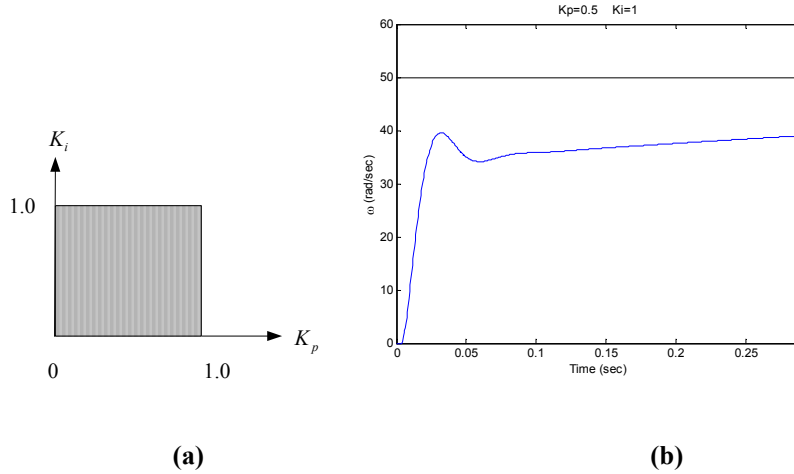


Figure 4-3. (a) Searching region for (K_p, K_i) , (b) Step response of the network-based controlled DC motor with off-line direct PI controller parameters tuning.

As Fig. 4-3(b) shows, the step response takes a long time to settle to its steady state value. With a bigger searching region shown in Fig. 4-4(a), and using incremental values of $\Delta K_p = \Delta K_i = 0.1$, the typical optimal (K_p, K_i) pair is (0.5,5). A better step response of the network-based controlled DC motor subject to tracking a reference signal of 50 (rad/sec) is obtained as shown in Fig. 4-4(b).

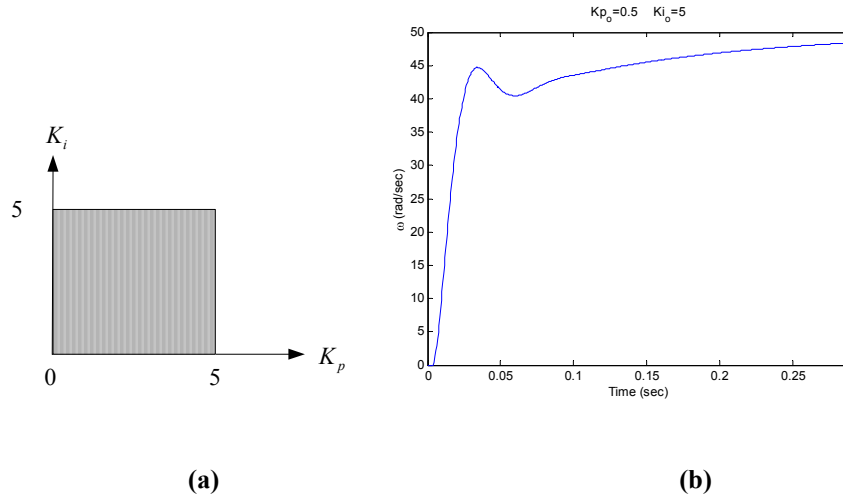


Figure 4-4. (a) Searching region for (K_p, K_i) , (b) Step response of the network-based controlled DC motor with off-line direct PI controller parameters tuning.

As shown in this example, the off-line direct PI controller parameters selection process is highly related to whether the searching region contains the optimal (K_p, K_i) pair or not; also the tuning algorithm is dependent on the incremental values of (K_p, K_i) used in searching for the optimal (K_p, K_i) pair in the searching region such that we don't jump over the optimal pair due to a bad incremental value choice.

4.1.4 Conclusion

Based on the above discussion, we see that by using on-line direct updating for the PI controller parameters in the network-based control systems; we violate the heuristics of PI

controller action. The oscillation (near oscillation) behavior of the network-based controlled plant cannot be compensated for using this direct updating scheme; thus the convergence of the plant output cannot be achieved which leads to destabilizing the plant. For the off-line direct tuning, the major drawback of this approach is the weak definition of the searching region for (K_p, K_i) as stated earlier.

All these drawbacks are good motivations to seek an alternate scheme/algorithm that updates the PI controller parameters on-line such that they can compensate for the network-induced time delay effects. From the controller designer viewpoint, the plant of the network-based control system includes not only the plant to be controlled but also the network that connects the system components. This network part of the plant is quite difficult to handle, because the network system traffic is stochastic in nature, and nonlinear time varying behaviors are inherently included. There is still no accurate mathematical model to describe the network-based control systems behavior.

Fuzzy logic control [69] has been used for control of plants where the plant modeling is difficult, and when conventional control methods have shown limited successes. Fuzzy logic controllers usually work well in an environment with a large uncertainty and unknown variation in plant parameters and structure, when an accurate mathematical model for the system under control is not available, and when human experts are available to provide linguistic fuzzy control rules or linguistic fuzzy descriptions about the system. Due to these reasons, fuzzy logic control becomes a very attractive method for controller design in network-based control systems. In this thesis, we will use a fuzzy logic controller to modulate the PI controller output in order to compensate for the network-induced time delays.

4.2 Fuzzy logic modulator: Design

In this section, a fuzzy logic modulator for network-based PI controller is designed to compensate for the network-induced time-delay effects in the network-based PI control problem [70]. This approach is based on modulating the control signal provided by the PI controller with a single parameter. The input to the fuzzy modulator is the error signal between the reference signal and the plant output signal, and the output of the fuzzy modulator is the modulation parameter used to compensate for the effect of the network-induced time delay in the network-based PI control system.

4.2.1 Network-induced time-delay fuzzy compensation

Our objective is to compensate for the effect of the network-induced time-delay on the network-based PI control without completely redesigning the controller. The approach taken is to use fuzzy logic to modify the nominal PI controller output. A general block diagram of the modified network-based control system is shown in Fig. 4-5. A network-based controlled DC motor described in chapter 3 is used as an example to illustrate the proposed fuzzy logic compensation technique.

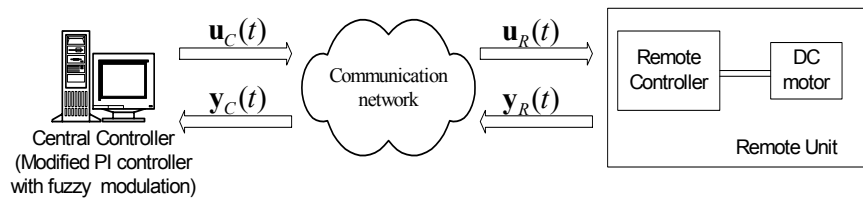


Figure 4-5 A modified network-based PI control system with fuzzy modulation.

In order to compensate and improve the performance of the nominal PI control under different network delays, we introduce a parameter β such that the new control signal provided by the central controller is

$$u_C(t) = \beta u_{PI}(t) \quad (4.16)$$

where $u_C(t)$ is the modulated value of the control signal provided by the PI controller with parameter β . In this work, a methodology based on fuzzification of the new parameter β is presented in order to improve the performance of the network-based PI control. The block diagram of the modified network-based PI control with fuzzy modulation applied to DC motor is shown in Fig. 4-6.

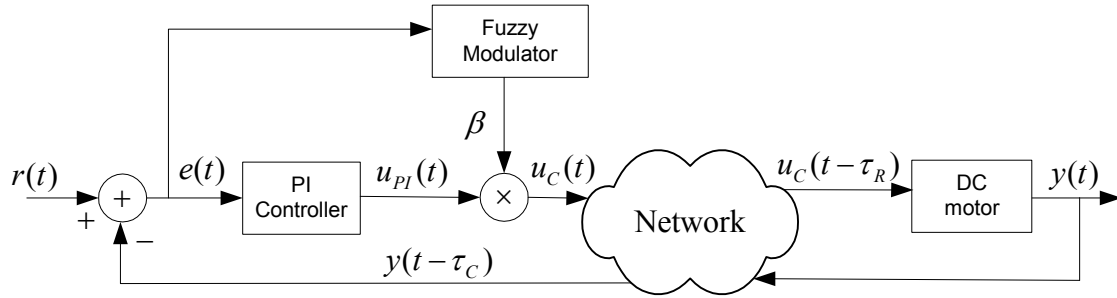


Figure 4-6 Network-based controlled DC motor with fuzzy modulation.

To see the effect of the new modulation parameter on the DC motor state trajectory, the solution of the system described by

$$\dot{\mathbf{x}}_R(t) = \mathbf{A} \mathbf{x}_R(t) + \mathbf{B} u_R(t) \quad (4.17)$$

is

$$\begin{aligned} \mathbf{x}_R(t) &= \int_0^t e^{\mathbf{A}(t-s)} \mathbf{B} u_C(s - \tau_R) ds \\ &= \beta \int_0^t e^{\mathbf{A}(t-s)} \mathbf{B} u_{PI}(s - \tau_R) ds \end{aligned} \quad (4.18)$$

From (4.18), we see that β has a direct relation to the state trajectories and thus the output of the network-based controlled DC motor.

As stated earlier, the modulation parameter β is given by:

$$\beta = \mathbf{h}_{fz}(e(t)) \quad (4.19)$$

where $\mathbf{h}_{fz}$ is a nonlinear function that describes the input/output relation of the fuzzy modulator, and

$$\begin{aligned} e(t) &= r(t) - y_R(t - \tau_C) \\ &= r(t) - y_C(t) \end{aligned} \quad (4.20)$$

is the error signal.

Based on the observation of the DC motor performance w.r.t. the delays simulated in section 3.7, the fuzzy modulator is composed of the following two rules:

$$\begin{aligned} \text{if } e \text{ is Small, then } \beta &= \beta_1. \\ \text{if } e \text{ is Large, then } \beta &= \beta_2. \end{aligned} \quad (4.21)$$

In the first rule, where e is **Small**, we need to modulate the control signal provided by the PI controller $u_{PI}(t)$ by parameter $\beta = \beta_1$ in order to compensate for the network effect. In the second rule, where e is **Large**, in this case, the modulator parameter $\beta = \beta_2$ will not be of the same strength as in the first rule (i.e., $\beta_2 > \beta_1$) since e is **Large** in this case and the system needs higher control signal value $u_{PI}(t)$ to reduce the error signal.

In order to achieve these heuristics we set the following condition:

$$0 < \beta_1 < \beta_2 < 1 \quad (4.22)$$

where $\beta_{i,i=1,2}$, are the consequent parameters corresponding to the modulation parameter β .

4.3 Fuzzy logic modulation: Membership functions tuning

Two membership functions associated with the input variable, e , to the fuzzy modulator are used, namely **Small**, and **Large**. The membership functions for the error signal are defined over the universe of discourse $e \in (-\infty, r]$, where r is the reference signal. The initial setting for the membership functions used for the input variables e to the fuzzy modulator, and the output variable β are shown in Fig. 4-7.

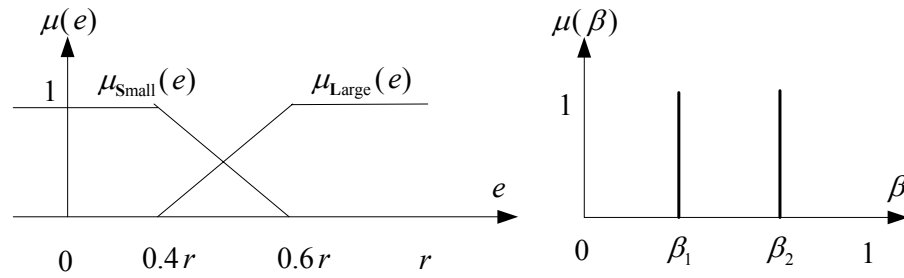


Figure 4-7 Initial setting for the membership functions associated with input variable e and the output variable β .

The output of the fuzzy modulator β is given by the central defuzzification rule [69, 71]:

$$\beta = \frac{\beta_1 \mu_{\text{Small}}(e) + \beta_2 \mu_{\text{Large}}(e)}{\mu_{\text{Small}}(e) + \mu_{\text{Large}}(e)} \quad (4.23)$$

where $\mu_{\text{Small}}(e)$ and $\mu_{\text{Large}}(e)$ are the membership functions describe the input variable e .

As shown in (4.21), we use singleton membership functions in the consequent part of the fuzzy rules, i.e., β_1 and β_2 are variables, the reason is that by using singleton membership functions we reduced the computational time at the modulator, and simplified the partial adaptation process since only two parameters need to be adapted.

4.3.1 PI parameterization

With this type of modulation, we moved from the nominal PI controller with fixed parameters to a new type of PI controllers with variable parameters:

$$\begin{aligned} u_c(t) &= \beta u_{PI}(t) \\ &= \beta \left(K_p e(t) + K_i \int e(t) dt \right) \\ &= \beta K_p e(t) + \beta K_i \int e(t) dt \end{aligned} \tag{4.24}$$

we thus use a new PI controller with variable parameters $(\beta K_p, \beta K_i)$ as a new proportional gain and new integral gain pair, where the parameterization variable $\beta = \mathbf{h}_{fz}(e(t))$ is the output of the fuzzy modulator. It should be noticed that by decreasing gains of the PI controller, a large settling time and rise time are obtained, which in some cases is unacceptable, where a fast response with a rapid rise time and settling time to a step reference command is required such as in unmanned vehicles. In order to properly tune the fuzzy modulator parameters, an adaptation scheme that tunes the fuzzy logic modulator parameters based on minimizing certain performance measure is needed.

4.3.2 Fuzzy logic modulation: Simulation

To demonstrate the effectiveness of this fuzzy modulator, we use an educated guess approach to select β_1 and β_2 , while fixing the membership functions parameters as in the initial setting shown in Fig. 4-7. In this case, we choose $\beta_1 = 0.10$, $\beta_2 = 0.12$, and $r = 50$ (rad/sec). The membership functions associated with the input variable e to the fuzzy modulator and the output variable β are shown in Fig. 4-8.

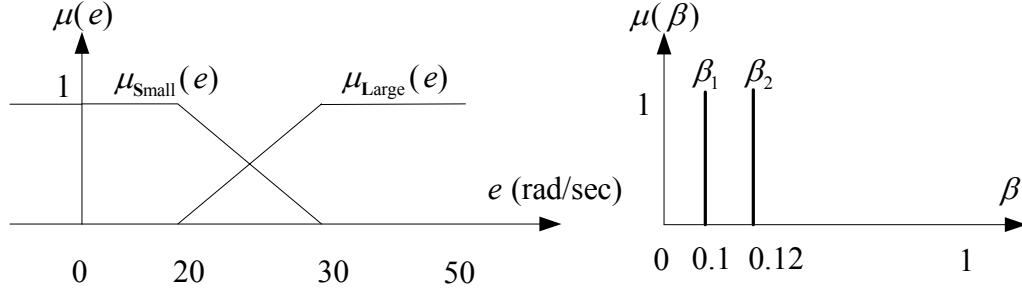


Figure 4-8 Membership functions associated with input variable e and output variable β .

In order to show the effectiveness of the proposed fuzzy modulator, two cases will be examined: smaller time delay, and longer time delays. In smaller time delays, both τ_C and τ_R are less than the sampling time h . Let us set the sampling time of 10 (msec), and set both time delays (τ_C and τ_R) as a gaussian random number with mean value 6,7,8,9 (msec) and variance 1.6×10^{-6} . Fig. 4-9(a) shows a plot of the typical step response of the network-based controlled DC motor subject to tracking a 50 (rad/sec) reference signal with a nominal PI control ($K_{po} = 1.5$, and $K_{io} = 42.5$) modified by fuzzy modulator. For the longer time delays, at least one or both τ_C and τ_R are longer than the sampling time h . Let us set the sampling time of 10 (msec), and set both time delays (τ_C and τ_R) as a gaussian random number with mean value 11,12,13,14 (msec) with variance 1.6×10^{-6} . Fig. 4-9(b) shows a plot of the typical step response of the network-based controlled DC motor subject to tracking a 50 (rad/sec) reference signal with a nominal PI control ($K_{po} = 1.5$, and $K_{io} = 42.5$) modified by fuzzy modulator.

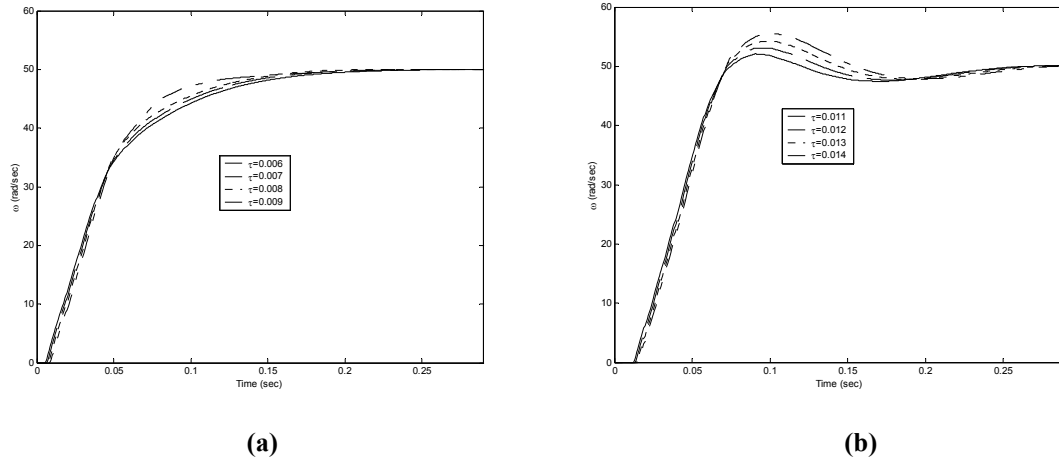


Figure 4-9 Step response of the network-based controlled DC motor with fuzzy modulation, (a) Smaller time delays, (b) Longer time delays.

We see from Fig. 4-9 how the proposed PI controller with fuzzy modulator improves the performance and tends the stability of the network-based controlled DC motor. Also we can see how this approach can handle effectively the cases of smaller as well as longer network-induced time delays. The reason behind that is the following: by reducing/modulating the input voltage value, i.e., reducing the PI parameter gains, this prevent the network-based controlled DC motor from going to unstable region, and at the same time an acceptable step response is obtained.

The cost of this approach is the tuning process, i.e., an educated guess, which is a time consuming process and does not guarantee the optimal setting for the consequent parameters. This is a major motivation to seek an adaptation law/scheme that can tune the membership functions parameters and consequent parameters such that a certain performance index is minimized. In this section, only the educated guess approach is used for tuning the consequent parameters while fixing the membership functions parameters as in the initial

setting. In the next section, we will develop an adaptation law/scheme for the consequent parameters tuning such that the certain performance measure is minimized.

4.4 Partial adaptation: consequent parameters tuning

In this section, three consequent parameters tuning methods are presented [72]. The first two tuning methods are on-line, while the third one is off-line consequent parameters selection. In the online tuning methods, consequent parameters are tuned based on minimizing instantaneous cost function and moving window cost function, respectively. For the off-line consequent parameters selection process, a look-up table of the consequent parameters under different QoS variations is used to select the “optimal” (β_1, β_2) pair having the minimum value of a pre-defined/pre-computed cost function. The following approaches present the case in which only the consequent parameters are to be tuned, while the membership functions parameters of the input variable e are fixed with the initial setting (this is the reason we named this approach as a fuzzy logic modulator with partial adaptation). Adaptation on both membership functions parameters in the antecedent part of the fuzzy rules and the consequent parameters will be presented in chapter 5.

4.4.1 On-line tuning: Instantaneous cost function

In the on-line partial AFM, the consequent parameters are updated based on minimizing the instantaneous performance measure $J(k)$,

$$J(k) = e(k)^2 \quad (4.25)$$

where $J(k)$ places an instantaneous penalty on both long response time and poor convergence. The consequent parameters are adjusted via steepest descent algorithm:

$$\beta_1(k+1) = \beta_1(k) + 2\eta \frac{e(k) \mu_{\text{Small}}(e(k)) u_{PI}(k)}{\left(K_p(k) + \frac{K_I(k)h}{2}\right) (\beta_1(k) \mu_{\text{Small}}(e(k)) + \beta_2(k) \mu_{\text{Large}}(e(k)))} \quad (4.26)$$

$$\beta_2(k+1) = \beta_2(k) + 2\eta \frac{e(k) \mu_{\text{Large}}(e(k)) u_{PI}(k)}{\left(K_p(k) + \frac{K_I(k)h}{2}\right) (\beta_1(k) \mu_{\text{Small}}(e(k)) + \beta_2(k) \mu_{\text{Large}}(e(k)))} \quad (4.27)$$

where η is the rate of descent.

4.4.2 On-line tuning: Moving window cost function

Another consequent parameters on-line tuning method based on minimizing a cost function with moving window is presented. The cost function is given by:

$$J(k) = \sum_{i=k-m}^k e(i)^2 \quad (4.28)$$

where m is the window size. Applying the gradient descent minimization algorithm to tune the consequent parameters, we get:

$$\beta_1(k+1) = \beta_1(k) + 2\eta \sum_{i=k-m}^k \frac{e(i) \mu_{\text{Small}}(e(i)) u_{PI}(i)}{\left(K_p(i) + \frac{K_I(i)h}{2}\right) (\beta_1(i) \mu_{\text{Small}}(e(i)) + \beta_2(i) \mu_{\text{Large}}(e(i)))} \quad (4.29)$$

$$\beta_2(k+1) = \beta_2(k) + 2\eta \sum_{i=k-m}^k \frac{e(i) \mu_{\text{Large}}(e(i)) u_{PI}(i)}{\left(K_p(i) + \frac{K_I(i)h}{2}\right) (\beta_1(i) \mu_{\text{Small}}(e(i)) + \beta_2(i) \mu_{\text{Large}}(e(i)))} \quad (4.30)$$

4.4.3 Off-line consequent parameters selection

This section proposes a method of selecting the fuzzy modulation consequent parameters based on searching for the minimum of a pre-defined/pre-computed cost function. In order to select fuzzy membership functions parameters and consequent parameters, the following cost function [67] is defined:

$$\mathbf{J} = \lambda \mathbf{J}_1 + (1 - \lambda) \mathbf{J}_2 \quad (4.31)$$

where λ is the weighting parameter between the two sub-cost functions $\mathbf{J}_1$ and $\mathbf{J}_2$,

$$\mathbf{J}_1(\beta_1, \beta_2) = \frac{\sum_{k=0}^N e(k)^2}{\|\mathbf{J}_1(\beta_1, \beta_2)\|_\infty}, \quad \mathbf{J}_2(\beta_1, \beta_2) = \frac{\sum_{i=1}^M \Delta e_b(i)^2}{\|\mathbf{J}_2(\beta_1, \beta_2)\|_\infty}, \quad (4.32)$$

$0 < \beta_1 < \beta_2 < 1$, and $\{\Delta e_b(i)\} = \{\Delta e(k) \mid e(k)\Delta e(k) > 0\}$.

The algorithm is based on fixing the membership functions parameters of the input variable e as in the initial setting shown in Fig. 4-7, and then using a look-up table to select the proper β_1 and β_2 corresponding to the minimum cost function value. The flowchart of the off-line consequent parameters selection process is shown in Fig. 4-10.

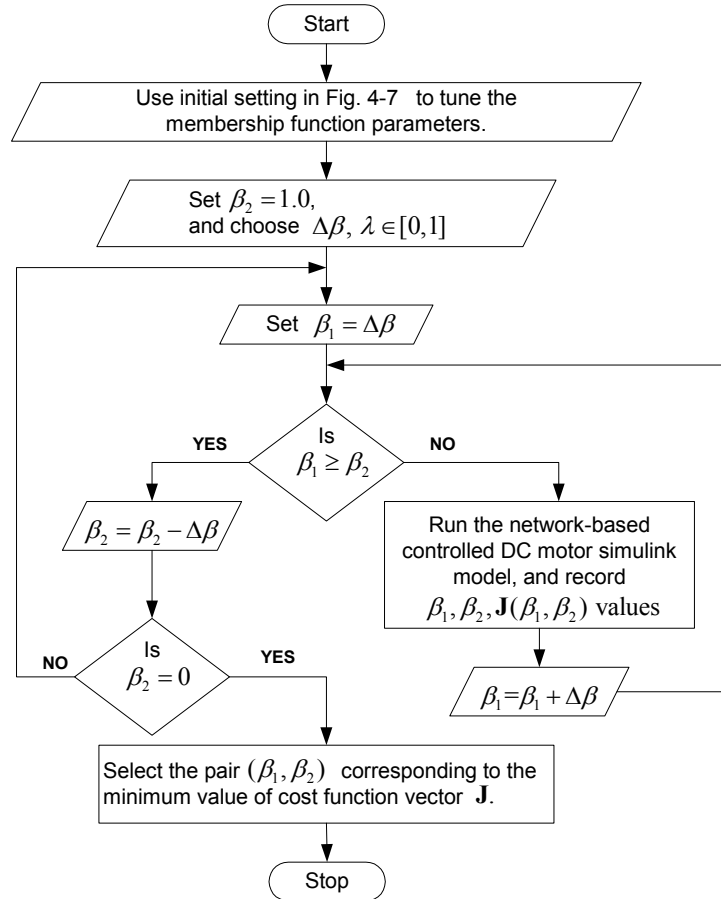


Figure 4-10. Flowchart of the off-line consequent parameters selection process.

In Fig. 4-10, $\Delta\beta$ is the increment value used for the variables β_1 and β_2 during the off-line calculation. We should mention here that unlike the direct off-line PI controller parameters selection process presented in section 4.1.3, the searching region for the optimal (β_1, β_2) pair is well defined as shown in Fig. 4-11.

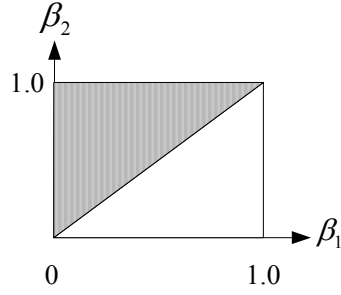


Figure 4-11. Searching region for the optimal (β_1, β_2) pair in partial AFM.

Five cases corresponding to different values of weighting parameter λ are illustrated. In all cases, a sampling time of 1 (msec), a random network-induced time delay of mean 5 (msec) and variance 1.6×10^{-6} , and an increment value $\Delta\beta = 0.01$ are used. Table 4-3 lists the weighting parameter λ and the corresponding optimal (β_1, β_2) pair, while Fig. 4-12 depicts a plot of the typical step response of the network-based controlled DC motor subject to tracking a 50 (rad/sec) reference signal with a modified PI control ($K_{po} = 1.5$, and $K_{io} = 42.5$) with fuzzy modulator corresponding to different values of λ .

Table 4-3. Different values of λ and the corresponding optimal (β_1, β_2) pair

λ	(β_1, β_2)
0	(0.21,0.24)
0.25	(0.24,0.29)
0.5	(0.24,0.30)
0.75	(0.25,0.30)
1	(0.26,0.30)

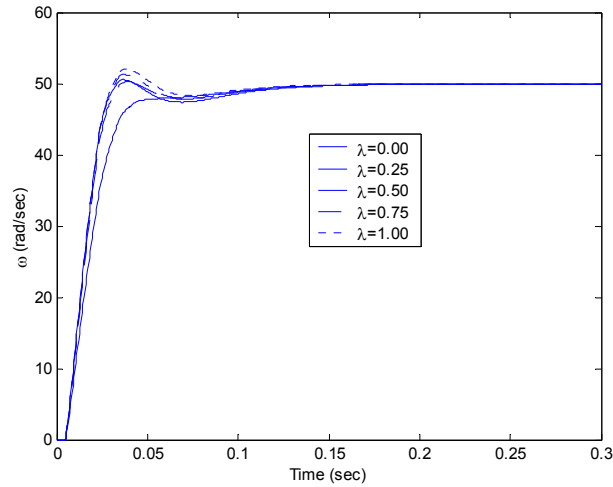


Figure 4-12. Step response of the network-based controlled DC motor with fuzzy modulation corresponding to different values of λ .

We can see by comparing step responses in Figs. 4-9 with Fig. 4-12, that the fuzzy logic modulator with partial adaptation improves the transient response over the fuzzy logic modulator with educated guess tuning. The transient response is improved by reducing both rise time and settling time. For the percentage maximum overshoot, it is possible to improve

it by putting more cost function weight on the overshoot, undershoot, and oscillatory behaviors, (e.g., $\lambda = 0$, as shown in Fig. 4-12).

4.4.3.1 Compensation technique

The adaptive fuzzy logic compensator is described as follows: The controller monitors the network QoS via appropriate middleware, and then adjusts the fuzzy logic consequent parameters (β_1, β_2) to the consequent parameters pair corresponding to the minimum cost function with a fixed initial membership functions parameters. Look-up table technique is used to perform searching on the optimal consequent parameters (β_1, β_2) . For every (h, τ) pair there is an optimal consequent parameters pair (β_1, β_2) corresponding to the minimum cost function with a given λ . Linear interpolation is needed when the exact entry cannot be found from the table with the given inputs.

4.5 Simulation Results and Discussion

The network-based controlled DC motor system is simulated using MATLAB/SIMULINK. The motor state space description in (3.3)-(3.4) is used as the plant, and is controlled by the PI controller in (3.14). The network-induced time delays as described in (3.12)-(3.13) are simulated using gaussian random number generators.

4.5.1 On-line tuning with instantaneous cost function

Using the adaptation laws in (4.6)-(4.7) to on-line tune (K_p, K_i) pair, and adaptation laws in (4.26)-(4.27) for on-line (β_1, β_2) tuning, the typical step responses of the network-based controlled DC motor are shown in Fig. 4-13.

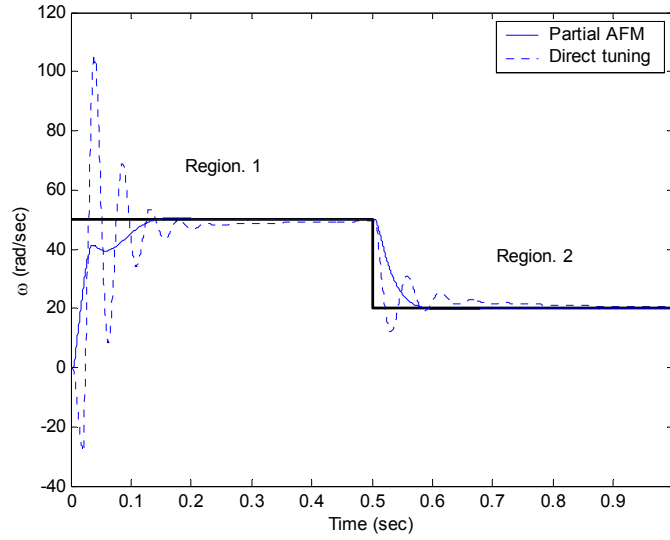


Figure 4-13. Step response of the network-based controlled DC motor with on-line updating based on minimizing instantaneous cost function.

The simulation results clearly indicate the improvement in the step response performance when we use the proposed partial AFM over the direct PI controller parameters tuning. With the proposed partial AFM, we obtained faster settling time with no overshoot and with zero steady state error. In Fig. 4-13, 14, 15, Region.1 and Region.2 describe different reference signal (r), sampling time (h), and delay time (τ) values used in the simulation:

Region. 1: $r = 50(\text{rad/sec})$, $h = 1(\text{msec})$, and $\text{mean}(\tau) = 5 (\text{msec})$.

Region. 2: $r = 20(\text{rad/sec})$, $h = 3(\text{msec})$, and $\text{mean}(\tau) = 6 (\text{msec})$.

4.5.2 On-line tuning with moving window cost function

With the adaptation laws for tuning (K_p, K_i) pair in (4.9)-(4.10), and adaptation laws for tuning (β_1, β_2) pair in (4.29)-(4.30), the typical step responses of the network-based

controlled DC motor with on-line updating based on minimizing the moving window cost function with different window sizes are shown in Fig. 4-14.

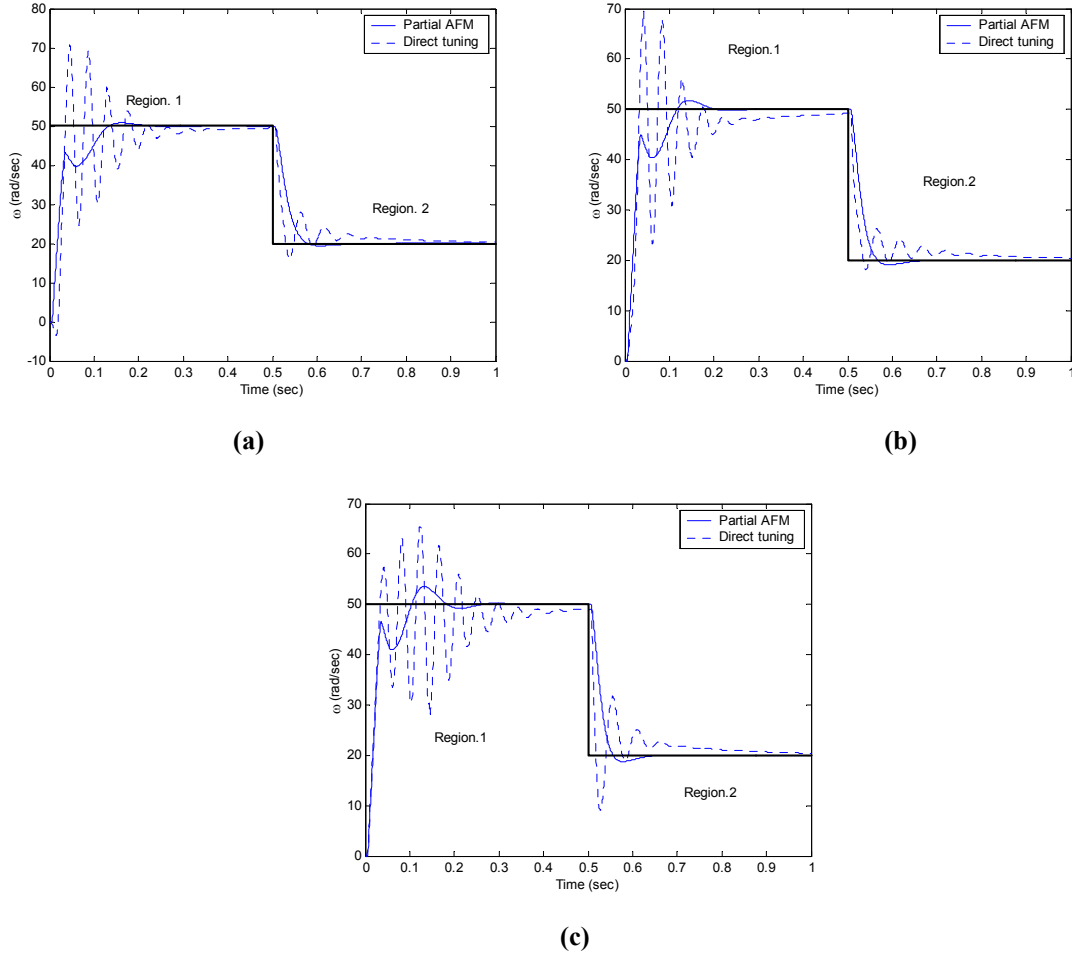


Figure 4-14. Step response of the network-based controlled DC motor with on-line updating based on minimizing moving window cost function: (a) $m=5$, (b) $m=10$, (c) $m=20$.

With the proposed partial AFM, and among these step responses, the best step response performance obtained is when window size $m=5$. By best step response performance, we mean the fastest rise time, fastest settling time, and minimum overshoot. As m increases, the step responses become worse (relative to $m=5$) in both adaptive

controllers. This implies that window size selection plays an important role in this on-line tuning algorithm that needs to be carefully chosen.

4.5.3 Off-line controller parameters selection

The adaptive controllers presented in Section 4.1.3 and 4.4.3 work as follows: The controller monitors the network QoS via appropriate middleware, and then adjusts the controller parameters (either PI controller parameters or fuzzy modulator consequent parameters) to the pair corresponding to the minimum pre-defined/pre-computed cost function. A look-up table technique is used to perform searching for the optimal controller parameters. Linear interpolation is needed when the exact entry cannot be found from the table with the given inputs. The typical step responses of the network-based controlled DC motor are shown in Fig. 4-15. In order to select the PI controller parameters, we use the following searching region: $K_p \in (0.01, 0.6)$ with $\Delta K_p = 0.01$, and $K_i \in (0.1, 3)$ with $\Delta K_i = 0.1$, while for the AFM parameters selection, we apply the proposed off-line algorithm in section 4.4.3 with $\Delta\beta = 0.01$.

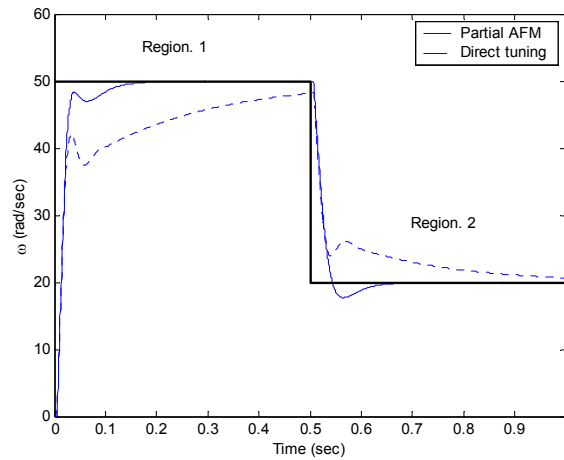


Figure 4-15. Step response of the network-based controlled DC motor with off-line selection based on minimizing pre-computed cost function.

The simulation results clearly indicate the advantages and superb performance of using partial AFM to stabilize the closed-loop system as well as to maintain the system performance while the network QoS is varying.

4.6 Conclusion

In this chapter, we proposed the use of real-time PI gain adaptation to compensate for the network effects in the NCS applications. We first showed an adaptation law to directly tune the PI controller parameters, and then we proposed an intelligent controller using fuzzy logic to compensate for the network-induced time delay in time-delay sensitive NCS applications. A partial adaptation scheme for this fuzzy logic modulator is proposed to tune the consequent parameters. This solved part of the adaptation problem associated with the fuzzy logic modulator parameters tuning since only the consequent parameters are to be tuned, while the membership functions parameters are fixed with the initial setting. In Chapter 5, we will extend this idea to a fully adaptive AFM where both consequent parameters and membership functions parameters in the antecedent part are adaptively tuned.

5 Adaptive Fuzzy Modulation for networked PI control systems: Full Adaptation

In chapter 4 of this thesis, we on-line/off-line adapt the consequent parameters in the fuzzy rules (i.e., (β_1, β_2) pair). In this chapter, both the consequent parameters and the membership functions parameters in the antecedent parts of the fuzzy rules will be tuned adaptively.

In this chapter, we propose a full adaptive fuzzy modulation and compare it with two other tuning methods to compensate for the network-induced time delays in networked PI control systems. In the first tuning method, we adaptively tune the PI controller parameters directly. In the second tuning method, we use partial Adaptive Fuzzy Modulation (AFM), i.e., tune the consequent parameters in the fuzzy rules only with fixed membership function parameters in the antecedent part and fixed PI controller parameters to their nominal values. In the full AFM, not only are the consequent parameters be tuned adaptively, but also the membership functions parameters in the antecedent part are tuned adaptively. Numerical simulation of a network-based controlled DC motor is used to illustrate the effectiveness of the proposed full AFM over other controllers.

5.1 Full adaptation

In this section, three tuning methods for the adaptive fuzzy modulator parameters are presented [73]. The first two tuning methods are on-line, while the third one is off-line AFM parameters selection. In the online tuning methods, the AFM parameters are tuned based on minimizing instantaneous cost function and moving window cost function, respectively. For

the off-line AFM parameters selection process, a look-up table of the AFM parameters under different QoS variations is used to select the “optimal” AFM parameters having the minimum value of a pre-defined/pre-computed cost function. The parameters that need to be tuned adaptively in the full adaptation process are the fuzzy membership functions parameters in $\mu_{\text{Small}}(e)$ and $\mu_{\text{Large}}(e)$, and the consequent parameters β_1 and β_2 . The membership functions for the error signal are defined over the universe of discourse $e \in (-\infty, r]$, where r is the reference signal. The fuzzy membership functions with adjustable parameters associated with the input variable e and the output variable β are shown in Fig. 5-1.

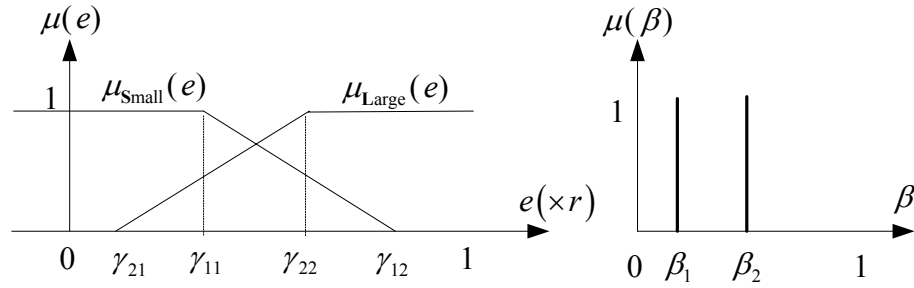


Figure 5-1. Membership functions associated with the input variable e , and the output variable β .

In [74, 75], some heuristic constraints on the membership functions were derived. The heuristic constraints are:

C1. Existence of prototype point constraint, i.e.,

$$\max(\mu_{\text{Small}}(e)) = 1, \max(\mu_{\text{Large}}(e)) = 1$$

C2. Convex Constraint: the membership functions $\mu_{\text{Small}}(e)$ and $\mu_{\text{Large}}(e)$ satisfy the convex constraint as shown in Fig. 5-1.

C3. Leftmost/Rightmost membership function constraint, i.e.,

$$\mu_{\text{Small}}(0) = 1, \mu_{\text{Large}}(r) = 1$$

C4. Overlap constraint: Usually the membership functions should overlap at least 25%. In order to achieve that, the following inequality should satisfy:

$$1 - \frac{\gamma_{21} + (\gamma_{22} - \gamma_{21})\alpha_c}{\gamma_{12} - (\gamma_{12} - \gamma_{11})\alpha_c} \geq \varepsilon \quad (5.1)$$

where $\varepsilon \geq 0.25$ is the overlapping level value, and α_c is the α -level cut value [71].

Some of these constraints are inherently satisfied (C1-C3) due to the chosen structure of the membership functions in Fig. 5-1. In order to satisfy the above heuristic constraints on the membership functions during the adaptation process, the following constraints on the membership functions parameters in the antecedent part of the fuzzy rules and consequent parameters should satisfy:

$$\begin{aligned} 0 < \gamma_{21} < \gamma_{11} < \gamma_{22} < \gamma_{12} < 1, \quad \text{and} \\ 0 < \beta_1 < \beta_2 < 1 \end{aligned} \quad (5.2)$$

5.1.1 On-line tuning: Instantaneous cost function

In the on-line full AFM parameters tuning, the AFM parameters are updated based on minimizing the instantaneous performance measure $J(k)$,

$$J(k) = e(k)^2 \quad (5.3)$$

where $J(k)$ places an instantaneous penalty on both response time and poor convergence.

The AFM parameters are adjusted via steepest descent algorithm:

$$\begin{aligned} \gamma_{11}(k+1) = \gamma_{11}(k) + 2\eta \frac{e(k) \mu_{\text{Large}}(e(k)) u_{PI}(k) (\beta_1(k) - \beta_2(k)) (\gamma_{12}(k) - \gamma_{11}(k))}{\left(K_p(k) + \frac{K_I(k)h}{2}\right) (\beta_1(k) \mu_{\text{Small}}(e(k)) + \beta_2(k) \mu_{\text{Large}}(e(k))) (\mu_{\text{Small}}(e(k)) + \mu_{\text{Large}}(e(k))) (\gamma_{12}(k) - \gamma_{11}(k))^2} \\ \Pi\left(\frac{e(k)}{r(k)}, \gamma_{11}(k), \gamma_{12}(k)\right) \end{aligned} \quad (5.4)$$

$$\gamma_{12}(k+1) = \gamma_{12}(k) + 2\eta \frac{e(k) \mu_{\text{Large}}(e(k)) u_{PI}(k) (\beta_1(k) - \beta_2(k)) \left(\frac{e(k)}{r(k)} - \gamma_{11}(k) \right)}{\left(K_p(k) + \frac{K_I(k)h}{2} \right) (\beta_1(k) \mu_{\text{Small}}(e(k)) + \beta_2(k) \mu_{\text{Large}}(e(k))) (\mu_{\text{Small}}(e(k)) + \mu_{\text{Large}}(e(k))) (\gamma_{12}(k) - \gamma_{11}(k))^2} \quad (5.5)$$

$$\Pi\left(\frac{e(k)}{r(k)}, \gamma_{11}(k), \gamma_{12}(k)\right)$$

$$\gamma_{21}(k+1) = \gamma_{21}(k) + 2\eta \frac{e(k) \mu_{\text{Small}}(e(k)) u_{PI}(k) (\beta_2(k) - \beta_1(k)) \left(\frac{e(k)}{r(k)} - \gamma_{22}(k) \right)}{\left(K_p(k) + \frac{K_I(k)h}{2} \right) (\beta_1(k) \mu_{\text{Small}}(e(k)) + \beta_2(k) \mu_{\text{Large}}(e(k))) (\mu_{\text{Small}}(e(k)) + \mu_{\text{Large}}(e(k))) (\gamma_{22}(k) - \gamma_{21}(k))^2} \quad (5.6)$$

$$\Pi\left(\frac{e(k)}{r(k)}, \gamma_{21}(k), \gamma_{22}(k)\right)$$

$$\gamma_{22}(k+1) = \gamma_{22}(k) + 2\eta \frac{e(k) \mu_{\text{Small}}(e(k)) u_{PI}(k) (\beta_2(k) - \beta_1(k)) \left(\frac{e(k)}{r(k)} - \gamma_{21}(k) \right)}{\left(K_p(k) + \frac{K_I(k)h}{2} \right) (\beta_1(k) \mu_{\text{Small}}(e(k)) + \beta_2(k) \mu_{\text{Large}}(e(k))) (\mu_{\text{Small}}(e(k)) + \mu_{\text{Large}}(e(k))) (\gamma_{22}(k) - \gamma_{21}(k))^2} \quad (5.7)$$

$$\Pi\left(\frac{e(k)}{r(k)}, \gamma_{21}(k), \gamma_{22}(k)\right)$$

$$\beta_1(k+1) = \beta_1(k) + 2\eta \frac{e(k) \mu_{\text{Small}}(e(k)) u_{PI}(k)}{\left(K_p(k) + \frac{K_I(k)h}{2} \right) (\beta_1(k) \mu_{\text{Small}}(e(k)) + \beta_2(k) \mu_{\text{Large}}(e(k)))} \quad (5.8)$$

$$\beta_2(k+1) = \beta_2(k) + 2\eta \frac{e(k) \mu_{\text{Large}}(e(k)) u_{PI}(k)}{\left(K_p(k) + \frac{K_I(k)h}{2} \right) (\beta_1(k) \mu_{\text{Small}}(e(k)) + \beta_2(k) \mu_{\text{Large}}(e(k)))} \quad (5.9)$$

where η is the rate of descent, and

$$\Pi(x, x_1, x_2) = \begin{cases} 1 & x_1 \leq x \leq x_1 \\ 0 & \text{otherwise} \end{cases} \quad (5.10)$$

5.1.2 On-line tuning: Moving window cost function

Another on-line tuning method for AFM parameters based on minimizing a cost function with moving window is presented. The cost function is given by:

$$J(k) = \sum_{i=k-m}^k e(i)^2 \quad (5.11)$$

where m is the window size. Applying the gradient descent minimization algorithm to tune the AFM parameters, we get:

$$\gamma_{11}(k+1) = \gamma_{11}(k) + 2\eta \sum_{i=k-m}^k \frac{e(i) \mu_{\text{Large}}(e(i)) u_{PI}(i) (\beta_1(i) - \beta_2(i)) \left(\frac{e(i)}{r(i)} - \gamma_{11}(i) \right)}{\left(K_p(i) + \frac{K_I(i)h}{2} \right) (\beta_1(i) \mu_{\text{Small}}(e(i)) + \beta_2(i) \mu_{\text{Large}}(e(i))) (\mu_{\text{Small}}(e(i)) + \mu_{\text{Large}}(e(i))) (\gamma_{12}(i) - \gamma_{11}(i))^2} \quad (5.12)$$

$$\Pi\left(\frac{e(i)}{r(i)}, \gamma_{11}(i), \gamma_{12}(i)\right)$$

$$\gamma_{12}(k+1) = \gamma_{12}(k) + 2\eta \sum_{i=k-m}^k \frac{e(i) \mu_{\text{Large}}(e(i)) u_{PI}(i) (\beta_1(i) - \beta_2(i)) \left(\frac{e(i)}{r(i)} - \gamma_{11}(i) \right)}{\left(K_p(i) + \frac{K_I(i)h}{2} \right) (\beta_1(i) \mu_{\text{Small}}(e(i)) + \beta_2(i) \mu_{\text{Large}}(e(i))) (\mu_{\text{Small}}(e(i)) + \mu_{\text{Large}}(e(i))) (\gamma_{12}(i) - \gamma_{11}(i))^2} \quad (5.13)$$

$$\Pi\left(\frac{e(i)}{r(i)}, \gamma_{11}(i), \gamma_{12}(i)\right)$$

$$\gamma_{21}(k+1) = \gamma_{21}(k) + 2\eta \sum_{i=k-m}^k \frac{e(i) \mu_{\text{Small}}(e(i)) u_{PI}(i) (\beta_2(i) - \beta_1(i)) \left(\frac{e(i)}{r(i)} - \gamma_{22}(i) \right)}{\left(K_p(i) + \frac{K_I(i)h}{2} \right) (\beta_1(i) \mu_{\text{Small}}(e(i)) + \beta_2(i) \mu_{\text{Large}}(e(i))) (\mu_{\text{Small}}(e(i)) + \mu_{\text{Large}}(e(i))) (\gamma_{22}(i) - \gamma_{21}(i))^2} \quad (5.14)$$

$$\Pi\left(\frac{e(i)}{r(i)}, \gamma_{21}(i), \gamma_{22}(i)\right)$$

$$\gamma_{22}(k+1) = \gamma_{22}(k) + 2\eta \sum_{i=k-m}^k \frac{e(i) \mu_{\text{Small}}(e(i)) u_{PI}(i) (\beta_2(i) - \beta_1(i)) \left(\gamma_{21}(i) - \frac{e(i)}{r(i)} \right)}{\left(K_p(i) + \frac{K_I(i)h}{2} \right) (\beta_1(i) \mu_{\text{Small}}(e(i)) + \beta_2(i) \mu_{\text{Large}}(e(i))) (\mu_{\text{Small}}(e(i)) + \mu_{\text{Large}}(e(i))) (\gamma_{22}(i) - \gamma_{21}(i))^2} \quad (5.15)$$

$$\Pi\left(\frac{e(i)}{r(i)}, \gamma_{21}(i), \gamma_{22}(i)\right)$$

$$\beta_1(k+1) = \beta_1(k) + 2\eta \sum_{i=k-m}^k \frac{e(i) \mu_{\text{Small}}(e(i)) u_{PI}(i)}{\left(K_p(i) + \frac{K_I(i)h}{2} \right) (\beta_1(i) \mu_{\text{Small}}(e(i)) + \beta_2(i) \mu_{\text{Large}}(e(i)))} \quad (5.16)$$

$$\beta_2(k+1) = \beta_2(k) + 2\eta \sum_{i=k-m}^k \frac{e(i) \mu_{\text{Large}}(e(i)) u_{PI}(i)}{\left(K_p(i) + \frac{K_I(i)h}{2} \right) (\beta_1(i) \mu_{\text{Small}}(e(i)) + \beta_2(i) \mu_{\text{Large}}(e(i)))} \quad (5.17)$$

5.1.3 Off-line tuning: Using lookup table

This section proposes a method of selecting the fuzzy logic modulator parameters based on searching for the minimum of a pre-defined/pre-computed cost function. In order to select fuzzy logic modulator parameters, the following cost function [67] is defined:

$$\mathbf{J} = \lambda \mathbf{J}_1 + (1 - \lambda) \mathbf{J}_2 \quad (5.18)$$

where λ is the weighting parameter between the sub-cost functions $\mathbf{J}_1$ and $\mathbf{J}_2$,

$$\mathbf{J}_1(p) = \frac{\sum_{k=0}^N e(k)^2}{\|\mathbf{J}_1(p)\|_\infty}, \quad \mathbf{J}_2(p) = \frac{\sum_{i=1}^M \Delta e_b(i)^2}{\|\mathbf{J}_2(p)\|_\infty}, \quad (5.19)$$

and $p = [\beta_1, \beta_2, \gamma_{11}, \gamma_{12}, \gamma_{21}, \gamma_{22}]$, $\{\Delta e_b(i)\} = \{\Delta e(k) \mid e(k) \Delta e(k) > 0\}$.

Thus, by full adaptation, we mean that tuning the fuzzy logic membership function parameters in the antecedent part of the fuzzy rules $(\gamma_{11}, \gamma_{12}, \gamma_{21}, \gamma_{22})$, and the consequent

parameters (β_1, β_2) such that the performance measure in (5.18) is minimized. The flowchart of the off-line full AFM parameters selection process is shown in Fig. 5-2.

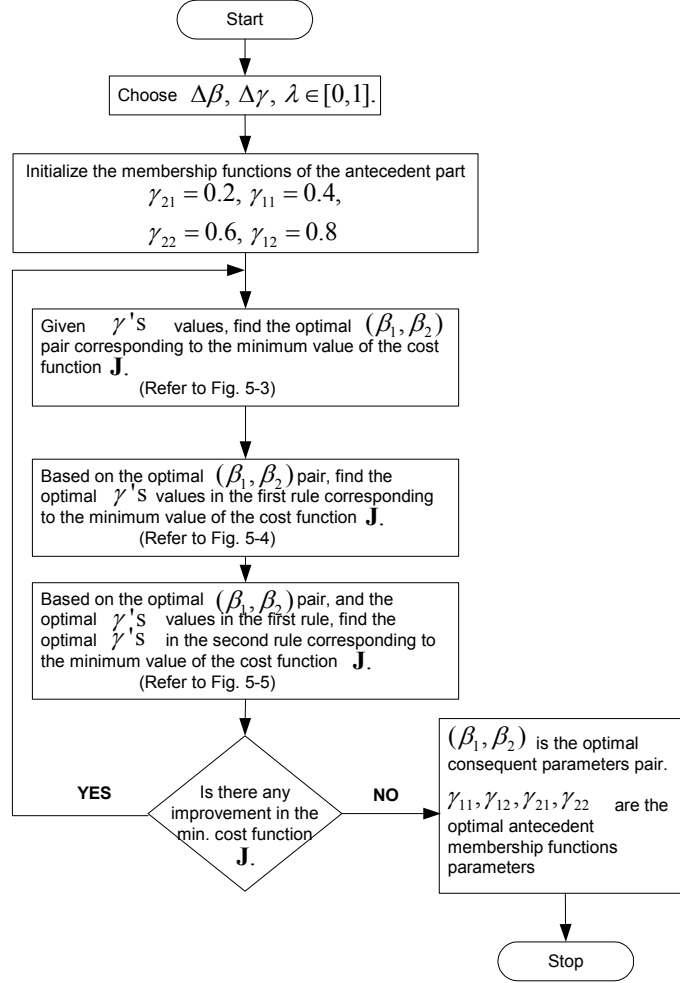


Figure 5-2. Flowchart of the off-line full AFM parameters selection process.

5.1.3.1 Optimal (β_1, β_2) pair selection process

This problem is similar to the partial adaptation problem proposed in chapter 4 of this thesis with small modifications in the adaptation process. The off-line consequent parameters selection process is shown in Fig. 5-3.

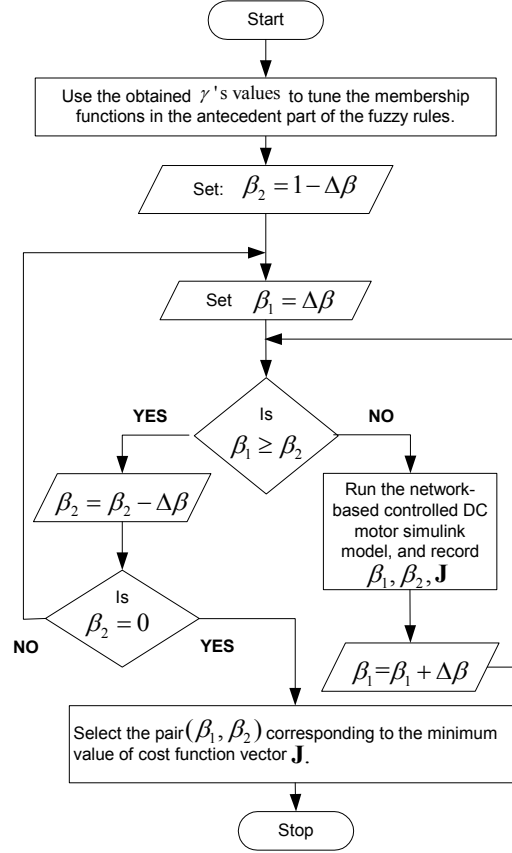


Figure 5-3. Flowchart of the off-line (β_1, β_2) selection process.

In Fig. 5-3, $\Delta\beta$ is the increment value used for the variables β_1 and β_2 during the off-line calculation.

5.1.3.2 Optimal Membership function parameters

The adaptation process to tune the fuzzy membership function parameters in the antecedent part of the first fuzzy rule (i.e., $(\gamma_{11}, \gamma_{12})$ pair) is shown in figure 5-4(a), while the adaptation process to tune the fuzzy membership function parameters in the antecedent part of the second fuzzy rule (i.e., $(\gamma_{21}, \gamma_{22})$ pair) is shown in figure 5-4(b).

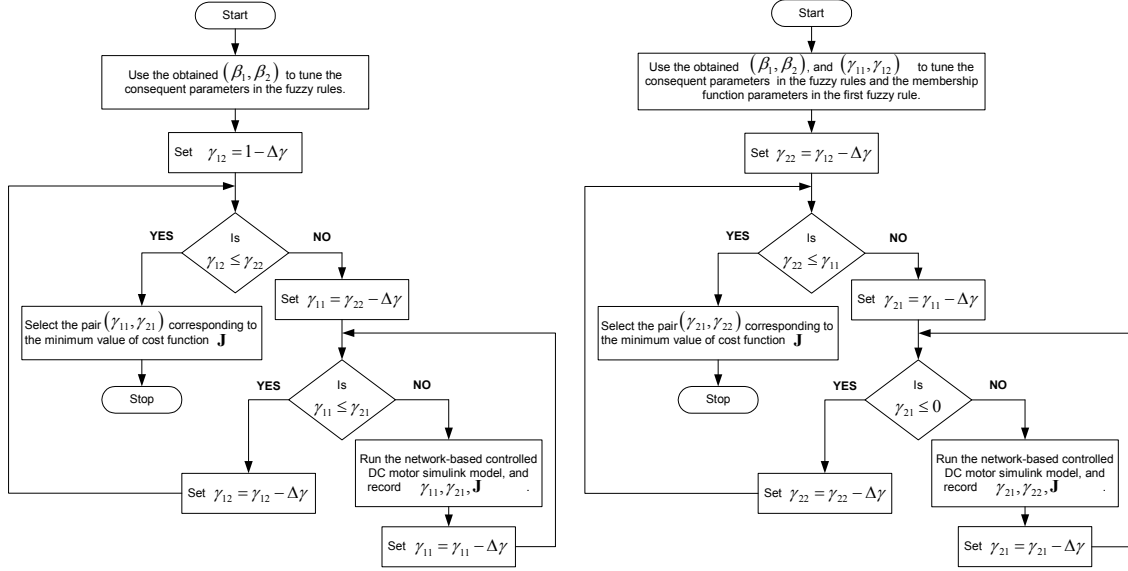


Figure 5-4. (a) Flowchart of the $(\gamma_{11}, \gamma_{12})$ adaptation process, (b) Flowchart of the $(\gamma_{21}, \gamma_{22})$ adaptation process.

In figure 5-4, $\Delta\gamma$ is the increment value used for the membership functions parameters during the off-line calculation. We should mention here that unlike the direct off-line PI controller parameters selection process presented in section 4.1.3, all searching space for the AFM parameters are bounded by (5.2) as shown in Fig. 5-5.

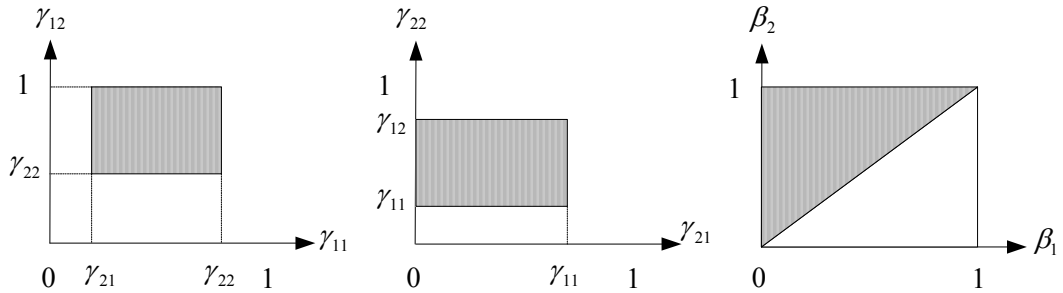


Figure 5-5. Searching regions in the off-line AFM parameters tuning.

5.2 Simulation results and discussion

The network-based controlled DC motor system is simulated using MATLAB/SIMULINK. The motor state space description in (3.4)-(3.5) is used as the plant, and is controlled by the PI controller in (3.15). The network-induced time delays as described in (3.13)-(3.14) are simulated using gaussian random number generators.

5.2.1 On-line tuning with instantaneous cost function

Using the adaptation laws in (4.6)-(4.7) to on-line tune (K_p, K_i) pair, and adaptation laws in (5.4)-(5.9) for on-line AFM parameters tuning, the typical step responses of the network-based controlled DC motor with on-line updating based on minimizing instantaneous cost function are shown in Fig. 5-6.

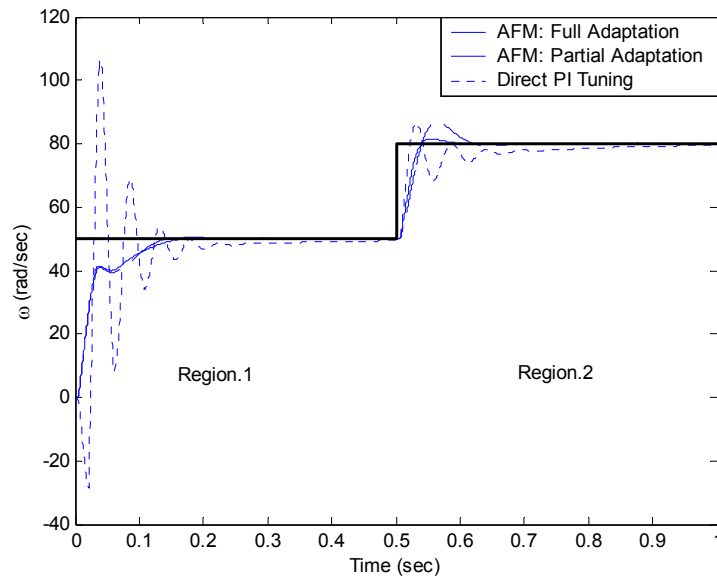


Figure 5-6. Step response of the network-based controlled DC motor with on-line updating based on minimizing instantaneous cost function.

Table. 5-1 summarizes the compute of the summation of squared error for all controllers.

Table 5-1 Comparison between full/partial AFM and direct PI tuning: On-line tuning with instantaneous cost function.

	Region.1			Region.2		
	Full AFM	Partial AFM	Direct PI	Full AFM	Partial AFM	Direct PI
Sum. of squared error, $\sum_{k=0}^N e(k)^2 (\times 10^4)$	8.7019	8.7863	30.048	2.7992	3.5843	3.2419

The full AFM provides the minimum summation of squared error values in both regions. The results are expected since with the full AFM we have more degrees of freedom in the controller parameters to adjust them adaptively (the membership functions parameters are not fixed as in partial AFM).

The initial and final membership functions used in the full AFM are shown in Fig. 5-7(a), while the overlapping between the membership functions during the adaptation and corresponding to $\alpha_c = 0.5$ is plotted in Fig. 5-7(b).

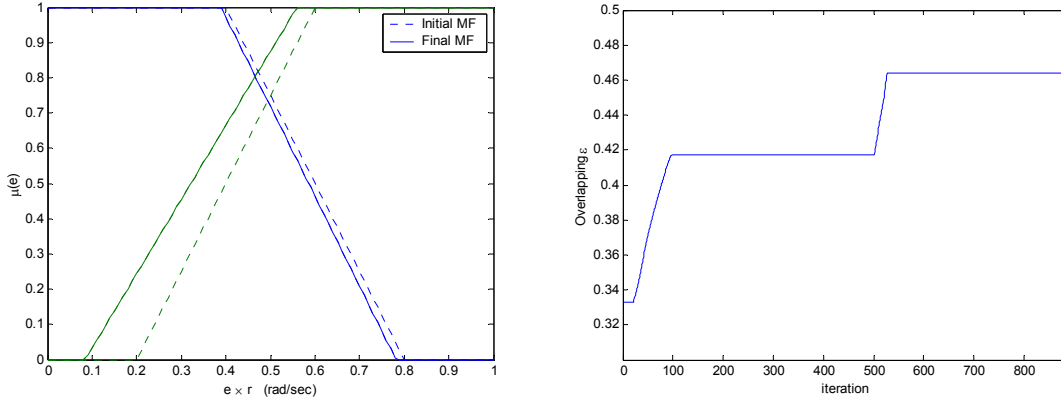


Figure 5-7. (a) Initial and final membership functions for full AFM, (b) Overlapping between membership functions during the adaptation process.

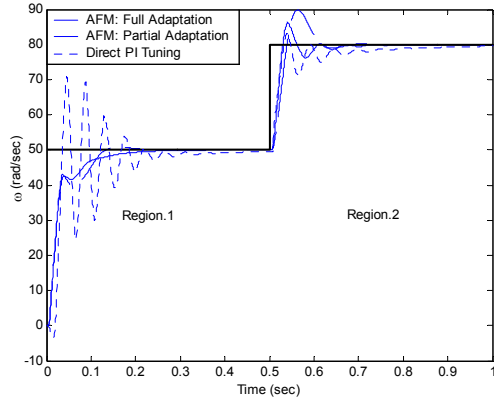
As shown from Fig. 5-7(b), the overlapping constraint (C4) in section 5.1 is satisfied. In Figs. 5-6, 5-8 and 5-12, Region.1 and Region.2 describe different reference signal (r), sampling time (h), and delay time (τ) values used in the simulation:

Region. 1: $r = 50$ (rad/sec), $h = 1$ (msec), and $\text{mean}(\tau) = 5$ (msec).

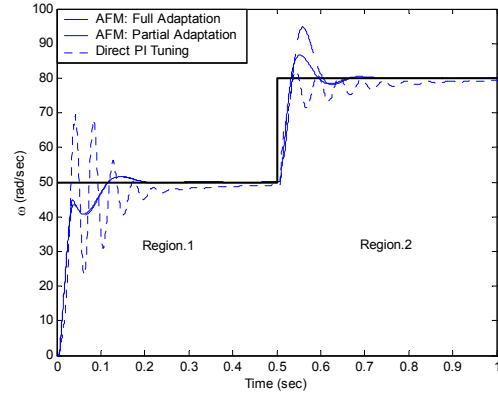
Region. 2: $r = 80$ (rad/sec), $h = 3$ (msec), and $\text{mean}(\tau) = 6$ (msec).

5.2.2 On-line tuning with moving window cost function

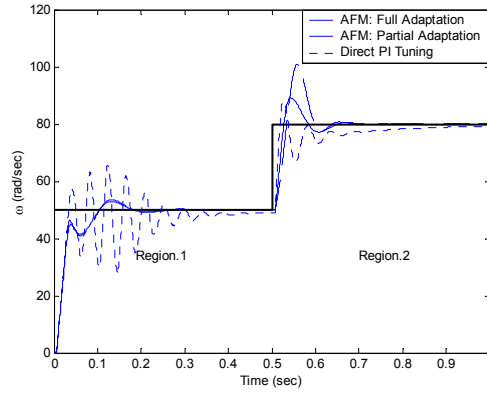
With the adaptation laws in (4.9)-(4.10) to on-line tune (K_p, K_i) pair, and adaptation laws in (5.12)-(5.17) for on-line AFM parameters tuning, the typical step responses of the network-based controlled DC motor with on-line updating based on minimizing the moving window cost function with different window sizes are shown in Fig. 5-8.



(a)



(b)



(c)

Figure 5-8. Step response of the network-based controlled DC motor with on-line updating based on minimizing moving window cost function: (a) $m=5$, (b) $m=10$, (c) $m=20$.

Table. 5-2 summarizes the computation of the summation of squared error for all controllers corresponding to different values of m .

Table 5-2 Comparison between full/partial AFM and direct PI tuning: On-line tuning with moving window cost function.

Sum. of squared error, $\sum_{k=0}^N e(k)^2 (\times 10^4)$	Region.1			Region.2		
	Full AFM	Partial AFM	Direct PI	Full AFM	Partial AFM	Direct PI
$m = 5$	8.7331	9.0478	17.856	2.9686	4.1621	3.1344
$m = 10$	8.7609	8.7841	14.249	3.1766	4.6930	3.3047
$m = 20$	8.5677	8.6289	12.2880	3.1235	5.7408	3.4062

From Table. 5-2, we noticed that the full AFM has the minimum summation of squared error over other controllers in both Region.1 and Region.2.

The initial and final membership functions used in the full AFM corresponding to $m=5,10,20$ are shown in Figs. 5-9(a), 5-10(a), 5-11(a), respectively. While the overlapping between the membership functions during the adaptation and corresponding to $\alpha_c = 0.5$ with $m=5,10,20$ are plotted in Figs. 5-9(b), 5-10(b), 5-11(b), respectively.

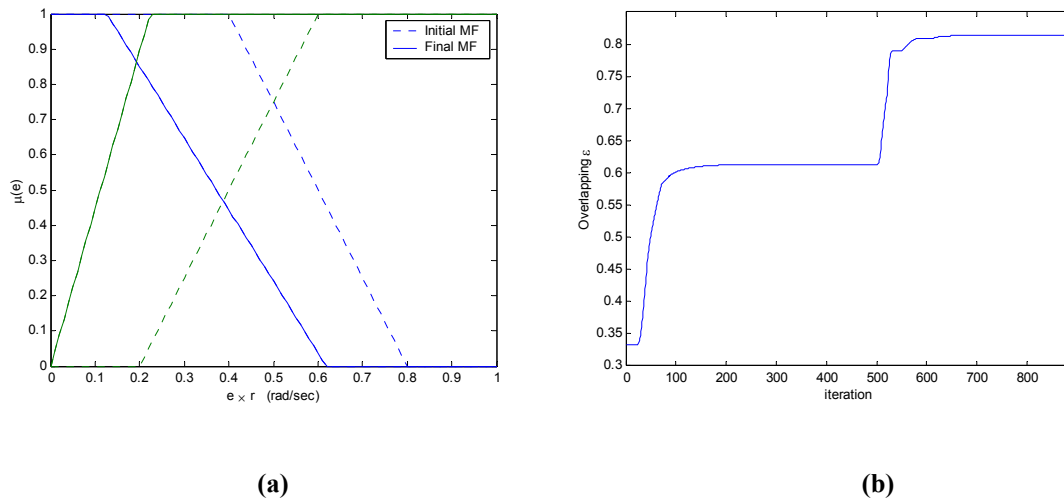


Figure 5-9. For $m=5$, (a) Initial and final membership functions for full AFM, (b) Overlapping between membership functions during the adaptation process.

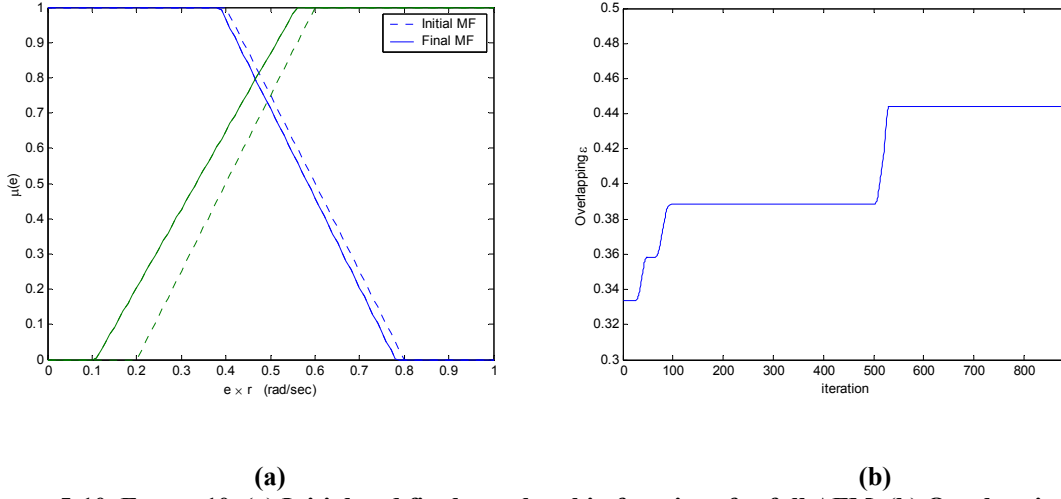


Figure 5-10. For $m=10$, (a) Initial and final membership functions for full AFM, (b) Overlapping between membership functions during the adaptation process.

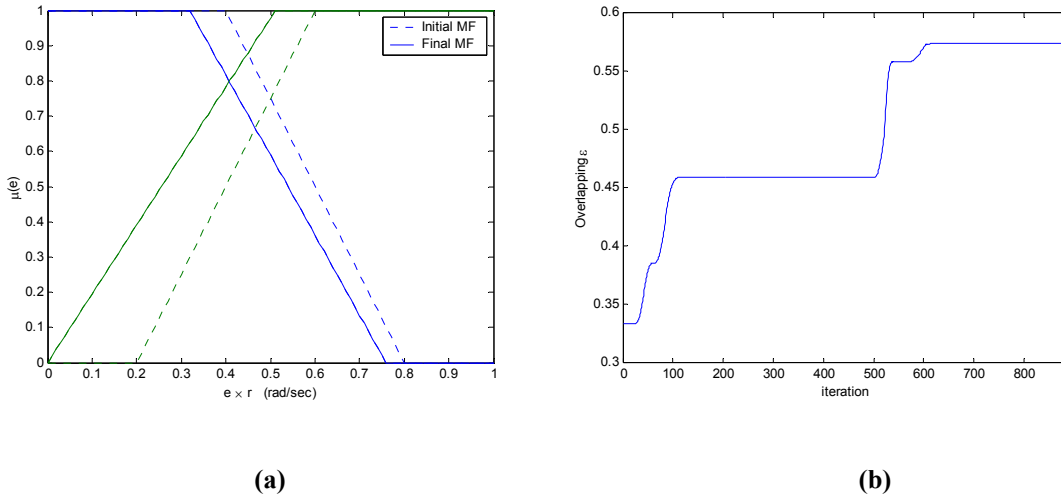


Figure 5-11. For $m=20$, (a) Initial and final membership functions for full AFM, (b) Overlapping between membership functions during the adaptation process.

As shown in Figs. 5-10(b), 5-11(b), and 15-12(b), the overlapping between the membership functions during the adaptation is satisfying condition C4 in section 5.1.

5.2.3 Off-line controller parameters selection

The adaptive controller presented in Section 5.1.3 is described as follows: The controller monitors the network QoS via appropriate middleware, and then adjusts the fuzzy

logic modulator parameters to the set of parameters corresponding to the minimum of a pre-defined/pre-computed cost function. A look-up table technique is used to perform searching on the optimal controller parameters. Linear interpolation is used when the exact entry cannot be found from the table with the given inputs. The typical step responses of the network-based controlled DC motor with off-line parameters selection are shown in Fig. 5-12. In order to select the PI controller parameters, we use the following searching region: $K_p \in (0.01, 0.6)$ with $\Delta K_p = 0.01$, and $K_i \in (0.1, 3)$ with $\Delta K_i = 0.1$, while for the AFM parameters selection, we apply the proposed off-line algorithm in section 5.1.3 with $\Delta\beta = 0.01$ and $\Delta\gamma = 0.02$.

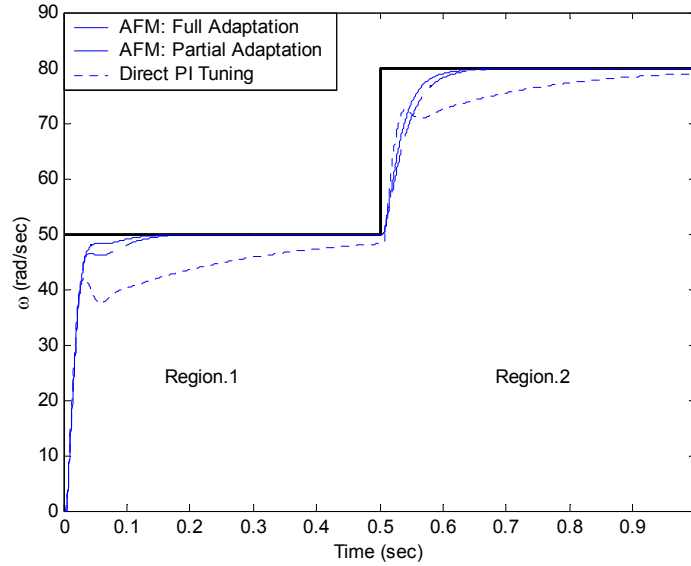


Figure 5-12. Step response of the network-based controlled DC motor with off-line selection based on minimizing pre-computed cost function.

By computing the sub-cost function $\mathbf{J}_1$ and $\mathbf{J}_2$ for all controllers, the results are shown in Table. 5-3.

Table 5-3 Comparison between full/partial AFM and direct PI tuning: Off-line tuning.

	Region.1			Region.2		
	Full AFM	Partial AFM	Direct PI	Full AFM	Partial AFM	Direct PI
sub-cost function $\mathbf{J}_1 = \sum_{k=0}^N e(k)^2 (\times 10^4)$	6.7439	6.8327	10.334	4.1412	4.794	5.524
Sub-cost function $\mathbf{J}_2 = \sum_{k=1}^M \Delta e_b(k)^2$	0	0.0081	0.5468	0	0	0.0747

The full AFM provides the minimum sub-cost functions values in both regions. The initial and final membership functions used in the full AFM are shown in Fig. 5-13(a), while the overlapping between the membership functions during the adaptation and corresponding to $\alpha_c = 0.5$ is plotted in Fig. 5-13(b).

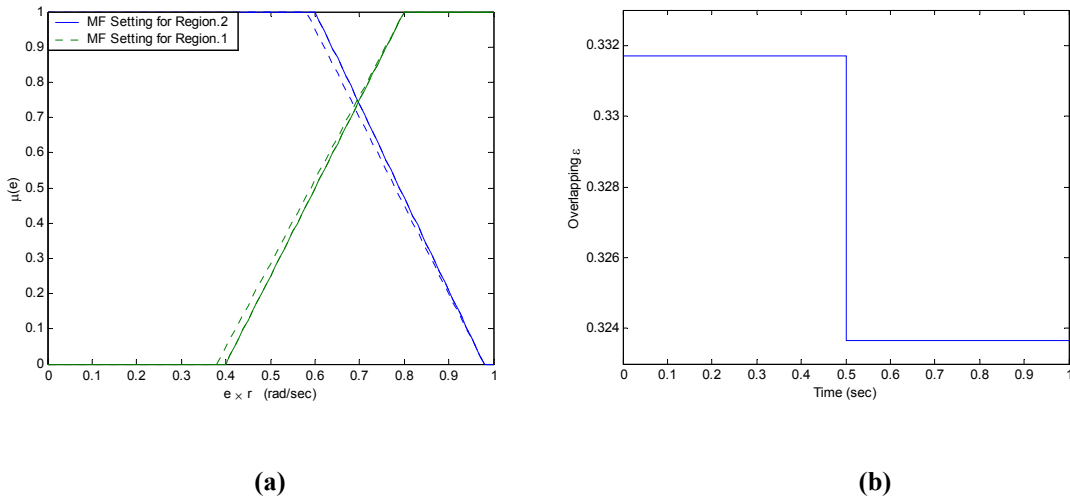


Figure 5-13. (a) Membership functions for full AFM in Region. 1 and 2, (b) Overlapping between membership functions during the adaptation process.

As shown from Fig. 5-13(b), the overlapping constraint (C4) in section 5.1 is satisfied.

5.2.4 On-line AFM as a continuous gain scheduling

The basic idea of gain scheduling is to linearize the system at different operating regions, then using linear control design approaches, certain controller parameters associated with each operating region are pre-computed. In our AFM, we can consider the off-line AFM selection process as a conventional gain scheduling since the controller monitors the network QoS via appropriate middleware, and then adjusts the fuzzy logic modulator parameters to the set corresponding to the minimum of a pre-defined/pre-computed cost function. While, in the on-line AFM, we can consider the controller as a supervisor controller that continuously monitors the state of the system and network QoS (i.e., identifies the operating region), then update the fuzzy logic modulator parameters via adaptation laws to improve the performance of the closed-loop system with the available network QoS.

To see the advantage and superb performance of using on-line AFM to stabilize the closed-loop system as well as to maintain the system performance while the network QoS is varying, we will use the step response of a network-based controlled DC motor subject to tracking a reference signal of 50 (rad/sec) and change the network QoS at steady-state (e.g., $t = 0.5$ sec) as an example, while keeping the reference signal unchanged. The typical step response of the network-based controlled DC motor subject to tracking a reference signal of 50 (rad/sec) and which the network QoS is changed from sampling time $h = 1$ (msec) and mean $(\tau) = 5$ (msec) to sampling time $h = 3$ (msec) and mean $(\tau) = 3$ (msec) at $t = 0.5$ sec is shown in Fig. 5-14.

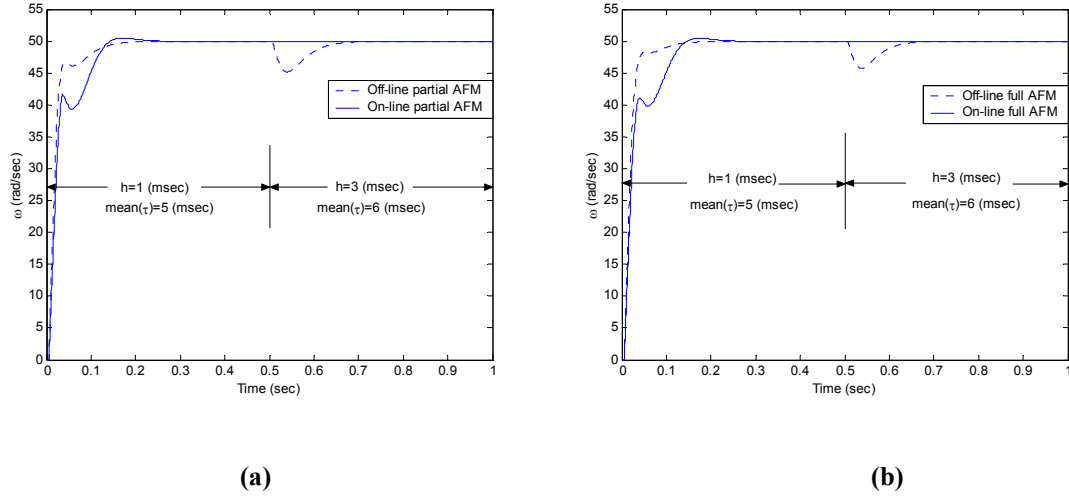


Figure 5-14. Step response of the network-based controlled Dc motor with off-line/on-line AFM under different network QoS. (a) Partial AFM, (b) Full AFM.

As shown in Fig. 5-14, with the on-line AFM, the effect of network QoS variation is compensated perfectly. While with the off-line AFM, we can see the adverse effect of the network QoS variation in terms of performance degradation since the error signal differs from steady-state value which is zero when the network QoS changes and takes time to settle again to its steady-state value.

To study the phenomenon in Fig. 5-14 when off-line AFM is used (i.e., the adverse effect when a changing in QoS is happened at steady state), we need to identify which cause this adverse effect on the step response of the network-based controlled DC motor. For example, in partial AFM we have two candidate sources causing this phenomenon to investigate: the QoS variation and the partial AFM parameters (i.e., (β_1, β_2) pair). First, let us define two QoS regions with their associated optimal (β_1, β_2) pairs as shown in Fig. 5-15.

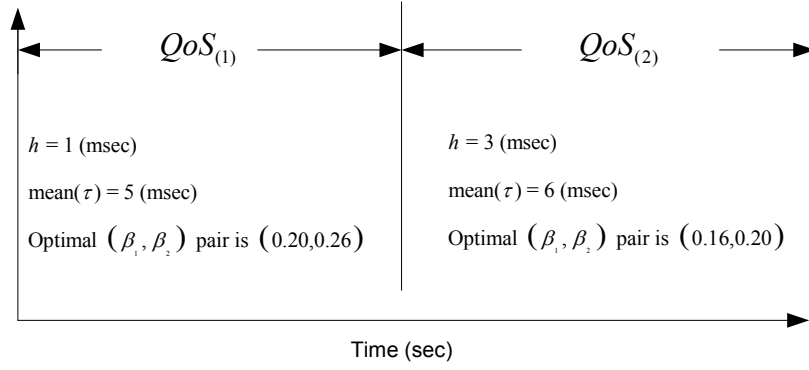


Figure 5-15 Two different QoS with their associated optimal (β_1, β_2) pair

The plot of the control signal provided by the central controller and the typical step response of the networked DC motor for which the QoS is changing from $QoS_{(1)}$ to $QoS_{(2)}$ at steady state is shown in Fig. 5-16.

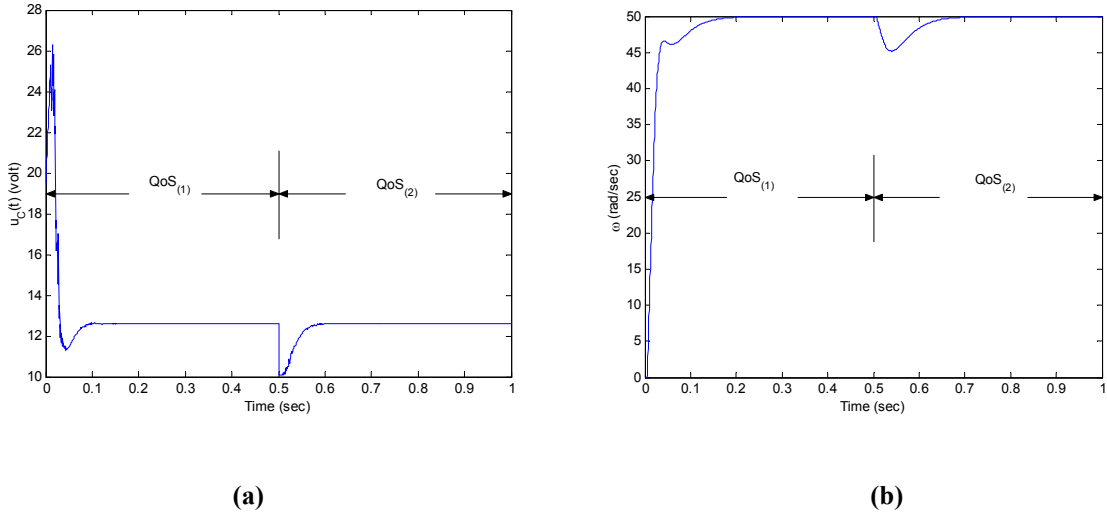


Figure 5-16 (a) $u_c(t)$, (b) $\omega(t)$.

Now if we reverse and change QoS from $QoS_{(2)}$ to $QoS_{(1)}$, the plot of the control signal provided by the central controller and the typical step response of the networked DC

motor for which the QoS is changing from $QoS_{(2)}$ to $QoS_{(1)}$ at steady state is shown in Fig. 5-17.

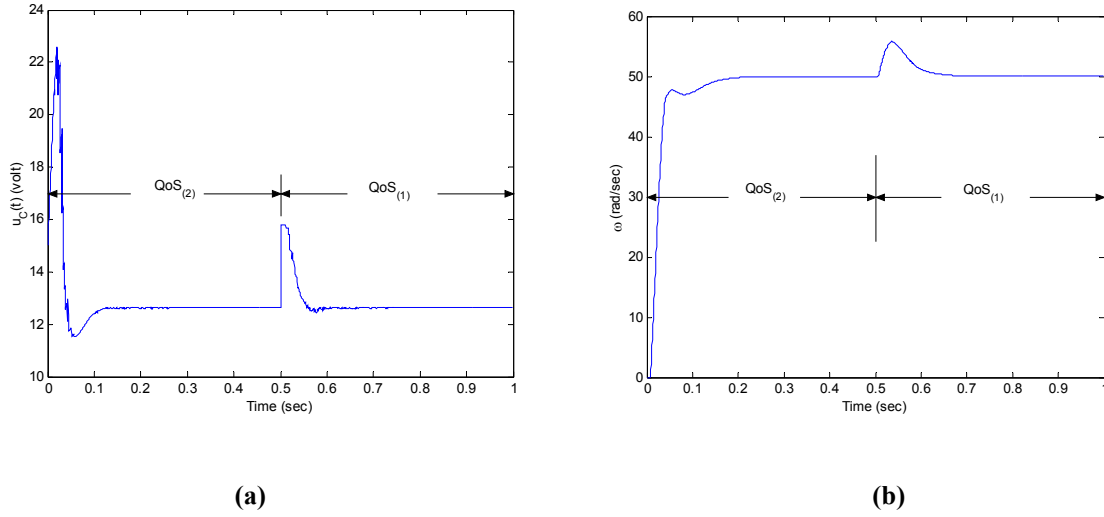
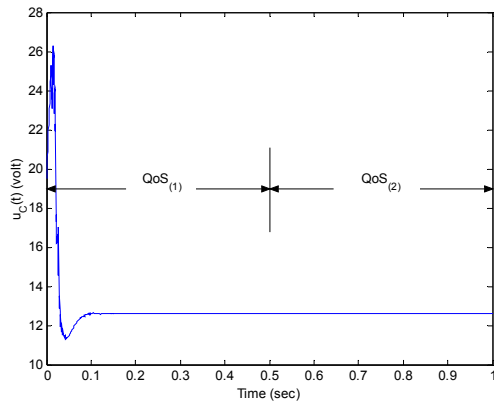
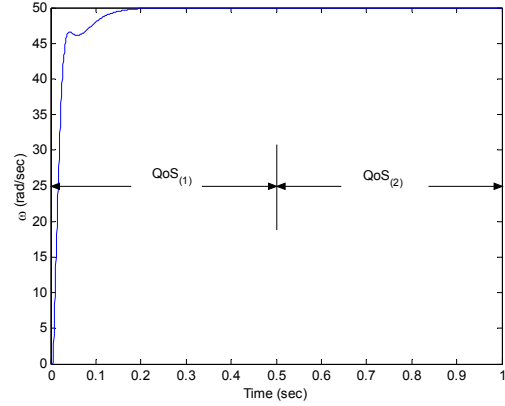


Figure 5-17 (a) $u_c(t)$, (b) $\omega(t)$.

On the other hand, let us change QoS from $QoS_{(1)}$ to $QoS_{(2)}$ at steady state without updating (β_1, β_2) pair to the pair corresponding to $QoS_{(2)}$, i.e., continue using the optimal (β_1, β_2) pair associated with $QoS_{(1)}$. The plot of the control signal provided by the central controller and the typical step response of the networked DC motor corresponding to this scenario is shown in Fig. 5-18.



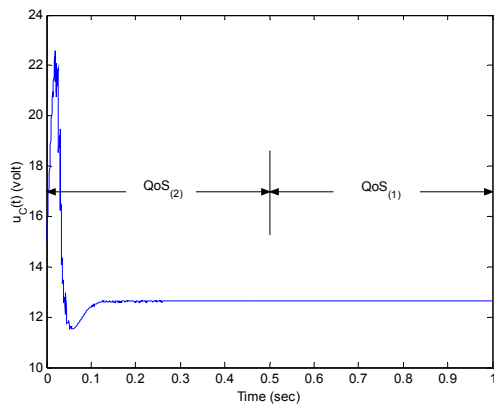
(a)



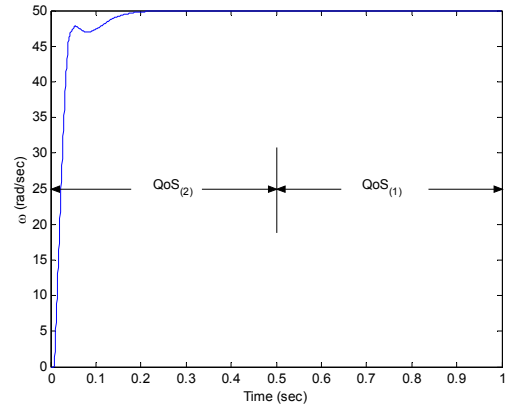
(b)

Figure 5-18 (a) $u_c(t)$, (b) $\omega(t)$.

Now, let us change QoS from $QoS_{(2)}$ to $QoS_{(1)}$ at steady state without updating (β_1, β_2) pair to the pair corresponding to $QoS_{(1)}$, i.e., continue using the optimal (β_1, β_2) pair associated with $QoS_{(2)}$. The plot of the control signal provided by the central controller and the typical step response of the networked DC motor corresponding to this scenario is shown in Fig. 5-19.



(a)



(b)

Figure 5-19 (a) $u_c(t)$, (b) $\omega(t)$.

If we set QoS to $QoS_{(1)}$ and change the (β_1, β_2) pair at steady state without changing QoS where the new (β_1, β_2) pair is the optimal (β_1, β_2) pair corresponding to $QoS_{(2)}$ although we will not change QoS to $QoS_{(2)}$, the plot of the control signal provided by the central controller and the typical step response of the networked DC motor corresponding to this scenario is shown in Fig. 5-20.

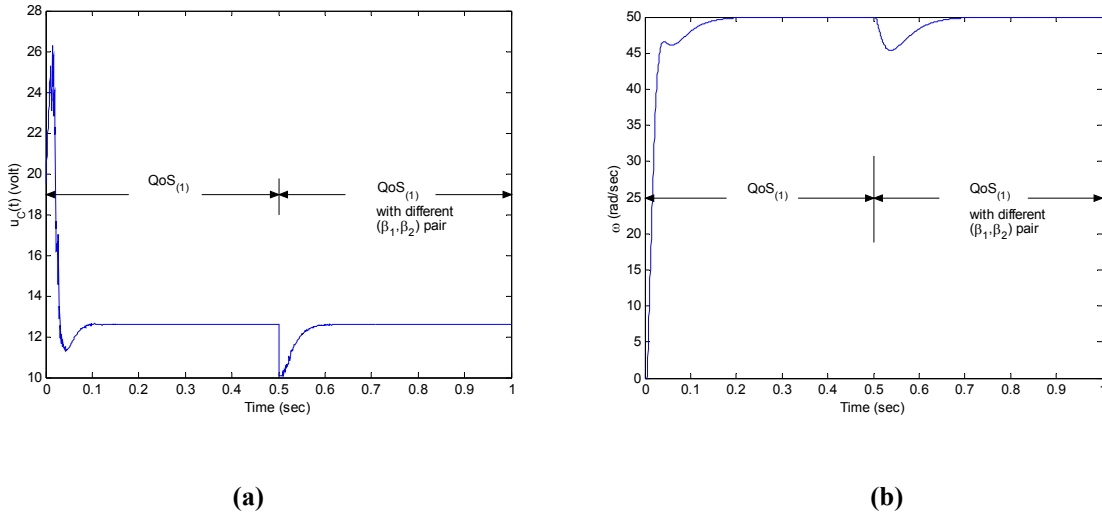


Figure 5-20 (a) $u_c(t)$, (b) $\omega(t)$.

Now we will set QoS to $QoS_{(2)}$ and change the (β_1, β_2) pair at steady state without changing QoS. The new (β_1, β_2) pair is the optimal (β_1, β_2) pair corresponding to $QoS_{(1)}$ although we will not change QoS to $QoS_{(1)}$. The plot of the control signal provided by the central controller and the typical step response of the networked DC motor corresponding to this scenario is shown in Fig. 5-21.

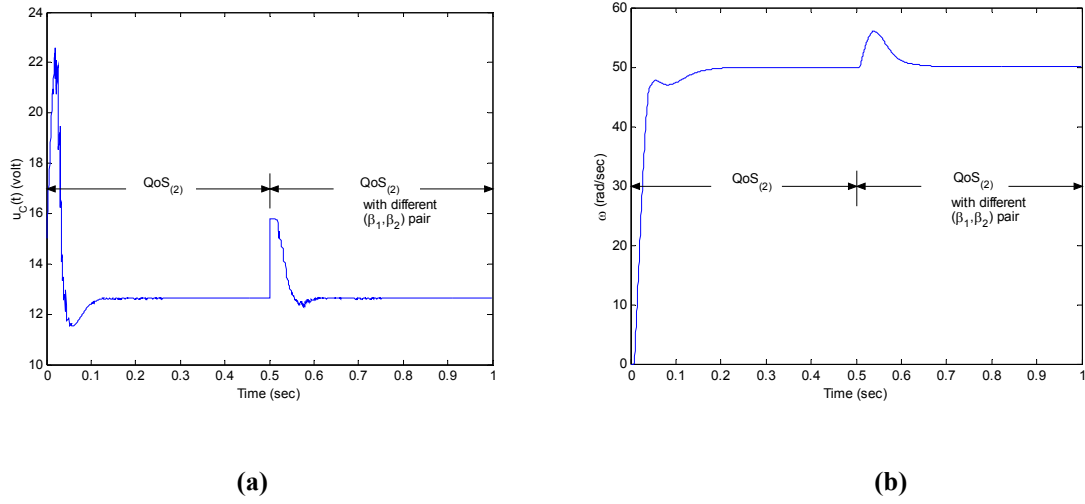


Figure 5-21 (a) $u_c(t)$, (b) $\omega(t)$.

From the simulated results, changing (β_1, β_2) pair at steady state cause the adverse effect on the step response when the QoS is change. The reason can be stated as follows: Once QoS is changed at steady state, and based on the off-line partial AFM, an optimal (β_1, β_2) pair associated with the new QoS is to chosen to compensate for this QoS changing. The new fuzzy logic modulator parameter β in this case is given by

$$\beta(k+1) = \frac{\beta_1(k+1)\mu_{\text{Small}}(e) + \beta_2(k+1)\mu_{\text{Large}}(e)}{\mu_{\text{Small}}(e) + \mu_{\text{Large}}(e)} \quad (5.20)$$

Three cases are to be addressed. The first case is when $\beta(k+1) < \beta(k)$; this implies that the control signal provided by the central controller will have a lower value than the steady state value, i.e., $u_c(k+1) < u_c(k)$ since $u_c(k+1) = \beta(k+1)u_{ss}$, where u_{ss} is the steady state value of the PI control signal. In this case the step response will be lower than the steady state value, which cause the PI controller with partial AFM to compensate for this degradation in performance as shown in Figs. 5-16, 5-20.

The second case is when $\beta(k+1) > \beta(k)$; this implies that the control signal provided by the central controller will have a larger value than the steady state value, i.e., $u_c(k+1) > u_c(k)$ since $u_c(k+1) = \beta(k+1)u_{ss}$. In this case the step response will be larger than the steady state value, which cause the PI controller with partial AFM to work again and compensate for this degradation in performance as shown in Figs. 5-17, 5-21.

The third case is when $\beta(k+1) = \beta(k)$; this implies that the control signal provided by the central controller will not change its steady state value, i.e., $u_c(k+1) = u_c(k)$ since $\beta(k+1) = \beta(k)$. In this case the step response will not affected by the changing of QoS at steady state as shown in Figs. 5-18, 5-19.

To compensate for this adverse effect one may choose not to change the controller parameters at steady state even when the network QoS is changed, but we have to mention here that this is a tread-off problem between changing controller parameters accordingly to QoS changes to compensate for the QoS deterioration effects; and fixing the controller parameters at steady state regardless of QoS changes to compensate for the performance degradation when the QoS is changed at steady state.

To see this trade-off problem, let us investigate two cases:

- In the first case the controller changes its parameters when network QoS is change at steady state, and
- In the second case the controller does not change its parameters at steady state even the network QoS is changed.

In both cases the reference signal changes its value at steady state but not at the same instant that the network QoS is changed. The typical step responses corresponding to the two cases are shown in Fig. 5-22.

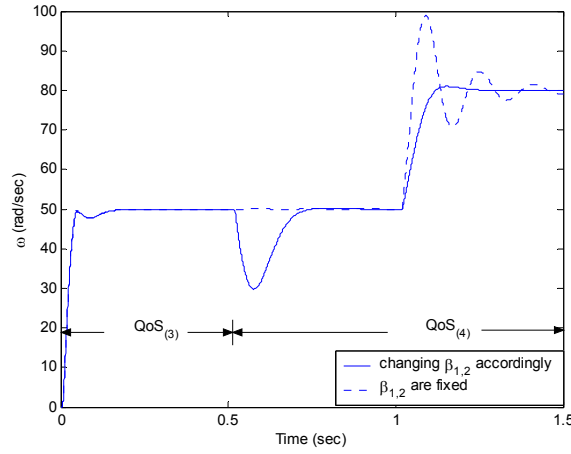


Figure 5-22 Illustration case of the tread-off problem in the off-line AFM.

where $QoS_{(3)}$ is corresponding to sampling time of 5 (msec) and a random time delay of mean 5 (msec) with the associated $(\beta_1, \beta_2) = (0.18, 0.22)$, and $QoS_{(4)}$ is corresponding to sampling time of 15 (msec) and a random time delay of mean 15 (msec) with the associated $(\beta_1, \beta_2) = (0.08, 0.10)$. From the illustrated case, we see the adverse effect when not changing the controller parameters accordingly to QoS changes by having higher overshoot and longer settling time which effect the performance of the network-based controlled DC motor.

For the on-line partial AFM, the consequent parameters are adapted by:

$$\beta_1(k+1) = \beta_1(k) + 2\eta \frac{e(k) \mu_{\text{Small}}(e(k)) u_{PI}(k)}{\left(K_p(k) + \frac{K_I(k)h}{2}\right) (\beta_1(k) \mu_{\text{Small}}(e(k)) + \beta_2(k) \mu_{\text{Large}}(e(k)))} \quad (5.21)$$

$$\beta_2(k+1) = \beta_2(k) + 2\eta \frac{e(k) \mu_{\text{Large}}(e(k)) u_{PI}(k)}{\left(K_p(k) + \frac{K_I(k)h}{2}\right) (\beta_1(k) \mu_{\text{Small}}(e(k)) + \beta_2(k) \mu_{\text{Large}}(e(k)))} \quad (5.22)$$

By using these adaptation laws and substituting for $e(k) = 0$, we get:

$$\begin{aligned}\beta_1(k+1) &= \beta_1(k) \\ \beta_2(k+1) &= \beta_2(k)\end{aligned}\tag{5.23}$$

which force the fuzzy logic modulator parameters not to change (i.e., $\beta(k+1) = \beta(k)$). In this case no change is happened to the control signal provided by the central controller which implies that the step response of the network-based controlled DC motor stay at its steady state value as shown in Fig. 5-23. The typical step response of the network-based controlled DC motor with on-line consequent parameters tuning is shown in Fig. 5-23.

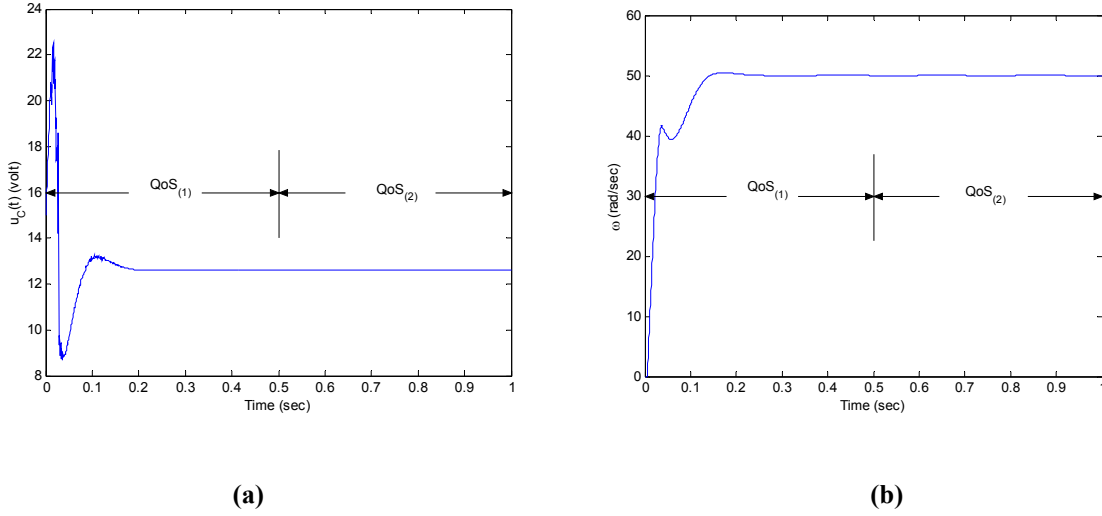


Figure 5-23 (a) $u_c(t)$, (b) $\omega(t)$.

5.3 Conclusion

In this chapter, we proposed a full adaptive fuzzy modulation and compared it with two other tuning methods to compensate for the network-induced time delays in networked PI control systems. In the first tuning method, we adaptively tune the PI controller parameters directly. In the second tuning method, we use partial adaptive fuzzy modulation, i.e., tune the consequent parameters in the fuzzy rules only with fixed membership function

parameters in the antecedent part and fixed PI controller parameters to their nominal values. While in the full adaptive fuzzy modulation, not only are the consequent parameters be tuned adaptively, but also the membership functions parameters in the antecedent part are tuned adaptively. Numerical simulation of a network-based controlled DC motor is used to illustrate the effectiveness of the full AFM over other controllers.

6 Stabilization of networked control system using fuzzy logic modulation

In this chapter, we will consider the problem of stabilizing the networked control system using PI controller with fuzzy logic modulation. Using a version of the Hermite-Biehler Theorem applicable to quasipolynomials, a complete analytical characterization of all stabilizing fuzzy logic modulator parameter values to stabilize the networked PI control system is provided. Numerical simulation of a network-based controlled DC motor is used to illustrate the proposed work.

6.1 Stability analysis in NCS

In networked control systems, and due to the network-induced time delay effects, the available techniques used to control the NCS have to maintain the performance and stability of the system.

Stability analysis of the NCS with network-induced time delay using the conventional state augmentation approach [42] is presented in [12, 35]. In this work, the authors used the augmented closed-loop system to derive the stability regions in cases of smaller and longer network-induced time delays for the simple integrator system. Usually, by using the state augmentation approach, we expect the complexity of the NCS to increase significantly and proportionally to the dimension of the states and inputs. Also the authors in [12, 35] used the hybrid systems stability analysis technique [76] to analyze the stability of the NCS with network-induced time delay in [12].

The problem of the stability in a first-order system with network-induced time delay is addressed in [77]. In this work, only the feedback path contains delay, which make it inapplicable for NCS with controller to actuator network-induced time delay. The effect of the network-induced time delay on the stability of a two-dimensional distributed system is analyzed in [78]. In [79], the stability condition for a time-variant uncertain discrete delay system is proposed. The proposed stability condition is necessary and sufficient and can be viewed as a modified combination of two existing conditions for time-variant systems with polytopic uncertainties [80].

The effect of network-induced time delay on the stability of an autonomous mobile robot was investigated in [81, 82]. The authors first design a control law that stabilizes the mobile robot assuming that there is no delay; then they highlight the problem of network-induced random time delay and design a predictor mechanism to build a control law taking into account the presence of this network-induced delay and stabilizing the mobile robot control system. In this work, the authors restrict their analysis to the case where the delay is less than the sampling period. In general, recent advances in control of time delay systems can be found in [83].

In [51, 84], the stability of the local NCS is proved if certain assumptions hold. The main assumptions are: 1) that the intertransmission interval is bounded, and the bound is sufficiently small, and 2) that the collision resolution is perfect, meaning the high error channel will always be transmitted [44, 48].

In [33], the authors present the network-induced maximal allowable delay bound for stability of the NCS. Two asymptotically stable criteria based on the Lyapunov function are used to find the maximal allowable delay of the system. The first criterion is concerned about

the delay independent stability, while the second criterion is concerned about the delay dependent stability.

The characterization of the set of all stabilizing PID parameters using a version of the Hermite-Biehler Theorem [85] for the first-order plant with time delay is presented in [86, 87]. The solution for the PID stabilization problem is based on first determining the range of the proportional parameter for which a stabilizing PID controller exists. Then, for a fixed value of the proportional parameter in this range, it is shown that the set of stabilizing integral and derivative gain values lies in a convex polygon (either a trapezoid, a triangle, or a quadrilateral). A similar approach is applied for determining the stabilizing feedback gains for the second-order systems with time delay [88].

In this chapter, we will consider the problem of stabilizing the networked PI control system using fuzzy logic modulator. Using a version of the Hermite-Biehler Theorem applicable to quasipolynomials, a complete analytical characterization of all stabilizing fuzzy logic modulator parameter values to stabilize the networked PI control systems is provided.

6.2 Fuzzy modulation for networked PI control system

In this section, we will briefly present the fuzzy logic modulator technique to compensate for the network-induced time-delay effects in the networked PI control systems. This approach is based on modulating the control signal provided by the PI controller with a single parameter. The input to the fuzzy modulator is the error signal between the reference signal and the plant output signal; the output of the fuzzy modulator is the modulation parameter used to compensate for the effect of the network-induced time delay. The block

diagram of the networked PI control system with fuzzy logic modulation applied to DC motor is shown in Fig. 6-1.

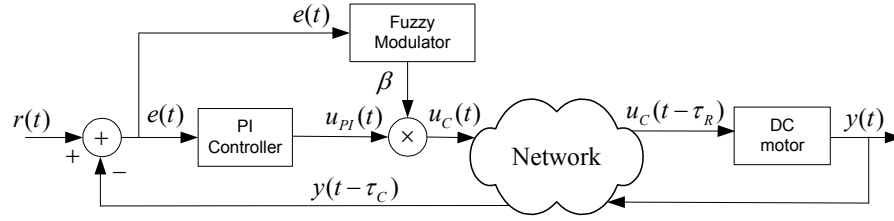


Figure 6-1. Network-based controlled DC motor with fuzzy modulation.

In Fig. 6-1, β is the modulation parameter that is the output of the fuzzy logic modulator system. In this chapter, we will use this fuzzy logic modulation parameter to stabilize the networked PI control system.

6.3 Hermite-Biehler Theorem applied to networked PI control system

In this section, we will apply the version of Hermite-Biehler Theorem applicable to quasipolynomials to prove the stability of the networked PI control system. We will use the network-based controlled DC motor as an illustration example. The block diagram of the networked PI control system is shown in Fig. 6-2.

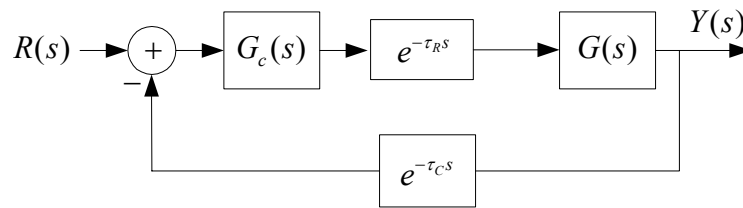


Figure 6-2. Block diagram of networked PI control system.

For stability analysis purposes, and using block diagram reduction techniques, we can lump the sensor to controller time delay τ_C and the controller to actuator time delay τ_R together in L , where $L \geq (\tau_C + \tau_R)$ is the upper bound on the network-induced time delays τ_C and τ_R as shown in Fig. 6-3.

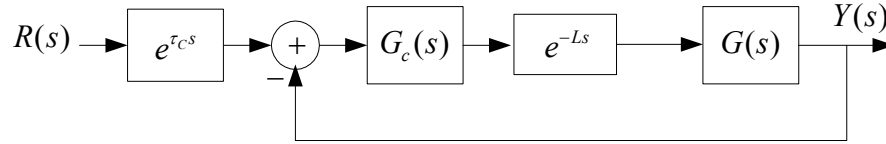


Figure 6-3. Simplified block diagram of networked PI control system.

In Fig. 6-3, the transfer function of the DC motor is

$$G(s) = \frac{k}{(T_1 s + 1)(T_2 s + 1)} \quad (6.1)$$

where $k = 3.9804$ represents the dc gain of the DC motor, $T_1 = 4.4783 \times 10^{-4}$ and $T_2 = 0.0353$ represents the DC motor time constants, and

$$G_c(s) = \beta \frac{k_p s + k_i}{s} \quad (6.2)$$

is the PI controller transfer function with fuzzy modulator transfer function β ; k_p and k_i are the nominal values of PI controller parameters tuned without considering network-induced time delay effects. The characteristic equation of the system in Fig. 6-3 is given by

$$\begin{aligned} \Delta(s) &= s(T_1 s + 1)(T_2 s + 1) + k\beta(k_p s + k_i)e^{-Ls} \\ &= d(s) + e^{-Ls}n_1(s) \end{aligned} \quad (6.3)$$

Due to the presence of the exponential term e^{-Ls} , the number of roots of (6.3) is infinite and this makes such a stability check extremely difficult [86]. Instead of using (6.3), we can consider the quasipolynomial [88]

$$\begin{aligned}\Delta^*(s) &= e^{Ls} \Delta(s) \\ &= s(T_1 s + 1)(T_1 s + 1)e^{Ls} + k\beta(k_p s + k_i)\end{aligned}\tag{6.4}$$

Since e^{Ls} does not have any finite zeros, the zeros of $\Delta(s)$ are identical to those of $\Delta^*(s)$.

We will use the extension of the Hermite-Biehler Theorem [85, 87] to check the stability of $\Delta(s)$ and thus the stability of the networked PI control system.

Theorem 1: (Hermite-Biehler Theorem applicable to quasipolynomials)

Let $\Delta^*(s)$ is given by (6.4), and write

$$\Delta^*(j\omega) = \delta_r(\omega) + j\delta_i(\omega)$$

where $\delta_r(\omega)$ and $\delta_i(\omega)$ represent respectively the real and imaginary parts of $\Delta^*(j\omega)$.

Under the following assumptions:

- A1. $\deg[d(s)] = n$ and $\deg[n_1(s)] \leq n$.
- A2. $L > 0$.

$\Delta^*(s)$ is stable if and only if

1) $\delta_r(j\omega)$ and $\delta_i(j\omega)$ have only simple real roots and these interlace.

2) $E(\omega_o) = \delta_i'(j\omega_o)\delta_r(j\omega_o) - \delta_i(j\omega_o)\delta_r'(j\omega_o) > 0$, for some $\omega_o \in (-\infty, \infty)$;

where $\delta_i'(j\omega_o)$ and $\delta_r'(j\omega_o)$ denote the first derivative with respect to ω of $\delta_i(j\omega_o)$ and $\delta_r(j\omega_o)$, respectively.

The proof of the theorem can be found in [89]. First, we will start testing the stability of the networked PI control system without including the network-induced time delays, i.e., $L=0$. The characteristic equation of the delay-free DC motor with PI controller is

$$\Delta(s) = s(T_1s + 1)(T_2s + 1) + kk_p s + kk_i \quad (6.5)$$

Using Routh stability criterion [39], the following conditions on the PI controller parameters k_p and k_i should be satisfied in order to stabilize the delay-free DC motor.

$$\begin{aligned} k_i &> 0 & \text{and} \\ k_p &> 0 \end{aligned} \quad (6.6)$$

Since the minimal requirement for any control design is that the delay-free closed-loop system be stable, the inequalities in (6.6) are assumed to hold through this chapter.

Now, we will test the stability of the networked PI control system in the presence of network-induced time delays. For our network-based controlled DC motor, and after several simple substitutions

$$\Delta^*(j\omega) = \delta_r(\omega) + j\delta_i(\omega) \quad (6.7)$$

where,

$$\delta_r(\omega) = \omega(\omega^2 T_1 T_2 - 1) \sin \omega L - \omega^2 (T_1 + T_2) \cos \omega L + k\beta k_i \quad (6.8)$$

$$\delta_i(\omega) = \omega(1 - \omega^2 T_1 T_2) \cos \omega L - \omega^2 (T_1 + T_2) \sin \omega L + \omega k\beta k_p \quad (6.9)$$

With the change of variables $z = \omega L$, the real and imaginary parts of $\Delta^*(j\omega)$ can be rewritten as:

$$\delta_r(z) = \frac{z}{L} \left(\frac{z^2}{L^2} T_1 T_2 - 1 \right) \sin z - \frac{z^2}{L^2} (T_1 + T_2) \cos z + k\beta k_i \quad (6.10)$$

$$\delta_i(z) = \frac{z}{L} \left(1 - \frac{z^2}{L^2} T_1 T_2 \right) \cos z - \frac{z^2}{L^2} (T_1 + T_2) \sin z + \frac{z}{L} k \beta k_p \quad (6.11)$$

By simple check, we can verify that $\delta_r(z)$ is an even function in z , while $\delta_i(z)$ is an odd function in z , i.e.,

$$\begin{aligned} \delta_r(-z) &= \delta_r(z), \quad \text{and} \\ \delta_i(-z) &= -\delta_i(z) \end{aligned} \quad (6.12)$$

From Theorem 1, we need to check two conditions to insure the stability of the quasipolynomial $\Delta^*(s)$.

Step.1, we need to check condition 2 of theorem.1.

$E(\omega_o) = \delta'_i(j\omega_o) \delta_r(j\omega_o) - \delta_i(j\omega_o) \delta'_r(j\omega_o) > 0$, for some $\omega_o \in (-\infty, \infty)$. Let us take

$\omega_o = z_o = 0$. Thus $\delta_i(z_o) = 0$ and $\delta_r(z_o) = k \beta k_i$. After a few simple derivations,

$$\begin{aligned} \delta'_i(z) &= \frac{1}{L} \cos z - \frac{z}{L} \sin z - 3 \frac{z^2}{L^2} T_1 T_2 \cos z + \\ &\quad \frac{z^3}{L^2} T_1 T_2 \sin z - \frac{z^2}{L^2} (T_1 + T_2) \cos z - \\ &\quad 2 \frac{z}{L} (T_1 + T_2) \sin z + \frac{1}{L} k \beta k_p \end{aligned} \quad (6.13)$$

$$\delta'_i(z_o) = \frac{1}{L} (1 + k \beta k_p) \quad (6.14)$$

and condition 2 becomes $\frac{1}{L} (1 + k \beta k_p) k \beta k_i > 0$ (6.15)

which is satisfied since $L > 0, k > 0, (k_p, k_i) > 0, \beta > 0$

Step.2, we now check condition 1 of Theorem 1: the interlacing of the simple real

roots of $\delta_r(z)$ and $\delta_i(z)$. From (6.11) we can compute the roots of the imaginary part,

i.e., $\delta_i(z) = 0$. This gives us the following equation:

$$\frac{z}{L} \left[\left(1 - \frac{z^2}{L^2} T_1 T_2 \right) \cos z - \frac{z}{L} (T_1 + T_2) \sin z + k \beta k_p \right] = 0 \quad (6.16)$$

Then either

$$z = 0 \quad \text{or}$$

$$\left(1 - \frac{z^2}{L^2} T_1 T_2 \right) \cos z - \frac{z}{L} (T_1 + T_2) \sin z + k \beta k_p = 0 \quad (6.17)$$

From this it is clear that one root of the imaginary part is $z_{io} = 0$. The other roots are difficult to find since we need to solve (6.17) analytically. However, we can plot the terms involved in (6.17) and graphically examine the nature of the solution [87]. For illustration, a given $L = 8$ (msec), and $\beta = 0.26$ values are used. A plot of the terms

$\left(1 - \frac{z^2}{L^2} T_1 T_2 \right) \cos z + k \beta k_p$ and $\frac{z}{L} (T_1 + T_2) \sin z$ is shown in Fig. 6-4.

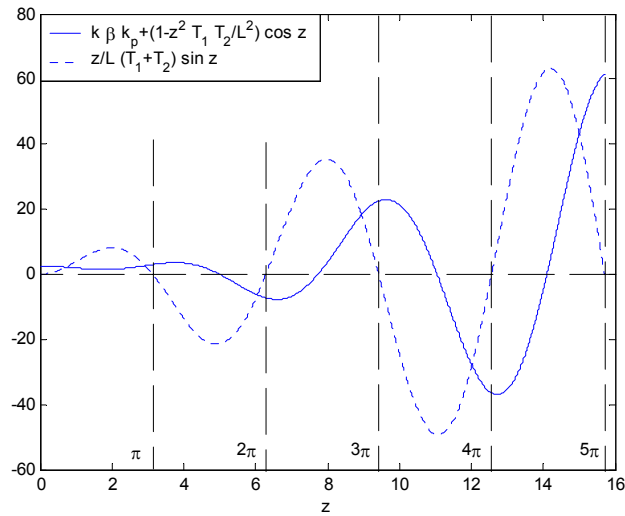


Figure 6-4. Plot of $\left(1 - \frac{z^2}{L^2} T_1 T_2 \right) \cos z + k \beta k_p$ and $\frac{z}{L} (T_1 + T_2) \sin z$.

The roots of (6.17) in $z \in [0, 5\pi]$ are:

$$z_{i1}=0.7325, z_{i2}=2.9385, z_{i3}=6.0495, z_{i4}=8.9505, z_{i5}=12.0175, z_{i6}=15.0065 \quad (6.18)$$

In order to check if $\delta_i(z)$ has only real roots; we need to use the following Theorem [87].

Theorem 2: Let M and N denote the highest powers of s and e^s respectively in $\Delta^*(s)$. Let η be an appropriate constant such that the coefficients of terms of highest degree in $\delta_r(\omega)$ and $\delta_i(\omega)$ do not vanish at $\omega = \eta$. Then for equations $\delta_r(\omega) = 0$ or $\delta_i(\omega) = 0$ to have only real roots, it is necessary and sufficient that in the intervals

$$-2l\pi + \eta \leq \omega \leq 2l\pi + \eta, \quad l = 1, 2, 3, \dots$$

$\delta_r(\omega)$ or $\delta_i(\omega)$ have exactly $4lN + M$ real roots starting with a sufficient large l .

The proof of Theorem 2 can be found in [90]. Using Theorem 2 to check if $\delta_i(z)$ has only real roots. Substituting $s_1 = Ls$ in the expression for $\Delta^*(s)$, we see that for the new quasipolynomial in s_1 , $M=3$ and $N=1$. Next we choose $\eta = \frac{\pi}{15} = 0.2094$ to satisfy the requirement such that the coefficients of terms of highest degree in $\delta_r(z)$ and $\delta_i(z)$ do not vanish at $z = \eta$.

Now from Fig. 6-4, we see that $\delta_i(z)$ has 4 roots in the interval $[0, 2\pi - \frac{\pi}{15}] = [0, 6.0737]$, including a root at the origin. Since $\delta_i(z)$ is an odd function of z , it follows that in the interval $[-2\pi + \frac{\pi}{15}, 2\pi - \frac{\pi}{15}] = [-6.0737, 6.0737]$, $\delta_i(z)$ will have 7 real roots. Also $\delta_i(z)$ have no real roots in the interval $[2\pi - \frac{\pi}{15}, 2\pi + \frac{\pi}{15}] = [6.0737, 6.4926]$. Thus $\delta_i(z)$ has $4N+M=7$ real roots in the interval $[-2\pi + \frac{\pi}{15}, 2\pi + \frac{\pi}{15}]$. From Fig. 2 it is clear

that $\delta_i(z)$ has two real roots in each of the intervals $\left[2l\pi + \frac{\pi}{15}, 2(l+1)\pi + \frac{\pi}{15}\right]$ and $\left[-2(l+1)\pi + \frac{\pi}{15}, -2l\pi + \frac{\pi}{15}\right]$ for $l=1,2,\dots$. Hence, it follows that $\delta_i(z)$ has exactly $4lN+M$ in the interval $\left[-2l\pi + \frac{\pi}{15}, 2l\pi + \frac{\pi}{15}\right]$. Hence, from Theorem 2, we conclude that $\delta_i(z)$ has only real roots.

Now we will evaluate $\delta_r(z)$ at the roots of the imaginary part $\delta_i(z)$. For $z_{io} = 0$, and using (6.10) we obtained

$$\delta_r(z_{io}) = 4.3767e+001 \quad (6.19)$$

for other $\delta_i(z)$ roots in (6.18), we get

$$\begin{aligned} \delta_r(z_{i1}) &= -2.3202e+002, \delta_r(z_{i2}) = 4.8480e+003, \delta_r(z_{i3}) = -2.1234e+004 \\ \delta_r(z_{i4}) &= 4.9414e+004, \delta_r(z_{i5}) = -9.5876e+004, \delta_r(z_{i6}) = 1.6213e+005 \end{aligned} \quad (6.20)$$

From (6.19)-(6.20), interlacing of roots of $\delta_r(z)$ and $\delta_i(z)$ is satisfied which is equivalent to

$$\delta_r(z_{io}) > 0, \delta_r(z_{i1}) < 0, \delta_r(z_{i2}) > 0, \dots \quad (6.21)$$

Also, we can see that roots of $\delta_r(z)$ and $\delta_i(z)$ are interlacing by checking roots of $\delta_r(z)$ in $z \in [0, 5\pi]$. A plot of $\delta_i(z)$ and $\delta_r(z)$ is shown in Fig. 6-5.

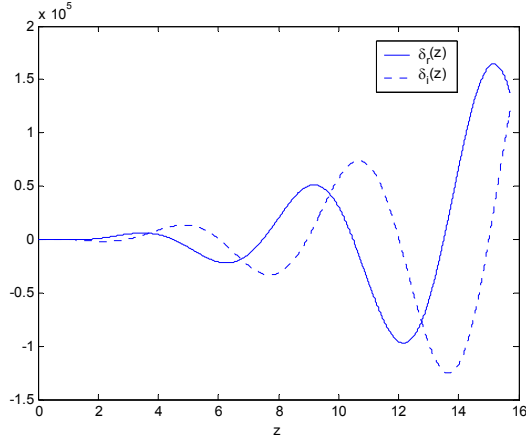


Figure 6-5. Plot of $\delta_i(z)$ and $\delta_r(z)$.

The roots of $\delta_r(z)$ in $z \in [0, 5\pi]$ are:

$$z_{r1} = 0.2575, z_{r2} = 1.5925, z_{r3} = 4.5185, z_{r4} = 7.4865, z_{r5} = 10.4865, z_{r6} = 13.5065 \quad (6.22)$$

It is clear from (6.18) and (6.22) that the interlacing of roots of $\delta_i(z)$ and $\delta_r(z)$ is satisfied [85]:

$$0 < z_{r1} < z_{i1} < z_{r2} < z_{i2} < \dots \quad (6.23)$$

From the interlacing property of $\delta_i(z)$ and $\delta_r(z)$ in (6.23), and the fact that $\delta_i(z)$ has only real roots, we conclude that $\delta_r(z)$ also has only real roots. Thus all conditions of Theorem 1 are satisfied, and the quasipolynomial $\Delta^*(s)$ is stable for the given β and L values, i.e., the networked PI control system with fuzzy logic modulation is stable for the given β and L values.

6.4 Stabilization using fuzzy logic modulator parameter

Now, we will consider the problem of stabilizing the networked PI control system using a fuzzy logic modulator. Using a version of the Hermite-Biehler Theorem applicable to

quasipolynomials, a complete analytical characterization of all stabilizing fuzzy logic modulator parameter values to stabilize the networked PI control systems is provided. In this theorem we assume a small network-induced time delay variation.

Theorem 3: The range of the fuzzy logic modulator parameter β for which the networked PI control system can be stabilized using PI controller with fuzzy logic modulator is given by

$$0 < \beta < \frac{1}{kk_p} \left[(T_1 + T_2) \frac{\alpha}{L} \sin \alpha - \left(1 - \frac{\alpha^2}{L^2} T_1 T_2 \right) \cos \alpha \right]$$

such that the interlacing condition on the roots of $\delta_i(z)$ and $\delta_r(z)$ is satisfied, where α is the solution of the equation

$$\tan \alpha = - \frac{\left(\frac{T_1 + T_2}{L} \right) + \left(2 \frac{T_1 T_2}{L^2} \right)}{\left(\frac{T_1 + T_2}{L} \right) + \left(1 - \frac{\alpha^2 T_1 T_2}{L^2} \right)} \alpha$$

Proof:

First, from the definition of fuzzy logic modulation parameter in chapter 4, this implies that $\beta > 0$. Then, we need to show that the imaginary part of $\Delta^*(j\omega)$ has only simple real roots if and only if

$$\beta < \frac{1}{kk_p} \left[(T_1 + T_2) \frac{\alpha}{L} \sin \alpha - \left(1 - \frac{\alpha^2}{L^2} T_1 T_2 \right) \cos \alpha \right] \quad (6.24)$$

Rewrite (6.17),

$$\frac{\left(1 - \frac{z^2}{L^2} T_1 T_2 \right) \cos z + k\beta k_p}{\sin z} = (T_1 + T_2) \frac{z}{L} \quad (6.25)$$

Now we define β_u as the largest number so that the plot of $\frac{\left(1 - \frac{z^2}{L^2} T_1 T_2\right) \cos z + k \beta k_p}{\sin z}$ intersects the line $(T_1 + T_2) \frac{z}{L}$ twice in the interval $(0, \pi)$. For the case $\beta > \beta_u$ and using Theorem 2, one can easily argue that in this case all roots of $\delta_i(z)$ will not be real, thereby ruling out closed-loop stability.

From the definition of β_u ; it follows that if $\beta = \beta_u$ the plot of $\frac{\left(1 - \frac{z^2}{L^2} T_1 T_2\right) \cos z + k \beta k_p}{\sin z}$ intersects the line $(T_1 + T_2) \frac{z}{L}$ only once in the interval $(0, \pi)$. Let α denote the value of z for which this intersection occurs. Then for $z = \alpha \in (0, \pi)$ we have

$$\frac{\left(1 - \frac{\alpha^2}{L^2} T_1 T_2\right) \cos \alpha + k \beta_u k_p}{\sin \alpha} = (T_1 + T_2) \frac{\alpha}{L} \quad (6.26)$$

Rewrite (6.26),

$$\left(1 - \frac{\alpha^2}{L^2} T_1 T_2\right) \cos \alpha + k \beta_u k_p = (T_1 + T_2) \frac{\alpha}{L} \sin \alpha \quad (6.27)$$

Moreover, at $z = \alpha$, the line $(T_1 + T_2) \frac{z}{L}$ is tangent to the plot of $\frac{\left(1 - \frac{z^2}{L^2} T_1 T_2\right) \cos z + k \beta_u k_p}{\sin z}$.

Thus

$$\frac{d}{dz} \left[\frac{\left(1 - \frac{z^2}{L^2} T_1 T_2\right) \cos z + k \beta k_p}{\sin z} \right]_{z=\alpha} = \frac{(T_1 + T_2)}{L} \quad (6.28)$$

After few lines of derivations,

$$\frac{-\left(1 - \frac{\alpha^2}{L^2} T_1 T_2\right) - 2 \frac{T_1 T_2}{L^2} z \sin z \cos z - k \beta_u k_p \cos z}{\sin^2 \alpha} = \frac{(T_1 + T_2)}{L} \quad (6.29)$$

Rewrite (6.29),

$$-\left(1 - \frac{\alpha^2}{L^2} T_1 T_2\right) - 2 \frac{T_1 T_2}{L^2} \alpha \sin \alpha \cos \alpha - k \beta_u k_p \cos \alpha = \frac{(T_1 + T_2)}{L} \sin^2 \alpha \quad (6.30)$$

Eliminating $k \beta_u k_p$ between equations (6.27), (6.30) by multiplying (6.27) with $\cos \alpha$ and add to it (6.30). After a few simple derivations, we conclude that $\alpha \in (0, \pi)$ can be obtained as a solution of the following equation:

$$\tan \alpha = - \frac{\left(\frac{T_1 + T_2}{L}\right) + \left(2 \frac{T_1 T_2}{L^2}\right)}{\left(\frac{T_1 + T_2}{L}\right) + \left(1 - \frac{\alpha^2 T_1 T_2}{L^2}\right)} \alpha \quad (6.31)$$

Once α is determined, the parameter β_u can be obtained using (6.27):

$$\beta_u = \frac{1}{k k_p} \left[(T_1 + T_2) \frac{\alpha}{L} \sin \alpha - \left(1 - \frac{\alpha^2}{L^2} T_1 T_2\right) \cos \alpha \right] \quad (6.32)$$

In order to check the interlacing condition on roots of $\delta_i(z)$ and $\delta_r(z)$, equation (6.21) is used.

For the fuzzy logic modulator parameter β in this range, i.e., $\delta_i(z)$ has only real roots, and from the satisfaction of interlacing condition of roots $\delta_i(z)$ and $\delta_r(z)$, we conclude that $\delta_r(z)$ also has only real roots. Thus all conditions of Theorem 1 are satisfied and the quasipolynomial $\Delta^*(s)$ is stabilized by the given fuzzy logic modulator parameter β , i.e.,

the networked PI control system is stabilized by the fuzzy logic modulator parameter. This completes the proof of the Theorem 3.

In the following sections, we will highlight some remarks on the stability of the networked PI control system.

6.4.1 Increasing in stability region

With the fuzzy logic modulation, we increased the stability region of the network-based controlled DC motor as shown in Fig. 6-6.

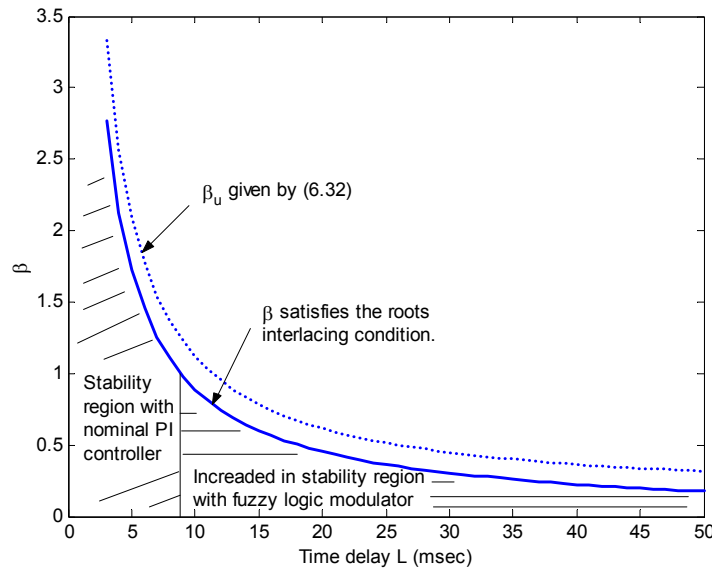


Figure 6-6. Plot of the stabilizing fuzzy logic modulator β as a function of L .

From Fig. 6-6, it is clear that the nominal PI controller (i.e., $\beta = 1$) can stabilize the networked PI control system up to $L = 9$ (msec), while for a network-induced time-delays larger than this upper bound value, the nominal PI controller cannot stabilize the networked PI control system. On the other hand, with the fuzzy logic modulator, the stability region of the networked PI control system is increased as shown in Fig. 6-6.

6.4.2 Delay-free closed loop system stability

With $L = 0$, i.e., no time delay, equation (6.31) reduced to

$$\tan \alpha = \frac{2}{\alpha} \quad (6.33)$$

can be solved analytically to obtain $\alpha = 1.08$. Substituting in (6.32),

$$\begin{aligned} \beta_u &= \frac{\alpha^2 T_1 T_2}{k k_p L^2} \cos \alpha \\ &= \infty \end{aligned} \quad (6.34)$$

which is consistent with the result in (6.6), i.e., both k_p and k_i are less than ∞ .

6.5 Simulation results

The network-based controlled DC motor system is simulated using MATLAB/SIMULINK. The motor state space description in (3.4)-(3.5) is used as the plant, and is controlled by the PI controller in (3.15). The network-induced time delays as described in (3.13)-(3.14) are simulated using gaussian random number generators. In this section we will simulate the dynamic responses of the network-based controlled DC motor corresponding to stable region, unstable region, and on border of stable/unstable regions.

6.5.1 Stable region

In this region, and based on Theorem 3, the fuzzy logic modulator parameter should be less than its upper bound value provided by Theorem 3. For example: with network-induced time-delays of $\text{mean}(\tau_C) = \text{mean}(\tau_R) = 7.5$ (msec), the step response of the network-based controlled DC motor subject to tracking a reference signal of 50 (rad/sec) is

shown in Fig. 6-7(a). The corresponding fuzzy logic modulator satisfies Theorem 3 (where the upper bound value of the fuzzy logic modulator is 0.59) is shown in Fig. 6-7(b).

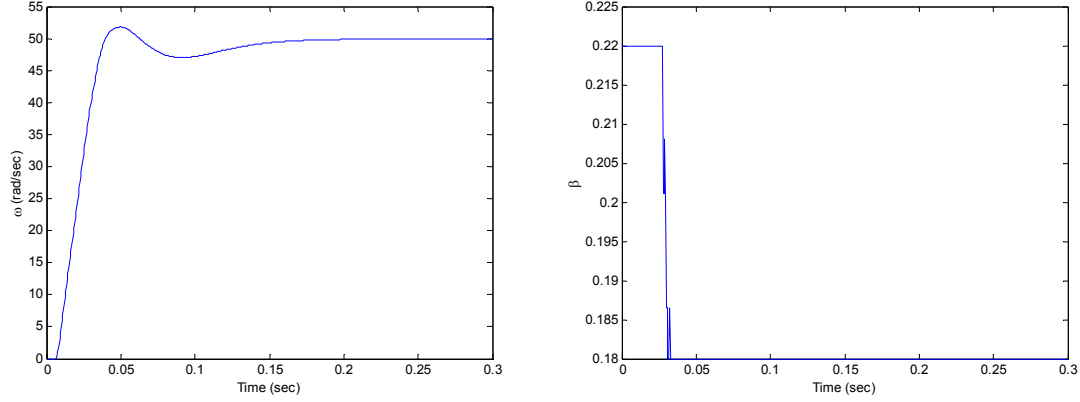


Figure 6-7. (a) Typical step response of the network-based controlled DC motor, (b) Corresponding fuzzy logic modulator parameter β .

Thus, by having the fuzzy logic modulator parameter less than the upper bound value provided by Theorem 3, we guarantee the stability of the networked PI control system with fuzzy logic modulator.

6.5.2 Unstable region

In this region, the fuzzy logic modulator parameter is greater than its upper bound value provided by Theorem 3. For example: with network-induced time-delays of $\text{mean}(\tau_C) = \text{mean}(\tau_R) = 5$ (msec), the step response of the network-based controlled DC motor subject to tracking a reference signal of 50 (rad/sec) is shown in Fig. 6-8(a). The corresponding fuzzy logic modulator, which violates Theorem 3 (where the upper bound value of the fuzzy logic modulator is 0.89) is shown in Fig. 6-8(b).

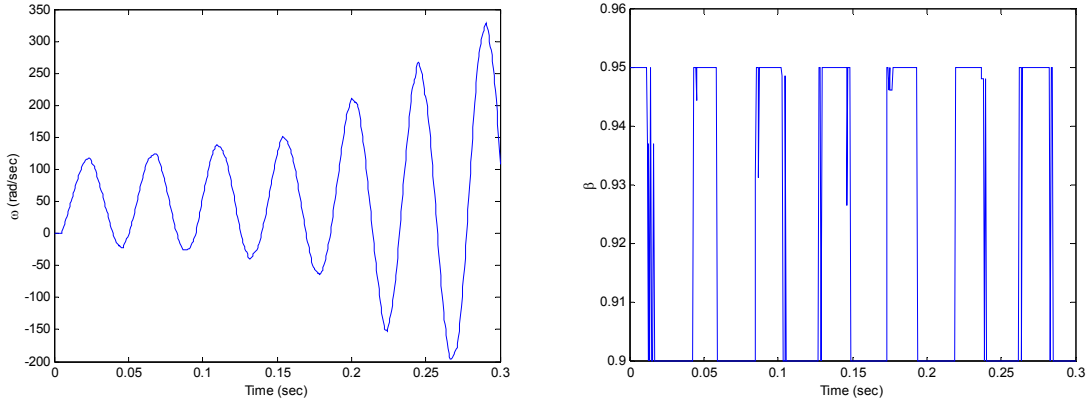


Figure 6-8. (a) Typical step response of the network-based controlled DC motor, (b) Corresponding fuzzy logic modulator parameter β .

The simulation results in Fig. 6-8 suggest that during the adaptation process of the adaptive fuzzy logic modulation, the upper bound of the fuzzy logic modulator provided by Theorem 3 should be taken into consideration; such that we adapt the fuzzy logic modulator and at the same time stabilize the networked PI control system.

6.5.3 On border of stable/unstable regions

In the boundary of stable/unstable region, the fuzzy logic modulator parameter is equal to its upper bound value provided by Theorem 3. For example: with network-induced time-delays of $\text{mean}(\tau_C) = \text{mean}(\tau_R) = 10$ (msec), the step response of the network-based controlled DC motor subject to tracking a reference signal of 50 (rad/sec) is shown in Fig. 6-9(a). The corresponding fuzzy logic modulator equals to the upper bound value on the fuzzy logic modulator of 0.4562 is shown in Fig. 6-9(b).

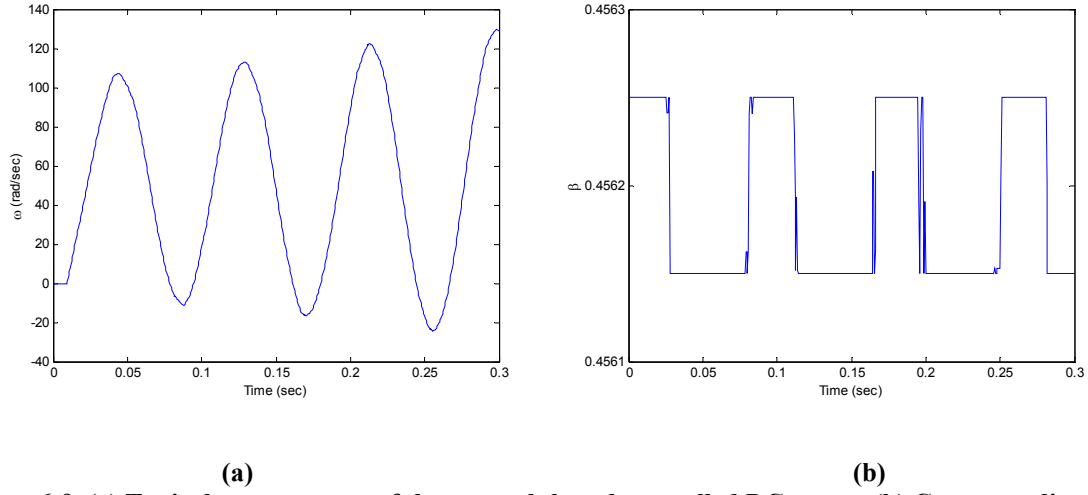


Figure 6-9. (a) Typical step response of the network-based controlled DC motor, (b) Corresponding fuzzy logic modulator parameter β .

As shown in Fig. 6-9, the networked PI control system has unstable oscillatory behavior since the fuzzy logic modulator parameter is equals to the upper bound value provided by Theorem 3.

6.6 Conclusion

In this chapter, we considered the problem of stabilizing the networked PI control system using a fuzzy logic modulator. Using a version of the Hermite-Biehler Theorem applicable to quasipolynomials, a complete analytical characterization of all stabilizing fuzzy logic modulator parameter values to stabilize the networked PI control is provided. Numerical simulation of a network-based controlled DC motor is used to illustrate the proposed work.

7 References:

- [1] Y. Halevi, and A. Ray, "Integrated communication and control systems: Part I - Analysis," *Journal of Dynamic Systems, Measurement, and Control*, vol. 110, pp. 367-373, 1988.
- [2] D. Radford, "Spread-spectrum data leap through AC power wiring," in *IEEE Spectrum*, vol. November, 1996, pp. 48-53.
- [3] R. S. Raji, "Smart networks for control," in *IEEE Spectrum*, vol. June, 1994, pp. 49-55.
- [4] U. Özgüner, H. Goktag and H. Chan, "Automotive suspension control through a computer communication network," presented at IEEE International Conference on Control and Applications, 1992.
- [5] A. Ray, "Performance evaluation of medium access control protocols for distributed digital avionics," *ASME Journal of Dynamic Systems, Measurements and Control*, vol. 109, pp. 370-377, 1987.
- [6] J. A. Sparks, "Low cost techniques for aerospace applications," *Microprocessors and Microsystems*, vol. 20, pp. 449-454, 1997.
- [7] S. Schneider, "Making Ethernet work in real time," in *Sensors*, vol. 17, 2000.
- [8] P. R. Kumar, "New technological vistas for systems and control," in *IEEE Control Systems Magazine*, vol. February, 2001, pp. 24-37.
- [9] G. Kaplan, "Ethernet's Winning Ways," in *IEEE Spectrum*, vol. 38, 2001, pp. 113-115.
- [10] Feng-Li Lian, J. R. Moyne, and Dwan M. Tilbury, "Performance evaluation of control networks," in *IEEE Control Systems Magazine*, vol. February, 2001, pp. 66-83.
- [11] P. Sink, "A Comprehensive guide to industrial networks," in *Sensors*, vol. 18, 2001.
- [12] Wei Zang, M. S. Branicky, and Stephen M. Phillips, "Stability of networked control systems," in *IEEE Control Systems Magazine*, vol. Feb., 2001, pp. 84-99.
- [13] Eitan Altman, T. Basar, and R. Srikant, "Congestion control as a stochastic control problem with action delays," *Automatica*, vol. 35, pp. 1937-1950, 1999.

- [14] S. Mascolo, "Classical control theory for congestion avoidance in high-speed Internet," presented at Conference on Decision & Control, Phoenix, Arizona USA, Dec. 1999.
- [15] L. Dugard, and Verriest, Erik I., *Stability and control of time-delay systems*, vol. 228. New York: Springer, 1998.
- [16] J. D. Spragins, *Telecommunications: protocols and design*, Repr. with corrections ed. Reading MA: Addison-Wesley, 1992.
- [17] W. Lawrenz, *CAN system engineering: from theory to practical applications*. New York: Springer, 1997.
- [18] A. Tanenbaum, *Computer Networks*, third ed. NJ: Prentic Hall PTR, 1996.
- [19] Michael Tangemann, and Klaus Sauer, "Performance analysis of the timed token protocol of FDDI and FDDI-II," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 271-278, Feb. 1991.
- [20] D. Anderson, *FireWire system architecture*, second ed. M.A.: Addison-Wesley, 1999.
- [21] Jamahl W. Overstreet and Anthony Tzes, "An Internet-based real-time control engineering laboratory," in *IEEE Control Systems Magazine*, 1999, pp. 19-34.
- [22] M. Exel, S. Gentil, F. Michau, and D. Ray, "Simulation workshop and remote laboratory: two web-based training approaches for control," presented at American Control Conference, Chicago, Illinois, June 2000.
- [23] D. A. Schoenwald, "AUVs: In space, air, water, and on the ground," in *IEEE Control Systems Magazine*, vol. 20, 2000, pp. 15-18.
- [24] F. Giulietti, L. Pollini, and M. Innocenti, "Autonomous formation flight," in *IEEE Control Systems Magazine*, vol. 20, 2000, pp. 34-44.
- [25] Thuan Nguyen, Ferit Yegenoglu, and Agatino Sciute, "Voice over IP service and performance in satellite networks," in *IEEE Communications Magazine*, 2001, pp. 164-171.
- [26] Dapeng Wu, Yiwei Thomas Hou, Wenwu Zhu, Ya-Qin Zhang, and Jon M. Peha, "Streaming video over the internet: approaches and directions," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, pp. 282-300, 2001.
- [27] J. F. Kurose, *Computer networking: a top-down approach featuring the Internet*. Boston: Addison-Wesley, 2001.

- [28] A. Ray, "Networking for real-time control of integrated manufacturing processes," presented at IEEE International Conference on Robotics and Automation, 1988.
- [29] A. Ray, "Network access protocols for real-time distributed control systems," *IEEE Trans. on Industrial Applications*, vol. 24, pp. 897-904, 1988.
- [30] J. Nilsson, B. Bernhardsson, and B. Wittenmark, "Stochastic analysis and control of real-time systems with random time delays," *Automatica*, vol. 34, pp. 57-64, 1998.
- [31] S. H. Hong, "Scheduling algorithm of data sampling times in the integrated communication and control systems," *IEEE Transactions on Control Systems Technology*, vol. 3, pp. 225-230, 1995.
- [32] Y. H. Kim, S. H. Park, and W. H. Kwon, "Stability and a scheduling method for network-based control systems," presented at IEEE IECON 22nd International Conference on Industrial Electronics, Control, and Instrumentation, 1996.
- [33] Y. H. Kim, S. H. Park, and W. H. Kwon, "A scheduling method for network-based control systems," presented at American Control Conference, 1998.
- [34] K. Zahr, and C. Slivinsky, "Delay in multivariable computer controlled linear systems," *IEEE Trans. on Automatic Control*, vol. 19, pp. 442-443, 1974.
- [35] Michael S. Branicky, S. M. Phillips, and Wei Zhang, "Stability of networked control systems: explicit analysis of delay," presented at American Control Conference, Chicago, Illinois, 2000.
- [36] V. B. Kolmanovskii, S.-L. Niculescu, and Keqin Gu, "Delay effects on stability: A survey," presented at Conference on Decision & Control, Phoenix, Arizona, 1999.
- [37] N. S. Nise, *Control systems engineering*, 2nd ed. Redwood, CA: Benjamin/Cummings, 1995.
- [38] Mo-Yuen Chow, and Y. Tipsuwan, "Network-based control adaptation for networks QoS variation," presented at MILCOM 2001, McLean, VA, 2001.
- [39] Gene F. Franklin, J. D. Powell, and Abbas Emami Naeini, *Feedback control of dynamic systems*. Reading, MA: Addison-Wesley, 1994.
- [40] Y. Tipsuwan, "Network-Based Control System: Direct Structure," North Carolina State University, Raleigh, Qualifying Review Report December 2000.
- [41] Mo-Yuen Chow and Y. Tipsuwan, "Network-Based Control Systems: A Tutorial," presented at The 27th Annual Conference of the IEEE Industrial Electronics Society, IECON' 01, Denver, CO, 2001.

- [42] K. J. Åström, and B. Wittenmark, *Computer-controlled Systems: theory and design*, third ed. Upper Saddle River, NJ: Prentice Hall, 1997.
- [43] R. Luck, and A. Ray, "An observer-based compensator for distributed delays," *Automatica*, vol. 26, pp. 903-908, 1990.
- [44] G. C. Walsh, O. Beldiman, and L. Bushnell, "Asymptotic behavior of networked control systems," presented at IEEE International Conference on Control Applications, 1999.
- [45] H. Chan, and U. Özgüner, "Closed-loop control of systems over a communication network with queues," *International Journal of Control*, vol. 62, pp. 493-510, 1995.
- [46] A. Ray, and Y. Halevi, "Integrated communication and control systems: Part II - Design considerations," *Journal of Dynamic Systems, Measurement, and Control*, vol. 110, pp. 374-381, 1988.
- [47] J. Nilsson, "Real-time control systems with delays," in *Department of Automatic Control*. Lund, Sweden: Lund Institute of Technology, 1998.
- [48] G. C. Walsh, O. Beldiman, and L. Bushnell, "Stability analysis of networked control systems," presented at American Control Conference, 1999.
- [49] L.-W. Liou, and A. Ray, "Integrated communication and control systems: Part III - Nonidentical sensor and controller sampling," *Journal of Dynamic Systems, Measurement, and Control*, vol. 112, pp. 357-364, 1990.
- [50] R. Luck, and A. Ray, "Experimental Verification of a delay compensation algorithm for integrated communication and control systems," *International Journal of Control*, vol. 59, pp. 1357-1372, 1994.
- [51] G. C. Walsh, "Scheduling of networked control systems," in *IEEE Control Systems Magazine*, 2001, pp. 57-65.
- [52] H. Khalil, *Nonlinear Systems*. New York: Maxwell Macmillan International, 1992.
- [53] G. C. Walsh, O. Beldiman, and L. Bushnell, "Error encoding algorithms for networked control systems," presented at IEEE Conference on Decision and Control, 1999.
- [54] T.-J. Su, and C-G Huang, "Robust stability of delay dependence for linear uncertain systems," *IEEE Transactions on Automatic Control*, vol. 37, pp. 1656-1659, 1992.
- [55] P.-L. Liu, and T.-J. Su, "On a sufficient condition for robust stability of delay systems," presented at American Control Conference, 1995.

- [56] T. Mori, N. Fukuman, and M. Kuwahara, "Simple stability criteria for single and composite linear systems with time delays," *International Journal of Control*, vol. 34, pp. 1175-1184, 1981.
- [57] Y. Tipsuwan, and Mo-Yuen Chow, "Network-Based Controller Adaptation Based on QoS Negotiation and Deterioration," presented at The 27th Annual Conference of the IEEE Industrial Electronics Society, IECON'01, Denver, CO, 2001.
- [58] B. Raymond and S. Jr., "Real-time motor testing and development over the internet," in *Sensors*, 2001, pp. 30-39.
- [59] R. C. Dorf and R. H. Bishop, *Modern control systems*. Upper Saddle River, NJ: Prentice-Hall, 2001.
- [60] W. L. Brogan, *Modern control theory*, 3rd ed ed. N.J.: Englewood Cliffs, Prentice Hall, 1991.
- [61] K. Ogata, *Modern control engineering*, 3rd ed ed. Upper Saddle Rive, NJ: Prentice Hall,, 1997.
- [62] K. J. Åström, and T. Hagglund, *PID controllers: Theory, Design, and Tuning*. Raleigh, NC: Research Triangle Park, 1995.
- [63] H. Saadat, *Computational aids in control systems using MATLAB*. New York: McGraw-Hill, 1993.
- [64] K. Zhou and J. C. Doyle, *Essentials of robust control*. Upper Saddle River, NJ.: Prentice Hall, 1998.
- [65] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press ed, 1981.
- [66] Gene F. Franklin, J. D. Powell, and Michael L. Workman, *Digital control of dynamic systems*, 3rd ed. Menlo Park, Calif: Addison Wesley, 1998.
- [67] J. T. Teeter, Mo-Yuen Chow, and J. Brickley, "A noval fuzzy friction compensation approach to improve the performance of a DC motor control system," *IEEE Trans. on Industrial Electronics*, vol. 43, pp. 113-120, 1996.
- [68] H.-S. Hwang, J.-N. Choi, W.-H. Lee, and J.-K. Kim, "A tuning algorithm for the PID controller utilizing fuzzy theory," presented at International Joint Conference on Neural Networks, IJCNN '99., 1999.
- [69] Mo-Yuen Chow, "Fuzzy logic-based control," in *The industrial electronics handbook*, vol. Chapter 39, 1996.

- [70] N. B. Almutairi, Mo-Yuen Chow, and Y. Tipsuwan, "Network-based controlled DC motor with fuzzy compensation," presented at The 27th Annual Conference of the IEEE Industrial Electronics Society, IECON'01, Denver, CO, 2001.
- [71] L.-X. Wang, *A course in fuzzy systems and control*. NJ: Prentic-Hall, 1996.
- [72] N. B. Almutairi and Mo-Yuen Chow, "PI Parameterization Using Adaptive Fuzzy Modulation (AFM) for Networked Control Systems- Part I: Partial adaptation," presented at IECON 2002, Sevilla, Spain, 2002.
- [73] N. B. Almutairi and Mo-Yuen Chow, "PI Parameterization Using Adaptive Fuzzy Modulation (AFM) for Networked Control Systems- Part II: Full adaptation," presented at IECON 2002, Sevilla, Spain, 2002.
- [74] S. Altug, Mo-Yuen Chow, and J. Trussell, "Heuristic constraints enforcement for training of and rule extraction from a fuzzy/neural architecture- Part II: Implementation and application," *IEEE Trans. on Fuzzy Systems*, vol. 7, pp. 151-159, 1999.
- [75] Mo-Yuen Chow, S. Altug, and J. Trussell, "Heuristic constraints enforcement for training of and rule extraction from a fuzzy/neural architecture- Part I: Foundation," *IEEE Trans. on Fuzzy Systems*, vol. 7, pp. 143-150, 1999.
- [76] M. S. Branicky, "Stability of hybrid systems: State of the art," presented at Conference on Decision & Control, San Diego, California, 1997.
- [77] C. Lorand, M. Sicitui, P. Bauer, and G. Schmidt, "Stability of first order discrete time systems with time-variant communication delays in the feedback path," presented at The 2000 IEEE Asia-Pacific Conference on Circuits and System, 2000.
- [78] P. H. Bauer, M. L. Sicitui, and K. Premaratne, "Stability of 2-D distributed processes with time-variant communication delays," presented at The 2001 IEEE International Symposium on Circuits and Systems, 2001.
- [79] M. L. Sicitui and P. H. Bauer, "Stability of discrete time-variant linear delay systems and applications to network control," presented at The 8th IEEE International Conference on Electronics, Circuits and Systems, 2001.
- [80] P. H. Bauer, K. Premaratne, and J. Duran, "A necessary and sufficient condition for robust asymptotic stability of time-variant discrete systems," *IEEE Transactions on Automatic Control*, vol. 38, pp. 1427-1430, 1993.
- [81] M. Wargui, K. Hentabi, M. Tadjine, and A. Rachid, "Effect of network induced delay on the stability of an autonomous mobile robot," presented at International Conference on Industrial Electronics, Control and Instrumentaion, IECON 97, 1997.

- [82] M. Wargui, A. Tayebi, M. Tadjine, and A. Rachid, "On the stability of an autonomous mobile robot subject to network induced delay," presented at IEEE International Conference on Control Applications, Hartford, CT, 1997.
- [83] K. Watanabe, E. Nobuyama, and A. Kojima, "Recent advances in control of time delay systems - A tutorial review -," presented at Conference on Decision & Control, Kobe, Japan, 1996.
- [84] G. C. Walsh, O. Beldiman, and L. G. Bushnell, "Asymptotic behavior of nonlinear networked control systems," *IEEE Transactions on Automatic Control*, vol. 46, pp. 1093-1097, 2001.
- [85] M.-T. Ho, A. Datta, and S. P. Bhattacharyya, "Generalization of the Hermite-Biehler theorem," *Linear Algebra and its Applications*, vol. 302-303, pp. 135-153, 1999.
- [86] G. J. Silva, A. Datta, and S. P. Bhattacharyya, "Stabilization of time delay systems," presented at American Control Conference, Chicago, Illinois, 2000.
- [87] G. J. Silva, A. Datta, and S. P. Bhattacharyya, "New results on synthesis of PID controllers," *IEEE Trans. on Automatic Control*, vol. 47, pp. 241-252, 2002.
- [88] G. J. Silva, A. Datta, and S. P. Bhattacharyya, "Determination of stabilizing feedback gains for secon-order systems with time delay," presented at American Control Conference, Arlington, VA, 2001.
- [89] S. P. Bhattacharyya, H. Chapellate, and L. H. Keel, *Robust Control: The Parametric Approach*. New Jersey: Prentice Hall, 1995.
- [90] R. Bellman and K. L. Cook, *Differential-Difference Equations*. London: Academic Press Inc., 1963.