國立交通大學

電 控 工 程 研 究 所

碩 士 論 文

網路控制系統之時間延遲控制與訊息估測之分析

Analysis of the Time-delay Control and the Message Estimation for Networked Control Systems

研 究 生： 張孟哲
指導教授： 徐保羅 博士

中 華 民 國 九 十 九 年 六 月

網路控制系統之時間延遲控制與訊息估測之分析

Analysis of the Time-delay Control and the Message Estimation for Networked Control Systems

研　究　生：張孟哲	Student: Meng-Zhe Zhang
指導教授：徐保羅 博士	Advisor: Dr. Pau-Lo Hsu

國立交通大學

電控工程研究所

碩士論文

A Thesis

Submitted to Institute of Electrical Control Engineering

College of Electrical and Computer Engineering

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master

in

Electrical Control Engineering

June 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年六月

網路控制系統之時間延遲控制與訊息估測之分析

學生： 張孟哲　　　　　　　　　　　　　指 導 教 授 ： 徐保羅 博士

國立交通大學電控工程研究所

摘要

網路控制系統(NCS)改進了傳統的點對點連接方式，提供了安裝、使用以及擴展方面的便利性。許多研究著重在網路訊號延遲和資料遺失這兩個主題上。本論文提出了一個針對PI網路控制器、以系統頻寬爲目標的參數自動調整方法。經由控制器參數的重新調整，可以在控制帶有變動因素的平臺時達到特定的系統頻寬要求，並且能夠降低因網路延遲時間變化產生的系統穩定性劣化程度。因爲PI控制器在網路控制上有其極限，本論文提出了一個採用H_∞演算法的網路控制器設計流程，並且演示於一經系統辨別而得的伺服控制系統模型上。在閉迴路的控制架構之外，額外加上了一個前饋式補償器，以符合對於系統頻寬及延遲容忍度雙方面的要求。在資料經網路傳送後遺失的情況發生時，本論文提出了以估測的方式重建訊息使系統得以維持正常運作，追跡效能也不至於過度劣化。在控制器下命令至驅動器的路徑上，使用了對過往命令作多項式近似的MFLSF訊息估測器；在感測器回傳系統動態資料至控制器的路徑上，則使用了建立在系統模型上的MBLSF訊息估測器。在以馬可夫鏈爲網路模型的前提之下，探討了這兩類型的訊息估測器，並在不同網路情境下討論其可用性與追跡效能。

Analysis of the Time-delay Control and the Message Estimation for Networked Control Systems

Student: Meng-Zhe Zhang Advisor: Dr. Pau-Lo Hsu

Institute of Electrical Control Engineering
National Chiao Tung University

ABSTRACT

Networked control systems (NCSs) offer advantages over the traditional point-to-point connections and has been studied mainly in the network-induced delay and the package dropout. Many theoretical approaches has been proposed to handle this two problems with various algorithms. In this thesis, a general PI controller auto-tuning method based on the optimization process is proposed. The specified system bandwidth can be achieved after the tuning process without knowing the exact plant model applied to NCS. To handle the network delay, actual procedures for H_∞ controller design for NCSs is presented under the mixed synthesis framework. The control structure with a forward compensator is used to meet specifications in both delay tolerance and bandwidth. For compensating network dropout in controller-to-actuator and sensor-to-controller paths, both the model-free least square fit (MFLSF) estimator and the model-based least square fit estimator (MBLSF) are used to improve the tracking performance, respectively, and stability criteria of both MFLSF and MBLSF estimators are provided with the Markov chain model. By switching MFLSF estimators according to the current identified network status, total tracking performances can be thus significantly improved.

Acknowledgment

這篇論文能夠完成，首先應該感謝我的指導教授徐保羅博士在我碩士生涯中的指點與鼓勵。沒有他不斷的引導，也許我會在找尋研究方向的迷途中放棄。

另外得感謝網路化監控系統實驗室裡的所有人。博士班的煒生、琮政、鎮洲、和建良學長以及碩士班的宗勝、醇偉、建龍、和也強學長，謝謝他們對我的指導。還有碩士班的同學與學弟們，我自覺付出的心力遠比不上從他們那裡所學習到的。在實驗室的這幾年如果我有了任何成長，他們必定是其中重要的因素。謝謝他們對我的包容與支持。再來是助理志民與慧霖，感謝他們在各項事務的處理上提供的協助。

最後要感謝我的父母，他們平日的關心和供給，讓我可以無後顧之憂地取得碩士學位。長久以來我一直認爲應該要依靠自己完成所有的事，但是這篇論文的完成絕非出於我一人之力。

感謝所有幫助過我的人，謝謝你們。

Contents

中文摘要 i
Abstract ii
Acknowledgment iii
Contents iv
List of Tables vii
List of Figures viii
List of Notations xi
Chapter 1 Introduction 1
1.1 Motivation 1
1.2 General Review 1
1.3 Problem Statement 2
1.4 Proposed Approaches 3
1.5 Organization of the Thesis 3
Chapter 2 Controller Auto-Tuning in the Frequency Domain 4
2.1 Estimation of the Frequency Responses 4
2.2 Optimization for Control Parameters 5
2.3 Simulation 9
2.4 Experiments 12
2.4.1 System Setup 12
2.4.2 Results 15

2.5 Auto-Tuning for the Network Delay 15

2.6 Summary 19

Chapter 3 H_∞ Design for the Network Delay Systems 23

3.1 Theoretical Structure 23

3.2 Weighting Functions 24

3.2.1 Template for W_s 25

3.2.2 Template for W_r 26

3.2.3 Template for W_t 26

3.3 H_∞ NCS Design Procedures 28

3.3.1 Performance Target 28

3.3.2 Design Weighting Functions 29

3.3.3 Adjust H_∞ Algorithm Property 29

3.3.4 Order Reduction for the Controller 31

3.3.5 Forward Compensator Design 33

3.3.6 Simulation 34

3.4 Summary 37

Chapter 4 Analysis of Message Estimation for the Dropout 38

4.1 Markov Chain as Network Model 38

4.2 System Stability with Markov Chain Network 40

4.3 Network Dropout Modeled by the Markov Chain 42

4.3.1 Network Dropout 42

4.3.2 Markov Model Measurement 42

4.3.3 Dropout Estimation 45

4.4 The MBLSF Estimation 47

4.4.1 Structure 47

4.4.2 Simulation Setup 49

4.4.3 Stability 50

4.4.4 Tracking Performance 51

4.5 The MFLSF Estimation 54

4.5.1 Structure 54

4.5.2 Stability 56

4.5.3 Tracking Performance 57

4.5.4 Estimator Switching Policy 59

4.6 Summary 62

Chapter 5 Conclusion 63

5.1 Conclusion 63

5.2 Future Work 63

References 65

List of Tables

2.1 Estimated magnitude and phase estimation in Fig. 2.2 with 100 trials. . . 5

2.2 Simulation results for PI auto-tuning with initial $K_p = 3$ and $K_i = 0.1$. . . 11

2.3 PI controller parameters auto-tuning results for different initial x. 15

2.4 Results of control parameters auto-tuning with extra delay $z^{-\tau}$ at ω_f =100(Hz). 20

3.1 Specification for system performances. 30

3.2 Performance comparison. 33

4.1 $\hat{Q}$ estimation with different Q. 45

4.2 $\mathcal{H}$ and $\mathcal{T}$ values for plant P with$W \leq 4$. 51

4.3 Tracking RMS error and the maximum error with different order of MBLSF estimations with $[q_1, q_2] = [0.1, 0.8]$ simulated with a noise v_k = uniform$[\pm 0.1]$. 51

4.4 $[p_i]$ values for $M \leq 3$. 57

4.5 RME error for MFLSF estimations with different $[q_1, q_2]$. 58

4.6 The maximum tracking error for MFLSF with different $[q_1, q_2]$. 59

4.7 Tracking performance for static MFLSF estimation and MFLSF with local Q estimation with (a)$[q_1, q_2] = [0.7, 0.2]$ (b)$[q_1, q_2] = [0.4, 0.2]$, and (c) $[q_1, q_2] = [0.7, 0.4]$ 62

List of Figures

2.1 An illustration for frequency response auto estimation from both the input and the output signals. 6

2.2 The I/O signals before and after applying the notch filter. 6

2.3 An illustration for PI control parameters auto-tuning process at frequency ω_f. 8

2.4 Step responses for PI controller auto-tuning with $\omega_f = 50$(Hz) at the k_{th} iteration. (Bold: turning result at $k = 100$.) 10

2.5 Step responses for systems with the tuned PI controllers at $\omega_f = 50$, 100, 200, and 400(Hz), respectively. 11

2.6 The transient values of ΔM, ΔP, K_p, and K_i during each iteration with $\omega_f = 100$(Hz). 12

2.7 Picture of the servo motor system. 13

2.8 Block diagram of the servo motor system. 13

2.9 The I/O signals before and after applying the notch filter. 14

2.10 The effects on M and P from different (a)K_p and (b)K_i. 14

2.11 Optimization starts with $M < M_0$ at $\omega_f = 200$(Hz). 16

2.12 Optimization starts with $M > M_0$ at $\omega_f = 200$(Hz). 16

2.13 Optimization starts with $M < M_0$ with different step size $c = 0.0625$, $c = 0.5$ and $c = 4$ at $\omega_f = 200$(Hz). 17

2.14 Optimization starts with $M < M_0$ and with 5 times extra loading at $\omega_f = 100$(Hz). 17

2.15 Optimization starts with $M < M_0$ at $\omega_f = 500$(Hz). 18

2.16 Optimization starts with $M < M_0$ at $\omega_f = 500$(Hz), repeated. 18

2.17 Step responses after auto-tuning at $\omega_f = 100$(Hz) for different delay time $z^{-\tau}$ with τ =1, 2, 4, and 8. The sampling time $h = 0.133$(ms). 20

2.18 Step responses before and after auto-tuning at $\omega_f = 50$(Hz) for the system with z^{-20}. 21

2.19 The transient values of ΔM, ΔP, K_p, and K_i during the tuning process for Fig. 2.18. 21

3.1 General linear fractional transformation(LFT) framework. 23

3.2 Mixed synthesis framework for H_∞ design. 24

3.3 Template for the weighting function W_s. 26

3.4 Template for the weighting function W_r. 27

3.5 Frequency response for W_t and ΔP with $\theta = 10$ and $\theta = 30$. 28

3.6 New NCS control structure with a forward compensator. 30

3.7 Sensitivity functions and weighting functions. (Solid: weighting functions; dotted: sensitivity functions.) 31

3.8 Frequency responses before and after controller order reduction. (Solid: before reduction; dotted: after reduction.) 32

3.9 Illustration for H_∞ NCS controller design. 33

3.10 System responses after a forward compensator applied. 34

3.11 Comparison for lead controller and H_∞ controller in both (a)the frequency response and (b)the time response. 35

3.12 Transient response for different controller, with a 1Hz sine wave input and a delay $T_d = 0.7$ after $t = 19$. 36

4.1 A NCS block diagram. 38

4.2 A two-state Markov chain with transition probability q_1 and q_2. 39

4.3 Dropout rate versus Q. 43

4.4 The two-state Markov chain with (a) $[q_1, q_2] = [0.5, 0.7]$ and $[0.7, 0.7]$ (dropout rate 0.5) (b) $[q_1, q_2] = [0.5, 0.2]$ and $[0.7, 0.52]$ (dropout rate 0.385). 44

4.5 Transient of estimated $\hat{Q}$ with $[q_1, q_2] = [0.5, 0.2]$ for the case shown in the top figure of Fig. 4.4(b). 46

4.6 Markov network with dropout and the forward and feedback estimation. . 46

4.7 Maximum eigenvalue for different Q in MBLSF estimation with (a) $W = 2$ (b) $W = 3$. 52

4.8 Time response for MBLSF with $W = 2$ at $[q_1, q_2] = [0.1, 0.8]$. 53

4.9 Time response for MBLSF with $W = 2$ and noise at $[q_1, q_2] = [0.1, 0.8]$. . . 53

4.10 Stability region on Q plane for MFLSF with different (M, N). 57

4.11 Time responses for MFLSF(1,0), MFLSF(2,1) and MFLSF(3,2) and $[q_1, q_2] = [0.4, 0.2]$. 58

4.12 2×2 reference points and the values of local $\bar{Q}$ with $j = 5$. 60

4.13 Local $\bar{Q}$ mapped into different reference points according to the maximum likelyhood. 60

4.14 The probability values for each local $\bar{Q}$ from 4 different reference points. . 61

List of Notations

$\lVert\cdot\rVert_\infty$	H_∞ norm
1_A	$\mathbb{E}(1_A) = \mathbb{P}(A)$, $\mathbb{E}(1_A\vert B) = \mathbb{P}(A\vert B)$, and $\mathbb{E}(f1_A) = \mathbb{P}(A)\mathbb{E}(f\vert A)$
vec$\mathcal{G}$	For $\mathcal{G} \in \mathbb{R}^{m\times n}$, vec$\mathcal{G} = \begin{bmatrix} \mathcal{G}^T_{1\sim m,1} & \mathcal{G}^T_{1\sim m,2} & \cdots & \mathcal{G}^T_{1\sim m,n} \end{bmatrix}^T \in \mathbb{R}^{mn\times 1}$
$\otimes$	For $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$, $A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{mp\times nq}$
I_N	identity matrix with size n by n
0_N	zero matrix with size n by n

Chapter 1 Introduction

1.1 Motivation

Recently, the use of network technology has dramatically increased. Control systems with network communication between loop paths are called networked control systems (NCS). The NCS structure offers advantages over the traditional point-to-point connections since it reduces wiring cost, increases configuration flexibility, and is generally simple in installation and maintenance. However, new problems and challenges arise [1,2]. Therefore, more and more researches focus on this area and different topics such as network-induced delay and package dropout have been analyzed to motivate this study.

1.2 General Review

The PI controllers are widely used in different types of real applications, including NCS control. Methods for tuning the PI control parameters are studied widely. A PI controller auto-tuning method based on the desired gain-margin and the phase-margin is proposed in [3] with the assumption that plant is limited to first or second order plus a dead-time to obtain approximated solutions. An analytical solution for PID control parameters based on exact plant frequency responses is provided in [4] and it leads to a difficult condition to be satisfied in real situations. Tuning procedures with complex algorithms, such as the neural network [5] or the genetic algorithm [6], need too much calculating time and are unsuitable to be realized on a simple DSP micro-processor. A DSP-based auto-tuning algorithm is realized in [7] using the gradient method, but with the time-domain indices like the maximum overshoot or the rise time, which are easily affected by sensor noises or system oscillations.

Discussions for NCS design with both the network delay and the package dropout are presented in [8,9], mainly with model-based design methods. Also, the estimator is realized with an observer with additional states, and state estimation convergence need to be considered. Nikolai Vatanski *et al.* proposed an approach with Smith predictor [10], and they point out some main drawbacks such as sensitivity to model mismatch, and it needs to be solved by applying adaptive procedures. Many researches also focus on NCS with H_∞ theorem [11–13], but few of them can be directly realized.

An FIR type estimator is proposed in [14] with model-based receding horizon state estimation and its stability criteria. For motion command estimation, there is no suitable pre-defined model and a model-free estimation technique such as the Taylor estimation or the least square estimation [15] are commonly applied. However, the stochastic nature of the network is usually not included in these estimators. The stochastic model for network is often used in stability analysis [16] with different criteria, but its restraints limits its application in NCS.

In [17] and [18], NCSs with servo motors and networks like CAN or Ethernet are concerned. All experiments and identified plants in this thesis are based upon them.

1.3 Problem Statement

1. Normally, remote controllers are designed separately by experts. When the network is involved, system responses are thus changed and controllers need to be re-tuned. Therefore, a simple yet efficient auto-tuning algorithm for NCS controllers that can be implemented with a DSP micro-controller is thus needed.

2. In a networked control system, data transmission between the controller and the plant acts like a time delay which is the key issue to be concerned in NCS. The delay effect on system performance is different in different NCSs, and all of them need to be considered in the controller design.

3. Network dropout occurs when packages collide or they arrive too slow in which it is considered outdated. When dropout happens, network transmission fails to pass data to the actuator input in the forward path or the controller input in the feedback path, the missing messages must be estimated to keep system working in a normal condition.

1.4 Proposed Approaches

1. A control parameters optimization algorithm is proposed with the frequency response measurement. A test signal is fed into system to excite responses in particular frequency bands and the measured magnitude responses are used in objective function during optimization procedures to render a desirable system bandwidth. It is assumed here that the network situation is with constant delay time.

2. In NCS controller design, the induced time delay is included with other specifications. Controller design with multiple performance requirements can be solved by applying the H_∞ algorithm. Each requirement is transformed into a weighting function and an overall synthesis is performed in mixed sensitivity framework. By adjusting weighting functions, a controller that fits all requirements can be thus found. It is assumed here that the network situation is with the maximum delay time.

3. The Markov chain theorem is used to model network transmission dropout behavior. When dropout happens, two least square estimators with and without the model are proposed to estimate missing network packages in both the feedback path and the forward path, respectively. Under the Markov network model, stability criteria for message estimators are provided.

1.5 Organization of the Thesis

The rest of this thesis is organized as follows: Chapter 2 proposed a PI controller autotuning technique for NCS; Chapter 3 describes how H_∞ algorithm is applied to NCS controller design with a forward compensator. In Chapter 4, analysis of the network dropout estimators are provided to verify the stability of NCS. Finally, Chapter 5 concludes this thesis.

Chapter 2 Controller Auto-Tuning in the Frequency Domain

2.1 Estimation of the Frequency Responses

In the tuning procedure of the PI control parameters, suitable indices must be determined to properly express system performance. Generally, there are two types of indices in either the time domain or the frequency domain. Because the time domain indices, such as the rise time, the maximum overshoot and the settle time, are easily affected by system steady-state oscillations and sensor noises in real applications, frequency-domain performance indices are used in this proposed approach.

A frequency response contains the magnitude and the phase parts, that is,

$$P(j\omega) = |P(j\omega)|\angle P(j\omega) \tag{2.1}$$

for a plant $P(s)$ at a certain frequency ω. A frequently used index is the bandwidth ω_b that

$$|P(j\omega_b)| = \frac{1}{\sqrt{2}} \tag{2.2}$$

and the phase margin ϕ at the gain-cross frequency ω_ϕ that

$$\phi = \pi - \angle P(j\omega_\phi) \tag{2.3}$$

where $|P(j\omega_\phi)| = 1$. The bandwidth is an index that represents how fast a system can respond, and is measured from magnitude response. The phase margin is an index to represent how stable a system is, and is measured from both magnitude and phase responses. Therefore, the bandwidth is adopted as an index in the tuning procedure because it is easy to measure. An alternative index for the phase margin is the closed-loop phase responses at the bandwidth frequency ω_b.

The discrete Fourier transform (DFT) is often adopted in the frequency response for analysis and design. However, it is not easy for a DSP micro-processor to perform this operation without particular hardware modules or a high precision sine table to calculate

Table 2.1: Estimated magnitude and phase estimation in Fig. 2.2 with 100 trials.

	M	avg. $\hat{M}$	variance
M	0.707	0.7065	0.0308
P	150°	149.08°	2.817°

the inverse tangent function. Thus, the frequency response is obtained from the time-domain here. To estimate the frequency response at a certain frequency ω_f, the desirable bandwidth, a test signal with frequency ω_f is fed into the system to excite transient responses. In practice, notch filters are applied to filter out specified signals other then ω_f. By comparing the input and output signals in the time domain, the frequency response of the system can be thus estimated, as shown in Fig. 2.1. Algorithm 2.1 shows how the auto estimator works in detail.

An example is shown in Fig. 2.2 with the original signals shown in the upper part and the filtered signals shown in the lower part. In the simulation, the sampling rate is 7.5(kHz) and the input signal is with frequency $\omega_f = 200$(Hz). The input signal is with amplitude 1, a random noise with uniform distribution between ± 1 and a resonance signal at 60(Hz) is added also with amplitude 1. A output signal is with amplitude 0.707, a phase shift 150°, a random noise with uniform distribution between ± 0.7, and a resonance signal at 60(Hz) with amplitude 0.7. The estimation result in the frequency domain for Fig. 2.2 are $\hat{M} = 0.697$ and $\hat{P} = 146.41°$. Average estimation results for 100 trials are listed in Table 2.1, in which sufficient accuracy can be achieved.

2.2 Optimization for Control Parameters

When the plant model changes, a re-design procedure for the controller is needed or the system stability is not guaranteed. Therefore, a controller auto-tuning algorithm is required to reduce manual controller adjustment while facing a changeable plant. This can be done by considering control parameters auto-tuning as an optimization problem, as shown in Fig. 2.3.

A typical constrained optimization problem is formed by the objective function f, a

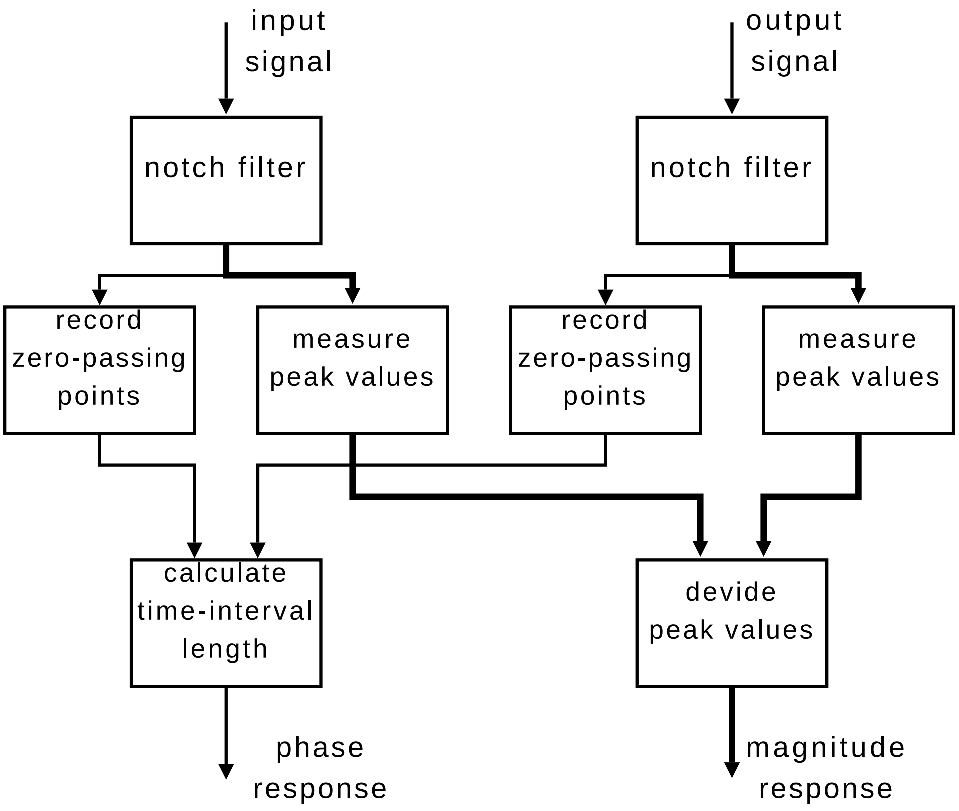


Figure 2.1: An illustration for frequency response auto estimation from both the input and the output signals.

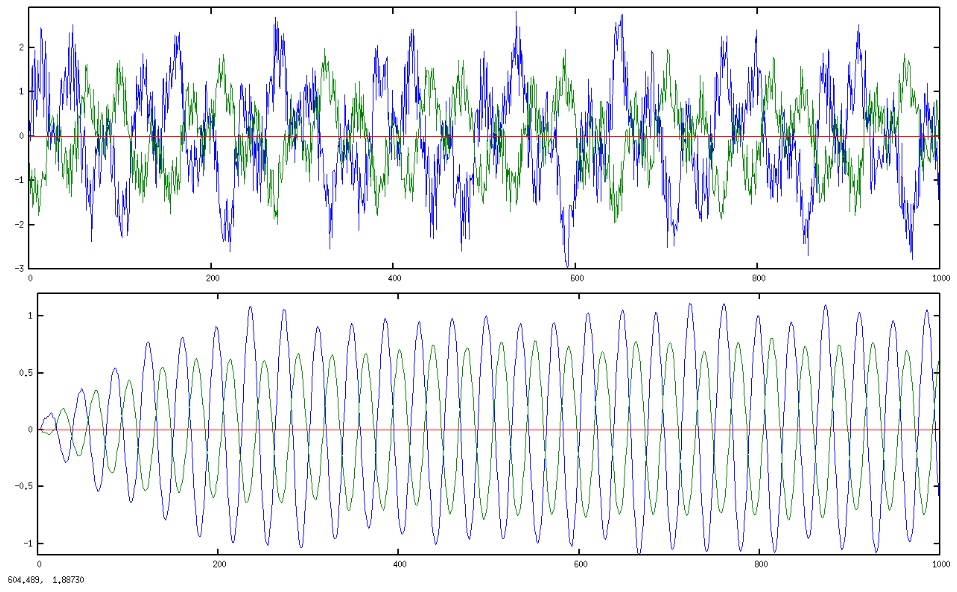


Figure 2.2: The I/O signals before and after applying the notch filter.

Algorithm 2.1 The frequency response estimation at ω_f by comparing sine waves.

1. A test signal with frequency ω_f is fed into system with two signals as input a and output b, and both signals are passed through notch filters N where

$$N(z^{-1}) = \frac{g - 2g\cos(2\pi\omega_f)z^{-1} + gz^{-2}}{1 - 2g\cos(2\pi\omega_f)z^{-1} + (2g-1)z^{-2}} \tag{2.4}$$

with parameters

$$g = \frac{1}{1+\beta},\ BW = \frac{2\pi\omega_f}{Q},\ \beta = \sqrt{\frac{1-\sigma^2}{\sigma}}\tan\frac{BW}{2}. \tag{2.5}$$

with the width factor Q and the notch gain β.

2. For signal a, measure the positive peak value $p_a^+(k)$ and zero-crossing $t_a(k)$ instance in the k_{th} period.

3. Perform the same operations on signal b.

4. The magnitude response $\hat{M}$ from a to b is estimated from calculating

$$\hat{M} = \frac{\sum\limits_i p_b^+(i)}{\sum\limits_i p_a^+(i)}. \tag{2.6}$$

5. The phase response $\hat{P}$ from a to b is estimated from calculating

$$\hat{P} = \frac{\sum\limits_i (t_a(i) - t_b(i))}{\frac{1}{2}\sum\limits_i (t_a(i+1) - t_a(i) + t_b(i+1) - t_b(i))}. \tag{2.7}$$

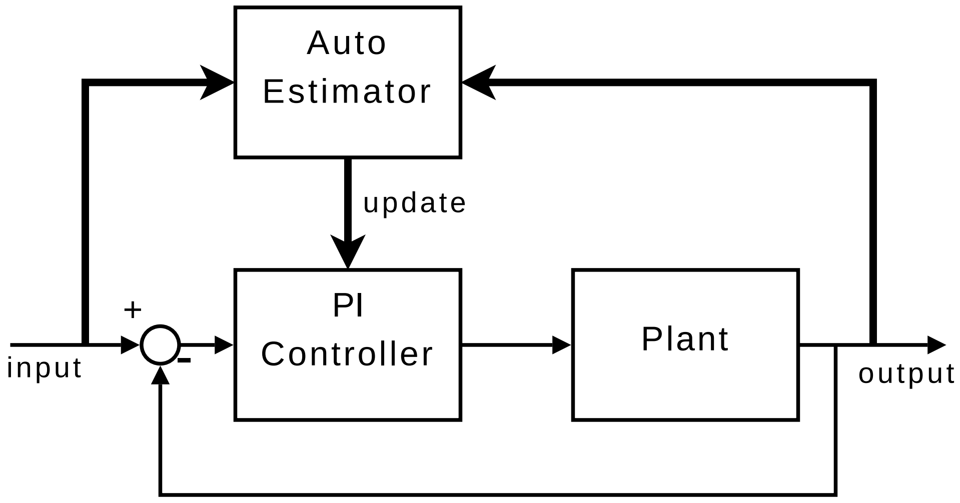


Figure 2.3: An illustration for PI control parameters auto-tuning process at frequency ω_f.

set of variable x, and the constraints S for x, and is stated as

$$\min f(x) \text{ s.t. } x \in S. \tag{2.8}$$

In this case of auto-tuning, control parameters are assigned to x, that is, for a PI controller,

$$x = \begin{bmatrix} K_p & K_i \end{bmatrix}^T. \tag{2.9}$$

The objective function f is defined from the closed-loop system frequency responses. When a frequency response estimation at ω_f mentioned in the previous section is performed, the closed-loop magnitude response M and the phase response P can be acquired and are treated as the system performance indices. The objective function f is defined as

$$f = \left\| \begin{bmatrix} M - M_0 \\ P - P_0 \end{bmatrix} \right\|^2 \tag{2.10}$$

where M_0 and P_0 are the objective closed-loop magnitude and phase response, respectively, and is a minimizer for f. Generally, M_0 is set to be -3dB so that after the optimization process converges, the system would have bandwidth ω_f. Note that the closed-loop frequency response is determined by control parameters x, therefore, M and P are also functions of x. A descent direction for f can be found by calculating the gradient value that

$$\nabla f = 2\Delta F \begin{bmatrix} \Delta M & \Delta P \end{bmatrix}^T \tag{2.11}$$

where

$$\Delta M = M - M_0,\ \Delta P = P - P_0,\ \Delta F = \begin{bmatrix} \frac{\partial M}{\partial K_p} & \frac{\partial P}{\partial K_p} \\ \frac{\partial M}{\partial K_i} & \frac{\partial P}{\partial K_i} \end{bmatrix} \tag{2.12}$$

and x is updated as

$$x(k+1) = x(k) - c\nabla f(k) \tag{2.13}$$

in each searching iteration with the step size c. Note that the initial condition $x(0)$ must stabilize the plant. The partial derivatives are approximated from the frequency responses variation near some specified work points. The constraints S for x are

$$0 < K_p < \max K_p,\ \ 0 < K_i < \max K_i \tag{2.14}$$

such that each $x(k)$ must stabilize the plant. If $x(k+1)$ exceed S, the invalid control parameter will be reset into its range, e.g., for variable K_i,

$$\begin{aligned} K_i(k) - 2c(\frac{\partial M}{\partial K_i}\Delta M + \frac{\partial P}{\partial K_i}\Delta P) < 0 \quad &\rightarrow \quad K_i(k+1) = 0, \\ K_i(k) - 2c(\frac{\partial M}{\partial K_i}\Delta M + \frac{\partial P}{\partial K_i}\Delta P) > \max K_i \quad &\rightarrow \quad K_i(k+1) = \max K_i. \end{aligned} \tag{2.15}$$

This is a so called two-degree-of-freedom PI tuning algorithm.

2.3 Simulation

Consider an identified model [18] that

$$P(z^{-1}) = 0.003864\frac{1+z^{-1}}{1-0.9989z^{-1}} \tag{2.16}$$

and a PI controller K_{pi}

$$K_{pi} = K_p + K_i\frac{1}{1-z^{-1}} \tag{2.17}$$

with the sampling time $h = 0.133$(ms). The initial control parameters are

$$K_p = 3,\ \ K_i = 0.1.$$

The frequency responses estimation is performed by using a MATLAB function '*freqresp*' with additional uncorrelated noises applied to both the magnitude and the phase responses with uniform distribution in [-0.05,0.05]. By observing the effects on M and P due to the variation of x, an approximation of the partial derivatives matrix ΔF can be found that

$$\Delta F = \begin{bmatrix} 0.0903 & 0.1238 \\ 0.03758 & -0.05258 \end{bmatrix}. \tag{2.18}$$

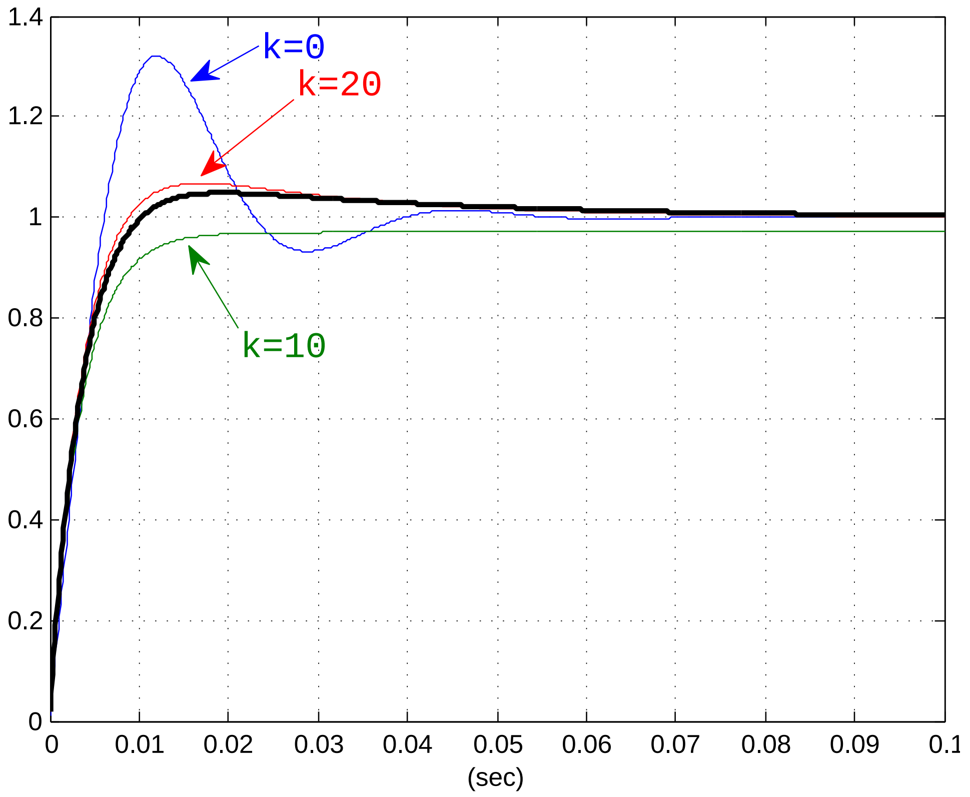


Figure 2.4: Step responses for PI controller auto-tuning with $\omega_f = 50$(Hz) at the k_{th} iteration. (Bold: turning result at $k = 100$.)

The desired frequency response is with

$$M_0 = \frac{1}{\sqrt{2}},\ P_0 = -50^\circ,$$

and a smaller P_0 results in larger overshoot. The initial step size c is 10 and it decreases as f decreases. The step responses before and after PI control parameters optimization tuned at $\omega_f = 50$(Hz) is shown in Fig. 2.4. The system step response with the tuned PI controller has nearly the same rise time but the maximum overshoot decreases from 32% to 5%. Step responses for systems tuned in different ω_f are also shown in Fig. 2.5, with similar transient responses and 5% the maximum overshoots. The tuned control parameters are listed in Table 2.2. Fig. 2.6 shows how control parameters and the frequency responses change during the optimization process. It can be found that ΔM and ΔP converge to zero after about 20 iterations, and also the proportional gain K_p. There are some variations in the trajectory of K_i caused by noises in the frequency responses estimation, but noise does not effect its estimation convergency.

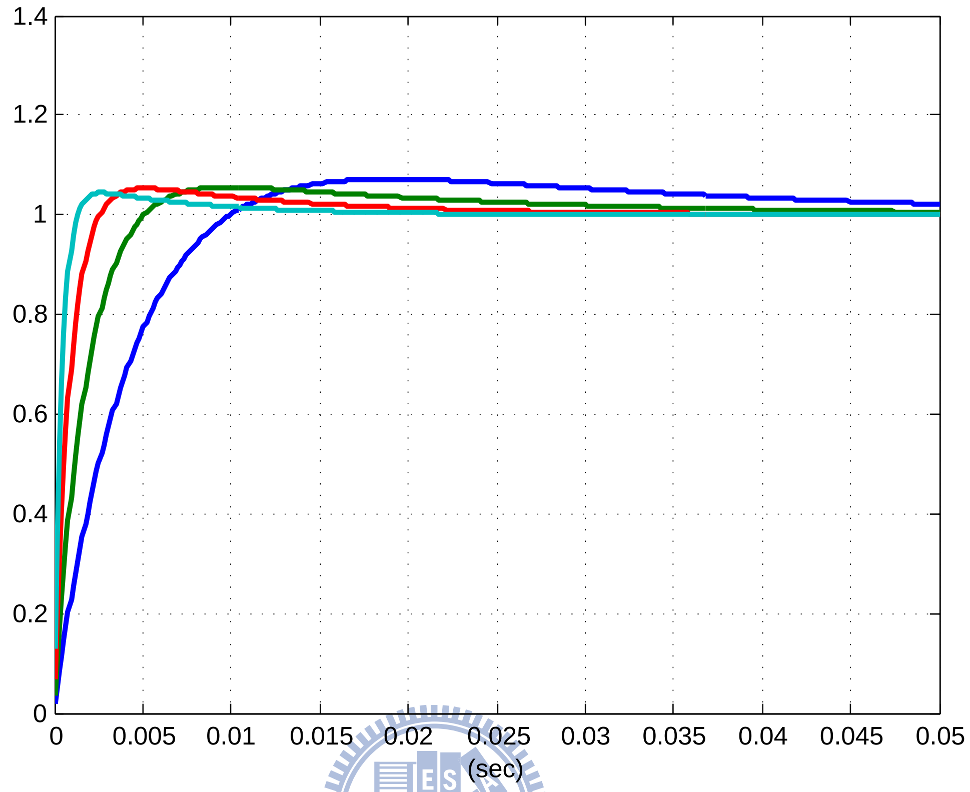


Figure 2.5: Step responses for systems with the tuned PI controllers at $\omega_f = 50$, 100, 200, and 400(Hz), respectively.

Table 2.2: Simulation results for PI auto-tuning with initial $K_p = 3$ and $K_i = 0.1$.

ω_f(Hz)	50	100	200	400
tuned K_p	4.937	9.856	18.938	38.030
tuned K_i	0.023	0.065	0.296	1.220

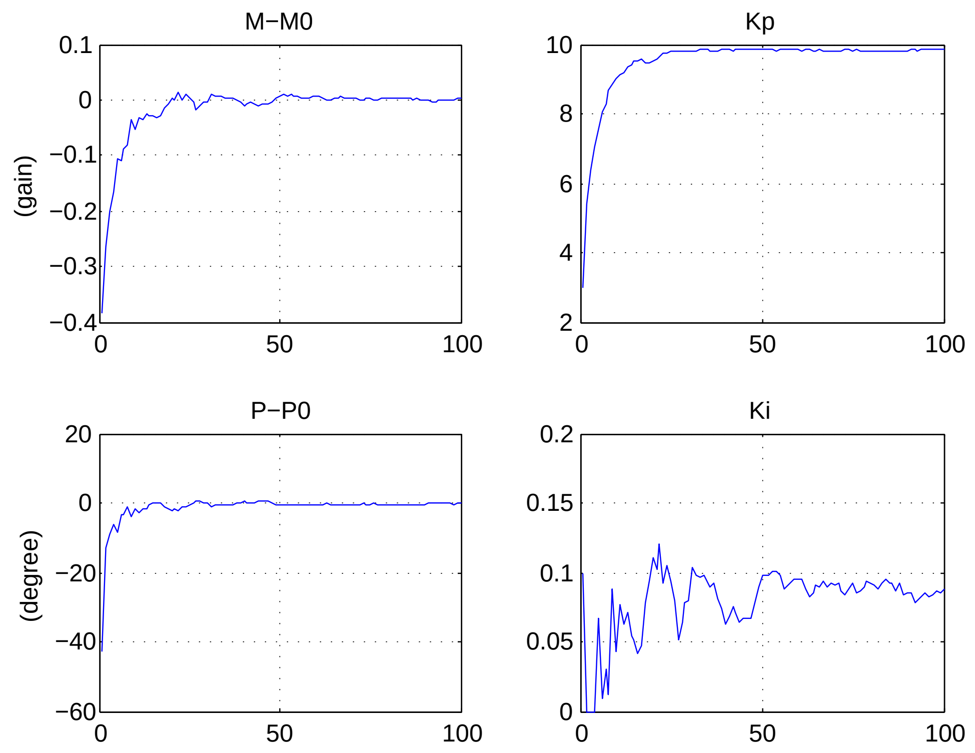


Figure 2.6: The transient values of ΔM, ΔP, K_p, and K_i during each iteration with $\omega_f = 100$(Hz).

2.4 Experiments

2.4.1 System Setup

Consider a servo motor system based on a TI TMS320F2812 DSP micro-controller [18], as shown in Fig. 2.7. Its block diagram is shown in Fig. 2.8, which contains multi-loop control structure such as the current loop, the velocity loop and the position loop. A PI controller auto-tuner as shown in Fig. 2.3 for the velocity-loop plant is implemented. The test signal adopted in the frequency responses estimation is a square wave signal, because it is easier to generate and is more steady in the network environment. Coefficients for the notch filter are chosen that $\sigma = \frac{1}{\sqrt{2}}$ and $Q = 4$. The input signal signal is a 300(rpm) speed command with a ± 30(rpm) square wave at a specified frequency about 200(Hz) and the output signal is the speed response of the closed-loop system. Both signals before and after applying the notch filter shown in Fig. (2.9). Relationships between M, P and different values of K_p and K_i measured at $\omega_f = 100$(Hz) are shown in Fig. 2.10(a) with $K_p \in [0.5, 5]$ and $K_i = 1.5$ and 2.10(b) with $K_p = 2.5$ and $K_i \in [0, 3]$. By using a

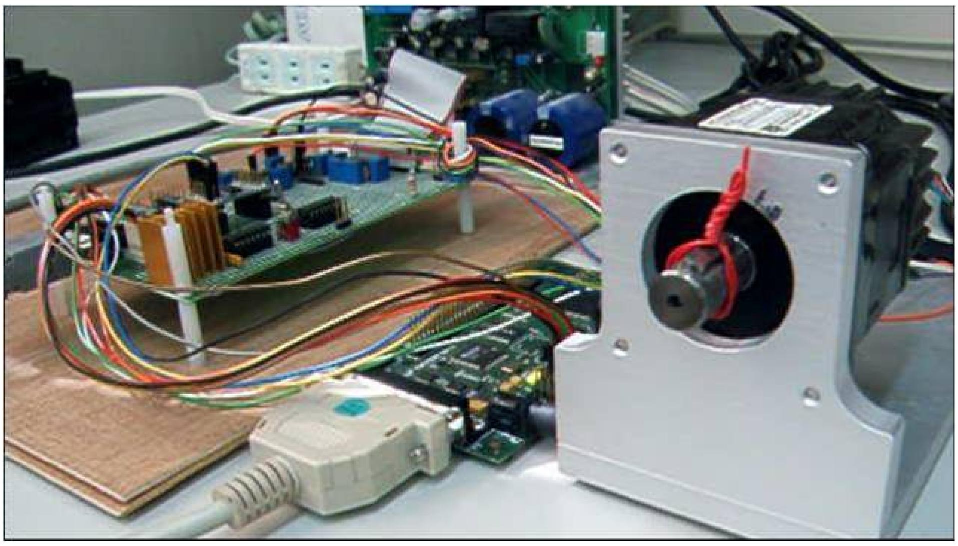


Figure 2.7: Picture of the servo motor system.

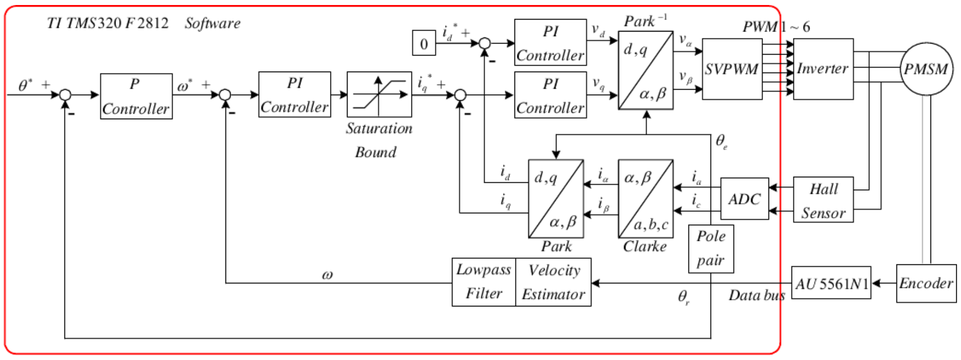


Figure 2.8: Block diagram of the servo motor system.

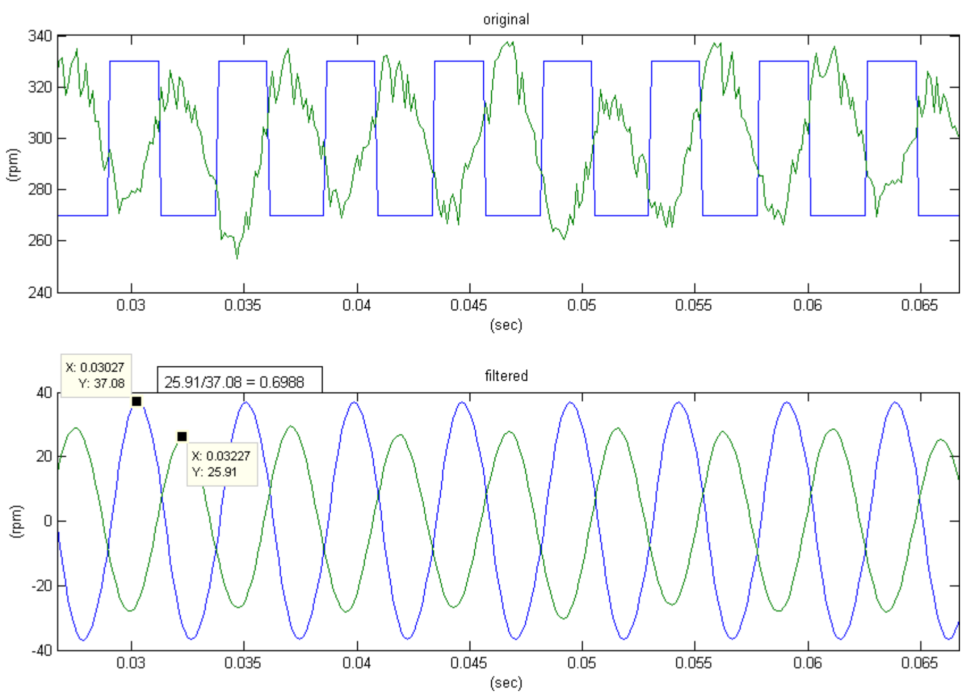


Figure 2.9: The I/O signals before and after applying the notch filter.

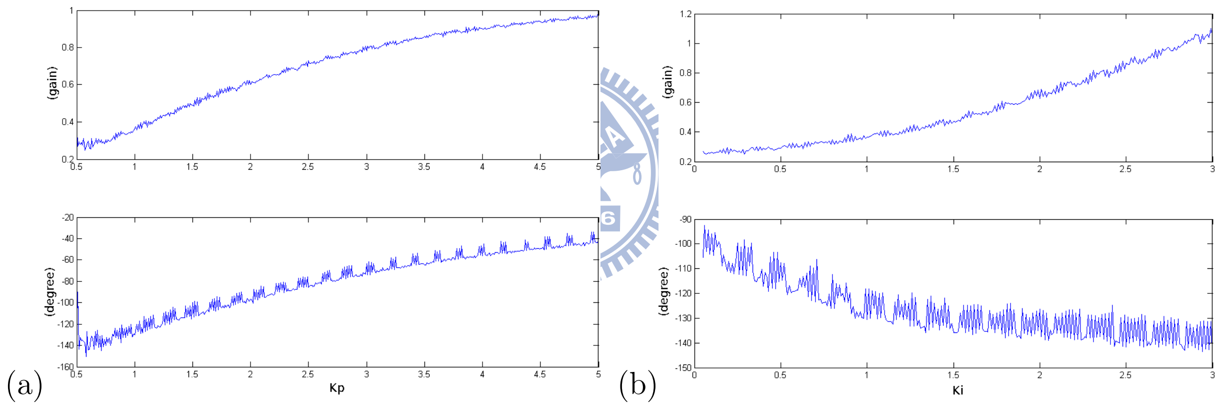


Figure 2.10: The effects on M and P from different (a)K_p and (b)K_i.

second-order polynomial approach, a numerical model for the partial derivatives can be found that

$$\Delta F = \begin{bmatrix} -0.0596Kp + 0.3258 & 0.0166Kp - 0.1064 \\ 0.166Ki + 0.2750 & -0.0306Ki + 0.0773 \end{bmatrix}. \tag{2.19}$$

The optimization process is performed with the update algorithm in Eq. (2.13).

Table 2.3: PI controller parameters auto-tuning results for different initial x.

	ω_f(Hz)	$K_p(0)$	tuned K_p	$K_i(0)$	tuned K_i
(a) $M < M_0$	200	3	2.037	3	9.089
(b) $M > M_0$	200	5	2.08	5	9.17
(c) $M < M_0$, loaded	100	2	3.96	2	12.521
(d) $M < M_0$	500	1	5.70	1	31.14
(e) $M < M_0$, repeated	500	1	5.70	1	31.42

2.4.2 Results

Because the main idea of this experiment is to verify the usability of the proposed control parameters optimization procedure, the property of convergence and repeatability are the most concerned. The optimization is performed at different target frequencies $\omega_f =$ 100, 200, and 500(Hz). The optimization process with starting $M < M_0$ and $M > M_0$ are shown in Fig. 2.11 and 2.12 with parameters listed in Table 2.3(a) and 2.3(b), respectively. Both of them converge to specified M_0 and P_0 with similar control parameters K_p and K_i. A converging rate test with different step size c is shown in Fig. 2.13. It shows that the optimization process converges faster with a larger c, but it also leads to larger oscillation of tuned control parameters. Too large step size will lead to divergence of the optimization process. A test with servo system with 5 times load of inertia and $\omega_f = 100$(Hz) is shown in Fig. 2.14 with parameters in Table 2.3(c). Although actual plant model is changed, the optimization process still converges. A test for the repeatability is shown in Fig. 2.15 with the initial $M < M_0$ at a relatively high frequency $\omega_f = 500$(Hz) and the process is repeated again, as shown in Fig. 2.16. Their tuning results are shown in Table 2.3(d) and (e), respectively, and are with similar control parameters. From the experiment results, the proposed optimization procedure presents reliable repeatability and convergency.

2.5 Auto-Tuning for the Network Delay

When the network induced delay occurs, the phase margin of the system decreases and the system stability degrades. The PI control parameters optimization process can be then

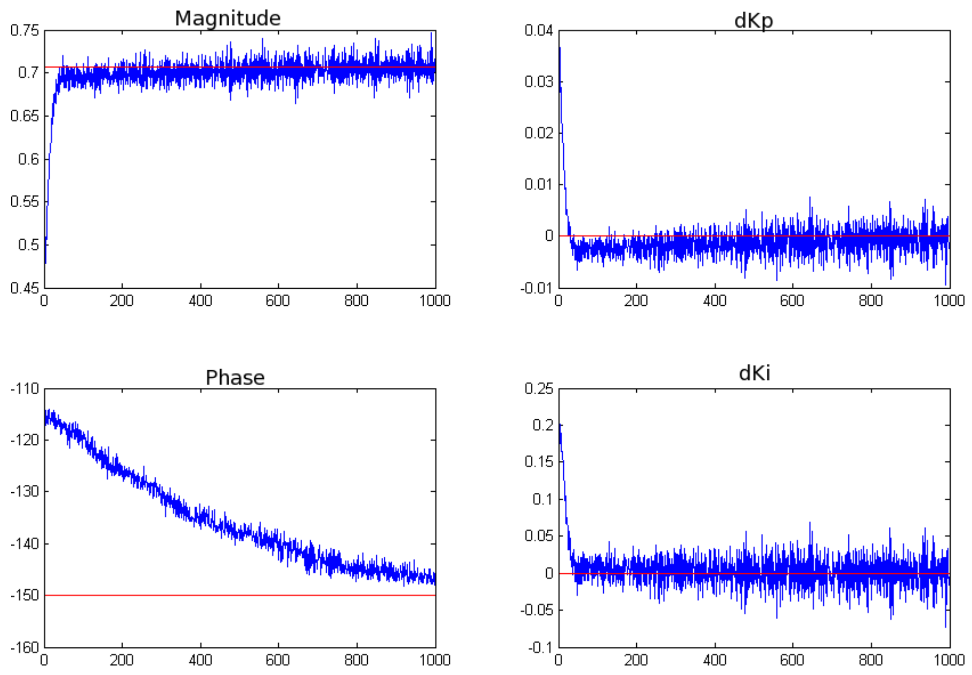


Figure 2.11: Optimization starts with $M < M_0$ at $\omega_f = 200$(Hz).

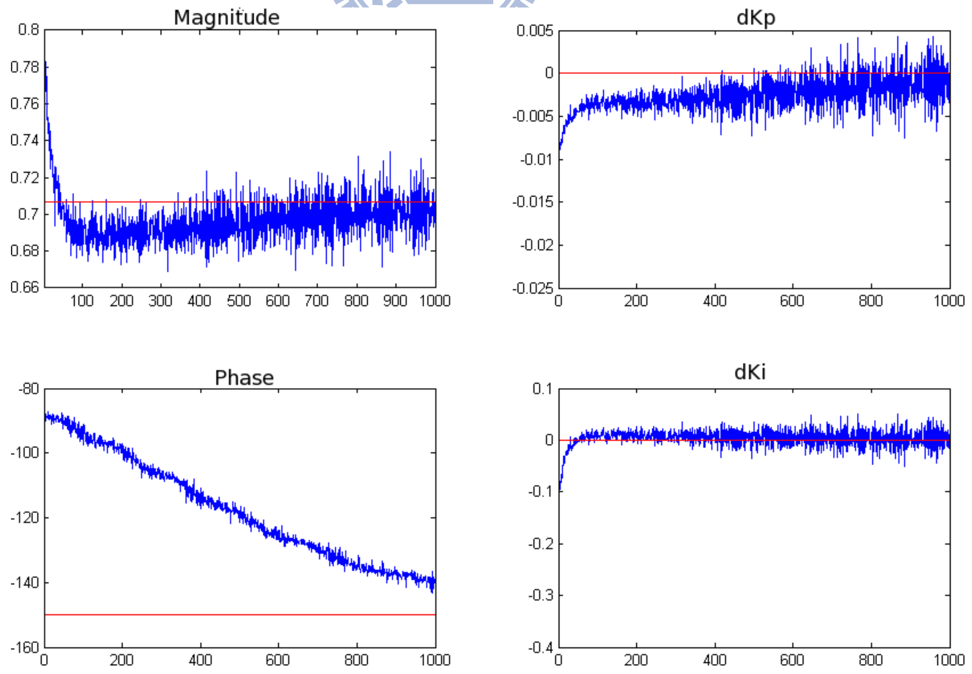


Figure 2.12: Optimization starts with $M > M_0$ at $\omega_f = 200$(Hz).

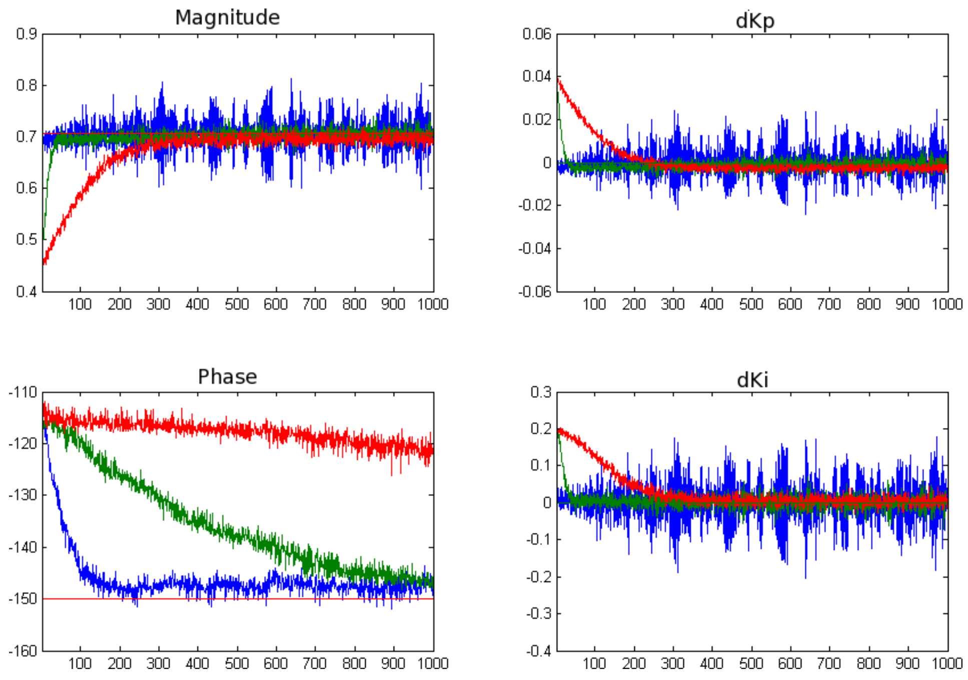


Figure 2.13: Optimization starts with $M < M_0$ with different step size $c = 0.0625$, $c = 0.5$ and $c = 4$ at $\omega_f = 200$(Hz).

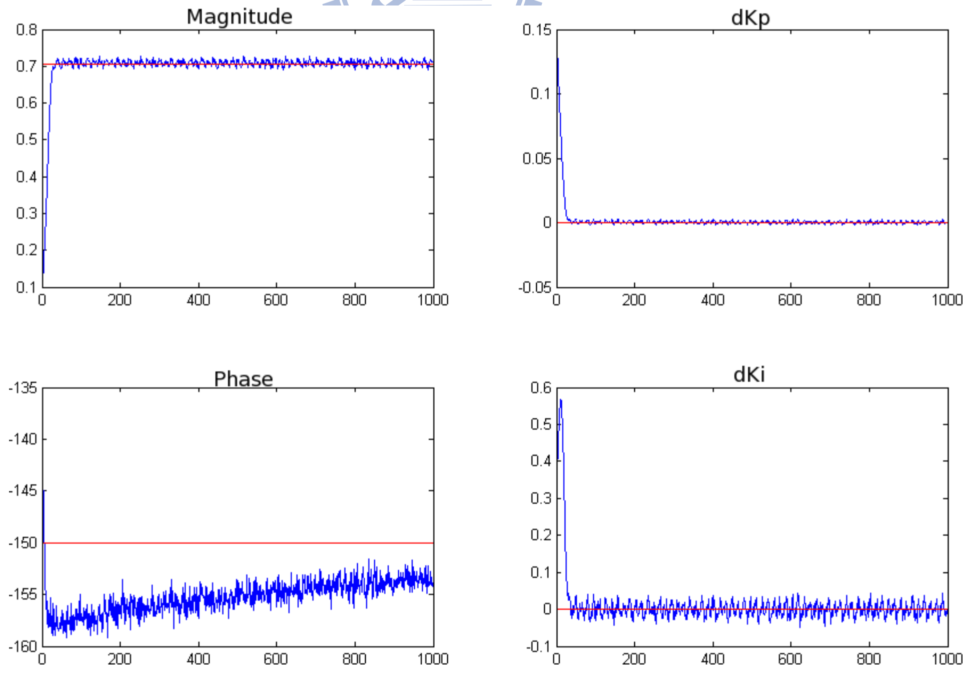


Figure 2.14: Optimization starts with $M < M_0$ and with 5 times extra loading at $\omega_f = 100$(Hz).

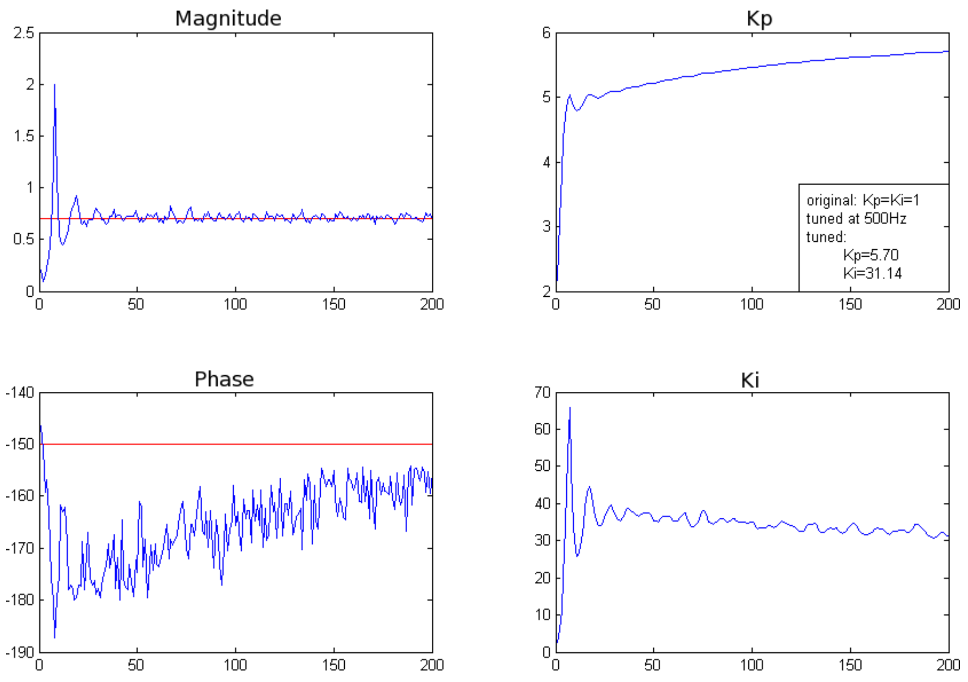


Figure 2.15: Optimization starts with $M < M_0$ at $\omega_f = 500(\mathrm{Hz})$.

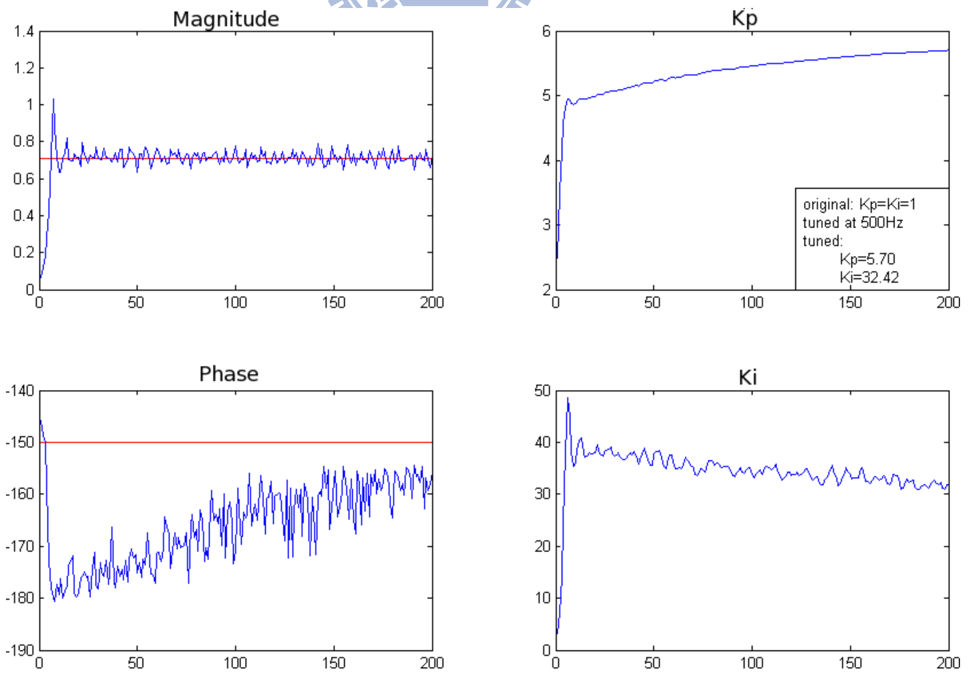


Figure 2.16: Optimization starts with $M < M_0$ at $\omega_f = 500(\mathrm{Hz})$, repeated.

applied to re-tune the NCS controller under the assumption of a constant network delay. Fig. 2.17 shows the step responses before and after auto-tuning for system in (2.16) with different network induced delay time $z^{-\tau}$ at ω_f =100(Hz). The original PI controller is tuned without the network delay as shown in Fig. 2.4 with $\tau = 0$, and clearly the network delay degrades the system performance. Therefore, the re-tuned controllers lead to better performances with the smaller maximum overshoots, and their control parameters are listed in Table 2.4. When the delay time τ is small, the optimization process works as well. For longer delay time, however, the integral gain K_i drops to zero during the optimization process, and there are steady-state errors in the closed-loop step responses. The reason is that for a typical PI controller K_{pi}, the phase response is

$$\angle K_{pi} = \begin{cases} -90^\circ, & \text{as } \omega \to 0, \\ 0^\circ, & \text{as } \omega \to \infty \end{cases} \tag{2.20}$$

and it will decrease the value of open-loop phase responses in nearly all frequency. Therefore, the effects of the integral gain K_i actually decreases the phase margin, and make the system more unstable under the networked situation. Its benefit for eliminating the steady-state error is not considered in the objective function f in Eq. (2.10). The system controlled with only the P controller is more stable than the original PI controller, and the steady-state error problem could be solved by applying a forward gain with value of 1.0142. It should be noted that when the delay time τ is too large, e.g., $\tau = 20$ as shown in Fig. 2.18, the minimal value of the objection function f will not be zero. The tuned PI control parameters are

$$K_p = 4.972,\ K_i = 0$$

and their trajectory are shown in Fig. 2.19, with the final values that

$$\Delta M = 0.576,\ \Delta P = -24.23^\circ$$

and $f = 0.512$.

2.6 Summary

In this chapter, the following results can be summarized as:

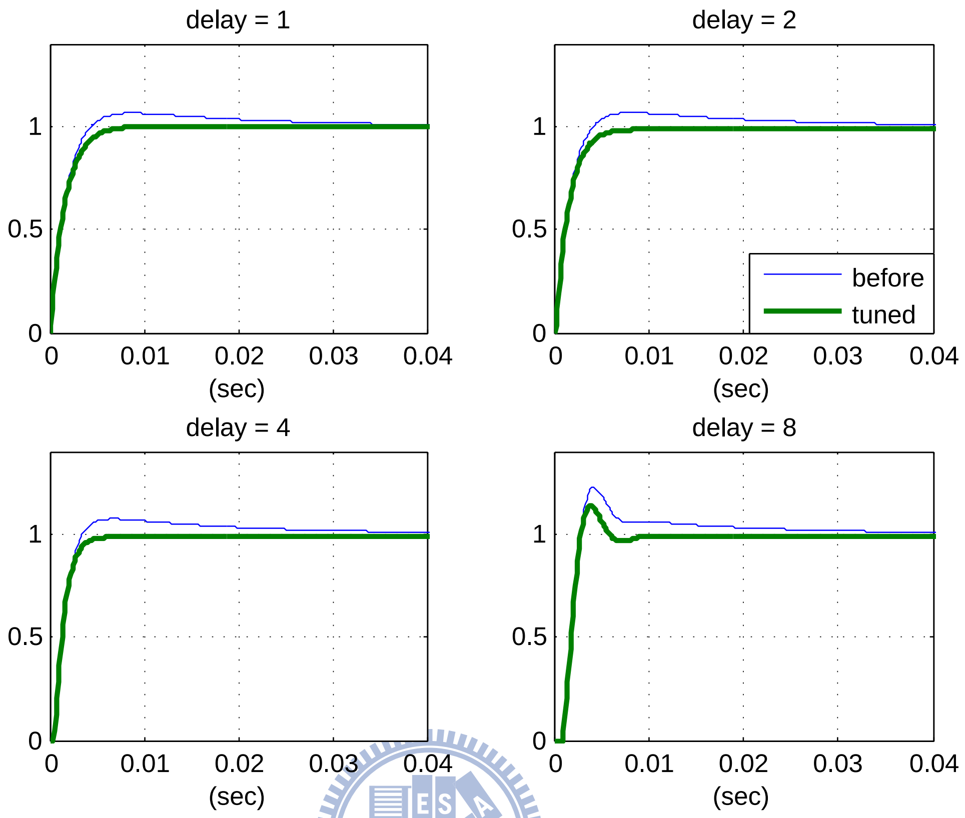


Figure 2.17: Step responses after auto-tuning at $\omega_f = 100$(Hz) for different delay time $z^{-\tau}$ with τ =1, 2, 4, and 8. The sampling time $h = 0.133$(ms).

Table 2.4: Results of control parameters auto-tuning with extra delay $z^{-\tau}$ at ω_f =100(Hz).

$\backslash\tau$	0	1	2	4	8
tuned K_p	9.880	9.963	10.174	10.476	10.332
tuned K_i	0.0809	0.0112	0	0	0
DC gain	1	1	0.986	0.986	0.986

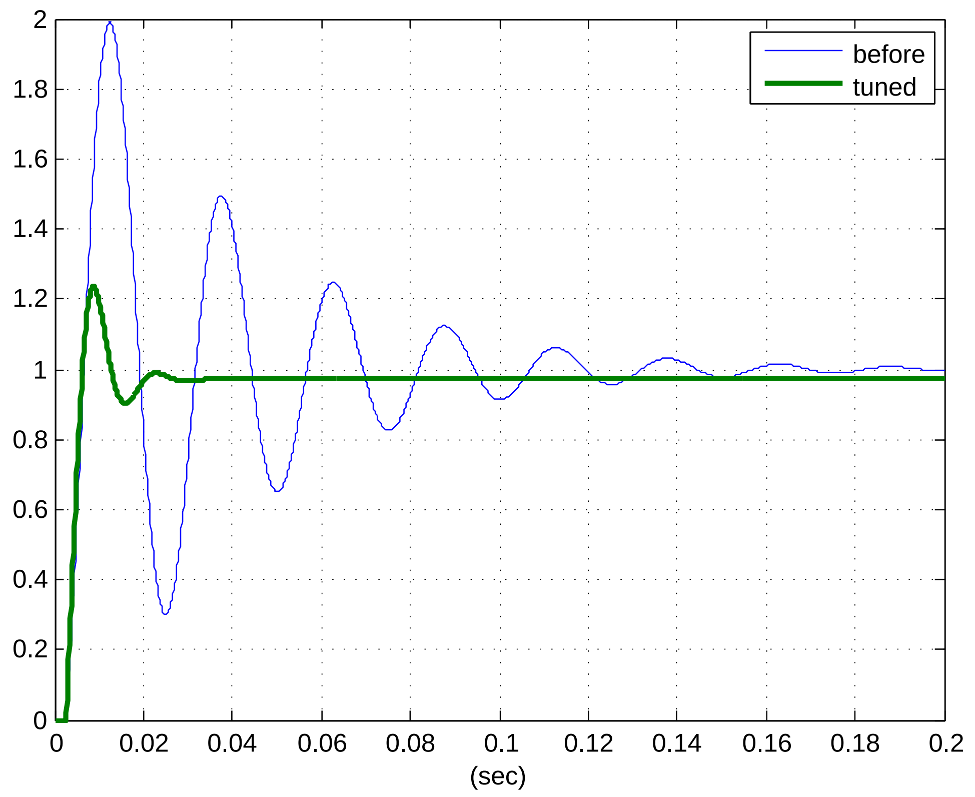


Figure 2.18: Step responses before and after auto-tuning at $\omega_f = 50$(Hz) for the system with z^{-20}.

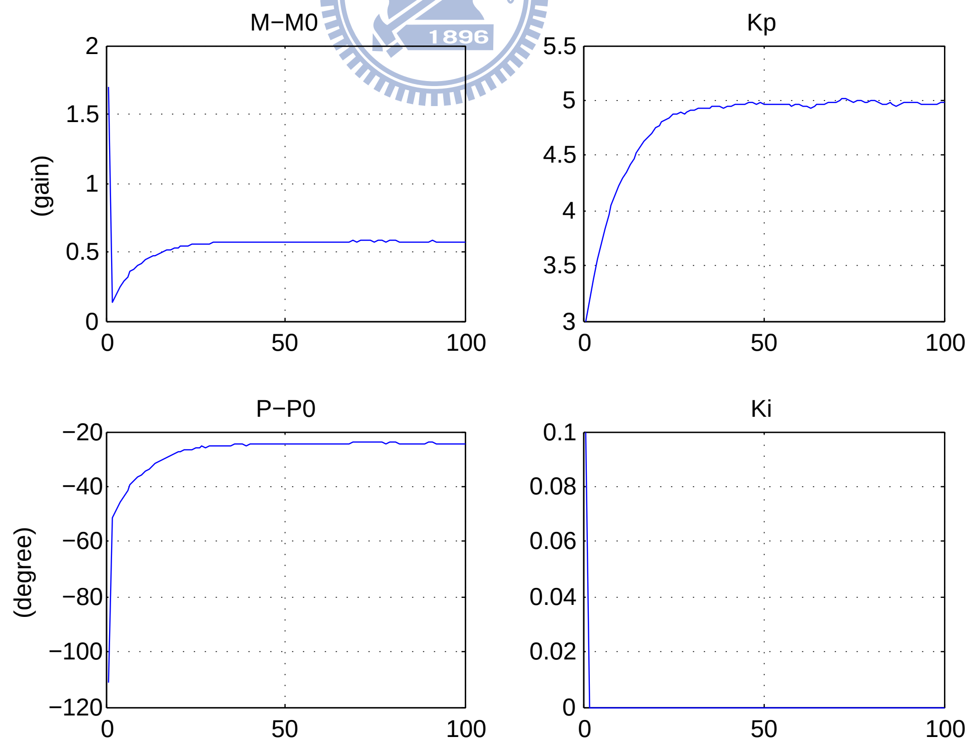


Figure 2.19: The transient values of ΔM, ΔP, K_p, and K_i during the tuning process for Fig. 2.18.

1. An algorithm for the frequency responses estimation in the time-domain is proposed. This method can be easily implemented in a DSP micro-processor and is with sufficient accuracy for further optimization process.

2. A general PI control parameters optimization procedure is described and is verified in both simulation and experiment to prove its usability and repeatability. With this algorithm, the control performance in bandwidth can be maintained with model variation caused by the network.

Chapter 3 H_∞ Design for the Network Delay Systems

3.1 Theoretical Structure

A general framework for H_∞ design is the linear fractional transformation (LFT), as shown in Fig. 3.1. Define the following notations as

w	:	external input, such as $\begin{bmatrix} d & n & r \end{bmatrix}^T$
y	:	controller input
z	:	weighted output
u	:	controller output

and denote T_{ab} as the transfer function from a to b, then the following relationship holds:

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} T_{zw} & T_{zu} \\ T_{yw} & T_{yu} \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix} \tag{3.1}$$

One objective of the H_∞ algorithm is to minimize the value of $\|T_{zw}\|_\infty$ where

$$\|T_{zw}\|_\infty = \sup \max_{\omega} \sigma(T_{zw}(\omega)), \;\; \omega \in \mathbb{R}. \tag{3.2}$$

Let $\|w\|_2^2$ and $\|z\|_2^2$ represent energies of input and output signal, respectively, then

$$\|z\|_2^2 \leq \|T_{zw}\|_\infty \|w\|_2^2 . \tag{3.3}$$

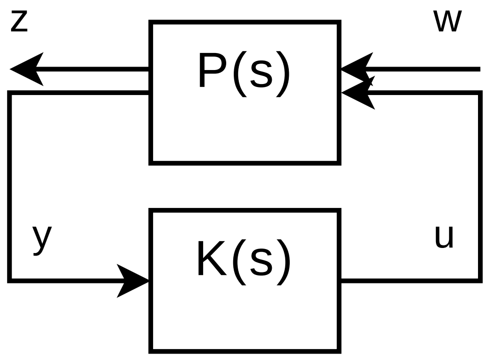


Figure 3.1: General linear fractional transformation(LFT) framework.

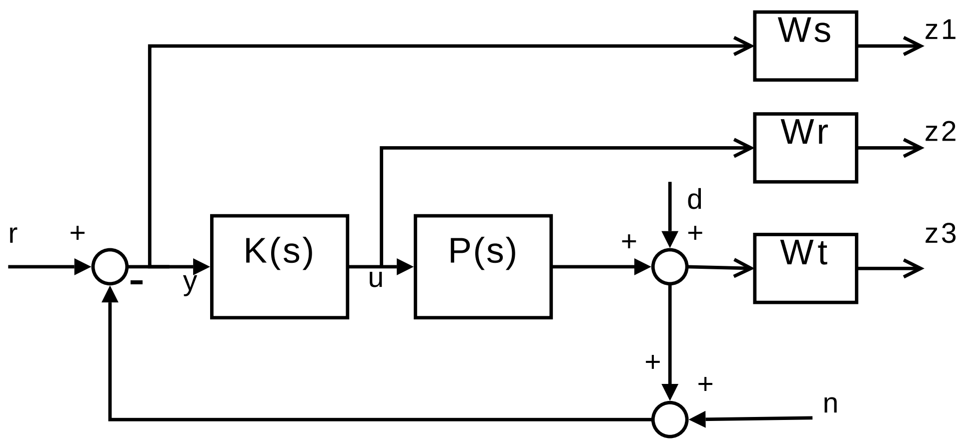


Figure 3.2: Mixed synthesis framework for H_∞ design.

It means that the maximum energy transfer ratio for the system is bounded, and an upper bound γ is

$$\|T_{zw}\|_\infty < \gamma. \tag{3.4}$$

When γ is smaller, variations caused by the input w to the output z are smaller.

There are several general frameworks for H_∞ NCS controller design, as discussed in [11]. However, they are also too general that it is hard to apply time-domain or frequency-domain performance requirements to them. To achieve specified system performances, weighting functions would be applied to LFT framework to shape frequency responses within a particular frequency band. Rules for choosing weighting functions are different because of various purposes. A frequently used structure is the mixed sensitivity design framework [19], as shown in Fig. 3.2.

3.2 Weighting Functions

The design purpose in the mixed synthesis framework is to satisfy

$$\left\| \begin{bmatrix} W_s S & W_r R & W_t T \end{bmatrix} \right\|_\infty < \gamma \tag{3.5}$$

where S is the sensitivity function, R is the control sensitivity function, and T is the complementary sensitivity function where

$$S = \frac{1}{1+KP}, \; R = \frac{K}{1+KP}, \; T = \frac{KP}{1+KP} \tag{3.6}$$

with weighting functions W_s, W_r, and W_t, respectively. Because Eq. (3.5) is not for explicit design purposes, a set of inequalities with similar effects are chosen to approach it where

$$\|S\|_\infty < \|W_s\|_\infty^{-1}, \ \|R\|_\infty < \|W_r\|_\infty^{-1}, \ \|T\|_\infty < \|W_t\|_\infty^{-1} \tag{3.7}$$

The selection of weighting functions in Eq. (3.7) is to specify the shape of each sensitivity function in the frequency domain. When a generated controller tends to satisfy Eq. (3.7), the γ value in Eq. (3.5) will have smaller value. Generally, there are no straight forward rules in mixed synthesis that can generate a perfect controller at the first try, and weighting functions need to be adjusted to meet performance requirements if there is needed. For each sensitivity function, the requirement is different, and its weighting function is designed separately with a template function. Practically, some low-pass or high-pass filters are applied to the template function to further shape its sensitivity functions.

3.2.1 Template for W_s

A requirement for the sensitivity function S is to have lower magnitude response in low frequency parts, which means the system is designed to be insensitive for external variations. It can be written as in the following statement that

$$\|S\| \text{ is } \begin{cases} < \varepsilon_s, & \text{when } \omega \text{ is small,} \\ \to 1, & \text{when } \omega \text{ is large,} \end{cases}$$

where ε_s is an arbitrarily small value. An usual template for W_s is

$$W_s = \frac{s/M_s + \omega_s}{s + \omega_s \varepsilon_s}, \tag{3.8}$$

as shown in Fig. 3.3. The value of ω_s is selected for the low-pass frequency range, and M_s is selected for proper peak value of S. The upper bound of sensitivity function is determined by ε_s.

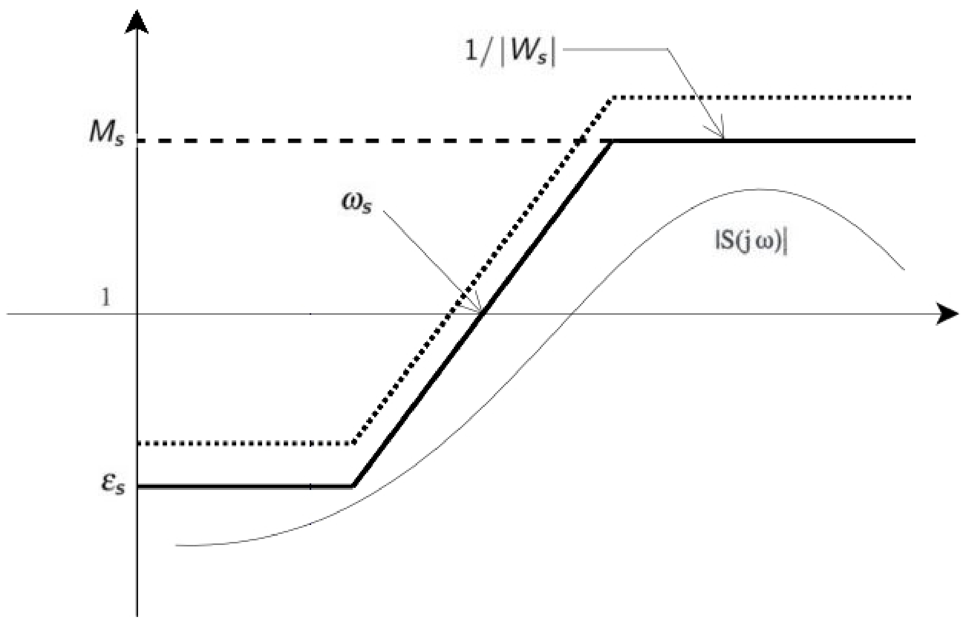


Figure 3.3: Template for the weighting function W_s.

3.2.2 Template for W_r

A requirement for the control sensitivity function R is to reduce effects from the high-frequency sensor noise n to the controller output u as the the noise rejection. A common template function for W_r is expressed as

$$W_r = \frac{s + \omega_r/M_r}{\varepsilon_r s + \omega_r} \tag{3.9}$$

where the value of ω_r decides the range of working frequency, and M_r decides the upper bound of magnitude response, as shown in Fig. 3.4.

3.2.3 Template for W_t

A template for the complementary sensitivity function T can be found from the small gain theorem. When a system has a multiply uncertainty, for a nominal system P_0, the actual system acts like

$$P = P_0(I + \Delta P), \tag{3.10}$$

a criterion for the system to achieve internally stability is

$$\left\|P_0K(I + P_0K)^{-1}\right\|_\infty \|\Delta P\|_\infty \equiv \|T\|_\infty \|\Delta P\|_\infty < 1. \tag{3.11}$$

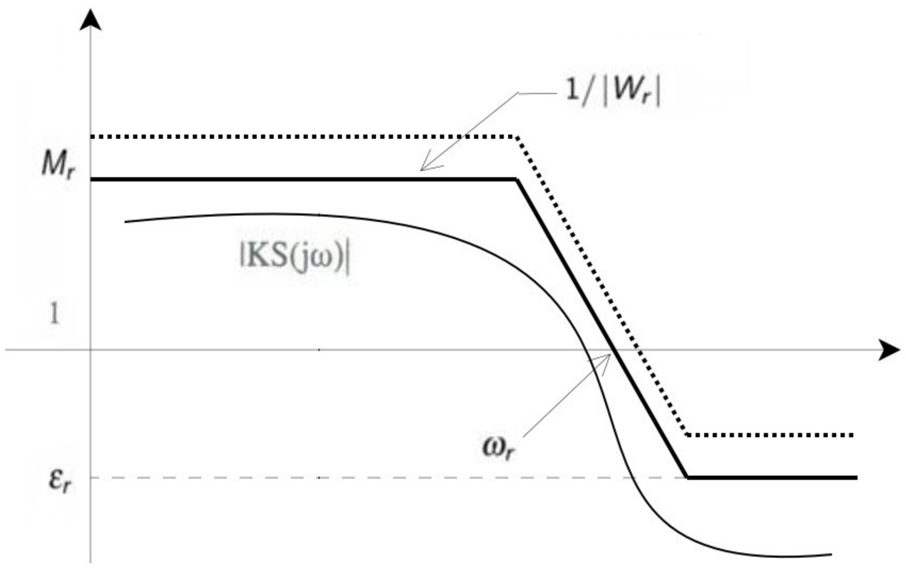


Figure 3.4: Template for the weighting function W_r.

Consider system uncertainty mainly in the network delay as

$$P = P_0 e^{-j\omega\theta} \tag{3.12}$$

where θ is the delay time. Therefore,

$$\Delta P(\theta) = \frac{P}{P_0} - 1 = e^{-j\omega\theta} - 1 \tag{3.13}$$

Let $\|W_t\|_\infty \geq \|\Delta P\|_\infty$, and

$$\|T\|_\infty < \|W_t\|_\infty^{-1} \leq \|\Delta P\|_\infty^{-1}, \tag{3.14}$$

system stability is thus guaranteed. From Eq. (3.13),

$$\lim_{\omega\to 0} \|\Delta P\| \to \omega\theta, \ \max\left(\lim_{\omega\to\infty} \|\Delta P\|\right) = 2. \tag{3.15}$$

A first-order realization for Eq. (3.15) is

$$W_t(\theta) = \frac{2\theta s}{\theta s + 2}. \tag{3.16}$$

From Eq. (3.16), it has the following property that

$$\theta_1 > \theta_2 \to \|W_t(\theta_1)\| > \|W_t(\theta_2)\| \tag{3.17}$$

as shown in Fig. 3.5. Therefore, when choosing θ for W_t, the maximum delay time $\theta_{\max}$ is chosen that

$$W_t = W_t(\theta_{\max}). \tag{3.18}$$

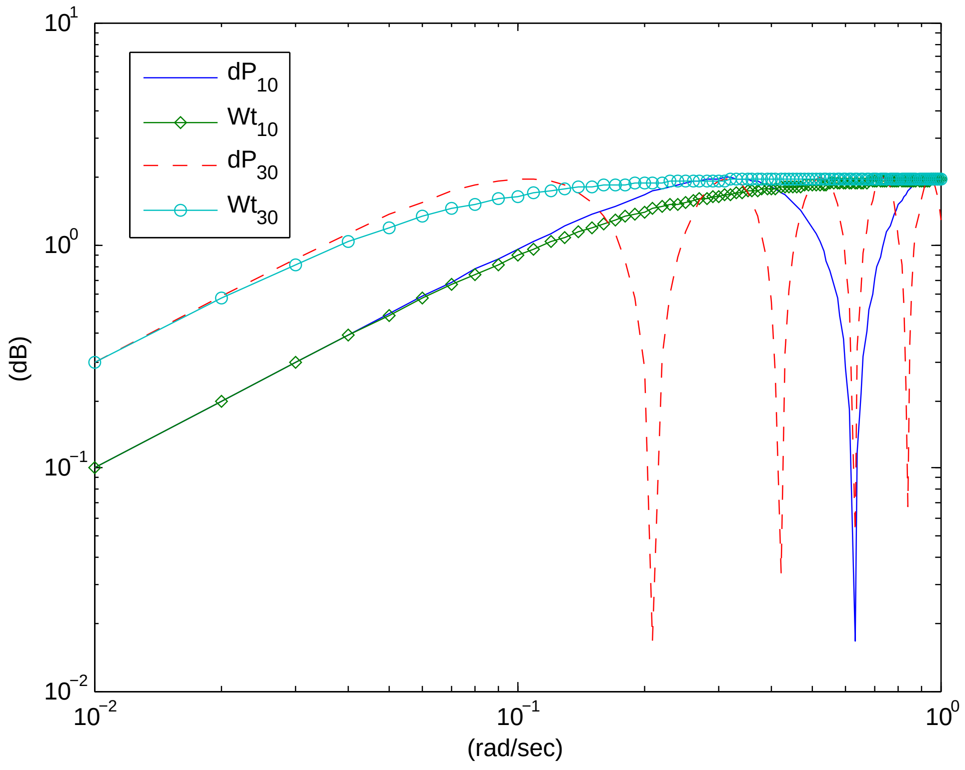


Figure 3.5: Frequency response for W_t and ΔP with $\theta = 10$ and $\theta = 30$.

3.3 H_∞ NCS Design Procedures

3.3.1 Performance Target

A design flow for H_∞ NCS controller is applied to a servo motor system [17] with an identified plant P that

$$P(s) = V(s) \times \frac{1}{s} = \frac{180.1s + 10000}{s^2 + 190s + 1000} \times \frac{1}{s} \tag{3.19}$$

where $V(s)$ is the build-in velocity-loop plant. The actuator generates output voltage calculated from the controller output to the motor and the rotor position is fed back from sensor. When the measurement of the rotor position is transmitted through network to a remote controller, an equivalent delay is determined by network transmission. Before H_∞ controller is designed, since delay decreases the phase margin and causes instability, a lead controller is used to restore phase margin. Let a lead controller be designed as

$$K_{\text{lead}} = 6.442\frac{s + 7.7495}{s + 18.36244} \tag{3.20}$$

that it satisfies system performance specifications listed in Table 3.1. An H_∞ controller is designed to further improve performance of the lead controller K_{lead} by specifying performance specifications.

3.3.2 Design Weighting Functions

Specification of this example is provided in Table 3.1. From Eq. (3.8), since the larger peak value of $\|S\|$ easily causes ripples, the maximum value is set as $M_s = 1$. According to the requirement that the bandwidth must be larger than 2.4(Hz), ω_s is set to be 2.32 with $\varepsilon_s = 10^{-6}$ so that the sensitivity function approaches zero at a low frequency. The calculated W_s becomes

$$W_s = \frac{s + 2.32}{s + 2.32 \times 10^{-6}}. \tag{3.21}$$

According to the requirement of disturbance rejection and Eq. (3.9), let $\omega_r = 72$ and assign ε_s an arbitrary small number with $M_r = 1$. Besides, a low-pass filter is cascaded to template W_r because there is an integrator already in $P(s)$. Thus, the weighting function W_r becomes

$$W_r = \frac{s + 72}{1e - 3s + 72} \times \frac{s/10 + 1}{s + 0.01}. \tag{3.22}$$

From Eq. (3.16), let $\theta_{\max} = 0.6 \times \frac{180}{\pi}$ and

$$W_t = \left.\frac{2\theta_{\max} s}{\theta_{\max} s + 2}\right|_{\theta_{\max} = 0.6 \times \frac{180}{\pi}} = \frac{1.2 \times 180s}{0.6 \times 180s + 2\pi}. \tag{3.23}$$

Gather the designed weighting functions and apply to MATLAB function 'mixsyn' with plant transfer function P, a H_∞ controller is generated that

$$K_{\text{hinf}} = \frac{7.727\text{e} - 4s^5 + 5564s^4 + 3.332\text{e}6s^3 + 6.39\text{e}5s^2 + 6063s + 0.06057}{s^6 + 781.9s^5 + 1.238\text{e}5s^4 + 8.632\text{e}6s^3 + 3.994\text{e}6s^2 + 3.916\text{e}4s + 0.09083} \tag{3.24}$$

with $\gamma = 1.956$. System performances are listed in Table 3.1. It can be found that the system with the controller K_{hinf} fits the bandwidth specification, but delay tolerance and disturbance rejection requirements are unsatisfied. This means weighting functions need to be further adjusted.

3.3.3 Adjust H_∞ Algorithm Property

Generally when the stability of a NCS is considered, a control strategy is to maintain a proper phase margin for the closed-loop system since the delay time may vary in a NCS.

Table 3.1: Specification for system performances.

	K_{lead}	desired	K_{hinf}
freq. crossover(rad/sec)	2.87		12.5
phase margin	101.3°		80.2°
max T_d(sec)	0.616	0.6	0.11
bandwidth(rad/sec)	2.39	>2.4	15.6
$\|KS(j100)\|$(dB)	16.28	<0	19.1

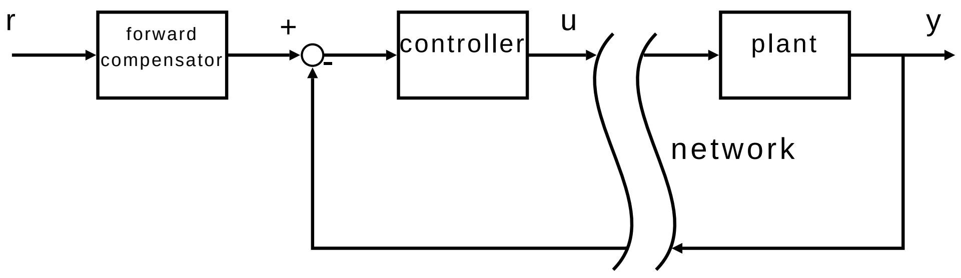


Figure 3.6: New NCS control structure with a forward compensator.

However, due to the fact that the phase lagged caused by network delay as

$$\angle e^{-j\omega\theta} = -j\omega\theta, \tag{3.25}$$

the delay-induced phase lag increases as frequency increases. To maintain the same phase margin, it needs a lead compensation in the higher frequency range. Controllers designed by H_∞ algorithm only considers magnitude response and it actually decreases open-loop gain to maintain system stability. When open-loop gain within some frequency band is lower than one, the input sinusoidal signals within this frequency band can tolerate arbitrarily large network delay without causing system instability in closed-loop system responses.

However, reducing open-loop gain slows system response and there is a trade-off between delay tolerance and bandwidth, even with the H_∞ algorithm. To conquer this problem, a forward compensator is applied before system input node, as shown in Fig. 3.6. Therefore, H_∞ algorithm is used to satisfy the delay tolerance requirement, and a forward compensator is designed to improve system performance.

Firstly, adjust weighting functions to meet stability requirement. Let $\omega_s = 0.1$ and

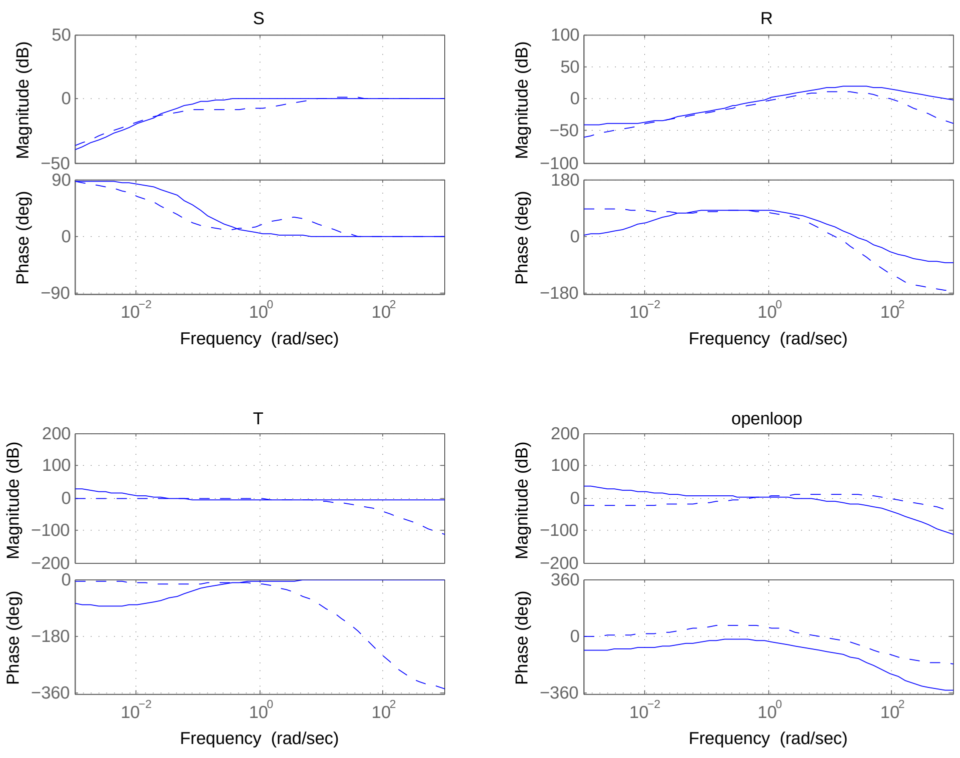


Figure 3.7: Sensitivity functions and weighting functions. (Solid: weighting functions; dotted: sensitivity functions.)

$\theta_{\max} = 36.4$, and a new controller is generated that

$$K_{\text{hinf\_mod}} = \frac{\sum a_i s^i}{\sum b_i s^i} \tag{3.26}$$

with $\gamma = 1.4527$ and

$$\begin{aligned} a_i &= \begin{bmatrix} 0.0015 & 1.082e4 & 2.056e6 & 1.083e8 & 7.026e6 & 5.949e4 & 0.5942 \end{bmatrix}, \\ b_i &= \begin{bmatrix} 1 & 299.6 & 3.409e6 & 1.747e6 & 3.37e7 & 6.505e7 & 8.373e5 & 0.08372 \end{bmatrix}. \end{aligned} \tag{3.27}$$

System performance indices are listed in Table 3.2. It can be found that delay tolerance is enlarged with the trade-off as a low bandwidth. Frequency responses for sensitivity and weighting functions are shown in Fig. 3.7.

3.3.4 Order Reduction for the Controller

The order of controller designed by applying the H_∞ algorithm equals to the summation of orders of plant P and all weighting functions W_s, W_r and W_t, and is generally much

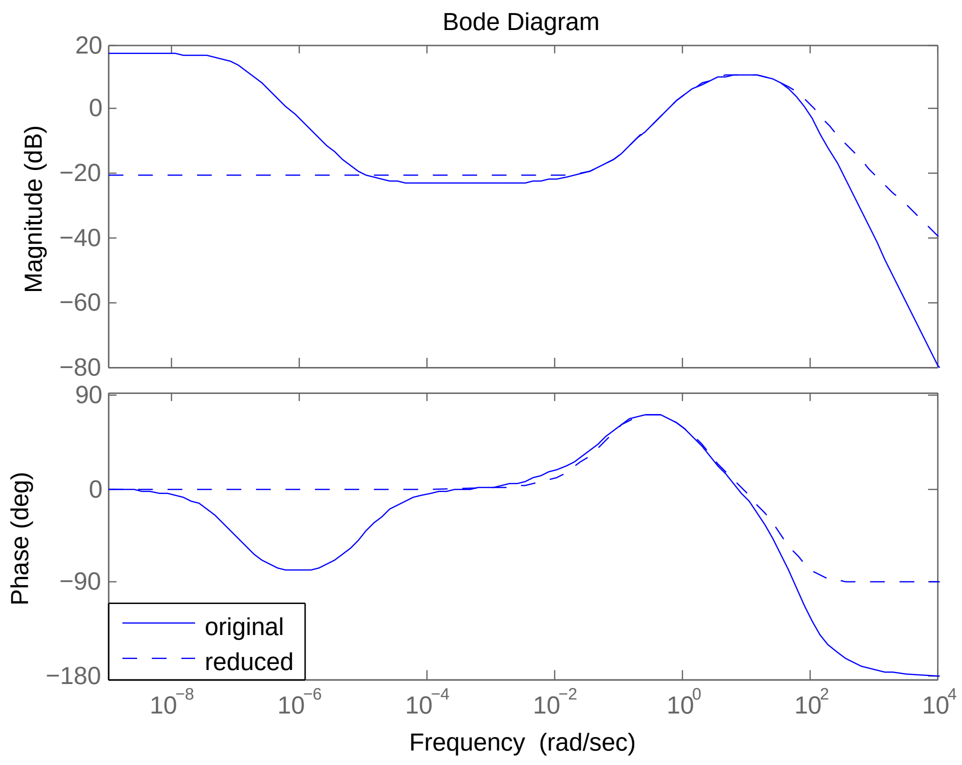


Figure 3.8: Frequency responses before and after controller order reduction. (Solid: before reduction; dotted: after reduction.)

higher than it requires. By observing magnitude in the frequency response, it can be found that not all poles/zeros play major on control performance or system stability. Therefore, other reduction methods can be also applied, such as pole-zero cancellation. From the Bode plot of the controller, there are two pairs of nearby pole and zero in high frequency($> 10^2$(rad/sec)) and low frequency($< 10^{-4}$(rad/sec)) that they have no significant effects on system performance and stability. Performing pole zero cancellation to these two pairs of pole/zero can reduce the controller order without degrading system responses. After pole-zero cancellation, a fourth-order controller is acquired

$$K_{\text{hinf\_loop}} = \frac{104.9764(s+0.05495)(s^2+190s+10000)}{(s+40.84)(s+2.147)(s^2+153.3s+7128)} \tag{3.28}$$

The frequency responses for controller and after order reduction is shown in Fig. 3.8. Their similarity within specified frequency bands can be found.

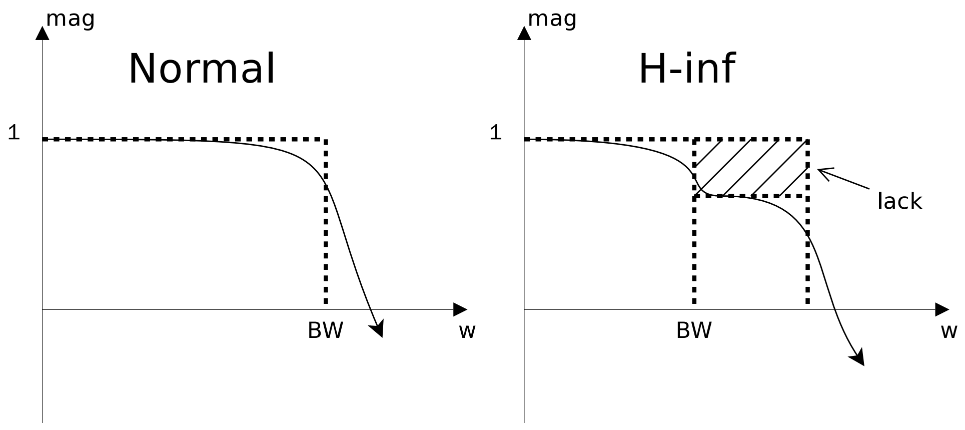


Figure 3.9: Illustration for H_∞ NCS controller design.

Table 3.2: Performance comparison.

	desired	without pre-filter	with pre-filter
freq. crossover(rad/sec)		2.74	2.74
phase margin		121°	121°
max T_d(sec)	0.6	0.772	0.772
bandwidth(rad/sec)	>2.4	0.07	6.42
$\lvert KS(j100)\rvert$(dB)	<0	-1.6	-1.6

3.3.5 Forward Compensator Design

To enhance delay tolerance, the H_∞ algorithm tents to lower open-loop gain and causes a magnitude lack in the high frequency region, as shown in Fig. 3.9. It can be compensated by a forward compensator. After H_∞ controller order reduction, a forward compensator is then designed that

$$K_{\text{hinf\_pre}} = \left(\frac{s/0.009561 + 1}{s/0.01 + 1}\right)\left(\frac{s/0.03578 + 1}{s/0.05495 + 1}\right) \tag{3.29}$$

which is basically a high-pass filter with two pole-zero pairs as shown in Fig. 3.10, and aims on compensating magnitude response lack within some frequency range. The maximum gain value is 4.12(dB), and system performance with a forward compensator is listed in Table 3.2.

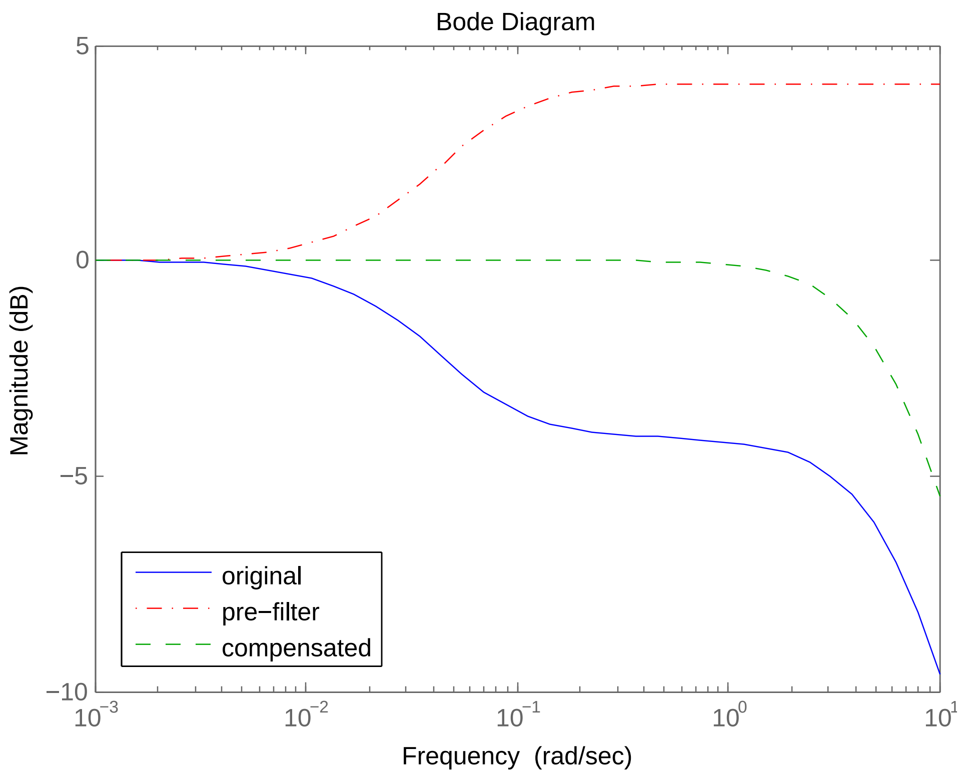


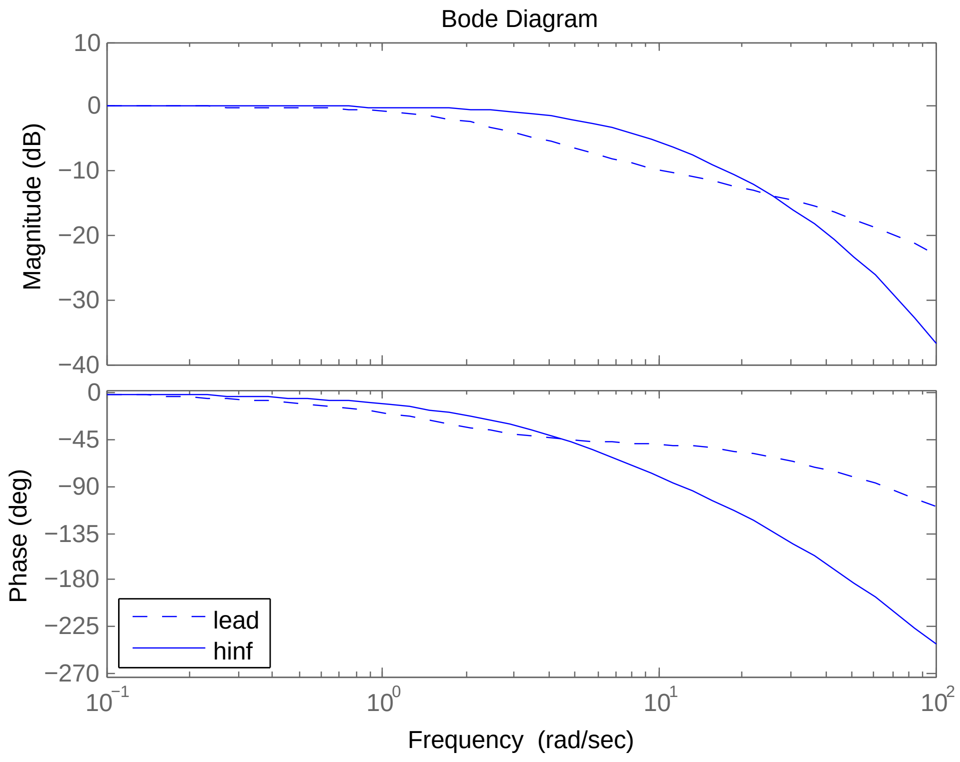
Figure 3.10: System responses after a forward compensator applied.

3.3.6 Simulation

Comparison in the system bandwidth for different controllers are shown in Fig. 3.11. It can be found that the H_∞ algorithm with modified weighting functions design enlarges the bandwidth from 2.39(Hz) to 6.42(Hz) compared to the original lead controller design. The rising time also decreases from 0.915(sec) to 0.338(sec).

The stability improvement can be observed by comparing transient responses under different network delay conditions. Fig. 3.12 shows the system responses with different controllers for a 1(Hz) input signal with varying network delay. With the delay time $T_d = 0$, both controllers have similar transient responses, but system controller by lead controllers has smaller amplitude due to its smaller bandwidth. When a network delay $T_d = 0.7$(sec) is applied at $t = 0.7$, the delay tolerance of system controlled by applying the lead controller is not held anymore and the system becomes unstable. (See Table 3.1). Meanwhile, although the tracking performance for H_∞ control is thus degraded, the system still remains stable.

(a)



(b)

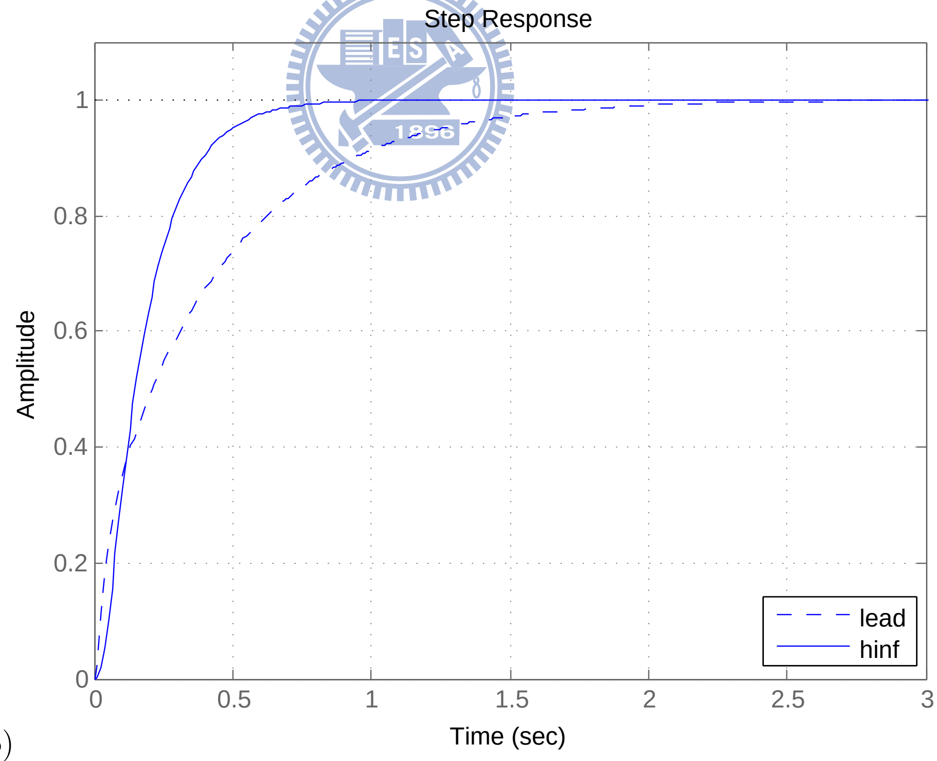


Figure 3.11: Comparison for lead controller and H_∞ controller in both (a)the frequency response and (b)the time response.

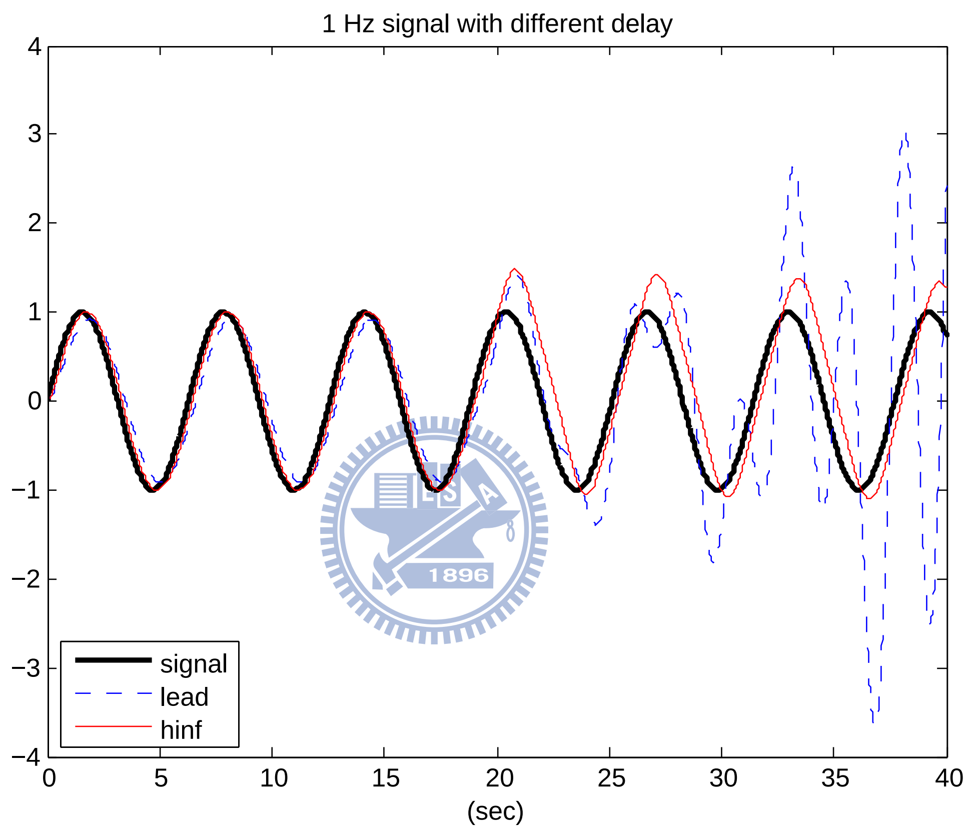


Figure 3.12: Transient response for different controller, with a 1Hz sine wave input and a delay $T_d = 0.7$ after $t = 19$.

3.4 Summary

In this chapter, the following results can be summarized as:

1. The rules for designing template of weighting functions under the mixed synthesis framework is described, and the steps for H_∞ NCS controller designed is illustrated with the identified plant from a servo motor system.
2. The control structure with a forward compensator is provided that it improves the system performances while still keeping its stability. Simulations shows the H_∞ design lead to larger delay tolerance and higher bandwidth.

Chapter 4 Analysis of Message Estimation for the Dropout

4.1 Markov Chain as Network Model

The network structure between the controller and the plant is shown in Fig. 4.1. The plant with the actuator and the sensor is considered as a continuous linear system, and a DSP micro-controller adopted as a network node with a sampling period h. Let $\tau_k = \begin{bmatrix} \tau_k^{sc} & \tau_k^{ca} \end{bmatrix}^T$, and τ_k^{sc}, τ_k^{ca} be the k_{th} term of sensor-to-controller delay time and controller-to-actuator delay time, respectively. Suppose that the delay τ_k is determined, a Markov state r_k is then assigned and

$$r_k \in \left\{ 1, \quad \cdots, \quad s \right\}. \tag{4.1}$$

Denote Y_k as a set that contains every previous Markov states and delay time, that is,

$$Y_k = \left\{ \tau_0 \sim \tau_k, \quad r_0 \sim r_k \right\} \tag{4.2}$$

The property of a Markov chain for every measurable set F has the relationship that

$$\mathbb{P}(\tau_k \in F | Y_{k-1}, r_k) = \mathbb{P}(\tau_k \in F | r_k) \tag{4.3}$$

Eq. (4.3) means the probability that delay time τ_k appears can be derived directly from current Markov state r_k. Let $Q = \{q_{ij}\}$ be the Markov transition matrix where

$$q_{ij} = \mathbb{P}(r_{k+1} = j | r_k = i), \ i, j \in \left\{ 1, \quad \cdots, \quad s \right\} \tag{4.4}$$

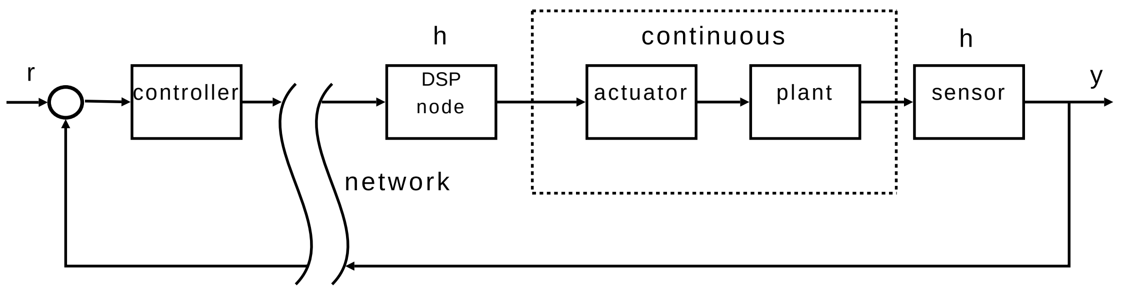


Figure 4.1: A NCS block diagram.

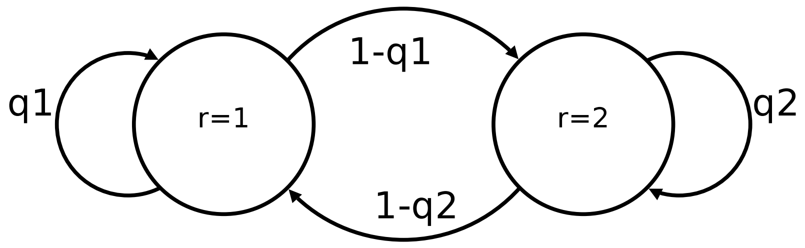


Figure 4.2: A two-state Markov chain with transition probability q_1 and q_2.

and let $\pi_i(k) = \mathbb{P}(r_k = i)$ be the probability of each Markov state, and denote $\pi(k)$ as the Markov probability distribution vector that

$$\pi(k) = \begin{bmatrix} \pi_1(k) & \pi_2(k) & \cdots & \pi_s(k) \end{bmatrix}. \tag{4.5}$$

A single step transition of Markov chain can be represented as

$$\begin{aligned} \pi(0) &= \pi_0 \\ \pi(k+1) &= \pi(k)Q. \end{aligned} \tag{4.6}$$

A simple illustration of a two-state Markov chain is shown in Fig. 4.2.

When Markov chain model is used for network analysis, it is generally assumed to be stationary and regular and the limit of probability distribution

$$\pi^\infty = \lim_{k\to\infty} \pi(k) \tag{4.7}$$

exists and is independent with $\pi(0)$. After a long transition time and suppose $\pi(k)$ is in a steady state π^∞, then

$$\pi^\infty Q = \pi^\infty, \text{ and } \sum_{i=1}^{s} \pi_i^\infty = 1, \tag{4.8}$$

and π^∞ can be derived uniquely from the equation above. For example, consider a two-state Markov chain with state transition probability

$$\begin{aligned} \mathbb{P}(r_k = 1 | r_{k-1} = 1) &= q_1 \quad (4.9) \\ \mathbb{P}(r_k = 2 | r_{k-1} = 2) &= q_2 \end{aligned}$$

or in a matrix form

$$Q = \begin{bmatrix} q_1 & 1-q_1 \\ 1-q_2 & q_2 \end{bmatrix}. \tag{4.10}$$

Generally, network is on when the first data received, so $\pi_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}$. From Eq. (4.8), π^∞ can be derived by calculating the eigenvector e with eigenvalue $\lambda = 1$ of the matrix $Q^T - I$. In this case,

$$e = \begin{bmatrix} 1 & \frac{q_1-1}{q_2-1} \end{bmatrix}^T \text{ with } \lambda{=}1 \tag{4.11}$$

and the steady-state probability distribution π^∞ is

$$\pi^\infty = \begin{bmatrix} \frac{q_2-1}{q_1+q_2-2} & \frac{q_1-1}{q_2+q_1-2} \end{bmatrix}. \tag{4.12}$$

4.2 System Stability with Markov Chain Network

Consider a discrete plant

$$\begin{aligned} x_{k+1} &= \Phi x_k + \Gamma u_k + v_k \\ y_k &= C x_k + w_k \end{aligned} \tag{4.13}$$

where $x(t) \in \mathbb{R}^n$, $y_k \in \mathbb{R}^p$, and v_k, w_k are uncorrelated Gaussian noise with average 0 and covariance matrices R_1 and R_2. A general form of controller K_c is

$$\begin{aligned} x^c_{k+1} &= \Phi^c(\tau_k, r_k)x^c_k + \Gamma^c(\tau_k, r_k)y_k \\ u_k &= C^c(\tau_k, r_k)x^c_k + D^c(\tau_k, r_k)y_k \end{aligned} \tag{4.14}$$

which is a time-varying system. If network delay occurs, Φ^c and Γ^c change with different value of r_k and τ_k, the Markov state and the delay time, separately. Combine Eq. (4.13) and Eq. (4.14) into a new state space equation as

$$z_{k+1} = \bar{\Phi} z_k + \bar{\Gamma} e_k \tag{4.15}$$

where

$$z_k = \begin{bmatrix} x_k^T & (x_k^c)^T \end{bmatrix}^T, \; e_k = \begin{bmatrix} v_k^T & w_k^T \end{bmatrix}^T, \tag{4.16}$$

and

$$\bar{\Phi} = \begin{bmatrix} \Phi + \Gamma D^c C & \Gamma C^c \\ \Gamma^c C & \Phi^c \end{bmatrix}, \; \bar{\Gamma} = \begin{bmatrix} I & \Gamma D^c \\ 0 & \Gamma^c \end{bmatrix} \tag{4.17}$$

with the covariance matrix

$$R = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix}. \tag{4.18}$$

Define the conditional state covariance $P_i(k)$ as

$$P_i(k) = \mathbb{E}_{Y_{k-1}}(z_k z_k^T | r_k = i) \tag{4.19}$$

and

$$\tilde{P}_i(k) = P_i(k)\pi_i(k) = \mathbb{E}_{Y_{k-1}}(z_k z_k^T 1_{r_k=i}). \tag{4.20}$$

If the covariance matrix in Eq. (4.19) is bounded, the closed-loop system in Eq. (4.15) can be thus considered as BIBO stable. It can be shown that the vectorized form of $\tilde{\boldsymbol{P}}(k)$ satisfies the recursion as [20]

$$\tilde{\boldsymbol{P}}(k+1) = (Q^T \otimes I)\text{diag}(\mathcal{A}_i)\tilde{\boldsymbol{P}}(k) + (Q^T \otimes I)(\text{diag}(\pi_i(k)) \otimes I)\boldsymbol{G} \tag{4.21}$$

where

$$\begin{aligned} \tilde{\boldsymbol{P}}(k) &= \begin{bmatrix} \text{vec}\tilde{P}_1(k) & \text{vec}\tilde{P}_2(k) & \cdots & \text{vec}\tilde{P}_s(k) \end{bmatrix}^T \\ \boldsymbol{G} &= \begin{bmatrix} \text{vec}\mathcal{G}_1 & \text{vec}\mathcal{G}_2 & \cdots & \text{vec}\mathcal{G}_s \end{bmatrix}^T \end{aligned} \tag{4.22}$$

and

$$\begin{aligned} \mathcal{A}_i &= \mathbb{E}_{\tau_k}(\bar{\Phi}(\tau_k) \otimes \bar{\Phi}(\tau_k) | r_k = i) \\ \mathcal{G}_i &= \mathbb{E}_{\tau_k}(\bar{\Gamma}(\tau_k) R \bar{\Gamma}^T(\tau_k) | r_k = i). \end{aligned} \tag{4.23}$$

From Eq. (4.21), if the stability matrix

$$V = (Q^T \otimes I)\text{diag}(\mathcal{A}_i) \tag{4.24}$$

has all its eigenvalues inside the unit circle, $\tilde{\boldsymbol{P}}(k)$ is bounded, thus the covariance matrix $P_i(k)$ is also bounded. Therefore, the closed-loop system is stable in the BIBO sense. For example, the stability matrix V of a NCS with network modeled by a two-state Markov chain is

$$V = \begin{bmatrix} q_1\mathcal{A}_1 & (1-q_2)\mathcal{A}_2 \\ (1-q_1)\mathcal{A}_1 & q_2\mathcal{A}_2 \end{bmatrix} \tag{4.25}$$

where

$$\mathcal{A}_i = \mathbb{E}_{\tau_k}(\bar{\Phi}(i) \otimes \bar{\Phi}(i)), \; i \in \{1,2\}. \tag{4.26}$$

Moreover, if $\bar{\Phi}(r_k)$ is not a function of τ_k,

$$\mathcal{A}_i = \bar{\Phi}(i) \otimes \bar{\Phi}(i), \; i \in \{1,2\}. \tag{4.27}$$

This happens when the network dropout is modeled as in a Markov chain mentioned later in section 4.4 and section 4.5.

4.3 Network Dropout Modeled by the Markov Chain

4.3.1 Network Dropout

Consider an NCS with network transmission dropout modeled by a a two state Markov chain [9]. Define the Markov states r_k for the k_{th} network transmission as

$$r_k = \begin{cases} 1, & k_{\text{th}} \text{ data received,} \\ 2, & k_{\text{th}} \text{ data dropout,} \end{cases} \tag{4.28}$$

as shown in Fig. 4.2. Generally this happens when τ_k is near h and the processing time left is not enough. The state transition matrix Q is in the same form as Eq. (4.10). Let $\pi_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}$, the average dropout rate $\mathbb{P}_d$ can be derived from the stead-state probability distribution π^∞ as in Eq. (4.12) as

$$\mathbb{P}_d = \frac{q_1 - 1}{q_2 + q_1 - 2}. \tag{4.29}$$

The relationship between Q and the average dropout is shown in Fig. 4.3, which the stair curves for constant dropout rate are straight lines that end up in $[q_1, q_2] = [1, 1]$. For instance, the plot in 4.4(a) has $\mathbb{P}_d = 0.5$ and 4.4(b) has $\mathbb{P}_d = 0.385$. Clearly, even with identical $\mathbb{P}_d$ values, the network situations could still be different with different Q.

4.3.2 Markov Model Measurement

When using Eq. (4.24) to determine the stability of a specified NCS, it is crucial to know the value of Markov transition matrix Q. Generally, it can be calculated from a series

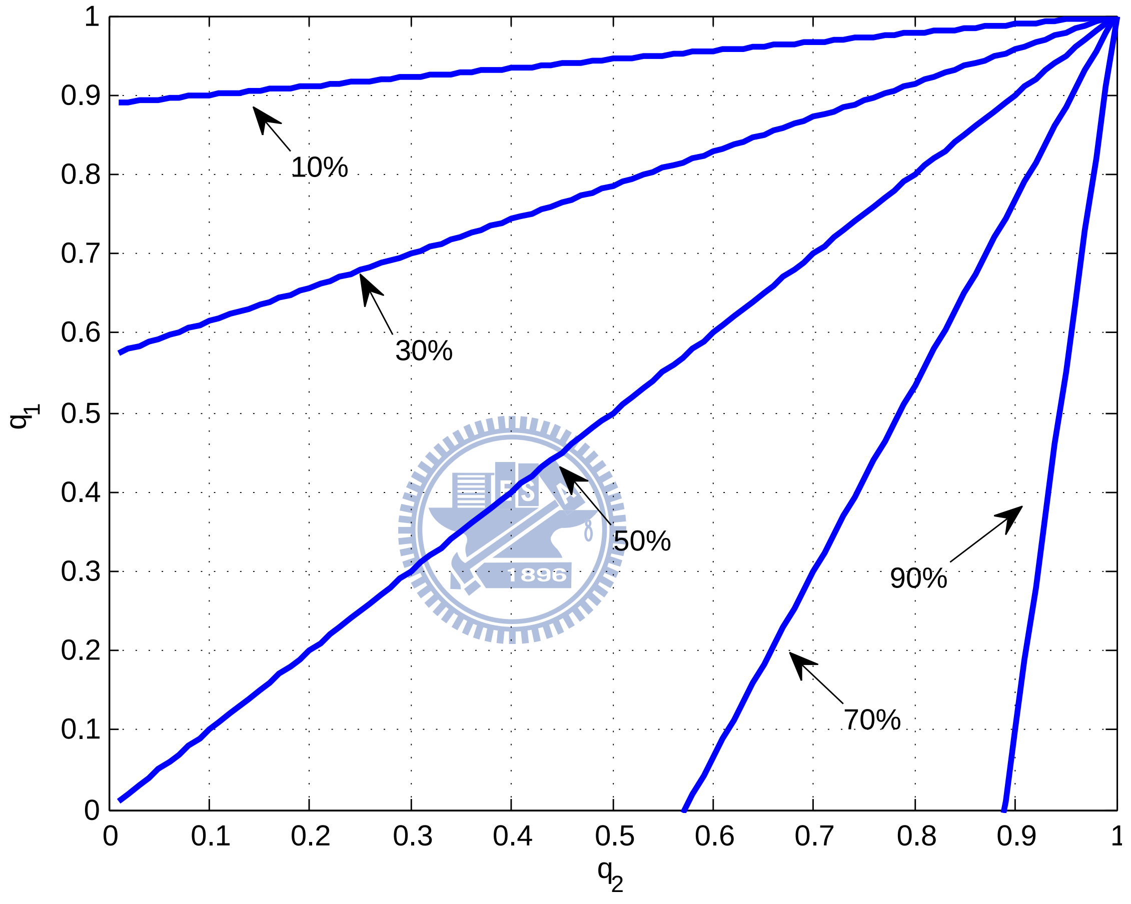
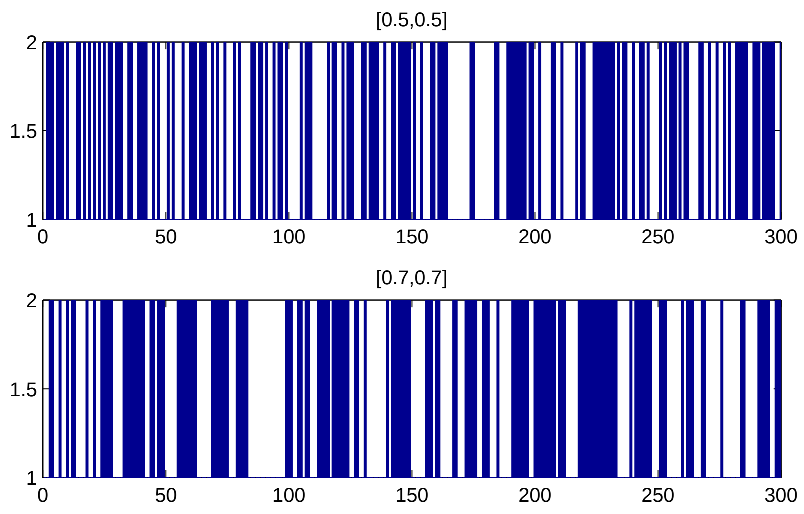
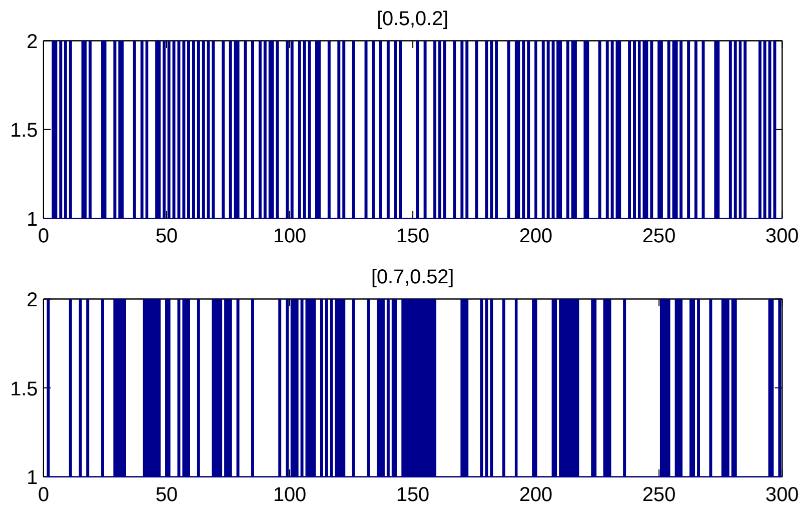


Figure 4.3: Dropout rate versus Q.



(a)



(b)

Figure 4.4: The two-state Markov chain with (a) $[q_1, q_2] = [0.5, 0.7]$ and $[0.7, 0.7]$ (dropout rate 0.5) (b) $[q_1, q_2] = [0.5, 0.2]$ and $[0.7, 0.52]$ (dropout rate 0.385).

of network transmission data with enough data length. Several examples for a two-state Markov chain are shown in Fig. 4.4 with a data length 300. The values of steady-state probability distribution π^∞ are identical in both Fig. 4.4(a) and 4.4(b), and obviously that different Q values cause different distribution for r_k.

However, the exact value of Q is unknown, and it needs to be found from measured data set as in Fig. 4.4. A general algorithm to estimate Q is shown in Algorithm 4.1. Perform this estimation for 100 times with different Q and the mean error are shown in Table 4.1. It can be found that for data length equal to 300, the mean error for estimated $\hat{Q}$ is around 3%. Fig. 4.5 shows that $\hat{Q}$ converges after about 150 samples. In $q_1 - q_2$

Algorithm 4.1 Estimating $\hat{Q}$

1. Assign a Markov state r_k to each network instance k.

2. q_{ij} equals to the count of state transition from r_i to r_j is divided by total appearance and

$$q_{ij}(k) = \frac{\sum_k (r_i \rightarrow r_j)}{\sum_k (r_i)}$$

3. Form the matrix $\hat{Q}$ with elements in the i_{th} row and j_{th} column be $q_{ij}(\bar{k})$, where $\bar{k}$ is the maximum data length.

plane, the measured value will stay in a small region near the actual Q. The measurement precision increases as the data length increases. Normally, larger q_{ii} converges faster in estimation.

Table 4.1: $\hat{Q}$ estimation with different Q.

	case 1		case 2		case 3		case 4	
$[q_1, q_2](\%)$	50	50	70	70	50	20	70	52
mean error(%)	3.2	3.3	2.8	3.1	3.0	3.1	2.7	3.8

4.3.3 Dropout Estimation

To overcome network transmission dropout problem, data estimation methods are used. From Fig. 4.6, there are two separate estimations needed in the controller-to-actuator path and the sensor-to-controller path. Since the tracking signal r is arbitrarily given, the estimated signal can also be regarded as both the model-based and the model-free signals. These two kinds of signal are estimated in different ways using the model-based least square fit (MBLSF) and the model-free least square fit (MFLSF) methods, respectively. For simplification, each situation is analyzed independently. The network connection is treated as a single direction only in one way because of the complexity in real analysis.

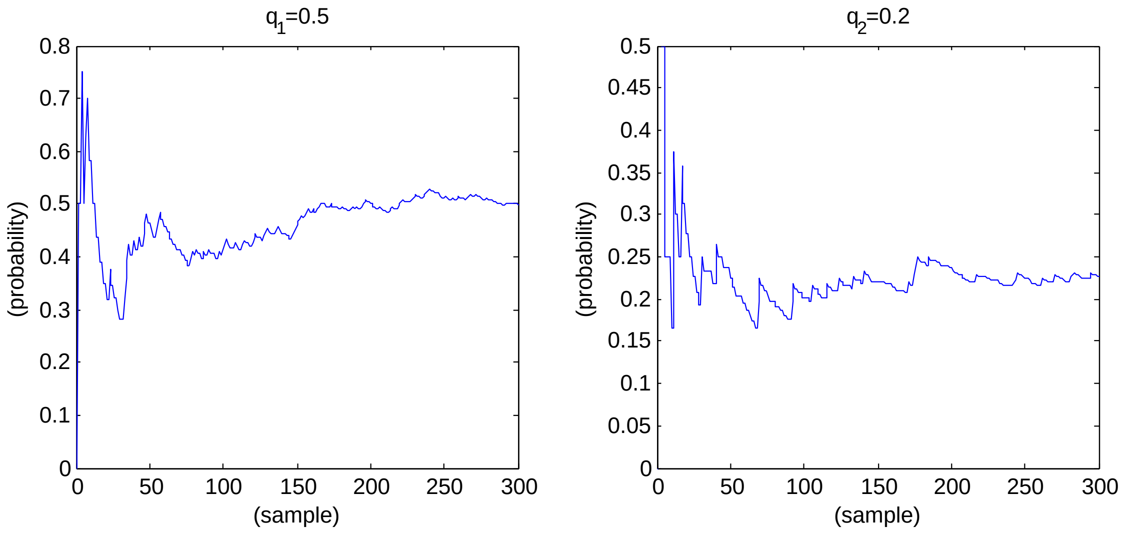


Figure 4.5: Transient of estimated $\hat{Q}$ with $[q_1, q_2] = [0.5, 0.2]$ for the case shown in the top figure of Fig. 4.4(b).

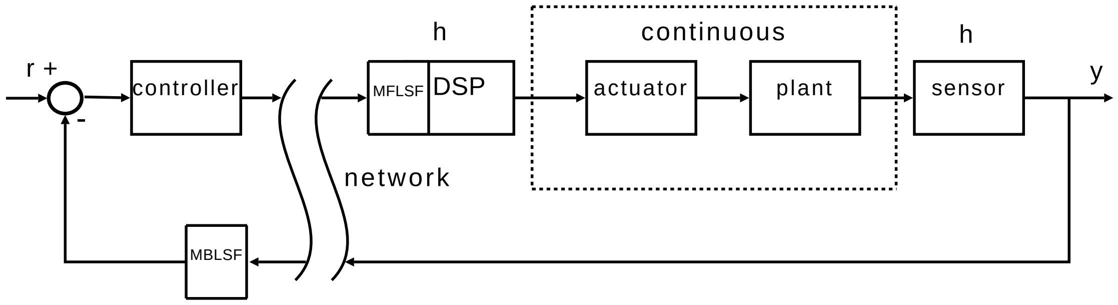


Figure 4.6: Markov network with dropout and the forward and feedback estimation.

4.4 The MBLSF Estimation

4.4.1 Structure

Consider the NCS with a plant P, the controller K_c with feedback control that

$$u = -K_c y. \tag{4.30}$$

Assume that the network connection is only between sensor-to-controller path. Therefore, the plant P becomes

$$\begin{cases} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{cases} \tag{4.31}$$

and controller K_c becomes

$$\begin{cases} x^c_{k+1} &= A_c x^c_k + B_c(-\bar{y}_k) \\ u_k &= C_c x^c_k + D_c(-\bar{y}_k) \end{cases}. \tag{4.32}$$

When network dropout occurs, an estimation method is applied to the controller input signal such that

$$\bar{y}_k = \begin{cases} y_k, & \text{if } r_k = 1(\text{normal}), \\ \hat{y}_k, & \text{if } r_k = 2(\text{dropout}), \end{cases} \tag{4.33}$$

where $\hat{y}_k$ is the estimated value. Since plant input u_k and output y_k are fully known in the controller side, $\hat{y}_k$ can be also estimated from model-based estimation methods. From the past W points, y and u has the following relationship [14] :

$$\mathbf{y}_{k,W} = \mathcal{C}_W x_{k-W} + \mathcal{D}_W \mathbf{u}_{k,W} \tag{4.34}$$

where

$$\begin{aligned} \mathbf{u}_{k,W} &= \begin{bmatrix} u^T_{k-1} & u^T_{k-2} & \cdots & u^T_{k-W} \end{bmatrix}^T \\ \mathbf{y}_{k,W} &= \begin{bmatrix} y^T_{k-1} & y^T_{k-2} & \cdots & y^T_{k-W} \end{bmatrix}^T \end{aligned} \tag{4.35}$$

and

$$
\begin{aligned}
\mathcal{C}_W &= \begin{bmatrix} (CA^{W-1})^T & (CA^{W-2})^T & \cdots & (CA)^T & (C)^T \end{bmatrix}^T \\
\mathcal{D}_M &= \begin{bmatrix} D & CB & CAB & \cdots & CA^{W-3}B & CA^{W-2}B \\ 0 & D & CB & \cdots & CA^{W-4}B & CA^{W-3}B \\ 0 & 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & D & CB & CAB \\ 0 & 0 & \cdots & 0 & D & CB \\ 0 & 0 & \cdots & 0 & 0 & D \end{bmatrix}.
\end{aligned} \tag{4.36}
$$

The value of x_{k-W} can therefore be estimated from a model-based least square fit estimation that

$$\hat{x}_{k-W} = (\mathcal{C}_W^T\mathcal{C}_W)^{-1}\mathcal{C}_W^T(\mathbf{y}_{k,W} - \mathcal{D}_W\mathbf{u}_{k,W}). \tag{4.37}$$

and $\hat{y}_k$ can be calculated from $\hat{x}_{k-W}$ that

$$
\begin{aligned}
\hat{y}_k &= CA^W\hat{x}_{k-W} + \sum_{i=1}^{W} CA^{i-1}Bu_{k-i} + Du_k \\
&= \mathcal{H}\mathbf{y}_{k,W} + (\mathcal{B}_W - CA^W(\mathcal{C}_W^T\mathcal{C}_W)^{-1}\mathcal{C}_W^T\mathcal{D}_W)\mathbf{u}_{k,W} + Du_k
\end{aligned} \tag{4.38}
$$

where

$$
\begin{aligned}
\mathcal{H} &= CA^W(\mathcal{C}_W^T\mathcal{C}_W)^{-1}\mathcal{C}_W^T, \\
\mathcal{B}_W &= \begin{bmatrix} CB & CAB & \cdots & CA^{W-2}B & CA^{W-1}B \end{bmatrix}.
\end{aligned} \tag{4.39}
$$

Note that W must be larger or equal to the order of x_k, otherwise $\mathcal{C}_W^T\mathcal{C}_W$ will be singular and $\mathcal{T}$ cannot be found. In other words, there must be more samples than unknowns. Rewrite Eq. (4.38) in a matrix form as

$$\hat{y}_k = \mathcal{H}\mathbf{y}_{k,W} + \mathcal{T}\mathbf{u}_{k,W} + Du_k \tag{4.40}$$

where

$$\mathcal{T} = \mathcal{B}_W - CA^W(\mathcal{C}_W^T\mathcal{C}_W)^{-1}\mathcal{C}_W^T\mathcal{D}_W \tag{4.41}$$

and a time-varying state space equation can be constructed from x_k, x_k^c, $\bar{\mathbf{u}}_{k,W}$ and $\bar{\mathbf{y}}_{k,W}$ which

$$\mathfrak{z}_{k+1} = \mathbf{\Phi}_1(r_k)\mathfrak{z}_k \tag{4.42}$$

where

$$\mathfrak{z}_k = \begin{bmatrix} x_k^T & (x_k^c)^T & \mathbf{u}_{k,W}^T & \bar{\mathbf{y}}_{k,W}^T \end{bmatrix}^T \tag{4.43}$$

and

$$\mathbf{\Phi}_1(r_k) = \begin{cases} \begin{bmatrix} A - BE_dD_cC & BE_dC_c & 0 & 0 \\ B_cE_DC & A_c - B_cDE_dC_c & 0 & 0 \\ -E_{1,W-1}^TE_dD_cC & E_{1,W-1}^TE_dC_c & E_{u,W-1} & 0 \\ -E_{1,W-1}^TE_DC & E_{1,W-1}^TDE_dC_c & 0 & E_{u,W-1} \end{bmatrix}, & r_k = 1, \\ \begin{bmatrix} A & BE_dC_c & -BE_dD_c\mathcal{T} & -BE_dD_c\mathcal{H} \\ 0 & A_c - B_cDE_dC_c & B_cE_D\mathcal{T} & B_cE_D\mathcal{H} \\ 0 & E_{1,W-1}^TE_dC_c & E_{u,W-1} - E_{1,W-1}^TE_dD_c\mathcal{T} & -E_{1,W-1}^TE_dD_c\mathcal{H} \\ 0 & E_{1,W-1}^TDE_dC_c & -E_{1,W-1}^TE_D\mathcal{T} & E_{u,W-1} - E_{1,W-1}^TE_D\mathcal{H} \end{bmatrix}, & \\ & r_k = 2, \end{cases} \tag{4.44}$$

with

$$E_{u,K} = \begin{bmatrix} 0_{1\times K} & 0 \\ I_K & 0_{K\times 1} \end{bmatrix}, \; E_{1,K} = \begin{bmatrix} 1 & 0_{1\times K} \end{bmatrix} \tag{4.45}$$

and

$$E_d = (I + D_cD)^{-1}, \; E_D = (DE_dD_c - I). \tag{4.46}$$

System stability can be thus determined by applying $\Phi_1(r_k)$ to Eq. (4.24) and check its maximum eigenvalues for different Markov transition matrix Q.

4.4.2 Simulation Setup

A plant P obtained from a real networked servo motor system [17] is adopted as

$$P(s) = \frac{8.04}{(1 + 0.00167s)} \times \frac{1}{s} \tag{4.47}$$

which is the velocity-loop model with an integrator by expending the velocity-loop plant into the position loop. A controller K_c is chosen as

$$K_c(s) = 10. \tag{4.48}$$

The sampling time h is set to be 2(ms). By discretizing the plant P into a state space form, the plant model becomes

$$A = \begin{bmatrix} 1.3019 & -0.6038 \\ 0.5 & 0 \end{bmatrix},\ B = \begin{bmatrix} 0.125 \\ 0 \end{bmatrix},\ C = \begin{bmatrix} 0.0537 \\ 0.0723 \end{bmatrix}^T \tag{4.49}$$

and $D = 0$, and the controller K_c becomes

$$A_c = 0,\ B_c = 0,\ C_c = 0,\ D_c = 10. \tag{4.50}$$

In each sampling period, a packet is transmitted through the network between controller at the remote site and the plant in the near site. If the packet does not arrive in time, the control algorithm will replace it with an estimated value. This process is modeled by a two-state Markov chain as in Eq. (4.28) with a pre-specified Q. The input signal r_k is chosen as

$$r_k = \sin(0.05k),\ k \in \{1, 2, \ldots, 1000\}. \tag{4.51}$$

4.4.3 Stability

Assume the network connection work as described in Eq. (4.31) and (4.32), a simulation is tested by applying $\Phi_1(r_k)$ to Eq. (4.24) and the maximum eigenvalue of V with different Q is also calculated. Because P is with an order 2, the minimum value for W is 2. Estimator coefficients $\mathcal{H}$ and $\mathcal{T}$ for $W \leq 4$ are listed in Table 4.2. For example, the estimation for y_k when $W = 2$ is

$$\hat{y}_k = 1.3019\bar{y}_{k-1} - 0.3019\bar{y}_{k-2} + 0.0067u_{k-1} + 0.0045u_{k-2}. \tag{4.52}$$

The maximum eigenvalues $\max|\lambda|$ for $W = 2,\ 3$ versus Q are shown in Fig. 4.7. For both cases, the values of $\max|\lambda|$ only exceed or equal to 1 when $q_2 = 100\%$, that is, when the network is fully occupied. Otherwise, the model-based LSF is always stable according the the present analysis.

Table 4.2: $\mathcal{H}$ and $\mathcal{T}$ values for plant P with$W \leq 4$.

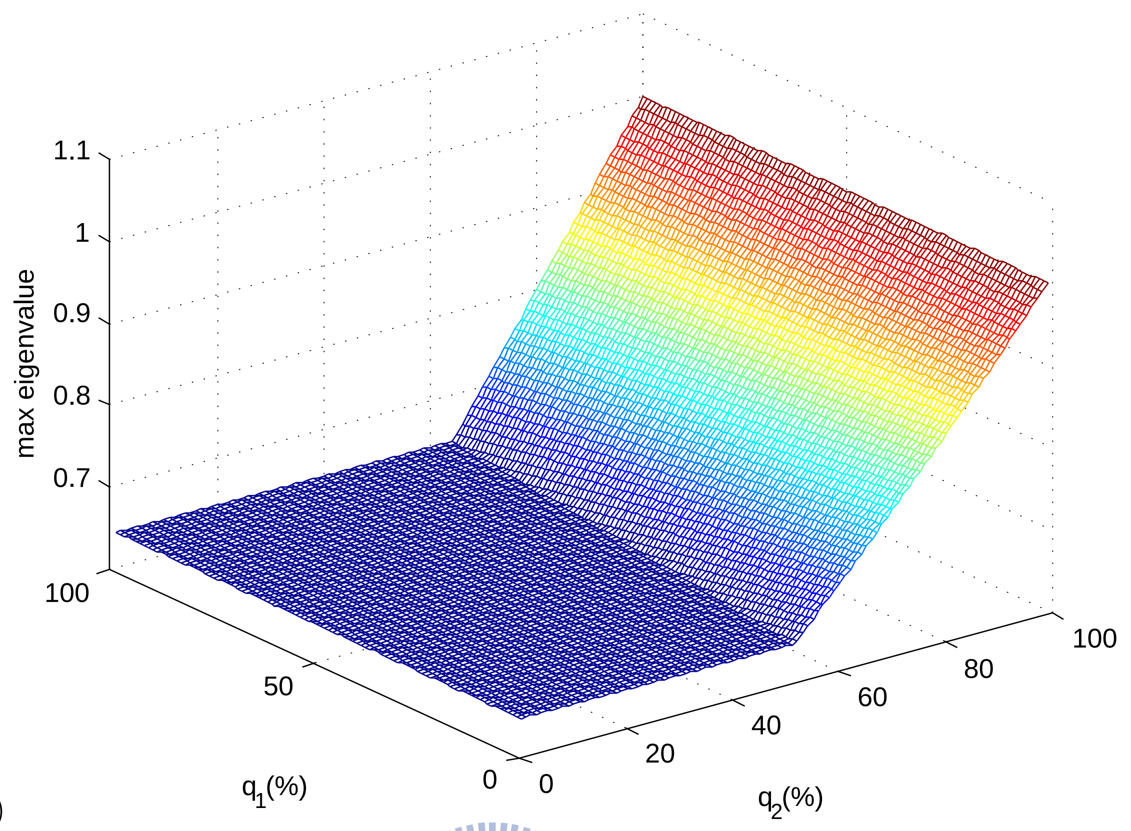
W	$\mathcal{H}$	$\mathcal{T}$
2	$[1.3019, -0.3019]$	$[0.0067, 0.0045]$
3	$[0.6936, 0.4901, -0.1837]$	$[0.0067, 0.0086, 0.0027]$
4	$[3.136, 3.057, 2.797, 1.934, -0.924]_{\times 10^{-1}}$	$[0.067, 0.111, 0.09, 0.051, 0.014]_{\times 10^{-1}}$

4.4.4 Tracking Performance

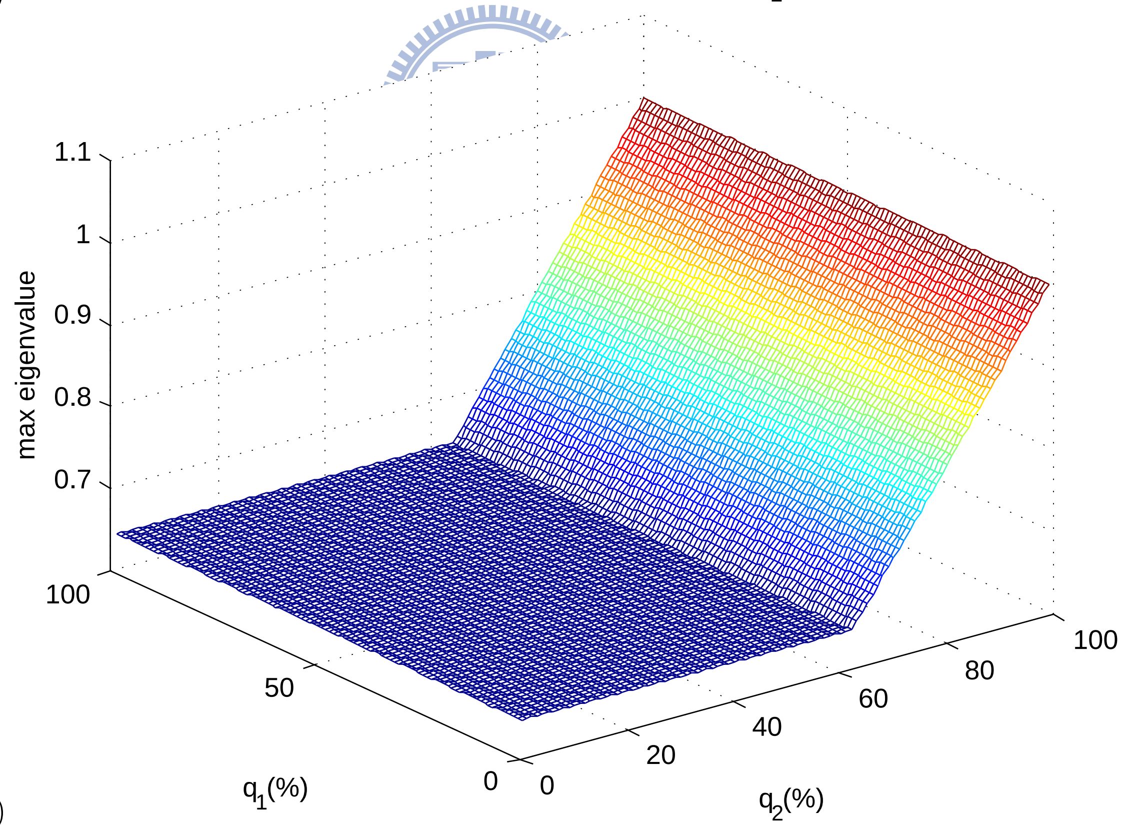
Time-domain illustrations are shown in Fig. 4.8 and Fig. 4.9 under a relatively undesirable network condition $[q_1, q_2] = [0.1, 0.8]$. Note that the system remains stable as predicted. The root mean square (RMS) error is 0.122×10^{-3}, which is neglectable. When noise v_k exists in sensor feedback, a larger W renders a smaller noise effect and it leads to a smaller RMS error. This is shown in Table 4.3. Therefore, W value should be properly chosen to minimize calculate efforts and noise effects with practical concern. In the present simulation, $W = 3$ is the best choice, as shown in Table 4.3.

Table 4.3: Tracking RMS error and the maximum error with different order of MBLSF estimations with $[q_1, q_2] = [0.1, 0.8]$ simulated with a noise $v_k = \text{uniform}[\pm 0.1]$.

W	RMS$\times 10^3$	$\max \lvert e \rvert \times 10^3$
2	78.974	289
3	68.287	192
4	65.739	187
5	63.604	181



(a)



(b)

Figure 4.7: Maximum eigenvalue for different Q in MBLSF estimation with (a) $W = 2$ (b) $W = 3$.

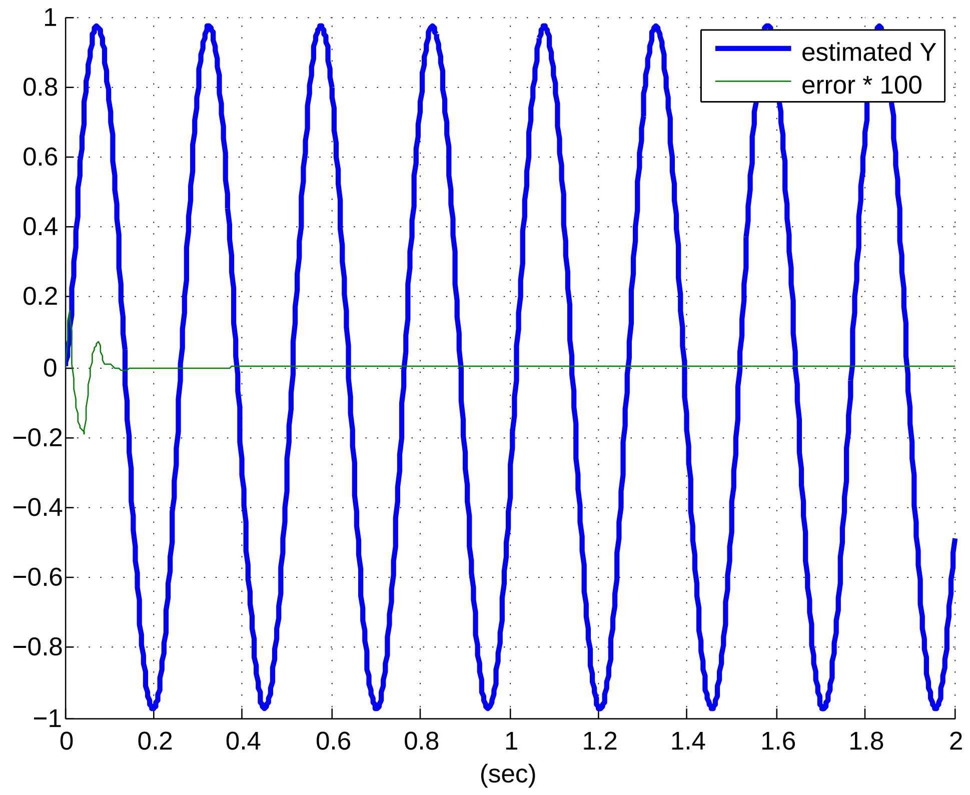


Figure 4.8: Time response for MBLSF with $W = 2$ at $[q_1, q_2] = [0.1, 0.8]$.

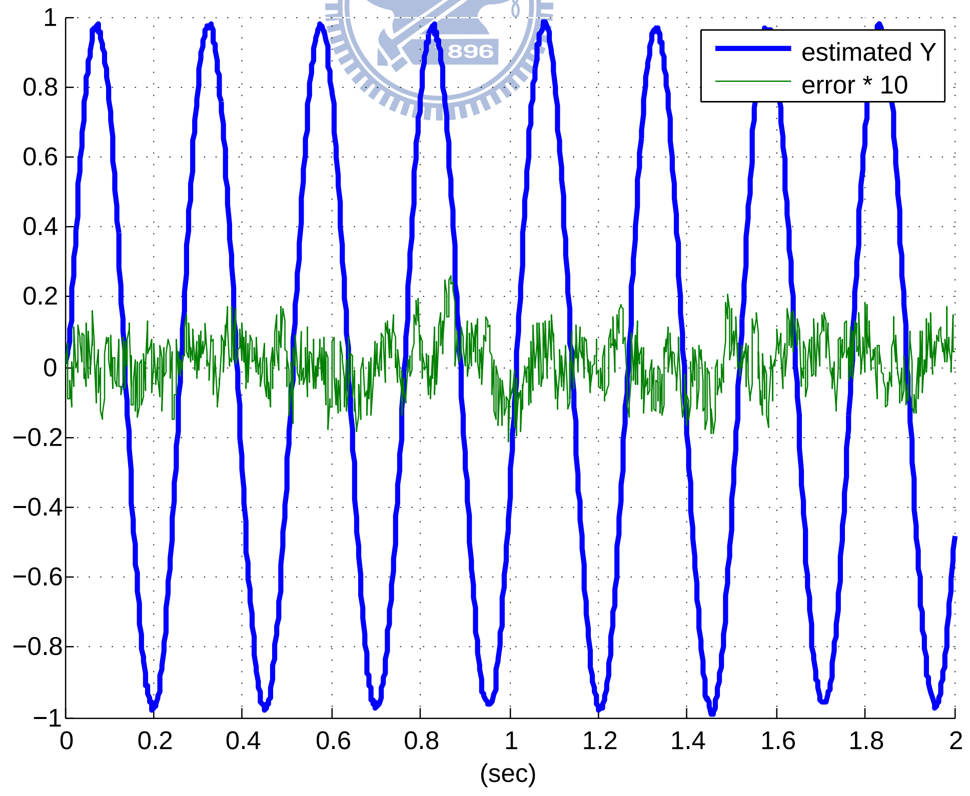


Figure 4.9: Time response for MBLSF with $W = 2$ and noise at $[q_1, q_2] = [0.1, 0.8]$.

4.5 The MFLSF Estimation

4.5.1 Structure

Assume that the network connection is only in the controller-to-actuator path. The plant P is discretized into discrete state space form that

$$\begin{cases} x_{k+1} &= Ax_k + B\bar{u}_k \\ y_k &= Cx_k + D\bar{u}_k \end{cases} \tag{4.53}$$

and controller $K_c(s)$

$$\begin{cases} x^c_{k+1} &= A_c x^c_k + B_c(-y_k) \\ u_k &= C_c x^c_k + D_c(-y_k) \end{cases} . \tag{4.54}$$

When network dropout occurs, an estimation method is applied to the controller output signal such that

$$\bar{u}_k = \begin{cases} u_k, & r_k = 1(\text{normal}), \\ \hat{u}_k, & r_k = 2(\text{dropout}), \end{cases} \tag{4.55}$$

where $\hat{u}_k$ is the estimated value. Since u_k contains information from r_k and can only be estimated from model-free estimation methods. Suppose that u_k is a N_{th} order polynomial function with variable t_k [15],

$$u_k = c_0 + c_1 t_k + c_2 t_k^2 + \cdots + c_N t_k^N \tag{4.56}$$

where $c_i \in \mathbb{R}$ and $t_k \in \mathbb{Z}$, because Eq. (4.56) is used in a discrete-time system. The k_{th} data can be then estimated by a model-free least square fit method from previous M points with N_{th} order estimation. Let $t_{k-M-1} = 0$, thus

$$\mathbf{u}_{k,M} = \mathbf{L}_{M,N}\mathbf{c}_N \tag{4.57}$$

where

$$\begin{aligned} \mathbf{u}_{k,M} &= \begin{bmatrix} u^T_{k-1} & u^T_{k-2} & \cdots & u^T_{k-M} \end{bmatrix}^T, \\ \mathbf{c}_N &= \begin{bmatrix} c_0 & c_1 & \cdots & c_N \end{bmatrix}^T, \end{aligned} \tag{4.58}$$

and

$$\mathbf{L}_{M,N} = \begin{bmatrix} 1 & M & \cdots & M^N \\ 1 & M-1 & \cdots & (M-1)^N \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1^N \end{bmatrix}. \tag{4.59}$$

The estimated $\hat{u}_k$ is

$$\hat{u}_k = \mathbf{b}_{M,N}(\mathbf{L}_{M,N}^T \mathbf{L}_{M,N})^{-1} \mathbf{L}_{M,N}^T \mathbf{u}_{k,M} \tag{4.60}$$

where

$$\mathbf{b}_{M,N} = \begin{bmatrix} 1 & M+1 & (M+1)^2 & \cdots & (M+1)^N \end{bmatrix}. \tag{4.61}$$

Rewrite Eq. (4.60) in a summation form that

$$\hat{u}_k = \sum_{j=1}^{M} p_j u_{k-j} \tag{4.62}$$

where p_i are coefficients calculated from the MFLSF estimation above. Combine the estimated data $\hat{u}_k$ when network dropout occurs and received data $\mathbf{u}_{k,M}$ into a vector

$$\bar{\mathbf{u}}_{k+1,M} = \mathcal{P} \bar{\mathbf{u}}_{k,M} \tag{4.63}$$

where

$$\mathcal{P} = \begin{bmatrix} p_1 & p_2 & \cdots & p_M \\ 1 & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix}. \tag{4.64}$$

For example, when $M = 3$ and $N = 2$, the matrix becomes

$$\mathcal{P} = \begin{bmatrix} \frac{4}{3} & \frac{1}{3} & -\frac{2}{3} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

A time-varying state space equation can be constructed from x_k, x_k^c, and $\bar{\mathbf{u}}_{k,M}$ that

$$z_{k+1} = \mathbf{\Phi}_2(r_k) z_k \tag{4.65}$$

where

$$z_k = \begin{bmatrix} x_k^T & (x_k^c)^T & (\bar{\mathbf{u}}_{k,M})^T \end{bmatrix}^T \tag{4.66}$$

and

$$\mathbf{\Phi}_2(r_k) = \begin{cases} \begin{bmatrix} A - BE_dD_cC & BE_dC_c & 0 \\ B_cE_DC & A_c - B_cDE_dC_c & 0 \\ -E_{1,N}^TE_dD_cC & E_{1,N}^TE_dC_c & E_{u,N} \end{bmatrix}, & r_k = 1, \\ \begin{bmatrix} A & 0 & BE_{1,N}\mathcal{P} \\ -B_cC & A_c & -B_cDE_{1,N}\mathcal{P} \\ 0 & 0 & \mathcal{P} \end{bmatrix}, & r_k = 2. \end{cases} \tag{4.67}$$

System stability can be thus determined by applying $\Phi_2(r_k)$ to Eq. (4.24) and check its maximum eigenvalues for different Markov transition matrix Q.

4.5.2 Stability

Assume that the network connection works as described in Eq. (4.53) and (4.54), the simulation is peformed with the same setup in section 4.4.2. Estimator coefficients for $M \leq 3$ are listed in Table 4.4. For example, the MFLSF with $M = 2$ and $N = 1$ is

$$\hat{u}_k = 2\bar{u}_{k-1} - \bar{u}_{k-2}. \tag{4.68}$$

Apply $\Phi_2(r_k)$ to equation Eq. (4.24) and calculate the maximum eigenvalue of V with different Q. The calculated results are shown in Fig. 4.10. Each line represents the boundary between $\max|\lambda| < 1$ (top-left) and $\max|\lambda| \geq 1$ (bottom-right), or the stable and the unstable regions, respectively. The stable area increases as M increases or as N decreases. Compared with Fig. 4.7, MFLSF is more likely to be unstable and is the dominant factor for dropout estimation stability. As stated in Eq. (4.24), system stability criteria defined here is BIBO stability. That is, if Q is located in the unstable region but not very far from the boundary, the system response will have bounded RMS error but unbounded maximum absolute error. Fig. 4.11 shows a NCS with the same Q value with different M and N. When N increases, the stable region becomes smaller. It is

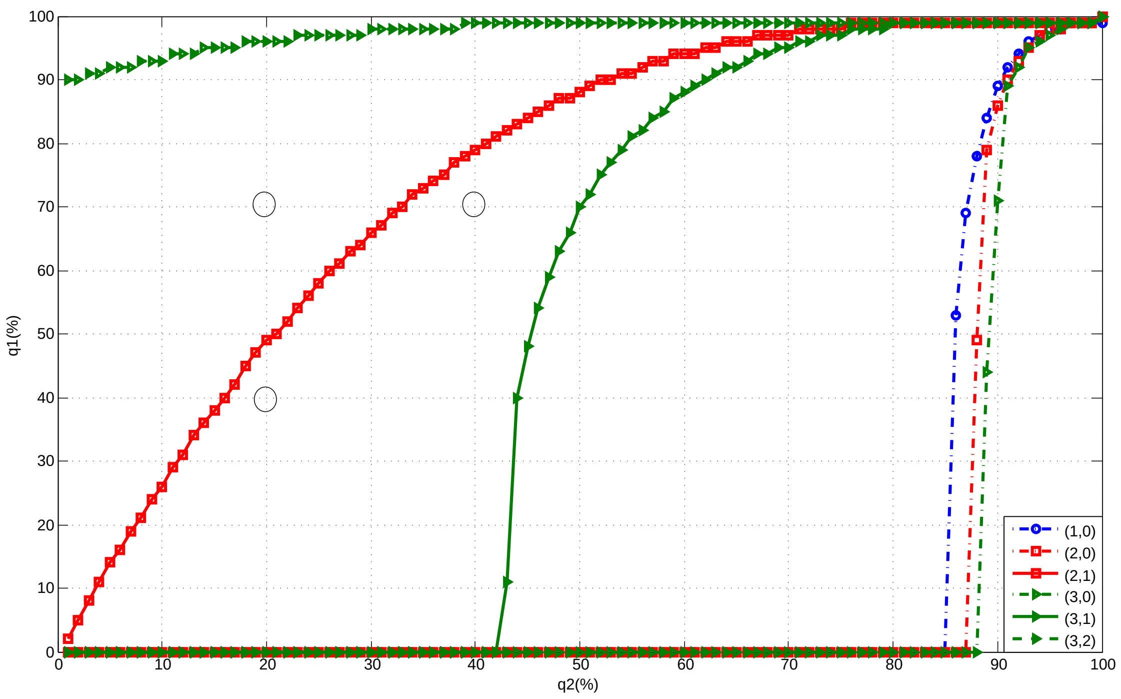


Figure 4.10: Stability region on Q plane for MFLSF with different (M, N).

possible for a specified Q being BIBO stable in MFLSF(1,0) but not in MFLSF(2,1) or MFLSF(3,2).

Table 4.4: $[p_i]$ values for $M \leq 3$.

$M \diagdown N$	0	1	2
1	$[\ 1\ ]$	$-$	$-$
2	$[\ \frac{1}{2}\quad \frac{1}{2}\ ]$	$[\ 2\quad -1\ ]$	$-$
3	$[\ \frac{1}{3}\quad \frac{1}{3}\quad \frac{1}{3}\ ]$	$[\ \frac{4}{3}\quad \frac{1}{3}\quad -\frac{2}{3}\ ]$	$[\ 3\quad -3\quad 1\ ]$

4.5.3 Tracking Performance

Generally, the value of N in MFLSF is with the most concern and M is set to be $(N+1)$. Theoretically, higher order of N can preserve higher-order natures of those data to be estimated and therefore, it has better tracking performance. RMS error with different

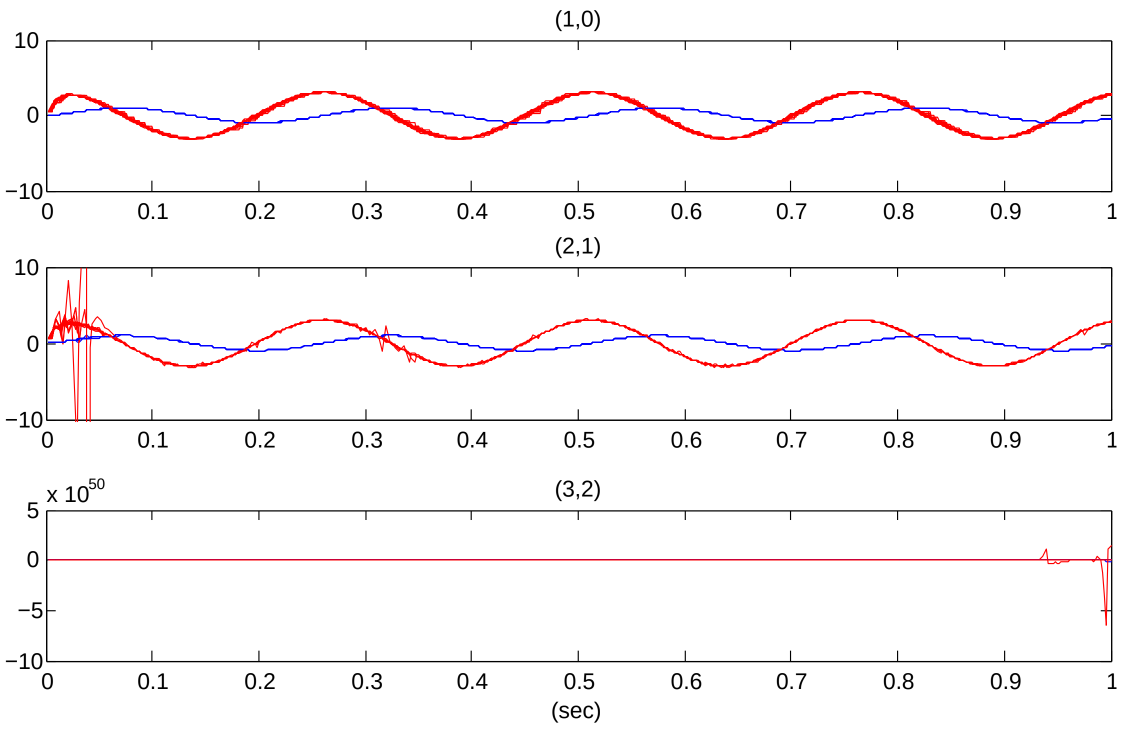


Figure 4.11: Time responses for MFLSF(1,0), MFLSF(2,1) and MFLSF(3,2) and $[q_1, q_2] = [0.4, 0.2]$.

MFLSF estimations are listed in Table 4.5, and they are in good agreement with predicted results. However, higher N also causes MFLSF to have smaller unstable area in Q plane, and it leads to unbounded maximum tracking error, as listed in Table 4.6.

Table 4.5: RME error for MFLSF estimations with different $[q_1, q_2]$.

$[q_1, q_2]$	MFLSF(1,0)	MFLSF(2,1)
[0.7,0.2]	4.34	0.54
[0.4,0.2]	6.19	1.59
[0.7,0.4]	8.06	2.00

Table 4.6: The maximum tracking error for MFLSF with different $[q_1, q_2]$.

$[q_1, q_2]$	MFLSF(1,0)	MFLSF(2,1)
[0.7,0.2]	7.36	13.02
[0.4,0.2]	9.13	327.89
[0.7,0.4]	19.23	137.88

4.5.4 Estimator Switching Policy

From the previous analysis, there is a trade-off in selecting N between RMS error and maximum tracking error. To conquer this problem, an estimator switching mechanism is proposed here. Initially, a set of reference points R_i is spreaded on the Q plane. Each point has a particular value of Q_i and its stability with any MFLSF estimator can be determined and a suitable estimator is assigned to it. A set of 2×2 reference points is shown in Fig. 4.12. When dropout occurs, the local Markov transition matrix $\bar{Q}$ can be measured from previous j points by applying Algorithm 4.1. For example, there are 16 possible values of local $\bar{Q}$ when $j = 5$, as shown in Fig. 4.12. For each reference point R_i, the probability can be found for a generated data set with length j has a local Markov transition matrix $\bar{Q}$, which is different from one another. For each local $\bar{Q}_i$, there exists a reference point R_i such that

$$R(\bar{Q}_i) = \max_j \mathbb{P}(\bar{Q}_i | R_j) \tag{4.69}$$

and is has the maximum likelihood. Normally, $\bar{Q}_i$ and R_i are nearby points on the Q plane, as shown in Fig. 4.13. A switching mechanism according to the maximum likelihood relationship is shown in Algorithm 4.2.

Consider the plant P and controller K_c in Eq. 4.47 and 4.48 with estimators MFLSF(2,1) and MFLSF(1,0), a local Q estimation with a set of 2×2 reference points and $j = 5$ is applied. System performance for local Q estimation compared with static MFLSF estimations is listed in Table 4.7. Clearly, the proposed algorithm possesses benefits from both estimator MFLSF(2,1) and MFLSF(1,0).

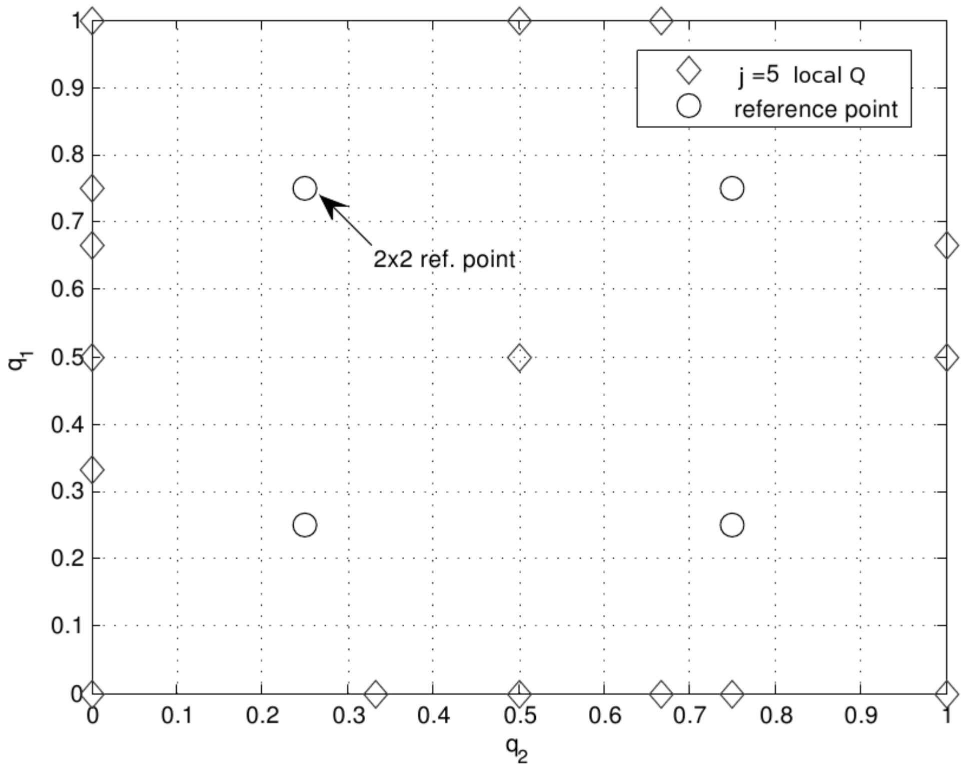


Figure 4.12: 2×2 reference points and the values of local $\bar{Q}$ with $j = 5$.

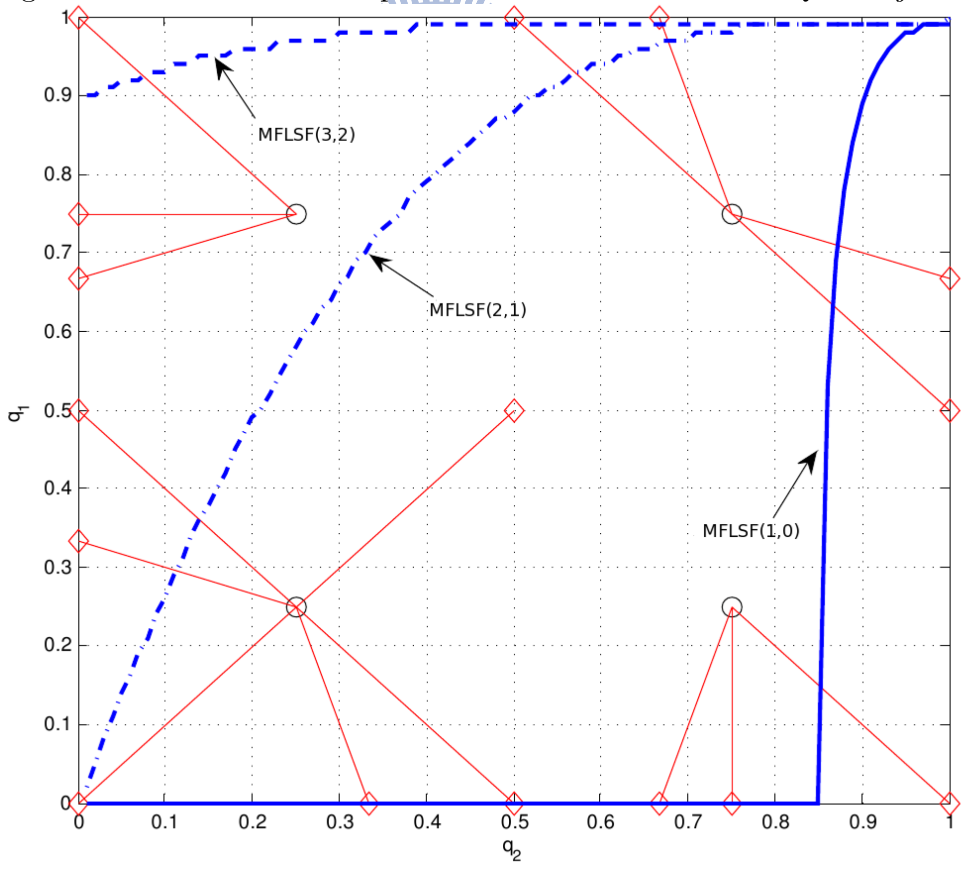


Figure 4.13: Local $\bar{Q}$ mapped into different reference points according to the maximum likelihood.

Algorithm 4.2 MFLSF estimation based on local $\bar{Q}$.

```
1. Calculate the stable region for each MFLSF estimator candidate to be switched.
2. Put $h \times h$ reference points $R_i$ uniformly on $Q$ plane and define the length $j$ for local $\bar{Q}$ estimation. For each $R_i$, there is a MFLSF estimator with the best tracking performance.
3. From the maximum likelihood relationship stated in Eq. (4.69), each $\bar{Q}$ is assigned a MFLSF estimator according to the reference point $R_j$, as shown in Fig. 4.14 and Fig. 4.13.
4. When dropout occurs, estimate local $\bar{Q}$ value from previous $j$-point dropout sequence.
5. Switch current estimator to a MFLSF estimator based on the estimated local $\bar{Q}$.
```

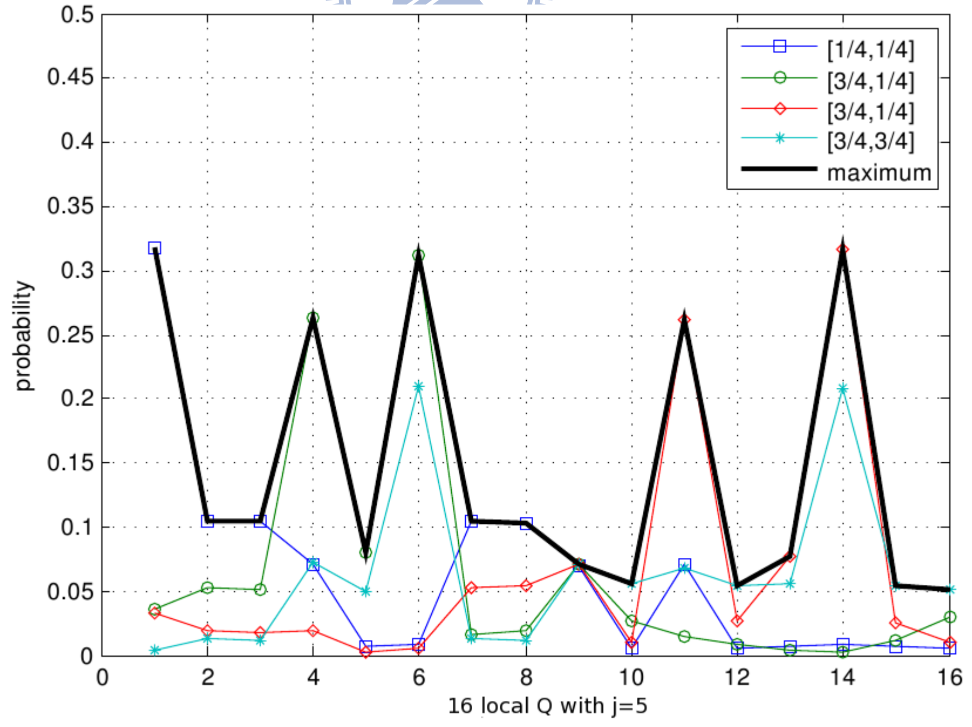


Figure 4.14: The probability values for each local $\bar{Q}$ from 4 different reference points.

Table 4.7: Tracking performance for static MFLSF estimation and MFLSF with local Q estimation with (a)$[q_1, q_2] = [0.7, 0.2]$ (b)$[q_1, q_2] = [0.4, 0.2]$, and (c) $[q_1, q_2] = [0.7, 0.4]$.

(a)

[0.7,0.2]	MFLSF(1,0)	local Q	MFLSF(2,1)
$\mathrm{RMS}_{\times 10^3}$	4.34	2.54	0.54
$\max \lvert\Delta y\rvert_{\times 10^2}$	7.36	9.13	13.02

(b)

[0.4,0.2]	MFLSF(1,0)	local Q	MFLSF(2,1)
$\mathrm{RMS}_{\times 10^3}$	6.19	5.15	1.59
$\max \lvert\Delta y\rvert_{\times 10^2}$	9.13	13.49	327.89

(c)

[0.7,0.4]	MFLSF(1,0)	local Q	MFLSF(2,1)
$\mathrm{RMS}_{\times 10^3}$	8.06	5.73	2.00
$\max \lvert\Delta y\rvert_{\times 10^2}$	19.23	19.17	137.88

4.6 Summary

In this chapter, the following results can be summarized as:

1. Two types of estimators MBLSF and MFLSF for the network dropout are proposed. With the network modeled by a two-state Markov chain, stability criteria for each estimator is also determined. Simulations with different Markov transition matrix Q verify the present results.

2. Simulations for the network dropout with different estimators are performed, and analysis of the tracking performances for estimators with different parameters M, N, and W are also provided. An MFLSF estimator switching mechanism based on local Q is proposed that it can maintain the tracking performances even under different network traffic situations.

Chapter 5 Conclusion

5.1 Conclusion

In this thesis, networked control systems under three different circumstances are discussed as follows:

1. A general method based on optimization process is proposed for PI controller auto-tuning. Once the bandwidth requirement is specified, control parameters can be automatically updated and finally being tuned up with the set bandwidth. Several simulation and experimental results have verified the usefulness and repeatability of the proposed algorithm for NCS.

2. Also, rules of H_∞ algorithm for NCS design are proposed. By considering the tolerated time delay, the proposed H_∞ design method has an advantage that several system performance requirements can be transformed into weighting functions and are used in the mix synthesis framework. A control structure with forward compensator to avoid delay tolerance and bandwidth trade-off is proposed to satisfy both requirements.

3. Finally, network dropout estimation methods in both controller and plant side of the network are proposed in this thesis. Two estimators MFLSF(M,N) and MBLSF(W) with criteria provided. The MBLSF estimator is almost guaranteed to be stable and with nearly zero tracking error due to its model-based nature. Different MFLSF(M, N) works with varied performance under differently Markov transition matrix Q. Moreover, a switching mechanism is proposed to reduce both RMS error and the maximum tracking error simultaneously.

5.2 Future Work

Although different methods are proposed to maintain system stability and performance requirements under different network situations, an overall structure that combines all of them is not provided. In practical situations, this should be implemented with the ability that different methods can be switched actively when the network situation changes. To do so, a general yet effective way to detect the current network situation is also needed, which is hard to achieve for different types of network.

References

[1] R. A. Gupta and M.-Y. Chow, "Networked control system: Overview and research trends," *Industrial Electronics, IEEE Transactions on*, vol. 57, pp. 2527–2535, July 2010.

[2] J. Baillieul and P. Antsaklis, "Control and communication challenges in networked real-time systems," *Proceedings of the IEEE*, vol. 95, pp. 9–28, January 2007.

[3] W. K. Ho, C. C. Hang, and L. S. Cao, "Tuning of PID controllers based on gain and phase margin specifications," *Automatica*, vol. 31, no. 3, pp. 497–502, 1995.

[4] Q.-G. Wang, H.-W. Fung, and Y. Zhang, "PID tuning with exact gain and phase margins," *ISA Transactions*, vol. 38, no. 3, pp. 243–249, 1999.

[5] R. Dhaouadi, Y. Al-Assaf, and W. Hassouneh, "A self tuning PID controller using wavelet networks," in *Power Electronics Specialists Conference, 2008. PESC 2008. IEEE*, pp. 773–777, 2008.

[6] A. Jones and P. de Moura Oliveira, "Auto-tuning of PI Smith predictor controllers using genetic algorithms," in *Control '96, UKACC International Conference on (Conf. Publ. No. 427)*, vol. 1, pp. 454–459, September 2-5 1996.

[7] M.-S. Wang, T.-C. Shau, and C.-M. Chang, "Dsp-based auto-tuning design of permanent magnet synchronous motor drives," in *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, pp. 1044–1048, November 5-8 2007.

[8] H. Li, Z. Sun, M.-Y. Chow, and B. Chen, "A simple state feedback controller design method of networked control systems with time delay and packet dropout," in *Proceedings of IFAC WC 2008 (the 17th IFAC World Congress)*, July 6-11 2008.

[9] X. Guan, S. Dai, and C. Long, "Controller design for networked control system with data packet dropout and transmission delays," *Journal of Control Theory and Applications*, vol. 5, pp. 227–232, 2007.

[10] N. Vatanski, J.-P. Georges, C. Aubrun, and E. Rondeau, "Networked control with delay measurement and estimation," *Control Engineering Practice*, vol. 17, no. 2, pp. 231–244, 2009.

[11] D. Yue, Q.-L. Han, and J. Lam, "Network-based robust H_∞ control of systems with uncertainty," *Automatica*, vol. 41, no. 6, pp. 999–1007, 2005.

[12] Y.-L. Wang and G.-H. Yang, "Output tracking control for continuous-time networked control systems with communication constraints," in *American Control Conference, 2009. ACC '09.*, pp. 531–536, June 10-12 2009.

[13] M. Roozbehani and C. Knospe, "Robust stability and H_∞ performance analysis of interval-dependent time delay systems," in *American Control Conference, 2005. Proceedings of the 2005*, vol. 6, pp. 4265–4270, June 8-10 2005.

[14] K. Ling and K. Lim, "Receding horizon recursive state estimation," *Automatic Control, IEEE Transactions on*, vol. 44, pp. 1750–1753, September 1999.

[15] R. Brown, S. Schneider, and M. Mulligan, "Analysis of algorithms for velocity estimation from discrete position versus time data," *Industrial Electronics, IEEE Transactions on*, vol. 39, pp. 11–19, February 1992.

[16] X. Zhang, Y. Zheng, and G. Lu, "Stochastic stability of networked control systems with network-induced delay and data dropout," in *Decision and Control, 2006 45th IEEE Conference on*, pp. 5006–5011, December 13-15 2006.

[17] C.-W. Cheng, "Analysis of the dynamic network control system with the time-delay effect," Master's thesis, National Chao Tung University, 2006.

[18] C.-W. Liu, "Precision motion control, auto-tuning, and remote monitoring of CNC servo motors," Master's thesis, National Chao Tung University, 2008.

[19] K. Zhou and J. C. Doyle, *Essentials of Robust Control.* Prentice Hall, 1998.

[20] J. Nilsson, "Real-time control systems with delays," 1998.