NETWORK-ENABLED MONITORING AND STABLE CONTROL OF DYNAMIC SYSTEMS

by

POI LOON TANG

B. Eng. (Hons.), Mechanical Engineering, University of Malaya, 1998
M. Eng., Mechanical Engineering, The National University of Singapore, 2001

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

(Mechanical Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA
July 2005
© Poi Loon Tang, 2005

Abstract

The integration of computers, communication, and control in modern distributed systems such as intelligent transit systems and industrial facilities has garnered a great deal of attention in recent years. This thesis presents an investigation into basic research, technology development, and implementation of monitoring and control strategies for networked-enabled dynamical systems. In view of the hierarchical nature of these systems, the thesis focuses on two important system layers; namely, networked control systems within the direct-control (servo) layer, and remote monitoring and supervisory control within the supervisory-layer.

In the present work, a networked control system (NCS) comprises the traditional components of a control system such as a plant, sensors, actuators, controllers, and signal conditioning and modification devices, but may be geographically distributed and linked through a communication network. In particular, in an NCS, the traditional feedback control loops are closed through a real-time communication network. A primary objective of the present thesis is to investigate, analyze, develop, implement and test stable control strategies to overcome the transmission problems between sensors, actuators, and controllers in an NCS; specifically, non-deterministic delay, losses, vacant sampling, and mis-synchronization of data/information. The term networked control is used in this thesis to mean the control of a networked system, and a networked system with a controller as an integral part is an NCS.

In this thesis, a new networked control strategy is developed which takes advantage of the potential capability of constrained Model Predictive Control (MPC) to compensate for anticipated data transmission problems. The constructive and computationally inexpensive strategy, which is developed here, uses predictions of future control actions and estimations of possibly erroneous sensory signals, to maintain good performance and stability of a closed-loop NCS, through an innovative method of information buffering. Network and computational loads are effectively minimized by incorporating variable prediction horizons. Performance is further improved through adaptive weight sequencing enhancements.

The analytical issues of system stability of the new networked control strategy are rigorously treated in the thesis. The analysis is carried out in two stages. In the first stage, prefect state measurement is assumed to develop the necessary and sufficient conditions for guaranteeing closed-loop asymptotic stability while utilizing predicted future control actions

that are suboptimal. Stability is achieved in the sense of Lyapunov by imposing a terminal cost and bounding it without end constraints, and projecting the instantaneous states to the terminal states. Feasibility, which in turn implies stability, is proven in the context of this suboptimal control strategy. The stability surfaces generated from the resulting theorems are particularly useful as design guidelines to determine the effective worst-case delay that can be sustained by the networked control strategy. The second stage of stability analysis assumes imperfect state measurement with the purpose of determining the amount of deviation between the actual states and the estimated states that can be sustained in stable manner by the developed networked control strategy. This is accomplished by establishing bounds for the evolution of future inputs and states, employing the concept of perturbation sensitivity in nonlinear programming, and based on Lyapunov's second method. The theorems that result from the second analytical stage serve as a precise criterion in the design of extended state observers for compensating possible transmission problems between sensors and controllers of a networked system.

The functionality of a networked control system may be further enhanced by integrating the capability of remote monitoring and supervisory control into the upper layers of the hierarchy of the system. As a contribution in this direction, a framework for developing a universal and scalable network infrastructure for web-based monitoring and supervisory control of dynamic systems is developed in the present work. Practical implementation of this framework is detailed, using a low cost and flexible two-tier client-server architecture, where a user is able to remotely carry out a variety of tasks including system operation, experimentation, process monitoring and supervision, task scheduling, system reconfiguration and tuning, control, and safety/emergency routines, through a web-browser interface. A single web-server provides smooth information flow using a robust and intelligent scheduling scheme. The flexibility and modularity of the developed network architecture provide the rationale for further incorporation of a multi-layered intelligent supervisory control structure with the objective of on-line improvement of the performance of a process. A remote supervisor, which incorporates knowledge-based decision making, continuously monitors the performance of the process to infer the best adaptive controller (and the best tuning actions, if needed) for the process under the existing conditions.

The strategies of networked-enabled monitoring and feedback control, as developed in the present work have been implemented on an industrial fish-processing machine using an Ethernet communication network. The developed and analyzed technologies are thoroughly evaluated and validated through experimentation under a variety of operating conditions in real-time, and are demonstrated to be practical, beneficial, and technically sound.

Table of Contents

List of Tables

List of Figures

Nomenclature

Notations

$\mathbb{R}$	set of real numbers
$\mathbb{R}^n$	vector space of n-tuples over $\mathbb{R}$
$\|\boldsymbol{x}\|$	Euclidean norm of vector $\boldsymbol{x} \in \mathbb{R}^n$ (equivalent to $\|\boldsymbol{x}\|_2^2$ or $\sqrt{x_1^2 + \cdots + x_n^2}$)
$\|\boldsymbol{x}\|_{\boldsymbol{P}}^2$	norm of $\boldsymbol{x} \in \mathbb{R}^n$ with weight $\boldsymbol{P}$ (equivalent to $\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x}$ or $\|\boldsymbol{P}\boldsymbol{x}\|_2$ in literature)
$\lceil \cdot \rceil$	nearest integer towards positive infinity
$\lfloor \cdot \rfloor$	nearest integer towards negative infinity
a_{fi}, b_{fi}	denominator and numerator coefficients of a Butterworth filter
a_i, b_i	denominator and numerator coefficients of a transfer function model
$\boldsymbol{A}, \boldsymbol{B}$	state-space representation of a system
$\boldsymbol{c}(\cdot)$	vector of constraint functions
c	general constant; intercept of curve
$\hat{c}$	estimate of the clock intercept
$\overline{c}$	scaling constant
C_C	control client node
C_S	control server node
$\mathrm{E}[\cdot]$	expectation (probabilistic) or mean value
f_s	sampling frequency
$G_{uy}(z^{-1})$	discrete-time transfer function from u to y
h	sampling period
H	general prediction horizon of a predictive controller
H_1	minimum prediction horizon of a predictive controller

H_2	maximum prediction horizon of a predictive controller
H_u	control horizon of a predictive controller
I_a	dead-band compensation constant
$\boldsymbol{I}_n$	identity matrix of size $n \times n$
k	discrete-time step
K_B	bounding gain of cost evolution
K_e	upper bounding gain of state estimation error
K_z	bounding gain of input-state parametric evolution
K_P	proportional gain
K_I	integral gain
K_D	derivative gain
m	number of system inputs; gradient of a curve
$\hat{m}$	estimate of clock gradient
M_p	maximum overshoot of a step response
n	number of system states; number counter
N	total number of elements in a set or series
p	number of system outputs
P_h	head-side cylinder pressure
P_r	rod-side cylinder pressure
$\boldsymbol{P}$	terminal weight matrix of a predictive controller
$\boldsymbol{P}_0$	modified terminal weight matrix of a predictive controller
$\boldsymbol{Q}$	state or error weighting matrix of a predictive controller
r, R	reference input (set-point)
$\boldsymbol{R}$	input weighting matrix of a predictive controller
t	continuous time
t_{abs}	absolute clock time
t_a^C	time-stamps at event a at node C.
t_p	time at the first peak of a step response

t_r	rise-time of a step response
t_s	settling-time of a step response
T_s	sampling interval
T_v	velocity filter time constant
$\boldsymbol{u}$	input vector of a system
$\boldsymbol{U}_\infty$	open-loop infinite weight matrix satisfying a discrete Lyapunov equation
$V_a(\cdot)$	MPC cost function evaluated over a prediction horizon of length a
$\boldsymbol{x}$	state vector of a system
y	piston position
$\dot{y}$	piston velocity
y_m, Y_m	response of reference model
y_s, Y_s	actual measured response of a system
δ^{CC}, δ^{CS}	clock skew
ε	tracking error, estimation error
λ	vector of Lagrange multipliers
$\overline{\lambda}(\cdot)$	maximum eigenvalue of a matrix
μ	mean message transmission time
$\hat{\mu}$	estimate of mean message transmission time
$\boldsymbol{\pi}_k$	series (set) of control input signals
$\boldsymbol{\Pi}_H$	control input vector over a prediction horizon of length H
ρ^{CC}, ρ^{CS}	clock drift
τ_{ca}^k	controller-to-actuator message transmission delay at time step k
$\tau_{\min}$	minimum message transmission time/delay
$\tau_{\max}$	maximum message transmission time/delay
τ_{rtt}	round-trip message transmission time/delay
τ_{sc}^k	sensor-to-controller message transmission delay at time step k
ω_d	damped natural frequency

ω_n	undamped natural frequency
ζ	damping ratio

Abbreviations

CAN	Control Area Network
CARIMA	Controlled Autoregressive Integrated Moving Average
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
GPC	Generalized Predictive Control
IP	Internet Protocol
ITAE	Integral Time Absolute Error
LAN	Local Area Network
LMI	Linear Matrix Inequality
MPC	Model Predictive Control
mpQP	multi-parametric Quadratic Programming
MTU	Maximum Transfer Unit
NCS	Networked Control System(s)
NIC	Network Interface Card
NTP	Network Time Protocol
PRBS	Pseudo-Random Binary Sequence
QP	Quadratic Programming
TCP	Transport Control Protocol
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
WAN	Wide Area Network

Acknowledgment

I wish to express my sincere appreciation to my supervisor, Prof. Clarence W. de Silva for his invaluable guidance, persistent advice and strong support throughout the entire process of my graduate studies. I thank him for taking an active interest in my work, his attentiveness and patience in editing all my publications, and continuously giving kind encouragement to keep me motivated. I am greatly indebted for the countless opportunities and career building experiences he had given me. These include, to name a few, offering to supervise me, funding me to attend conferences and workshops, recommendation for scholarships, and providing the opportunity to take a leadership role in organizing a conference. The aforementioned words are simply insufficient to justify the amount of time and effort Prof. de Silva has unconditionally spent on me.

I would also like to thank my previous Master's research supervisor, Prof. Aun-Neow Poo of the National University of Singapore, for his constant support and for introducing me to and arranging my prior research attachment with Prof. de Silva.

It has been a great honor to have Prof. Madan M. Gupta from the University of Saskatchewan, as the external examiner of my doctoral thesis. I would also like to express my gratitude to, Prof. Victor C.M. Leung and Prof. Mohamed S. Gadala, who have kindly served as my university examiners. My cordial appreciation goes to the members of my research committee, Prof. Elizabeth A. Croft and Prof. Farrokh Sassani, for their constructive comments and suggestions. This thesis would not have attained its final state without the positive feedback and invaluable suggestions I received from other faculty members, particularly the late Prof. Dale B. Cherchas and late Prof. Vinod J. Modi, who were members of the research committee, and Prof. Yusuf Altintas and Prof. Ian Yellowley, during our annual graduate research seminars. Their kind support is gratefully acknowledged.

Generous financial support during the course of my research has been provided through various sources. These include the various scholarships from the University of British Columbia in the form of University Graduate Fellowships (UGF), Ph.D. Tuition Fee Awards, and International Partial Tuition Scholarships; research assistantships and equipment funding from the Natural Sciences and Engineering Research Council (NSERC) of Canada and the

Canada Foundation for Innovation (CFI) through Prof. de Silva as the Principal Investigator; and supplemental stipend during the initial phase of the research from the National Research Council of Canada – Institute for Fuel Cell Innovation through the Research Officer, Dr. George X. Wang.

The kind assistance from the department staff; particularly, Ms. Lanna Lok, Ms. Sheilla Dagta, Ms. Chotivan Manglanond, Ms. Jan Marsden, Mr. Dan Miner, Ms. Barb Murray, Mr. Alan Steeves, Mr. Gord Wright, and Mr. Perry Yabuno, is kindly appreciated.

Thanks also go to Prof. Elizabeth Croft and her students; namely, Dr. Daniela Constantinescu, Ms. Dana Kulić, Mr. William Owen, and Mr. Tao Sang, for the morning meetings, paper discussions, and especially the constructive criticisms on my seminar dry runs that have greatly improved my presentation style and skills.

I would like to take this opportunity to thank my research group members, close colleagues and friends; especially, Dr. Yang Cao, Mr. Tao Fan, Mr. Richard McCourt, Mr. Ying Wang, Mr. Duminda Wijewardene, Mr. Kenneth Wong, and Mr. Jason J. Zhang, for extending their help whenever they were needed, and making my journey at the University of British Columbia a very memorable and enjoyable one.

As a personal note, I would like to thank my family for their constant support. Above all, I wish to express my exceptional appreciation to my mother, for her continuous love, support, and encouragement throughout my life. *To her, I dedicate this thesis.*

Chapter 1
Introduction

The maturity, robustness, and wide availability of the technology of high-speed networks and communication, particularly the Internet, have many positive implications. For example, these features have propelled the recent advances in the manufacturing environment, such as Flexible Manufacturing Systems (FMS), and Reconfigurable Manufacturing Systems (RMS) (Abdullah and Chatwin, 1994); notably the integration of computer, communication and control into different levels of factory automation and information processing. For example, an integrated system of workcells might include processing tools (e.g., manipulators, grippers, and positioners) with associated component controllers, supervisory controllers, tasks allocators, and intelligent monitoring devices. The communication medium is the backbone of such an advanced manufacturing environment. A modern and automated factory environment has to be efficient, flexible, modular, and reliable, with fast and convenient accessibility and exchange of information through a common networked communication medium. A unified network architecture provides the attractive features of easy installation, maintenance and reconfiguration of various entities of a plant, in addition to reducing the setup and maintenance costs. In particular, it has the advantage of reducing system wiring and maintenance costs by coexisting with an office network. Furthermore, by incorporating additional tools for fault sensing, detection and diagnosis, the health of the system can be monitored and resolved (Lian, *et al.*, 2001, Tang, *et al.*, 2002) from various locations of the network. Hazardous production facilities can be insolated from densely populated areas. Furthermore, engineers, managers, operators, and consultants can work together in a coordinated, interactive, and efficient manner without being physically present at one location.

Modern large-scale industrial systems such as high-speed paper production machines, power generation plants, food processing and packaging operations, and petrochemical processing facilities consist of an array of distributed sensors, actuators, and controllers, which are interconnected through a common network medium or bus. There is an emerging need for

distributed and smart sensors and actuators with network communication capability as well as wireless access to reduce expensive and tedious equipment wiring. Figure 1.1 shows a rather generic representation of a factory with two systems, both having a series of actuators and sensors connected to the factory network. This type of control systems falls into the class of Networked Control Systems (NCS) where control loops are closed through a real-time communication network. Specifically, an NCS is a system consisting of the traditional components of a control system such as a plant, sensors, actuators, controllers, and signal conditioning and interfacing hardware and software, which are all connected through a common communication network. It follows that a primary defining feature of an NCS is that information; e.g., reference inputs, plant outputs, and control inputs, is exchanged among the control system components (sensors, controllers, actuators, etc.) using a common network-communication medium. There is also an emerging interest in implementing remote feedback control over the Internet. For example, as shown in Figure 1.2, additional redundancy in safety-critical or fault-tolerant systems can be achieved by using network-based redundant controllers to keep a system running, should the local controllers malfunction.

Different types of control networks such as the CAN (Controller Area Network), FieldBus, Profibus, LonWorks and Modbus have been around for over two decades. Of late, there has

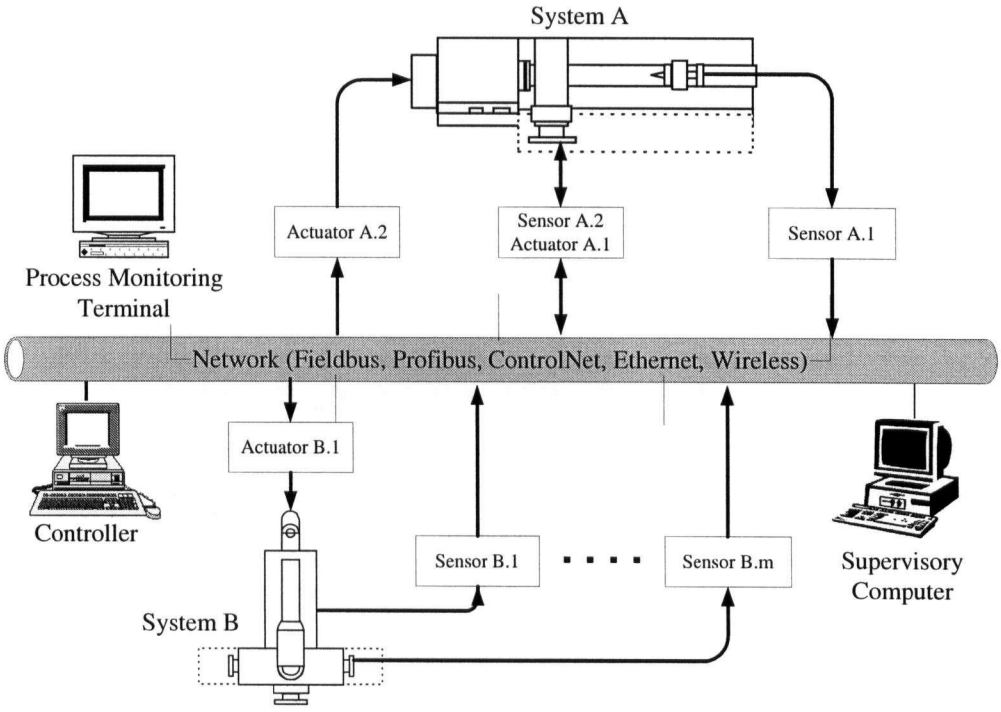


Figure 1.1: A generic factory network.

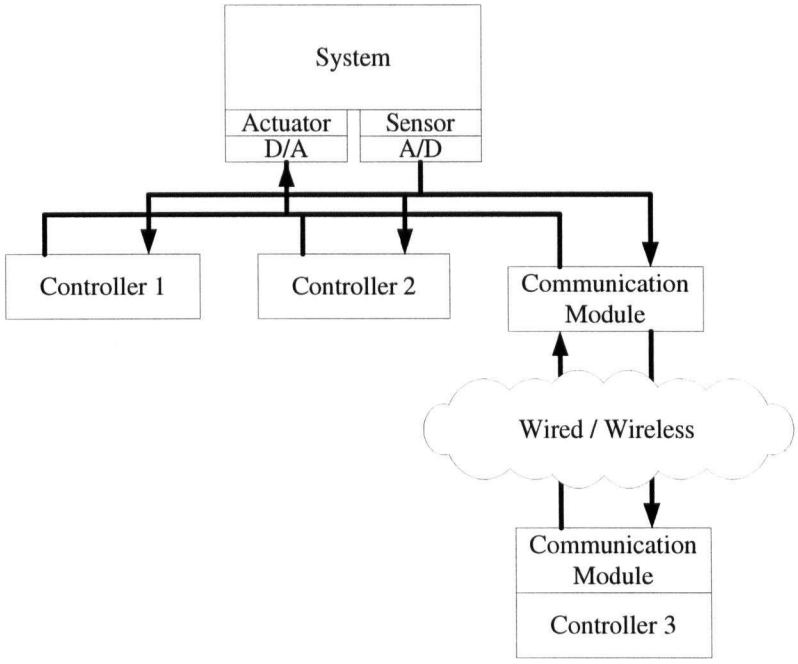


Figure 1.2: Additional redundancy in a fault-tolerant system.

been much interest in using Ethernet as a control network. Ethernet is not originally designed for real-time control. However, the popularity and versatility of the Internet has resulted in a steady development and improvement of Ethernet technology. As a result, Ethernet is fast becoming an attractive option to replace other control networks that are low bandwidth, high cost, propriety and aging. Besides, Ethernet is cost effective and widely deployed. It can share the same office network, and the same communication medium can be used for factory floor monitoring, diagnostics, and implementing feedback control loops.

While it has well-known advantages, implementing feedback control over Ethernet network, as with other control networks, presents the fundamental challenge of overcoming the unavoidable delays in data transmission between the various distributed NCS devices, due to the limited bandwidth and overhead in the communicating nodes and in the network itself. The performance, stability in particular, of a control system can be significantly degraded by the presence of communication delays. Furthermore, the packet-switching nature of the Ethernet causes the delay to be non-deterministic. There are also the frequent occurrence of data loss, vacant sampling, and mis-synchronization of sampled data depending on the type of transport protocol used in the data communication.

1.1 Goals of the Research

From the perspective of feedback control, the main challenge in the development of an NCS would be to overcome the adverse effects of inevitable time delays that incur by the communication network. The network-induced delays are time-varying and possibly stochastic. A control system with such varying and unpredictable delays will no longer be amenable to techniques of deterministic and time-invariant systems.

The main objective of the research reported in this thesis is not to redesign the communication protocol for reducing the transmission delay. Instead, it is to treat the network protocol and traffic as given conditions, and formulate robust feedback control strategies to realize good performance in a high bandwidth distributed NCS, by explicitly taking into account the crucial time delays and other network-control issues. The design and critical analysis, particularly the conditions for maintaining asymptotic stability, of feedback control strategies for NCS will be given primary attention in the thesis. The issues of NCS architecture and infrastructure development for practical implementation and experimental investigation also will be treated. A parallel objective of this study is to investigate the suitability of the Ethernet network, which is cost effective and widely deployed, for implementing networked control systems. Real-time experimental benchmarking of the developed NCS strategies will be carried out on an industrial application within an Ethernet network.

Within the theme of NCS, a parallel objective of the thesis is the development of a universal, reconfigurable and scalable network architecture, both hardware and software, for web-based remote monitoring and supervisory control of dynamic systems such as industrial plants, research facilities, and academic laboratories. This will be followed by further incorporation of a multi-layered intelligent supervisory control structure with the objective of on-line improvement of the performance of a system. A full-scale implementation of the developed approach will be made on the industrial application used for benchmarking the NCS.

1.2 Scope of the Study

The scope of the present study includes two main areas: networked control systems (NCS), and remote monitoring and supervisory control. In the area of NCS, a novel control solution is developed for a networked system, based on the potential framework of unconstrained and constrained Model Predictive Control (MPC). The idea is to render the dynamics of network

transmission "transparent" by incorporating a carefully devised strategy of pre-computing, buffering and sequencing the future control efforts at the actuators in order to anticipate for any transmission problems in the data streams between controllers and actuators. Compensation for transmission "errors" in the sensory data is achieved through a modified form of state or output estimators. Various enhancements to the basic framework of the developed NCS-MPC strategy is investigated and implemented. These enhancements are meant to directly or indirectly reduce the network congestion, reduce the computation load of the controller, adapt to time varying dynamics of the system, and increase the operating range of the control system.

Closed-loop stability and optimality of the developed NCS-MPC buffering and estimation strategy are given rigorous treatment in this study. The approach is to divide the stability analysis into two stages. The first stage assumes perfect state measurements of the estimated sensory data and focuses on establishing stability conditions for the buffering future control efforts within the MPC policy. Global closed-loop asymptotic stability in the sense of Lyapunov is guaranteed for the controller, by bounding the projected receding horizon costs by lower- and upper-bounding terms using a predetermined terminal cost. Stability theorems are developed, which provide a suboptimal measure for the controller in real-time, which is sufficient to estimate the worst-case transmission delay that can be handled by the developed control buffering strategy. The second analytical stage assumes imperfect state measurement of sensory data, which results in a robust stability problem with state mismatch. The problem then reduces to the following: given the set of parameters for a stable MPC policy as obtained from the first stage, determine the upper bound of the norm of the state estimation error (between the actual and the estimated states) for further maintaining the asymptotic stability of the system.

Within the context of realizing the Ethernet network as a platform for NCS, a new client-server NCS architecture is developed around a dual-axis electro-hydraulic manipulation system of an industrial fish-processing machine. This development includes the hardware setup; client-server communication protocols; multi-process and shared memory area communication and scheduling; and implementation of a robust clock synchronization algorithm. The real-time implementation of MPC on the servo systems of an industrial machine requires a fast quadratic optimization algorithm. Here, an off-line multi-parametric Quadratic Programming (mpQP) method is adopted. This requires a state-space partitioning algorithm and a binary search tree construction algorithm. Utilizing the networked client-server control architecture as developed here, real-time experimental evaluation of the developed NCS-MPC buffering and estimation

strategy is carried out with regard to tracking performance, effectiveness in handling various transmission problems, and validation of the stability theorems established in the research.

In the second closely-related research area of remote monitoring and supervisory control, a framework for developing a universal and scalable network infrastructure for web-based monitoring and supervisory control of dynamic systems is investigated and developed. The developed infrastructure employs a low cost and flexible (reconfigurable) two-tier client-server architecture where a user is able to remotely carry out a variety of tasks including system operation, experimentation, process monitoring and supervision, task scheduling, system reconfiguration and tuning, control, and safety/emergency routines, through a web-browser interface. A single web-server together with a mediating server provides smooth information flow using a robust and intelligent scheduling scheme. The flexibility and modularity of the developed network architecture form the rationale for further incorporation into it a multi-layered intelligent supervisory control structure with the objective of on-line improvement of the performance of a process. A remote supervisor, which incorporates knowledge-based decision making, continuously monitors the performance of the process to infer the best adaptive controller for the process under existing conditions. Full-scale implementation of the developed approach is made on the same industrial fish-processing machine as before, and is used to demonstrate the application of the developed technologies in an industrial environment for process monitoring and supervisory control.

1.3 Related Work

Four main research aspects have to be investigated in order to achieve the objectives of the thesis. They are: networked control systems (NCS), model predictive control (MPC), stability of model predictive control, and remote monitoring and supervisory control. The following sections survey some pertinent work that has been carried out in the past in these areas.

1.3.1 Networked Control Systems

The research of NCS may be considered to focus in two general directions. One direction involves the development of robust communication protocols to ensure constant delay or minimum jitter in the data stream (Farsi and Ratcliff, 1998, Tovar and Vasques, 1999, Xiao, *et al.*, 2001, Cena and Valenzano, 2002, Almeida, *et al.*, 2002, Conti, *et al.*, 2002). Under these

conditions, a controller can be designed without considering the dynamics of the network. The second direction in NCS research assumes the particular communication protocol as given, without any modification, and focuses on the development of feedback control methodologies to realize good performance. This latter direction is taken in the present research.

Halevi and Ray (1988) have considered a continuous-time plant and discrete-time controller to analyze an integrated communication and control system (ICCS) using a discrete-time approach. They have studied a clock-driven controller with mis-synchronization between the plant and the controller. The system was represented by an augmented state vector, consisting of past values of the plant and the controller. This has resulted in a less complex finite-dimensional, time-varying discrete time model as compared to a continuous model with time-varying delays. However, their work is limited to transmission delays that are smaller than the sampling period. Luck and Ray (1990) made a closed-loop NCS time-invariant by introducing buffers at the controller and actuator inputs. A linear time-invariant model was derived by setting the buffer size longer than the worst-case delay. An advantage of this method is that it handles transmission delays that are longer than the sampling interval of the controller. However, the buffering process requires unnecessarily large data packets, which increase the network traffic, and in turn causes longer delays than necessary. The most effective way to minimize the effect of time delay on the performance of an NCS is to reduce network traffic (Otanez, *et al.*, 2002b). Zhang, *et al.* (2001) looked into the stability of an NCS using the concepts of stability regions and hybrid systems. System models with packet dropout and multiple packet transmission were developed as asynchronous dynamical systems. The stability of an NCS with packet losses was analyzed by Azimi-Sadjadi (2003) but a delay-free network was assumed, which in fact defeated the purpose of the analysis. Xiao, *et al.* (2000) proposed to model an NCS as a "stochastic hybrid system" where the random communication delays would be modeled as finite-state Markov chains. Although the method is promising in that it can handle randomness in the delay, it has the limitation of assuming narrowly-bounded delay. The bound on the probability of packet losses was not addressed.

In order to minimize the effect of transmission delays on an NCS, several ideas have been explored with the objective of reducing or optimizing the network traffic. A common approach in this regard is to introduce estimators or predictors at the communication nodes. This is a form of data queuing, as proposed for example by Chan and Özgüner (1995), where queues are located at the sensor output and the controller input. A probabilistic state predictor is located

before the controller to estimate delayed sensor data. Otanez, *et al.* (2002a) devised a procedure to effectively reduce network traffic, where an adjustable deadband on the rate of change of a measured signal was used to determine the frequency of data transmission, while employing a simple controller such as proportional-integral (PI) or proportional-integral-derivative (PID), and a deadbeat compensator. However, the method unnecessarily introduces state uncertainty to the system, thereby reducing the stability boundary as well as the operating bandwidth of the system. It is also is not suitable, however, for high bandwidth systems where the dynamics can change quite rapidly. A similar approach has been investigated by Yook, *et al.* (2000) where each communicating node is retrofitted with an estimator. In this method, the states of each node would only be broadcasted when the estimated states differed from the actual measured states by a pre-specified tolerance, thereby limiting the network utilization. In addition, the network delay was assumed to approach zero, which is somewhat unrealistic. Consequently, the performance degradation caused by network delays could not be overcome. Quevedo, *et al.* (2003) minimized the network bandwidth by using a finite-set constraint for all control inputs as well as supplying control increments on an as-needed basis. In order to improve the performance of an NCS, Beldiman, *et al.* (2000) developed two structures of predictor to estimate the system outputs between successive transmission intervals. However, the delay in control actions was not addressed. In Walsh, *et al.* (1999, 2002), a continuous plant and a continuous controller have been considered. The control network, shared by other nodes, has been inserted between only the sensor nodes and the controller. They have introduced the notion of maximum allowable transfer interval (MATI), which supposes that successive sensor messages are separated by at most the MATI. The goal is to find that value of MATI, which would guarantee the desired performance and stability of the NCS.

Considerable amount of work has been done within the framework of optimal control, where a specific cost function is minimized. Nilsson, *et al.* (1998a, 1998b) proposed an optimal stochastic control strategy for networks with random delay. A few delay scenarios were studied, including both deterministic and stochastic transmission. It was found that by incorporating time-driven sampling together with an event-driven controller and actuator, a Linear Quadratic Gaussian (LQG) controller could guarantee system stability, as long as the network delay was shorter than one sampling interval. The limitation to cases where the network delay is within one sampling interval renders the method less useful. Lian, *et al.* (2002a) designed an optimal controller for an NCS that was capable of compensating for

multiple time delays. The key was to use a delay transformation procedure, which mapped the NCS model to a standard system model with delayed states. Sinopoli, *et al.* (2004a, 2004b) in a sequence of work introduced the idea of discrete-time Kalman filtering with intermittent observation in order to overcome data losses in a communication channel. The dependence of the expected estimation error covariance on the loss probability and the system dynamics was studied. In (Sinopoli, *et al.*, 2004a) it was shown that the separation principle (Khalil, 2002) of optimal control would hold in the presence of data losses. The analysis was done using an LQG problem in discrete time.

Some *ad-hoc* approaches have also been introduced. For example, Tipsuwan and Chow (2003) adopted a gain scheduling scheme on the controller parameters to maintain the system performance based on a Quality-of-Service (QoS) measurement, depending on the level of network traffic. This method is only able to handle small delay conditions and has a very limited range of operation. Furthermore, an additional middleware is required to monitor and measure the network QoS which may not be practical in some factory networks.

1.3.2 Model Predictive Control

Model Predictive Control (MPC) being able to incorporate various forms of constraints (on inputs, outputs, states, and so on), is applicable to multivariable control problems in their original form, and more importantly able to explicitly predict the system outputs or states at future time steps. These attractive features of MPC render it a viable control scheme for consideration in the current research and development of robust feedback control strategies for Networked control Systems (NCS).

The term MPC generally denotes a collection of controllers, which determine the control effort by minimizing a cost function (usually quadratic) in a receding horizon manner using an explicit model while satisfying some imposed constraints. Physical limitations of the system; e.g., valve saturations, may be represented by an input constraints. The state constraints are imposed for states or outputs that may not have set-points, but are required to remain within certain limits during the intended system operation. Early forms of MPC were developed for the industrial sector, specifically the chemical processing industry, with such names as Dynamic Matrix Control (Cutler and Ramaker, 1980), Extended Prediction Self-Adaptive Control (De Keyser and Van Cuawenberghe, 1985), and Generalized Predictive Control (GPC) (Clarke, *et al.*, 1987). The purpose of the present section is not to review the entire history of

MPC but to focus on the distinct contributions that have been made that are of particular relevance to the thesis, and more recent advancements, particularly on stability analysis.

Modifications have been made to the basic MPC with adaptive capability to cope with time-varying dynamics of a system. The transfer function variant of MPC; i.e., GPC[1] is distinctly suited for adaptive approaches. Bordons and Camacho (1998) reformulated the GPC algorithm to provide a direct relation between the parameters of GPC and the system to be controlled. Besides providing intuitive tuning means for plant operators, this formulation enables easy incorporation of higher level adaptive algorithms; e.g., model-referenced adaptive control (Åström and Wittenmark, 1995). However, it is only useful in systems that exhibit first-order-plus-dead-time dynamics. A similar approach of mapping the design parameters to the controller parameters has been proposed by Al-Ghazzawi, *et al.* (2001). This approach is more practical than the previous one in that the desired closed-loop performance in time-domain is directly related to the parameters of the controller. Using Laguerre and subspace identification, Huzmezan and Dumont (2000) replaced the norm of indirect adaptive algorithm of identifying the system parameters prior to control law redesign with fast least squares computation and simple algebraic manipulation. The resulting low dimensional formulations allow efficient implementation of direct adaptive MPC on high-order multivariable systems.

Alternative formulations based on Linear Matrix Inequality (LMI) (see Appendix A) have also been explored in order to deal with uncertainty in modeling that is lacking in typical Quadratic Programming (QP)-based MPC design. Kothare, *et al.* (1996) introduced a new technique to recast robustness properties of unconstrained and constrained MPC in the framework of LMI. This was done by employing the robust $\mathcal{H}_\infty$ control modeling paradigm (Skogestad and Postlethwaite, 1996) of either polytopic model or structured feedback uncertainty model. Robust stability was guaranteed by first assuming an upper bound for the MPC cost function with infinite prediction horizon prior to synthesizing the MPC control law. Kim, *et al.* (1998) proposed a similar approach but in the realm of GPC where the GPC problem was formulated as a min-max problem of minimizing the cost function while suppressing the maximum effect of disturbances. The stabilizing settings of the influential

[1] In this thesis, the standard term Generalized Predictive Control (GPC) is used to denote a variant of Model Predictive Control (MPC) where the formulation is done in the transfer function (s-domain) form (Camacho & Bordons, 1998). Unless stated otherwise, the GPC problem is generally unconstrained. On the other hand, MPC is the term that is commonly used for the "equivalent" state-space formulation and generally involves some form of constraints.

control parameters were obtained by solving a compact set of LMIs. In terms of computation speed, there is generally little or no gain in LMI-based optimization compared to QP-based optimization. Hence both approaches are limited to relatively low speed systems.

The digital control of high speed servo systems requires small sampling intervals. The use of traditional QP methods (Fletcher, 1987) to optimize the MPC cost function can be rather slow and besides feasibility of the solution may not be guaranteed, making these methods ineffective in high-speed servo implementations. Fast QP optimization algorithms have been developed. Chisci, *et al.* (1994) proposed an algorithm for fast online computation of the Generalized Predictive Control (GPC) optimization problem by simply transforming the GPC formulation to a condensed form using linear and hyperbolic polynomial rotation. Kouvaritakis, *et al.* (2002) proposed an alternative method of transforming the QP problem into an equivalent linear optimization problem using Newton-Raphson iteration, requiring only a fraction of the QP computation time. In this method, the convergence to a solution is guaranteed, and complexity of the problem only increases linearly with the number of constrained control moves. Significant amount of work has been done in multi-parametric Quadratic Programming (mpQP) where all the heavy computational load is brought off-line leaving simple functional evaluation during online optimization (Bemporad, *et al.*, 2000, 2001, 2002, Pistikopoulos, 2002, Tøndel, 2003). It is found that the space of the state variables can be partitioned into polyhedral regions where the optimal control effort of each region is given as a linear function of the state variables. This method is employed in the implementation of the constrained MPC optimization in the present thesis. Further investigation, enhancement, and issues encountered in the implementation of the algorithm are addressed in Chapter 4 and Appendix B.

Another popular trend in MPC research concerns nonlinear MPC. Developments in this realm can be quite valuable since practical systems are inherently nonlinear and since there is a fundamental need to operate a system over a broad range of operating conditions. Although a robust linear MPC scheme is capable of handling nonlinear systems, the existing robust linear MPC (Kothare, *et al.*, 1996) may be conservative or computationally intractable. In addition, nonlinear MPC has the potential of achieving better closed-loop performance. Findeisen, *et al.* (2003) looked into a different formulation of nonlinear MPC including output and state feedback, which allows the direct use of nonlinear models for prediction. Stability and robustness properties have also been established. Nevistić and Primbs (1996) extended the converse Hamilton-Jacobi-Bellman approach to create various categories of nonlinear systems

in which a deeper understanding into the properties of nonlinear MPC has been gained. Simulated evaluations and comparison have been carried out and potential issues have been explored on nonlinear MPC in comparison to MPC with feedback linearization. A quasi-infinite horizon nonlinear MPC has been proposed by Chen and Allgöwer (1998) where the input profile needs to be determined online only for a finite horizon. In this method, a terminal state penalty term has been added to the finite horizon MPC cost function to guarantee asymptotic stability.

1.3.3 Stability in Model Predictive Control

The slow adoption of MPC in the early years is mainly due to the lack of a solid theoretical foundation, specifically on stability analysis. This has motivated researchers to dedicate considerable attention to this topic in recent years. A key reference on the subject of stability in discrete-time MPC is the paper by Keerthi and Gilbert (1988). The pioneering work proved that the MPC cost function is a valid Lyapunov function for establishing closed-loop stability of MPC for a class of time-varying discrete-time systems. In order to guarantee closed-loop stability, various modifications to the MPC formulation have been explored (Mayne, *et al.*, 2000); e.g., terminal equality constraint, terminal inequality constraint, terminal constraint set, and terminal cost function.

Due to the feasibility problem in the method introduced by Keerthi and Gilbert (1998) requiring an exact solution at every sampling period, Michalska and Mayne (1993) proposed to replace the terminal (state) equality constraint with terminal (state) inequality constraint and employ a local asymptotically stabilizing controller within the positively invariant set of the inequality constraint. On the other hand, the MPC problem with terminal equality constraint is proved to be useful in the stabilization of systems when continuous feedback controllers fail (Meadows, 1994). In general, however, since typical MPC only utilizes the first control effort, imposing of terminal constraints is rather "artificial" (Kothare, *et al.*, 1996).

Bitmead, *et al.* (1990) proposed to impose a terminal cost to the MPC problem instead of terminal constraints. It was argued that by using a finite horizon Fake Algebraic Riccati Equation (FARE) to solve the weight matrix of the imposed terminal cost, infinite horizon Linear Quadratic stability was obtainable. However, the constraint handling capability of MPC was not retained in the formulation. Improvements have been made by Rawlings and Muske (1993) to formulate the MPC problem, which permits the integration of input and state

constraints. The underlying idea is to keep on-line a finite number of decision variables in the optimization problem that is solved. Primbs and Nevistić (2000) proposed a systematic approach of analyzing stability of the constrained MPC problem with terminal cost. Sufficient stability conditions can be obtained by bounding the terminal cost and relating the final state to the current state, with the knowledge of the upper and the lower bounds of the MPC cost function. Due to the systematic layout of this approach, it is used as the basis for analyzing close-loop stability of the random time-delayed NCS-MPC strategy developed in this study (see Chapter 4).

Following the work of Michalska and Mayne (1993), significant advancement has been made in using a sequence of sets as stabilizing constraints (terminal constraint set) by means of set invariance theory (Kerrigan, 2000). Scokaert, *et al.* (1999) relaxed the computation problems in optimal MPC with a terminal constraint by establishing conditions under which suboptimal MPC is stabilizing. They proved that under mild conditions, feasibility rather than optimality was sufficient for stability. Two suboptimal versions of MPC with relatively modest computational requirements were utilized to validate the idea. Limón Marruedo, *et al.* (2002a) combined terminal constraint and terminal cost to enlarge the domain of attraction of linear MPC, without increasing the prediction horizon. With the terminal cost as the Lyapunov function, the formulation converted the terminal region with a sequence of contractive control invariant sets which in turn converted the terminal constraint to a contractive terminal constraint. The formulation for nonlinear discrete-time system with bounded additive uncertainties is given in (Limón Marruedo, *et al.*, 2002b).

Other approaches have also been investigated, which are not in the mainstream. Stability of constrained MPC with model uncertainty was investigated by Zheng (1999) by posing the MPC problem as a convex optimization problem involving LMI. Costa and do Val (2003) proposed a method of ensuring stability of a nonlinear system controlled by MPC without imposing any terminal cost or terminal constraint. The positive definiteness of the cost function was also relaxed by only requiring "detectability" of the nonlinear system under control as well as the cost function. It was found that for a sufficiently long prediction horizon, the MPC control law was exponentially stabilizing if there would exist a uniform upper bound on the cost function. Nunes, *et al.* (2003) looked into characterizing the stability properties of unconstrained multivariable GPC using polynomial operators and coprime matrix factorizations resulting in a numerical solution of the closed-loop poles of the system. The parameters of the controllers could then be tuned accordingly, to obtain a stable system where the poles of the system would

be within a unit circle. However, extending this method to constrained problems may present difficulties.

1.3.4 Remote Monitoring and Supervisory Control

Tang, *et al.* (2002) have proposed an approach for remote monitoring of machines within a group of distributed and networked production plants using an existing extensible TCP/IP (Transmission Control Protocol/Internet Protocol). Their work focused on vibration monitoring of precision machines where vibration signals or signatures gathered from accelerometers that are mounted on the machine, were transmitted to a remote client. The client was able to monitor and control the machine through a web-browser via a CGI interface (Common Gateway Interface). Gu and de Silva (1997) developed an open-architecture robot controller for coordination of various components of a robotic workcell when carrying out workcell tasks. The system possessed the capabilities of low level, direct, manipulator control; process monitoring; kinematic transformation; and intelligent decision making; and incorporated a graphical user interface. Chen and Luo (1997) designed a remote supervisory control architecture, which combined a computer network and an autonomous mobile robot. An authorized user was able to directly control the mobile robot remotely using a web-browser and obtain video feedback captured by a CCD camera, which was mounted on the robot.

Due to the advancements of the Internet technology, recent years have also seen the introduction of tools that facilitate web-based academic research and teaching (Bhandari and Shor, 1998, Röhrig and Jochheim, 1999, Ko, *et al.*, 2000, 2001). Virtual laboratories facilitate students and research personnel to conduct experiments remotely through the Internet from anywhere and anytime. The concept is especially useful in situations where the laboratory resources are rather limited and costly but are regularly needed to serve a large number of users. With virtual laboratories, globally located collaborating institutes are able to share resources and expertise, resulting in clear economic advantages as well. In this manner, domain experts can divide their attention rather efficiently. With the objective of permitting experiments to be conducted through the Internet, Bhandari and Shor (1998) developed a distance learning application at the Control Engineering Laboratory of Oregon State University. Experiments were easily accessible for students using a Java™-enabled web-browser within the campus or even from home. Several German universities are pursuing concepts of distance education, as applied to laboratory experiments. Röhrig and Jochheim (1999) developed a network of remotely accessible laboratories called Virtual Lab using a client-server

architecture. Active and simultaneous interaction between students or researchers participating in a particular experiment was a key focus of the reported work. Ko, *et al.* (2000, 2001) have demonstrated the practicality and feasibility of implementing web-based laboratory experiments by successfully opening their experiments to over 1000 engineering students and later to the general public. A robust, general methodology has been presented.

Several commercial products exist; e.g., LabVIEW™ (National Instrument) and WebLab (Quanser), which provide monitoring and control of experimental apparatus remotely over the Internet. However, these products require one dedicated web-server to be associated with each process, which permits only one user to access the process at a given time. This limits the level of cooperation between the users who are associated with the specific process. In addition, such products are not cost effective since a sufficiently powerful server has to be installed for each process. This is the case because the entire computational load associated with data communication, user access, program timing and scheduling, and low-level control is handled by that same computer. It is also difficult to maintain a unified user access and the needed security level for all the available processes within a local establishment.

1.4 Contributions and Organization of the Thesis

The main contributions of this thesis are outlined below.

1. An original feedback control strategy based on the theory of model predictive control (MPC) is developed for networked control systems (NCS). The strategy predicts future control efforts and buffers them at the actuator in anticipation for data transmission problems between the controller and the actuators, while using an extended estimator to compensate for data transmission problems in the sensory information stream. This is an innovative and successful attempt to overcome problems of network delay and losses in the control effort, in the context of an NCS. The effectiveness of the developed NCS strategy is demonstrated through implementation on an industrial machine (two-axis electro-hydraulic manipulator of a fish-processing machine) through an Ethernet network, and carrying out rigorous real-time experimentation.
2. New stability theorems in the sense of Lyapunov are established in detail, with analytical proofs, and experimentally validated for the developed NCS-MPC strategy. The originality of this work lies in the analysis of feasibility, stability and optimality of using the sequences of future control effort in constrained MPC. The established results form a

useful tool to design an NCS-MPC controller with proper parameter settings that will maintain stability within a predetermined worst-case delay between the controller and the actuators.

3. With regard to the effect of state mismatch in the sensory data estimation on the closed-loop stability of the NCS-MPC system, an analysis is carried out to determine the upper bound of the norm of estimation error up to which stability is maintained. This stability bound which is indirectly related to the worst-case delay between the sensors and the controller proves to be a useful guideline in the design of any form of estimator for the developed NCS strategy.
4. A low cost, universal and scalable two-tier client-server network infrastructure (including hardware and software) for remote monitoring and supervisory control of industrial systems is established. User scheduling and data flow management methodology is developed. The practicality of the developed web-based architecture is demonstrated through implementation and experimentation on an industrial fish-processing machine.
5. The development of a networked, multi-layered, modular, and intelligent supervisory control structure is presented. Real-time implementation of the architecture shows that the incorporation of a knowledge-based autonomous supervisor for the lower-level local and remote adaptive controllers is vital in improving the system performance.

The research and development that leads to the summarized contributions are presented in the subsequent chapters of the thesis in sufficient detail. The organization of these chapters is as follows. **Chapter 2** presents the preliminary work necessary for the study of an NCS, which includes the development of a robust control network architecture within an Ethernet network, an accurate clock synchronization algorithm, characterization of network transmission properties, and modeling and identification of the experimental platform.

Chapter 3 describes the constructive NCS control strategy, which is central to the present investigation. The formulation of the strategy, which is based on multivariable predictive control, is detailed. In this chapter, only the unconstrained case is considered in order to focus on the underlying mechanics of the strategy. It is shown that the developed strategy renders transparent the dynamics of data transmission. Real-time experimental evaluation and performance benchmarking results are then presented and discussed. The central work from this chapter has been published in (Tang and de Silva, 2003a) and accepted for publication in (Tang and de Silva, 2005a).

Chapter 4 focuses on analyzing the asymptotic stability of the developed NCS-MPC

strategy, with the buffering of future predicted control effort. The analysis is accomplished in a broader realm of constrained MPC with equivalent state-space formulation. Perfect state measurement or estimation of sensor data is assumed. Validation of the developed stability results is carried out through computer simulation and experimentation. The main contribution of this chapter are reported in (Tang and de Silva, 2005b) and expected to be published in (Tang and de Silva, 2005c).

Further stability analysis of the developed NCS-MPC strategy is presented in **Chapter 5** in which the main consideration is the effect of imperfect state measurement from the estimator on the closed-loop stability. The problem formulation and characterization using the concept of sensitivity in nonlinear programming, leading to the main stability bounds, are carefully laid out.

After having resolved, with thorough analysis, simulation, and experimentation, the problems incurred in the data transmission at the low-level feedback control of a networked control system, the benefits of incorporating monitoring, diagnostic, and supervisory control capabilities into the system architecture is investigated. **Chapter 6** provides the framework for infrastructure development for web-based monitoring of a remote system. This includes the hardware commissioning and networking, and the software development of a two-tiered client-server communication architecture. The practicality of the developed and implemented infrastructure is demonstrated on an industrial machine. This work has been published in (Tang, *et al.*, 2002c) and (Tang, *et al.*, 2002d), and submitted for publication in (Tang and de Silva, 2005d).

Chapter 7 presents a new scheme for remote supervisory control of plant and the associated client-server network architecture, which has been developed in the present work. The scheme uses a remote knowledge-based supervisor to autonomously supervise both local and remote intelligent model-referenced adaptive controllers. The performance of the developed scheme is then evaluated through implementation on an industrial fish-processing machine and subsequent experimental case studies. The various control techniques developed in this chapter have been published, in part, in (Tang, *et al.*, 2002a), (Tang, *et al.*, 2002b), and (Tang and de Silva, 2003b). The most recent work has been submitted for publication in (Tang and de Silva, 2005d). **Chapter 8** concludes the thesis by summarizing the overall research that has been carried out, indicating some limitations of the work, and suggesting possible directions for future research in the area.

Chapter 2

Control Network Architecture and System Modeling

The main objective of this chapter is to present the preliminary work that is essential in the investigation of Networked Control Systems (NCS). A general architecture for a networked control system will be constructed using the Ethernet communication network. This is a practical architecture for implementing low-level control loops. It facilitates real-time experimental evaluation of system performance, stability in particular, which will be carried out in chapters 3 and 4, and also serves as the basic building block for remote monitoring and supervisory control, which will be explored in chapters 6 and 7.

An important element of any NCS setup is an accurate clock synchronization algorithm for establishing a common time reference for various interacting nodes (devices) across the communication network. The adopted algorithms and issues relating to clock synchronization will be explored in Section 2.2. Next, some insight into the data transmission characteristics of the Ethernet network will be provided. Section 2.4 will describe the dual-axis electro-hydraulic manipulator of an industrial fish-processing machine, which forms the main experimental system in this thesis. Model parameters of the electro-hydraulic manipulator will be identified experimentally using a time-domain correlation method for input-to-output and input-to-state responses.

2.1 Control Network Architecture

The simplified data flow architecture of the networked control system, when performing real-time experiments, is shown in Figure 2.1. This setup is connected to a general office Ethernet-based local area network (Kurose and Ross, 2003). Each node; i.e., control server, control client, and intermediate nodes, are connected to the building server room which house Ethernet switches. These switches direct data between all the nodes connected to them and to the outside world though an Internet backbone. The performance of a particular network can

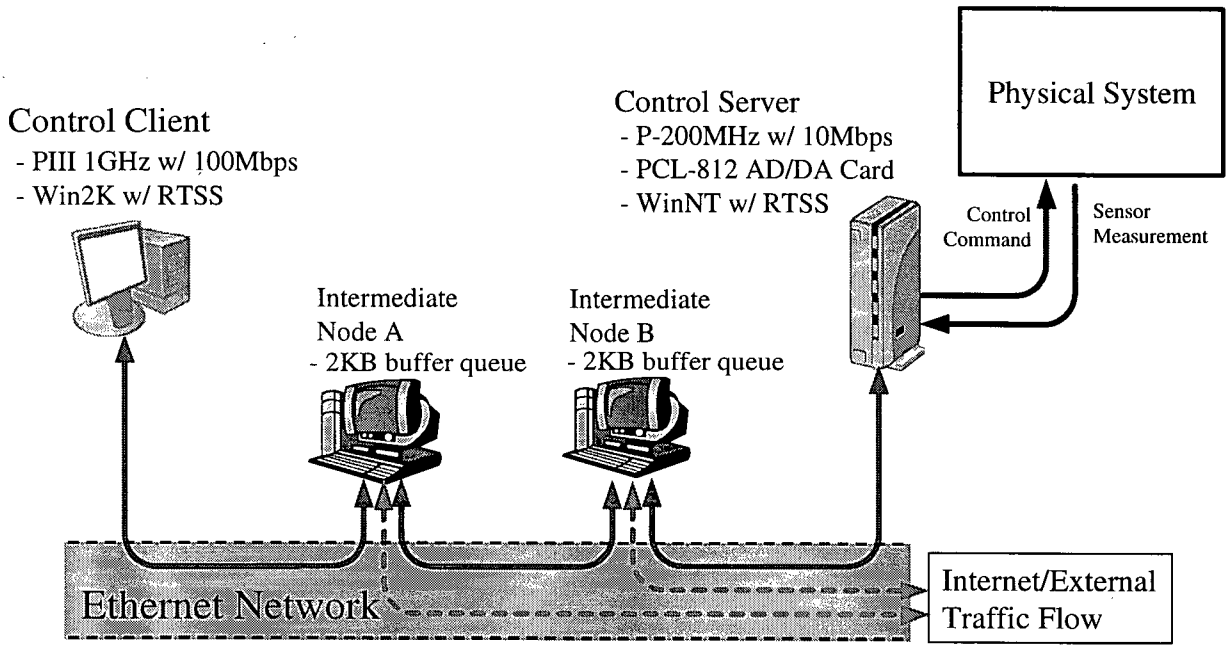


Figure 2.1: Simplified data flow layout of the developed control network architecture.

differ significantly depending on the type of interconnecting hardware between the nodes; e.g., between hubs and switches. The network control strategies developed in chapters 3 through 5 of this thesis are intended for general packet data networks. The physical system is directly connected to the control server, which hosts a 12-bit data acquisition board (PC-LabCard PCL-812) and is connected to the Ethernet network through a 10BaseT network interface card (NIC). It consists of a Pentium-200 MHz processor running on Windows NT with Real-time Sub-system. This control server handles sensor data sampling and control signal buffering.

The feedback control strategies developed in the present work are implemented at the control client. The control client has a higher processing power with a PIII-1GHz processor running on Windows 2000 with Real-time Sub-system. No data acquisition hardware is required at the control client except for a low cost NIC. Although a 100BaseT NIC is used in the control client, the bottleneck is at the control server, which has 10BaseT NIC.

In order to evaluate the effectiveness of the developed networked control strategies under various types of network behavior, two intermediate nodes are introduced between the control server and the client. These two nodes contain 2 kilobyte buffer queue and have external traffic flowing during experimental studies, to simulate the bandwidth consumption by other equipment that share the same Ethernet network. This external traffic flow is simply induced by running any of the widely available Point-to-Point (P2P) file-sharing programs. In addition, the

data packets traveling between the control server and client are "juggled" between the two intermediate nodes. This closely represents data hopping in routers and switches in wide area networks. It is to be noted that exact network utilization is not strictly managed. Similar data hopping behavior can be simulated using a store-and-forward queuing algorithm located at the control server. In case the control server is not suitable to run this algorithm (e.g., due to low computational power), it can be located at one of the intermediate nodes. The main objective here is to test the performance of the network control strategies under severe network conditions that may be experienced in large-scale local- or wide-area industrial networks with potentially noisy and long data lines.

The control server and client are implemented in multiple independent processes as shown in Figure 2.2. The number in each block represents the priority level of the corresponding process. Inter-process communication is carried out through a common, shared memory area. At the control server, the processes are of minimal functionality since most of the high computational tasks are located at the remote control client. The shared memory area of the

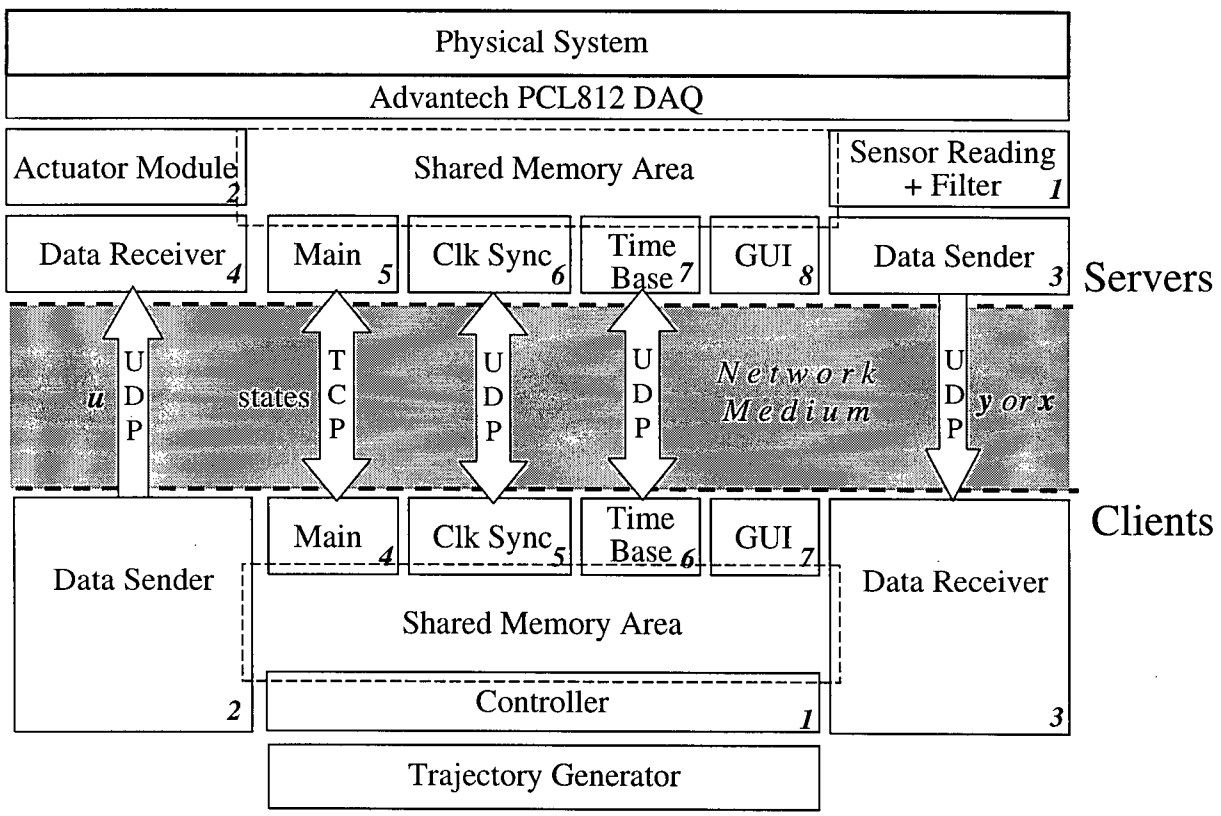


Figure 2.2: Schematic diagram of the end-to-end client-server control network architecture.

control server contains the variables to store filtered sensor readings, control inputs, time stamping information of the control inputs, time base for clock offset, and execution states of the system (e.g., start, stop, hold, and ready). The process module for sensor reading and filtering runs at a pre-specified interval at the highest priority. It triggers the data sender to time-stamp, and sends the sensor data packet as soon as a sensor measurement is ready. The data receiver is an event-driven process. Whenever a data packet is received, the data receiver time-stamps it and stores it in a buffer in the shared memory area awaiting retrieval by the actuator module. The actuator module, running at a pre-specified interval, reads and sorts the control inputs from the shared memory area before clearing the buffer and writing the appropriate control inputs to the data acquisition board. It should be noted that the data reading buffer in the shared memory area discussed here is not the actuator buffer of future control inputs of the network control strategy developed in this thesis. The present buffer is implemented merely to handle situations when more than one data packet is received between the intervals of the actuator module process. The GUI (Graphical User Interface) process allows basic operator interaction to the control system. The main process coordinates the execution and hand-shaking among the running processes. Similar role is performed by the corresponding processes on the remote control client. Control laws are implemented in the controller process. The clock synchronization and time base process pairs are implemented to maintain an identical time reference between the servers and clients, which is vital in time-stamping of data packets and feedback control. This is further discussed in the next section.

UDP (User Datagram Protocol) protocol is adopted for all data communication activities between the control server and the client, except for exchanging the execution states of the system where TCP (Transport Control Protocol) protocol is used. TCP is a connection-orientated protocol and provides reliable point-to-point data transfer. Data delivery is ensured using flow control, sequence numbering, retransmission, and timer. In addition, TCP also provides congestion control by regulating the rate at which data is injected into the network. However, the throughput of such a regulated network is unsuitable for real-time feedback control of servo systems. UDP, on the other hand, eliminates the overheads associated with TCP, making it a much preferred transmission protocol for feedback control. It is a connectionless service, which sends independent data packets (datagrams) from one node to another. Hence, data packets can be "pumped" as frequently as possible to the network, but at the cost of some data getting lost and not reaching the destination. All control and

communication modules are coded and compiled in Microsoft Visual C++ using standard ANSI C/C++ language.

2.2 Clock Synchronization Algorithm

In distributed systems, each node has its own internal clock, and typically these clocks are neither identical nor synchronous. This leads to a fundamental problem in distributed systems where it is crucial to accurately determine the time and the order of the occurring events. The time accuracy is even more crucial in asynchronous networked control systems (NCS) with sub-millisecond timing requirements.

Before proceeding, the definitions of three pertinent terms are given. First, clock *skew* is defined as the instantaneous difference between the recorded time and the actual absolute time. The instantaneous difference between the time readings of any two clocks is termed clock *offset*. Clock *drift* occurs in a crystal-based clock, if the counting frequency varies (drifts), causing a timing error in the particular clock and generates a diverging clock skew and clock offset. Since an NCS setup involves a finite number of pre-determined nodes, it is particularly important to determine the clock offsets between the nodes and make corrections for clock drifts.

2.2.1 Existing Algorithms

Clock synchronization, in typical network setups, is based on message passing. The clock requests are sent back-and-forth between communicating nodes with the requesting node adjusting its own clock relative to the responding node. In a synchronous system where the bounds for the drift rate of clocks, the maximum message transmission delay, and the time to execute each step of a process are known, the clocks can be easily synchronized based on the maximum and minimum message transmission delays, denoted by $\tau_{\max}$ and $\tau_{\min}$, respectively. Lundelius and Lynch (1984) state that if the requesting node sets its clock according to $t = t + \frac{1}{2}(\tau_{\max} + \tau_{\min})$, the optimum achievable bound on clock offset is $(1-\frac{1}{N})(\tau_{\max} + \tau_{\min})$ for a system with N nodes. It is not as straight forward in asynchronous systems where the message transmission is not bounded from above.

Assuming the round-trip delay (i.e., the time instant from clock request until the instant the requesting node receives back the response from the opposite node) can be equally split,

Cristian (1989) proposed the use of a time server to synchronize asynchronous systems. With the time server connected to an absolute time source of Coordinated Universal Time (UTC), the requesting node can simply advance or lapse its clock by half of the round trip time. This method is plausible only if the observed round-trip delays between the requesting clients and the time server are sufficiently short compared to the required clock accuracy.

The Berkeley algorithm (Gusella and Zattai, 1989) uses a coordinator node that periodically polls the clocks from the other nodes within a distributed system. The coordinator node then computes the required clock adjustment using a method similar to Cristian's algorithm, and forwards it to the corresponding node. It is claimed that any occasional readings associated with larger times than the nominal round-trip delay can be eliminated, and the scheme is able to achieve a synchronized accuracy of approximately 20-25 ms.

Since the algorithms of Cristian and Berkeley are intended only for the Intranet network, Mills (1995) proposed the Network Time Protocol (NTP) service to handle large networks (Internet) with large and variable message delays. NTP uses a network of servers arranged in a hierarchical tree, with the primary server connected directly to the UTC source and the user nodes functioning as the servers at the lowest level of the tree (i.e., at the leaves). NTP servers are synchronized with each other in one of the following three modes: multicast, procedure-call, and symmetric. Within a high-speed LAN (Local Area Network) with small delay, multicasting is used. An algorithm similar to that of Cristian is used in the procedure-call mode. In the symmetric mode, timing information is retained as part of an association between servers and is used when the highest level of clock accuracy is required. This higher accuracy is obtained by enforcing double message exchange pairing, resulting in four time-stamps.

2.2.2 Approach

As indicated before, NCS requires synchronized clocks with sub-millisecond accuracy, which is not achievable using the existing clock synchronization algorithms discussed above. In this thesis, the algorithm proposed by Nilsson (1998), which allows off-line or on-line clock corrections, is adopted. In the present study, the clock of the control server node will be used as the reference time, and on this basis a control client node initiates the clock synchronization algorithm to readjust its clock. Let C_C denote the control client node requesting the clock time and C_S denote the responding control server node. Three series of time-stamp data are

collected by the control client sending the clock time request messages to the control server, and the control server responds by a replying message to the control client, repeated over N data samples. Let $t_a^{C_C}(n)\Big|_{n=1}^{N}$ be the series of time-stamps when the control client sent the clock request, $t_b^{C_S}(n)\Big|_{n=1}^{N}$ be the series of time-stamps when the control server received and immediately replied to the clock request, and $t_c^{C_C}(n)\Big|_{n=1}^{N}$ be the series of time-stamps when the response to the clock request arrived back at the control client. Defining t_{abs} as the absolute time, $t_i^{C_C}$ as the local time of node C_C, $t_i^{C_S}$ as the local time of node C_S, and assuming both clocks in the control client and server have a linear growing skew to the absolute time, we have

$$t_i^{C_C} = t_{abs} + \delta^{C_C} + \rho^{C_C} t_{abs} \tag{2.1}$$

$$t_i^{C_S} = t_{abs} + \delta^{C_S} + \rho^{C_S} t_{abs} \tag{2.2}$$

where δ^{C_C} and δ^{C_S} are clock skews, and ρ^{C_C} and ρ^{C_S} are clock drifts. Combining (2.1) and (2.2) to eliminate t_{abs}, one obtains the following linear relation:

$$t_i^{C_C} = m t_i^{C_S} + c \tag{2.3}$$

in which

$$c = \delta^{C_S} - \left(\frac{1+\rho^{C_S}}{1+\rho^{C_C}}\right)\delta^{C_C} \tag{2.4}$$

and

$$m = \frac{1+\rho^{C_S}}{1+\rho^{C_C}} \tag{2.5}$$

With the message transmission times (based on the clock at node C_C) from the control client to the control server, and the control server to the control client defined as $t_n^{C_C C_S}$ and $t_n^{C_S C_C}$, respectively, it is common in network clock synchronization to assume their mean values to be equal; i.e., $\mathrm{E}\left[t_n^{C_C C_S}\right] = \mathrm{E}\left[t_n^{C_S C_C}\right] = \mu$ for $n \in [1,...,N]$. Since the time at which the control server handles a particular clock request is unknown to the control client, μ can be used as a

measure for the time duration (based on clock at node C_C) from the request is made, up to the time when it is handled by the control server, and is estimated as

$$\hat{\mu} = \frac{1}{2N}\sum_{n=1}^{N}\left(t_c^{C_C}(n) - t_a^{C_C}(n)\right) \tag{2.6}$$

The data points in the time series $t_b^{C_S}(n)\Big|_{n=1}^{N}$ and $t_c^{C_C}(n)\Big|_{n=1}^{N}$ can be written as

$$t_b^{C_S}(n) = m\left(t_a^{C_C}(n) + t_n^{C_C C_S}\right) + c \tag{2.7}$$

and

$$t_c^{C_C}(n) = t_a^{C_C}(n) + t_n^{C_C C_S} + t_n^{C_S C_C} \tag{2.8}$$

respectively. From equation (2.7), we obtain

$$t_b^{C_S}(n) = m\left(t_a^{C_C}(n) + \hat{\mu}\right) + c \tag{2.9}$$

Hence, by fitting a straight line through the data points of plot $t_b^{C_S}(n)\Big|_{n=1}^{N}$ versus $t_a^{C_C}(n)\Big|_{n=1}^{N} + \hat{\mu}$, its gradient $\hat{m}$ and intercept $\hat{c}$ can be determined, which are estimates for m and c, respectively. Here, the least squares line fitting algorithm (Ljung, 1987) is employed which is given as:

$$\begin{bmatrix}\hat{c}\\ \hat{m}\end{bmatrix} = \left[\left(\boldsymbol{T}_a^{C_C}\right)^T \boldsymbol{T}_a^{C_C}\right]^{-1}\left[\left(\boldsymbol{T}_a^{C_C}\right)^T \boldsymbol{t}_b^{C_S}\right] \tag{2.10}$$

where

$$\boldsymbol{T}_a^{C_C} = \begin{bmatrix} 1 & t_a^{C_C}(1) + \hat{\mu} \\ 1 & t_a^{C_C}(2) + \hat{\mu} \\ \vdots & \vdots \\ 1 & t_a^{C_C}(N) + \hat{\mu} \end{bmatrix} \tag{2.11}$$

and

$$\boldsymbol{t}_b^{C_S} = \begin{bmatrix} t_b^{C_S}(1) & t_b^{C_S}(2) & \cdots & t_b^{C_S}(N) \end{bmatrix}^T \tag{2.12}$$

Consequently, using the estimates of m, c and μ from (2.6) and (2.10), the clock reading in

the control server can be transformed to the clock reading in the control client through (2.3).

Since the smallest sampling period used in this study is 1 ms and the computers used have relatively low clock drift rates, clock synchronization is done off-line at an interval of 10 min with N chosen to be 100 data sets. The Real-time Sub-system employed here reports time in 100 ns clock ticks since 12:00 a.m. of January 1, 1601. At the current time of writing this thesis, the clock tick readings are in the order of 10^{17} ticks subjecting (2.12) to numerical errors. This can be overcome by finding a time base in which all subsequent clock readings would be subtracted for the purpose of reducing the order of the data values. The time base is set as the smaller clock value between the control server and the control client, through the time base process pairs in Figure 2.2. However, the resulting clock tick readings and $\hat{c}$ are still significantly greater than $\hat{m}$ which can cause a singularity in the matrix inverse of (2.12). This can be resolved by introducing a scaling constant $\bar{c}$ into (2.12) such that (2.12) becomes

$$\begin{bmatrix} \hat{c}/\bar{c} \\ \hat{m} \end{bmatrix} = \left[\left(\boldsymbol{T}_a^{C_C} \begin{bmatrix} \bar{c} & 0 \\ 0 & 1 \end{bmatrix} \right)^T \boldsymbol{T}_a^{C_C} \begin{bmatrix} \bar{c} & 0 \\ 0 & 1 \end{bmatrix} \right]^{-1} \left[\left(\boldsymbol{T}_a^{C_C} \begin{bmatrix} \bar{c} & 0 \\ 0 & 1 \end{bmatrix} \right)^T \boldsymbol{t}_b^{C_S} \right] \tag{2.13}$$

where $\bar{c}$ is close to $t_a^{C_C}(n)$. Here, $\bar{c}$ is set to 10^{10}. Using this clock synchronization approach, a synchronized clock accuracy of 1 μs is achieved.

2.3 Characteristics of Ethernet Communication

Control networks can be grouped into two main categories; namely, cyclic service network, and random access network. PROFIBUS (Process Fieldbus), ControlNet, FIP (Factory Instrumentation Protocol), and SAE token ring are cyclic service networks while Ethernet and DeviceNet (CAN) are random access networks. In cyclic service networks, the nodes are arranged logically into a ring and messages are transmitted in a cyclic order with deterministic behavior, because the maximum waiting time before sending a message is governed by the token rotation time. A token is passed from one node to a predetermined successive node. Only the node with the token is permitted to transmit messages within a limited number of data frames. In random access networks, the CSMA (Carrier Sense Multiple Access) protocol is usually used to regulate the network usage. A node listens (carrier sensing) to the channel before attempting to transmit its message. If the node detects that another node is transmitting a message, it waits for a random amount of time before attempting to retransmit its message. This

Table 2.1: Comparison of four typical communication networks.

	Ethernet	ControlNet	DeviceNet (CAN)	PROFIBUS
Data rate (Mbps)	10/100/1000	5	0.5	0.0096-12
Max. length (m)	100	1000	100	100-1200
Overhead per message (bytes)	38	7	5.9	9
Min. transfer unit (bytes)	46	7	8	0
Max. transfer unit (bytes)	1500	504	8	32-244
Max. number of nodes	∞	99	64	32
Deterministic?	No	No	Yes	Yes

random waiting time is the main reason for the non-deterministic nature of random access networks. Table 2.1 gives a basic comparison of four typical communication networks. Although, originally, Ethernet is not intended as a control network, it is the most widely deployed LAN (Local Area Network) technology due to its low cost and rapid technological advancement. Data can be transmitted at differently rates; most common are 10 Mbps, 100 Mbps, and 1000 Mbps, which are among the fastest for communication networks. The maximum cable length between two nodes without repeaters is 100 m (40 km with fiber optics). The allowable data rate for PROFIBUS depends on the maximum cable length, and it ranges from 9.6 kbps for 1200 m to 12 Mbps for 100 m. Although Ethernet has the longest overhead per message, performance is not affected due to its relatively high data rates. The maximum transfer unit (MTU) of Ethernet is 1500 bytes making it ideal for large-scale centralized control networks with a large amount of information per sample. With 8 bytes MTU, DeviceNet is particularly suitable for short messaging networks. A message larger than an MTU is fragmented into multiple packets, which pose a challenge for NCS. For example, a set of sensor readings may be fragmented into multiple packets, arriving at the controller node at different times. In terms of the total number of nodes that can be connected within a control network, the Ethernet has practically no limit, rendering it well suited for long-distance remote control.

Various types of Ethernet have the same frame structure (for an individual data packet) which comprises 6 fields; namely, preamble field, destination address, source address, type field, data field, and cyclic redundancy check (CRC). The function of the preamble field is to

"wake up" and alert the receiving node of an incoming data packet. The destination and source address field contains the unique 6 bytes physical LAN address of the target and transmitter nodes. The type of network-layer protocol is specified in the type field. The data field is the location where the IP datagram (actual message) resides and the size of the field is variable from 46 to 1500 bytes. The field is padded with additional filler data if the IP datagram is smaller than 46 bytes. If the datagram is larger than 1500 bytes, it has to be fragmented into additional data packets. The receiving node uses the CRC field to check for bit errors in the datagram.

The Ethernet is based on the decentralized and multiple access protocol called CSMA/CD (Carrier Sense Multiple Access with Collision Detection) specified in the IEEE 802.3 standard (IEEE Computer Society, 2002). The nodes within the network have no explicit coordination with each other. Before attempting to transmit a data packet, a node senses (through its line voltage level) whether the network channel is busy or idle. If the channel is busy, it waits until it senses that the channel is idle before starting to transmit the data packet. While transmitting, if the node senses that some other node is also transmitting (collision), it aborts the transmission of the data packet and transmits a 48-bit jamming signal. The node goes into a random waiting phase (exponential back-off phase) before attempting to retransmit the same data packet again. Such mechanism of the CSMA/CD protocol results in the nondeterministic nature of Ethernet transmission. The nondeterministic effect of CSMA/CD between communicating nodes is overcome with the use of *switched* Ethernet in most modern facilities as compared to the use of legacy hubs or coaxial bus (10base2). However, although network efficiency is increased, switching Ethernet transmission is still nondeterministic because the switches are subjected to packet queuing within a limited buffer and packets are dropped when the buffer overflows.

Ethernet-based discrete-time control networks are plagued with the following four classes of problems:

- ***Nondeterministic communication delay***. Communication delay is composed of processing delay caused by the time needed for destination determination and bit error checking, queuing delay due to a data packet waiting to be transmitted onto the network link, store-and-forward delay due to the time required to "push-out" all the bits of a data packet into the network link, and propagation delay due to the travel time of the data packet along the network link. It is also be caused by the type of transport protocol used;

e.g., TCP may initiate a resend if the data packet is lost or corrupted.

- ***Packet losses***. Packet losses are caused by the dropping of data packets from a finite queue in communication nodes when the queue is full. Data packets can also be lost, or rather unusable, from noisy network cabling which leads to data corruption.
- ***Vacant sampling***. Vacant sampling is classified as the situation where no data packet is received within a sampling interval due to loss or delay.
- ***Mis-synchronized (out-of-order) data arrival***. Data mis-synchronization occurs when the time sequence according to which data packets received is not the same as the time sequence in which they are transmitted.

Figure 2.3 shows the typical data communication delay of Ethernet networks. A histogram distribution is also shown on the side of each plot. Data are transmitted by UDP stream sockets locally via Ethernet and remotely through the Internet. The datagram size is 528 bytes per data packet, and transmitted at 200 ms intervals over 200 samples (Note: The 200 ms interval used here is not meant to be realistic but to give a general picture of the problem at hand. The transmission rate should be consistent with the sampling time that will be used in real-time feedback control (on the order of 1-10ms)). The information given in Figure 2.3(a) has been generated between two nodes from a LAN hopping between 6 nodes. The mean communication delay is 1.362 ms with 0.173 ms standard deviation. No packet loss is recorded. The information in Figure 2.3(b) has been generated between the University of British Columbia (UBC) in Vancouver and the University of Toronto in Toronto, Canada. The data packets transited 14 node hops along the route resulting in 72.207 ms mean delay with a standard deviation of 7.886 ms and 12 % packet loss rate which are indicated by the vertical falling signals down to the horizontal axis. Figure 2.3(c) is based on information exchange between UBC and the National University of Singapore (NUS) in Singapore over 20 node hops and reaching a mean delay of 239.157 ms with a standard deviation of 3.204 ms. A total of 15 % of the transmitted packets are lost. Clearly, the incurred communication delay increases with the distance between the source computer and the destination computer. Although there is variation in the transmission delay, its distribution is concentrated within a narrow band, as can be seen from the histograms.

Figure 2.4 shows the Ethernet transmission characteristics for tracking a 1 Hz sine wave of 1 unit magnitude. Data packets of 32 bytes each are transmitted at an interval of 4 ms using the UDP protocol. The response between two directly connected nodes is shown in Figure 2.4(a).

Obviously, no data packet is lost and the recorded mean delay is 0.459 ms. Figure 2.4(b) shows the tracking response obtained within a LAN setup where data packets are "juggled" between two intermediate nodes as discussed in Section 2.1. Each data packet is juggled 10 times before being directed to the target node. Mean delay of 11.790 ms with minimal data packet loss (<1%) is observed. Frequent vacant sampling and simultaneous data arrival are also observed. Next, an echo server is setup at NUS. The sine wave transmitter and receiver nodes are located locally at UBC. All data packets transmitted to the echo server are echoed back to the receiver. The resulting severe response is shown in Figure 2.4(c), which has a mean delay of 840.5 ms and a data packet loss rate of 34.4 %. The delay grew with time until the wave on the receiving end lagged approximately one complete cycle behind the source. This is mostly due to the limited storage space in the queuing buffers along the traversed nodes. In general, the communication delay depends on the number of nodes traversed, speed of each segment, size of data packets, network load, amount of congestion in the network traffic, size of queuing buffer on each traversed node, and the communication protocol of each node. In addition, it can be observed that the number of data packet losses increases with the distance between the two communicating nodes.

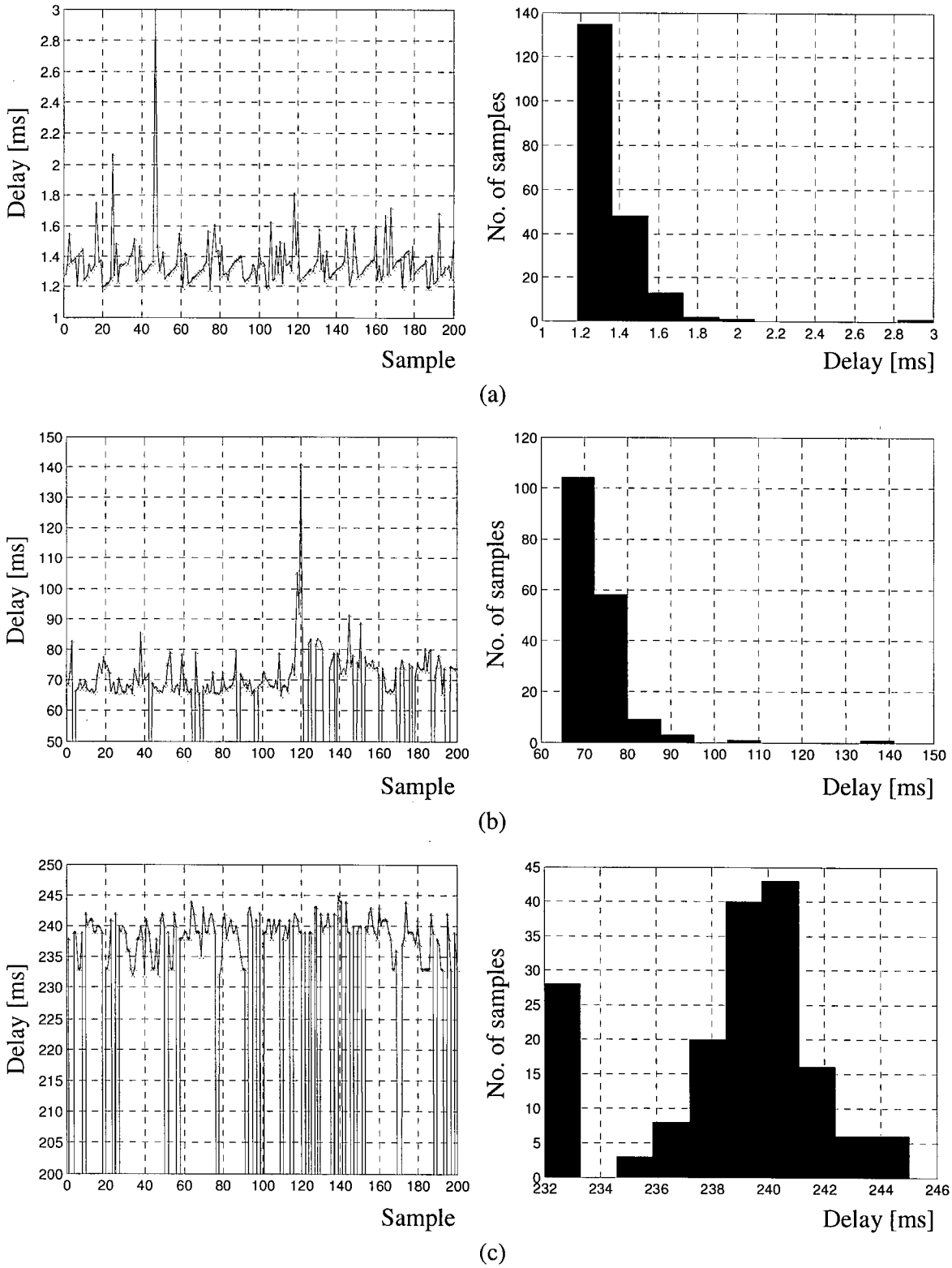


Figure 2.3: Communication delay in Ethernet networks. (a) Within a LAN; (b) Between the University of British Columbia and the University of Toronto; and (c) Between the University of British Columbia and the National University of Singapore.

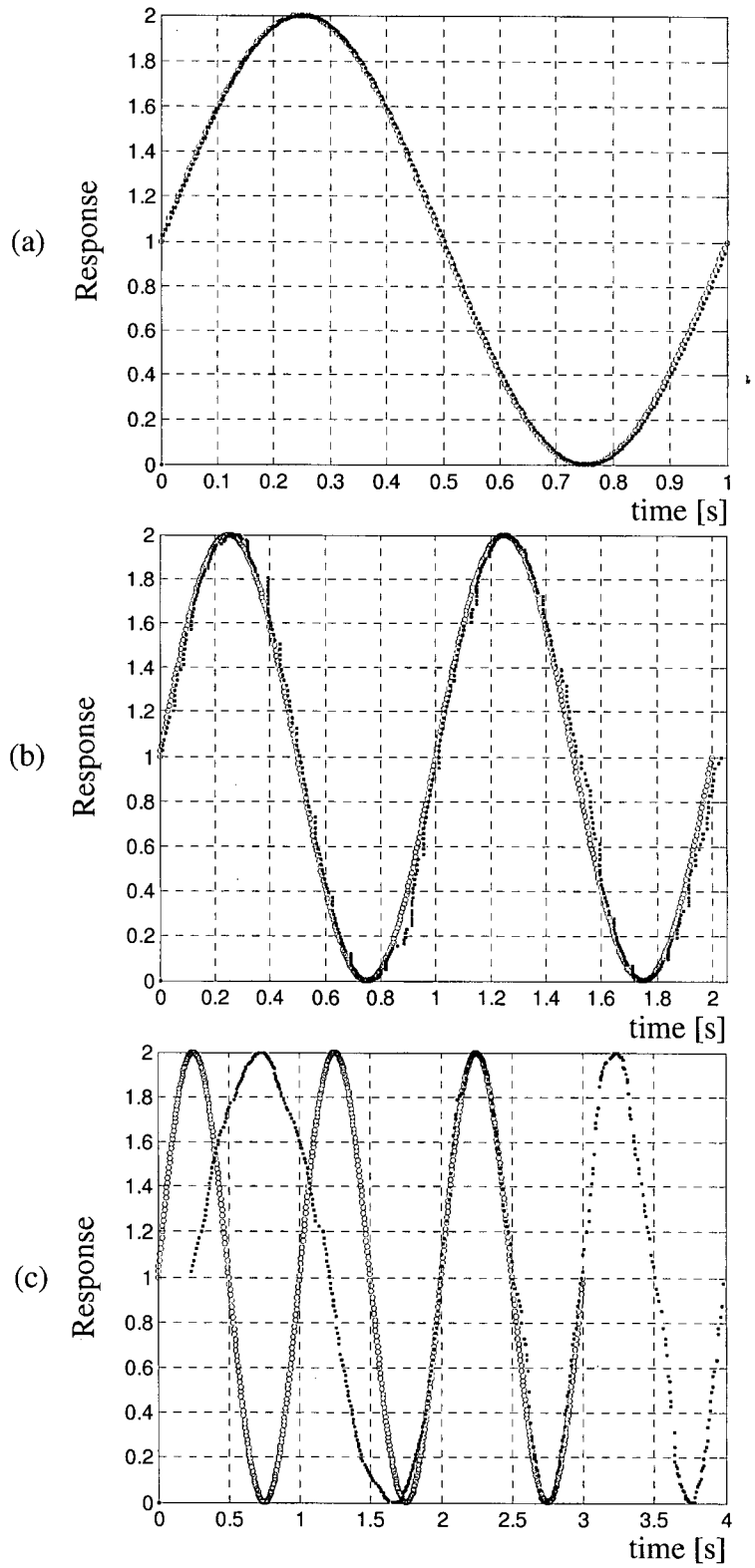


Figure 2.4: Sine wave tracking in Ethernet networks (circles – transmitted; and solid-squares – received). (a) Direct transfer within a LAN; (b) Using data "juggling" between intermediate nodes; and (c) Loop transfer with remote echo server at the National University of Singapore.

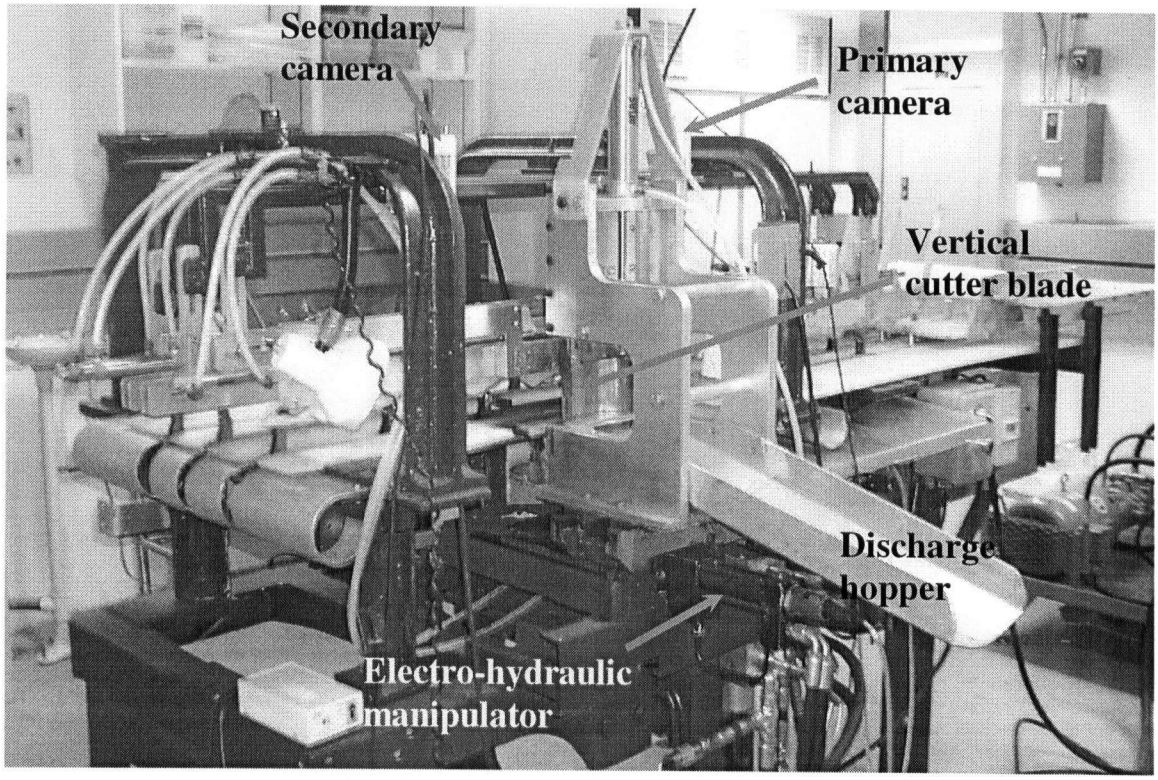
2.4 Modeling of the Electro-hydraulic Manipulator

This section describes the key experimental platform that is employed for real-time experimental evaluation and technological demonstration of the networked control and supervisory control strategies, and the remote monitoring architecture developed in the succeeding chapters. It is an industrial fish-processing machine (Tafazoli, *et al.*, 1998, de Silva, 2005), known as the "Intelligent Iron Butcher," developed in the Industrial Automation Laboratory of the University of British Columbia to automate the head cutting in fish, which is the first primary stage in the canning process. The goal is to realize fast and accurate cuts so as to minimize the wastage of useful meat and consequently to increase the recovery and throughput.

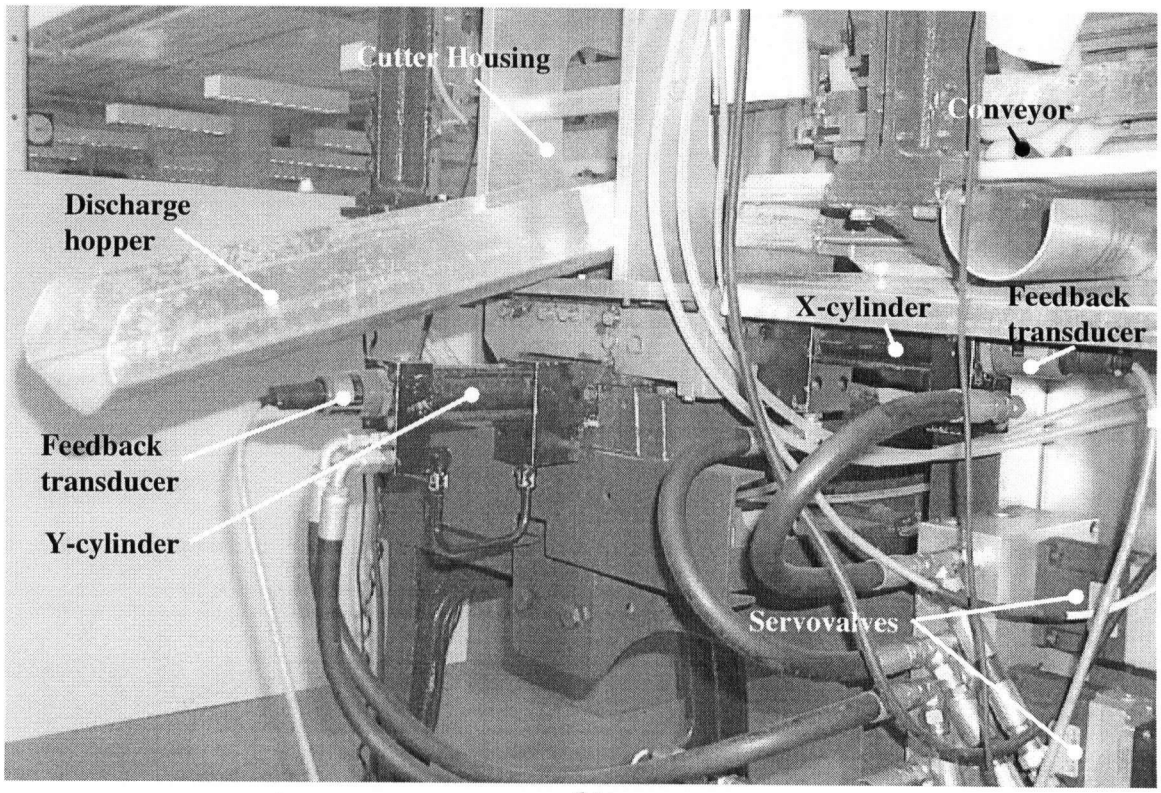
2.4.1 System Overview

The industrial fish-processing machine, as shown in Figure 2.5(a), operates using an ac motor-powered reciprocating, logic-driven indexing conveyor system to accurately move the fish from the feeder end to the vision module and the cutter assembly of the machine in an intermittent manner. The cutter assembly, as the crucial component of the machine shown as a close-up view in Figure 2.5(b), consists of a two-axis orthogonal electro-hydraulic planar manipulator on which a pneumatically operated heavy-duty guillotine blade is mounted. There are two vision sub-systems, each consisting of a CCD camera, a frame grabber, and an image processor. The first vision-based feature extraction sub-system is located just before the cutter assembly to analyze the geometry of each fish, particularly the position of the gill with respect to the reference frame of the cutter assembly, in order to determine the optimal X-Y positioning coordinates for moving the cutter blade. The motion command is streamed to the hydraulic manipulator in the form of a step input, for each cut. The second vision sub-system is located beyond the cutter assembly to inspect the quality of a cut, and it feeds the extracted information to the hierarchical supervisory controller of the machine for product quality assessment and for automatic tuning and performance enhancement of the machine (see Chapter 7).

The main experimental focus in the present work is accurate control of the electro-hydraulic positioning system of the cutter assembly. The cutter assembly consists of two orthogonal prismatic axes (X and Y axis). Each axis is actuated by a hydraulic cylinder as shown in the schematic representation in Figure 2.6. The hydraulic pistons have a maximum



(a)



(b)

Figure 2.5: The Intelligent Iron Butcher. (a) A view of the entire machine; and (b) Close-up view of the electro-hydraulic manipulator.

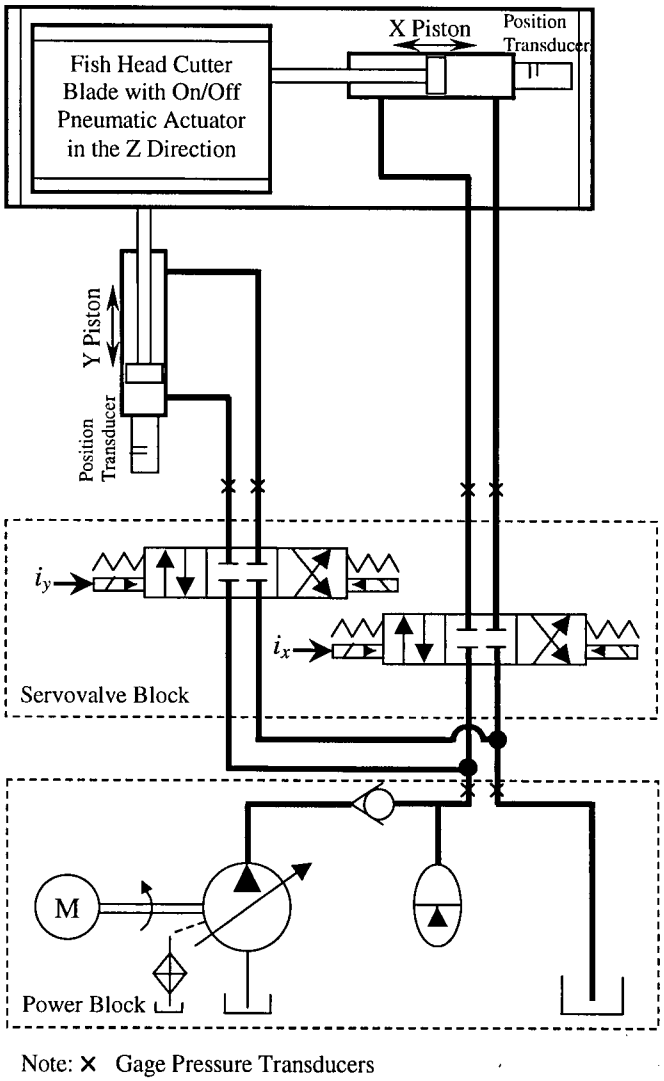


Figure 2.6: Schematic diagram of the electro-hydraulic manipulator.

stroke of 50.0 mm, and the fluid flow within the cylinder is controlled using a flow-controlled servo-valve, which has two stages; namely, the pilot stage which consists of a torque motor-actuated, double-nozzle flapper, and the booster stage consisting of a pressure actuated, spring-centered valve module with double spool configuration (Anderson and Li, 2002). The differential pressure from the pilot is applied across both spools of the boost stage and is balanced by the centering springs. This produces a spool position proportional to the differential pressure and therefore to the input current. The input current ranges between –42 mA to +42 mA. A custom built voltage-to-current converter is used to convert command voltages from the data acquisition board to input current within the pre-specified range. The operating bandwidth of this manipulator is estimated at 30 Hz.

Linear magnetostrictive displacement transducers are used for absolute position sensing of the hydraulic cylinders. They have a resolution of 0.025 mm when coupled with a 12-bit analog-to-digital converter. The pressures in the head and rod sides in each hydraulic cylinder are measured using gage fluid pressure transducers. The force applied to the blade assembly can be deduced from the chamber pressures of the cylinder.

2.4.2 System Identification and Model Validation

Since the networked control strategy investigated in the present thesis is model-based, sufficiently accurate parametric model for each axis of the electro-hydraulic manipulator is required. Linear, normalized, discrete-time transfer function and state-space models are acquired through experimental system identification method of correlation analysis (Ljung, 1987). Each axis of the system is excited by injecting a series of Pseudo-Random Binary Sequence (PRBS) current input u into the servo-amplifier of each axis. A choice of $|u| = 7$ mA at a PRBS step interval of 10 ms over a duration of 2048 steps sequence is found to be sufficiently exciting without causing the piston to hit its stroke limits and is used in the model identification procedure. The responses of the system recorded at 10 ms sampling intervals are the piston position y [mm], piston velocity $\dot{y}$ [mm/s], head-side pressure P_h [psi], and rod-side pressure P_r [psi]. Since the raw sensor readings are rather noisy, the position and pressure measurements are acquired at a sampling frequency of f_s = 1000 Hz and filtered through a low-pass 5th-order, discrete-time Butterworth filter with a cutoff frequency of 80 Hz, of the form:

$$x_{filtered} = \frac{b_{f0} + b_{f1}z^{-1} + b_{f2}z^{-2} + b_{f3}z^{-3} + b_{f4}z^{-4} + b_{f5}z^{-5}}{1 + a_{f1}z^{-1} + a_{f2}z^{-2} + a_{f3}z^{-3} + a_{f4}z^{-4} + a_{f5}z^{-5}} x_{raw} \tag{2.14}$$

in which $a_{f1} = -3.3780$, $a_{f2} = 4.7518$, $a_{f3} = -3.4397$, $a_{f4} = 1.2740$, $a_{f5} = -0.1924$, $b_{f0} = b_{f5} = 4.8944\times10^{-4}$, $b_{f1} = b_{f4} = 2.4472\times10^{-3}$, and $b_{f2} = b_{f3} = 4.8944\times10^{-3}$. Velocity is estimated from the position measurement through a low-pass filtered, discrete-time differentiator given as (Tafazoli, *et al.*, 1998):

$$\hat{\dot{y}} = \frac{1 - z^{-1}}{(\frac{1}{2}T_s + T_v) + (\frac{1}{2}T_s - T_v)z^{-1}} y \tag{2.15}$$

with the sampling interval $T_s = 1/f_s$, and the filter time constant $T_v = 1/500$ s.

The collected raw response data are pretreated to remove the high frequency disturbances beyond the frequency range of interest for the dynamics of the system, and also to remove the low frequency disturbances such as drift and steady-state offset. Drift is removed by passing the response data through a high-pass 5th-order Butterworth filter of the same structure as (2.14) with 0.5 Hz cut-off frequency, in which $a_{f1} = -4.8983$, $a_{f2} = 9.5985$, $a_{f3} = -9.4053$, $a_{f4} = 4.6085$, $a_{f5} = -0.9033$, $b_{f0} = -b_{f5} = 0.9504$, $b_{f1} = -b_{f4} = -4.7522$, and $b_{f2} = -b_{f3} = 9.5044$. The steady-state operating pressures of the cylinder (with the position and velocity at the origin) are 265.8 psi for the head-side, and 474.2 psi for the rod-side. In order to obtain normalized models of the system, the pretreated responses as well as the input data are scaled accordingly (to facilitate controller design and analysis, and to eliminate the need for agreement of units). The scaling for the input, position, velocity, and pressure data are 42 mA, 25 mm, 200 mm/s, and 500 psi, respectively.

Transfer Function Model Identification

Figure 2.7 shows the pretreated data set (before scaling) for the current input and the position response, as collected from the X-axis of the electro-hydraulic manipulator. It is used for discrete-time transfer function modeling of the following ARX (AutoRegressive eXternal input) model:

$$G_{uy}(z^{-1}) = z^{-n_c}\frac{B(z^{-1})}{A(z^{-1})} = z^{-n_c}\frac{b_0 + b_1 z^{-1} + \cdots + b_{n_b} z^{-n_b}}{1 + a_1 z^{-1} + \cdots + a_{n_a} z^{-n_a}} \tag{2.16}$$

The data set in the range of 300-800 samples is used for model building and the set in the range of 900-1700 samples is used for model cross-validation. A gross estimate of the impulse response function, obtained through the covariance function between u and y, is shown in Figure 2.8. It is observed that there is a lag of 2 sample intervals before a positive response is noted. Accordingly, a time delay of $n_c = 2$ is estimated in (2.16). The next step is to determine the model structure (order) by selecting n_a and n_b that will provide a "good" model fit, without arriving at a model order that is too high that would unnecessarily complicate the subsequent usage of the model; particularly in relation to the computational complexity, and the robustness and uncertainties in the model-based feedback controllers. Figure 2.9 shows the best

fit loss function criterion, in terms of the percentage of unexplained output variance (the ratio between the prediction error variance and the output variance, expressed as a percentage) calculated using the cross-validation data set, for different total number of parameters $(n_a + n_b + n_c)$. Only the model that has the best fit for the given total number of parameters is shown. In the present study, a total of 8 parameters, with $n_a = 2$, $n_b = 4$ and $n_c = 2$, is chosen after taking into consideration the high computational requirements of the model predictive control (MPC) approach for NCS. The resulting parameters of the ARX model of (2.16) for the X-axis of the electro-hydraulic manipulator are: $a_1 = -1.5620$, $a_2 = 0.5885$, $b_0 = 0.007691$, $b_1 = 0.01608$, $b_2 = 0.004824$, $b_3 = 0.006505$, and $b_4 = 0.002786$.

The resulting model is cross-validated through residual analysis (Figure 2.10), and comparison of predicted and measured output responses (Figure 2.11). Figure 2.10(a) shows the correlation function of residuals (prediction errors) from the output response, which is reasonably within a 99 % confidence level. In Figure 2.10(b), a clear negative correlation at lag 2, in agreement with the model, is observed in the cross-correlation function between the input and the output response. Figure 2.11 shows that the predicted output response from the identified model is able to sufficiently capture the dynamics of the measured output responses although the model seems to have a smaller gain, which can be easily compensated in controller design. The frequency response of the identified model is given in Figure 2.12, indicating a smooth stable model with an evidently low open-loop gain.

Similarly for the Y-axis of the electro-hydraulic manipulator, the following parameters of the ARX model are obtained: $a_1 = -1.66$, $a_2 = 0.6796$, $b_0 = 0.01153$, $b_1 = 0.02643$, $b_2 = 0.003392$, $b_3 = -0.01742$, and $b_4 = -0.005612$.

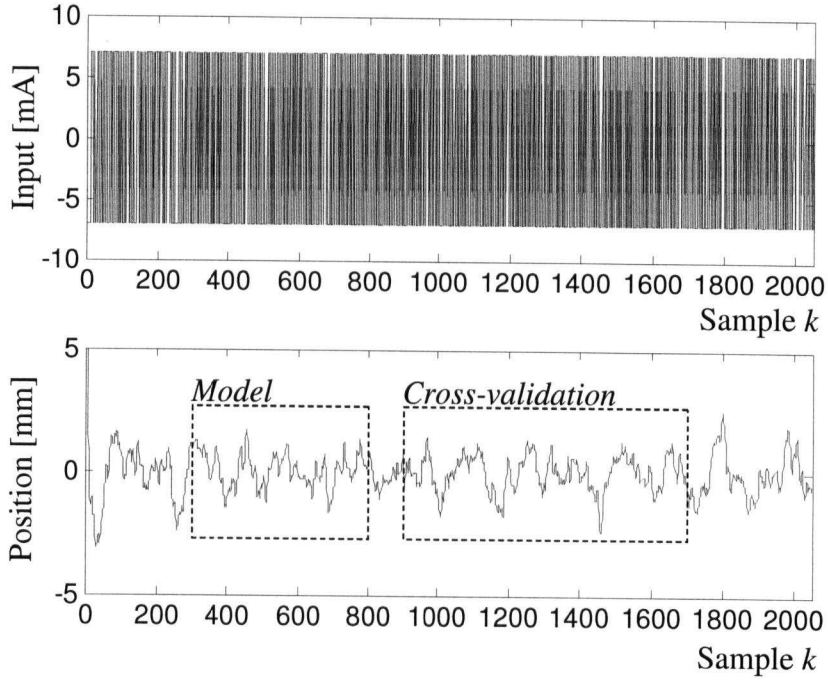


Figure 2.7: Pretreated input and position data collected for transfer function model identification. Figure shows the region used for correlation analysis and model cross-validation.

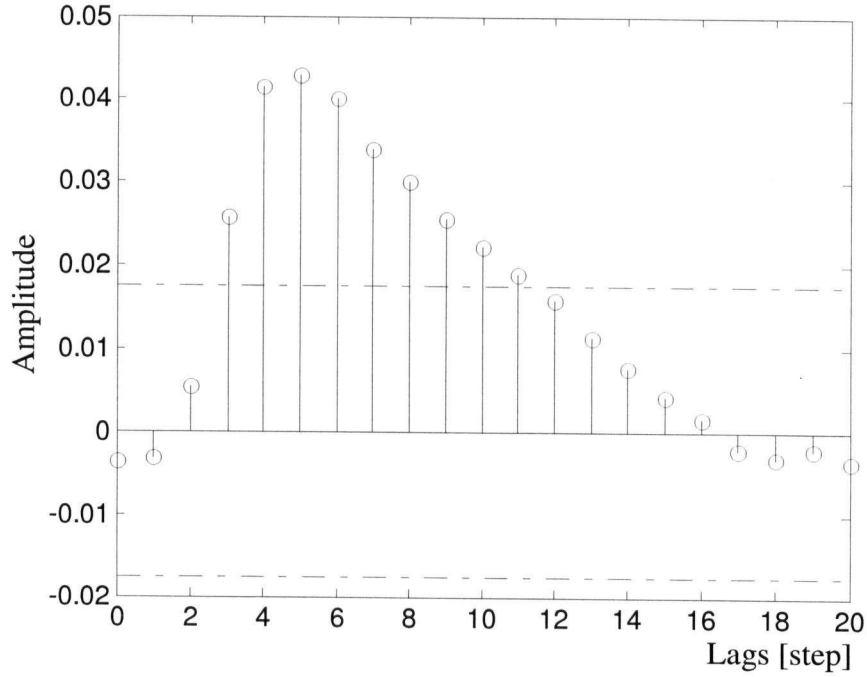


Figure 2.8: Scaled impulse response estimate for transfer function model identification.

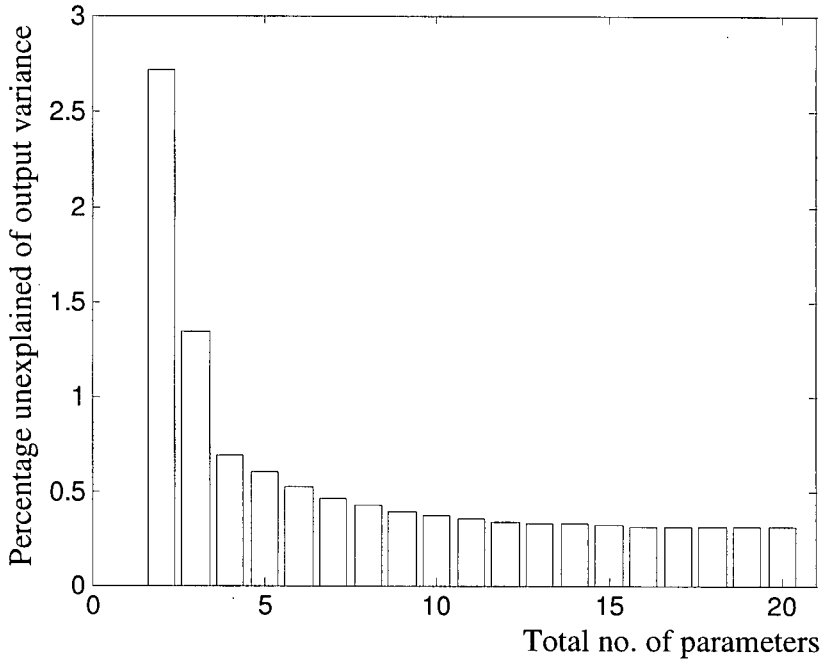


Figure 2.9: Model fitting for transfer function model identification.

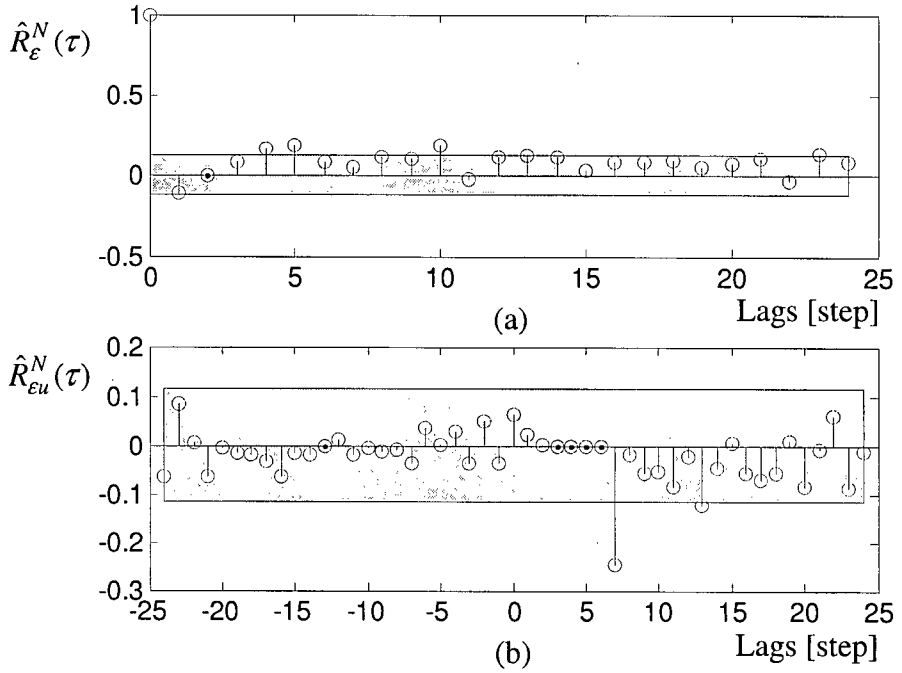


Figure 2.10: Residual analysis for transfer function model identification. Grey enclosures correspond to a confidence level of 99%. (a) Correlation function of residuals from output; and (b) Cross-correlation function between input and residuals from output.

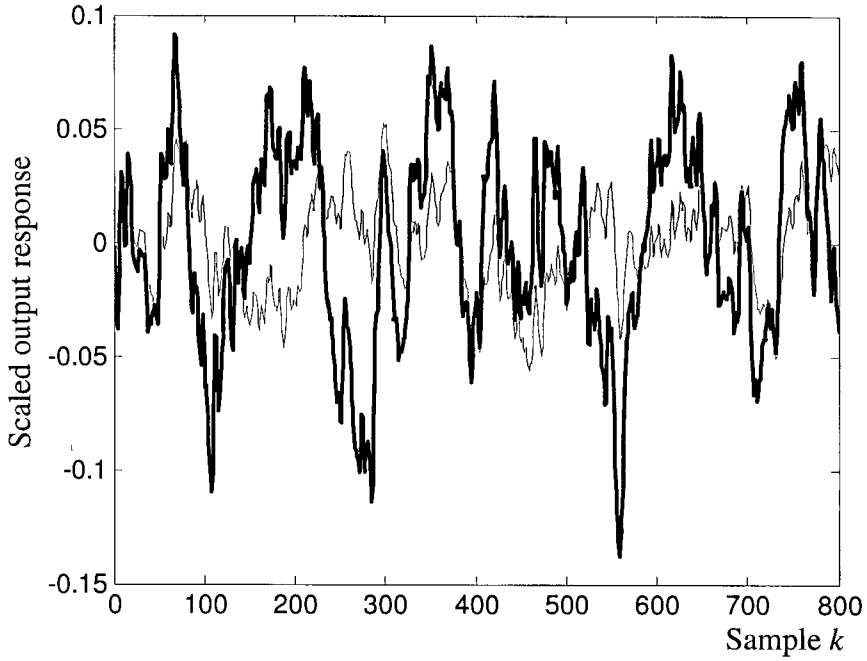


Figure 2.11: Transfer function model cross-validation through comparison of predicted scaled output of the model (thin-line) with the measured scaled output (thick-line) in the region indicated in Figure 2.7.

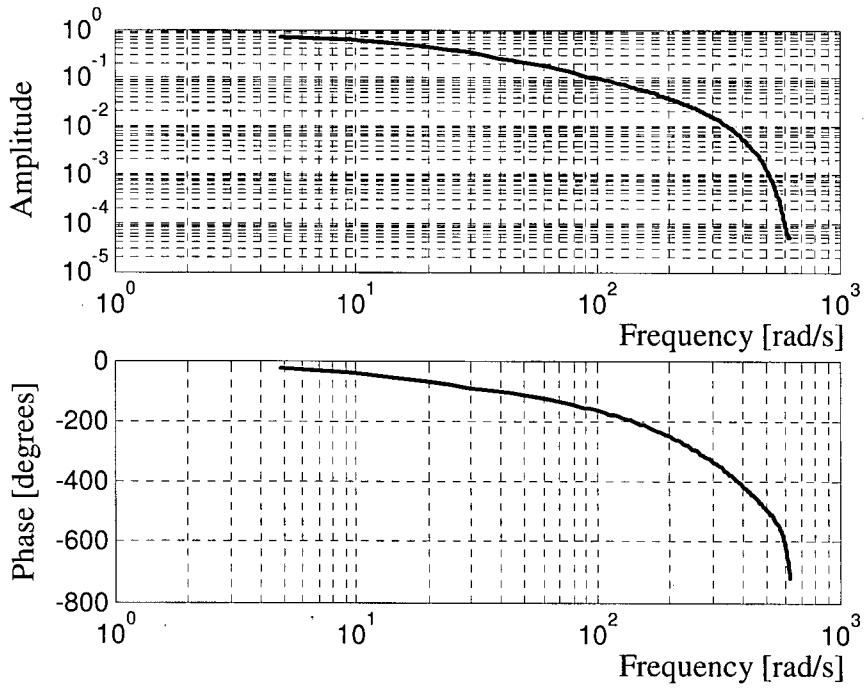


Figure 2.12: Bode plot of the identified transfer function model.

State-space Model Identification

The experimental evaluation of the networked control strategies developed in this study also requires a discrete-time state-space model of the electro-hydraulic manipulator, of the form:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \tag{2.17}$$

where $\boldsymbol{x}_k \in \mathbb{R}^n$ is the state vector of the system at time k, $\boldsymbol{u}_k \in \mathbb{R}^m$ is the control input vector at time k, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, and $\boldsymbol{B} \in \mathbb{R}^{n\times m}$. The state vector for the present system is defined as follows:

$$\boldsymbol{x} \triangleq [x_1 \quad x_2 \quad x_3 \quad x_4]^T = [y \quad \dot{y} \quad P_h \quad P_r]^T \tag{2.18}$$

The experimental identification of a normalized state-space model for the X-axis is carried out using a similar correlation analysis as before. Figure 2.13 shows the collected four pretreated state responses data (before scaling). The data set between 200 and 800 sample steps is used for modeling, and the resulting model is cross-validated with the data set between 1000 and 1400 sample steps. A black-box discrete-time state-space model is estimated using a standard prediction error, maximum likelihood method based on iterative minimization of a loss function criterion similar to that used in the transfer function modeling. The Matlab® System Identification toolbox is employed to facilitate the model estimation giving the state-model matrices of (2.17) as

$$\boldsymbol{A} = \begin{bmatrix} 1.06730 & -0.10423 & 0.13156 & 0.41143 \\ -0.02484 & 0.89404 & -1.51910 & 0.45326 \\ -0.00123 & 0.23267 & 1.15180 & -0.94321 \\ 0.30104 & 0.18348 & 0.19487 & 0.28778 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 0.00138 \\ 0.01027 \\ -0.00862 \\ -0.02929 \end{bmatrix}$$

The eigenvalues of this open-loop model are located at 0.7457±j0.3394 and 0.9547±j0.0791.

Figure 2.14 shows the autocorrelation of the state residuals (prediction errors) and the cross-correlation between the input and the state residuals for each of the four states. The horizontal dashed bars indicate a confidence level of 99 %. Although acceptable, some correlation is observed for most of the states. This is inevitable for a relatively low-order four state model fitted to represent a nonlinear system. It is evident that the internal immeasurable states are neglected. In order to reaffirm that the identified model is sufficient to capture the

dynamics of the system, the state responses of the model are compared with the actual measured state responses, as illustrated in Figure 2.15.

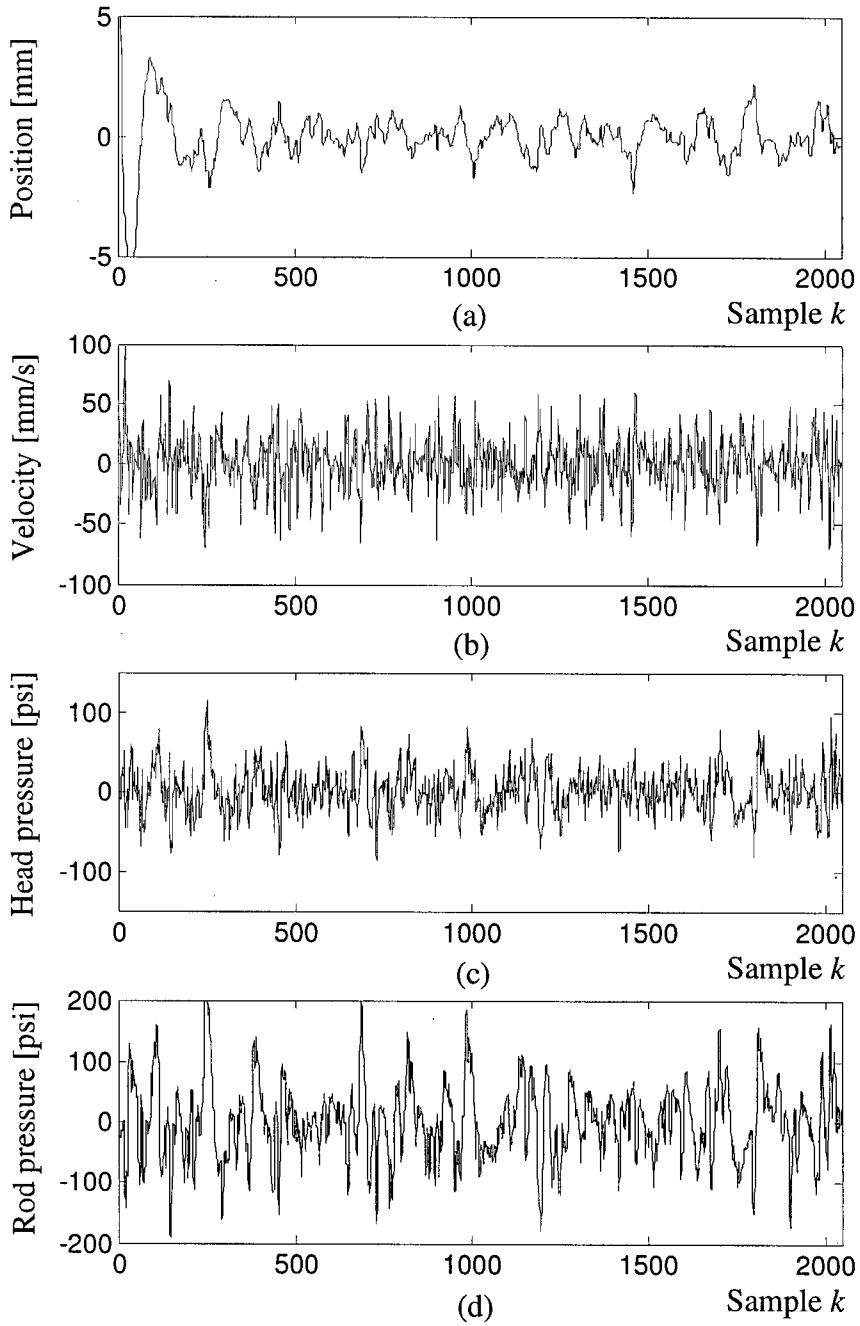


Figure 2.13: Pretreated state response data collected for the state-space model identification. (a) Position response; (b) Velocity response; (c) Head-side pressure; and (d) Rod-side pressure.

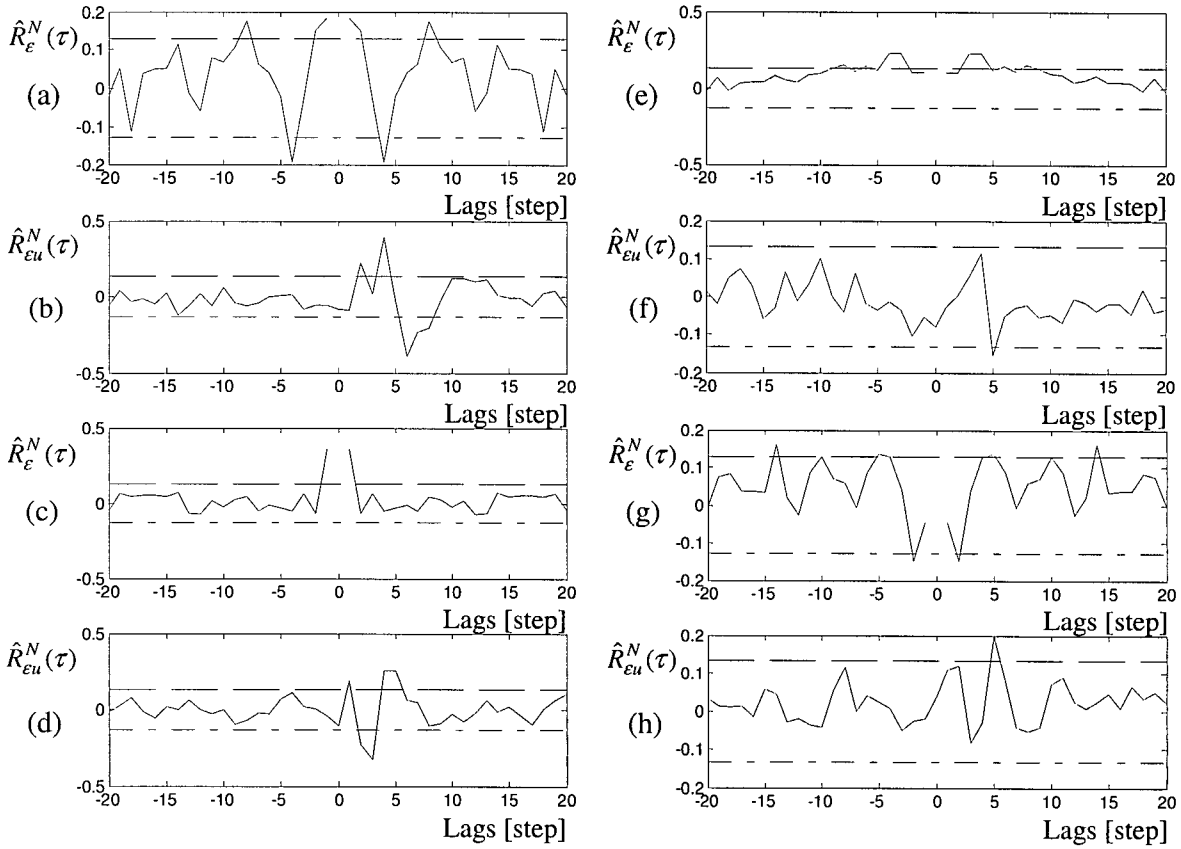


Figure 2.14: Residual analysis for state-space model identification. Horizontal dashed bars indicate 99% confidence levels. (a) Autocorrelation of residuals for state x_1; (b) Cross-correlation function between input and residuals from state x_1; (c) Autocorrelation of residuals for state x_2; (d) Cross-correlation function between input and residuals from state x_2; (e) Autocorrelation of residuals for state x_3; (f) Cross-correlation function between input and residuals from state x_3; (g) Autocorrelation of residuals for state x_4; and (h) Cross-correlation function between input and residuals from state x_4.

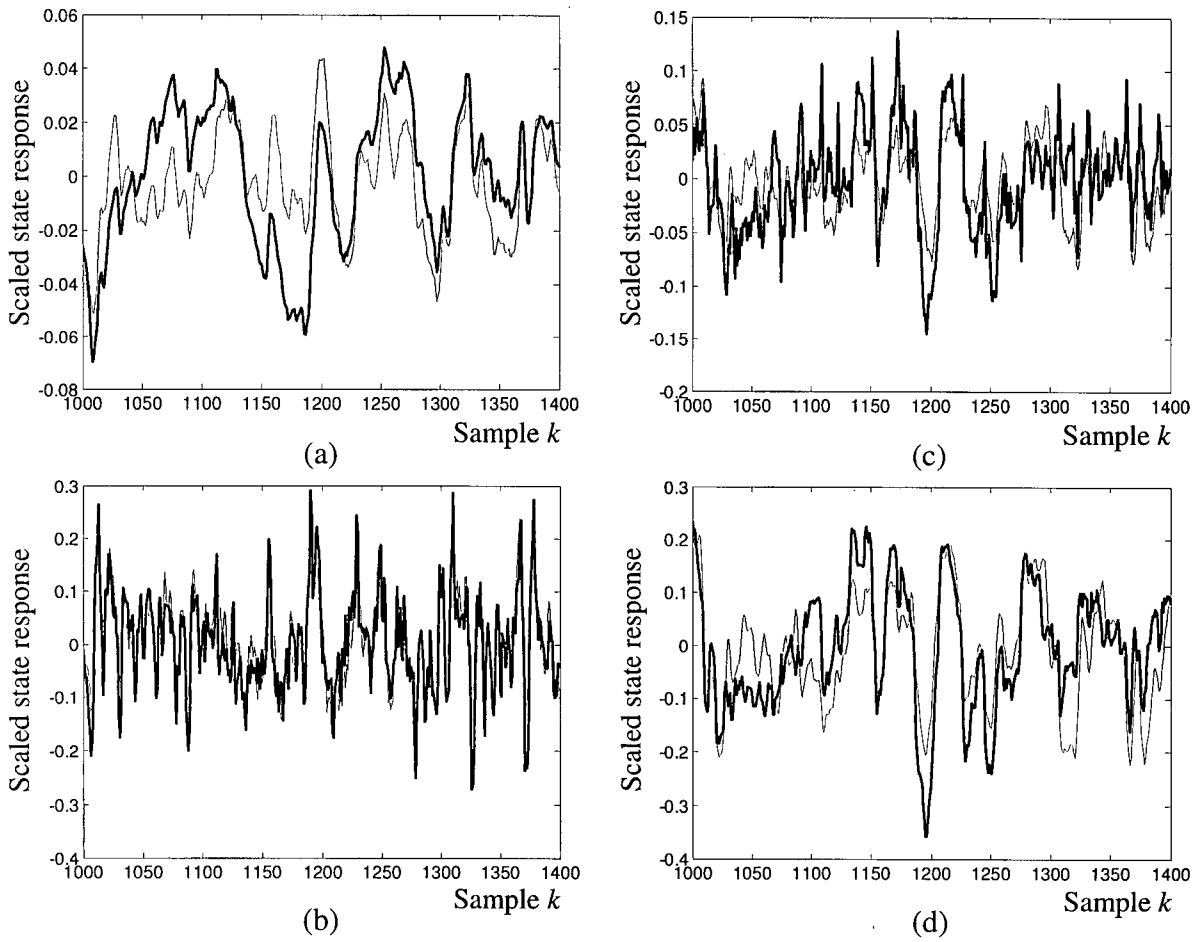


Figure 2.15: State-space model cross-validation through comparison of simulated scaled response of the model (thin-line) with the measured scaled response (thick-line). (a) State x_1; (b) State x_2; (c) State x_3; and (d) State x_4.

2.5 Summary

A control network architecture for networked control systems (NCS) was developed within an Ethernet network. The hardware interconnection and the software entities, particularly the client-server communication framework, were developed and implemented in detail. The integration of a precise off-line clock synchronization algorithm based on message passing and least-square line fitting would allow real-time implementation of feedback control strategies on high-speed servo systems over a communication network. Although plagued with nondeterministic communication delay, packet losses, vacant sampling, and data mis-synchronization, Ethernet has proved to be a viable communication network for control

applications. The inner working of a fish-processing machine which is the key experimental evaluation platform in this study was described with the emphasis of the electro-hydraulic manipulator sub-system. Sufficiently accurate discrete-time transfer function and state-space models of the manipulator were identified.

Chapter 3

Predictive Networked Control with Future Input Buffering

Distributed control networks encounter non-deterministic delays in data communication between sensors, actuators, and controllers including direct feedback control and higher-level supervisor control. This chapter presents a novel strategy, which extends the Model Predictive Control (MPC) algorithm to compensate for these data-transmission delays. In this context, the communication lines between the sensors and the controller, and the controller and the actuators are considered. The strategy developed in this chapter incorporates a minimum effort estimator to estimate the missing or delayed sensor data, and a variable horizon, adaptive predictive controller to predict the required future control actions to drive the plant to track a desired reference trajectory. Action buffers are introduced at the actuators to sequence the future control actions. The developed scheme is implemented on the dual-axis hydraulic position system of the industrial fish-processing machine, which was described in Chapter 2.

The present chapter focuses on developing control strategies for networked control systems (NCS), termed "networked control strategies" in this thesis, which will render the network transmission delays transparent to feedback controller design. The chapter is organized as follows. The next section focuses on the development of a novel control strategy based on efficient estimation and an extension of the multivariable predictive control problem. Section 3.2 addresses the implementation issues of real-time networked control. The performance and effectiveness of the developed strategy is evaluated in detail in Section 3.3, where key experimental results are presented and discussed.

3.1 The Networked Control Strategy

The transmission delay incurred in a communication network can fluctuate somewhat randomly depending on the nature and the level the information transmission activity and the level of congestion in the network at that particular time. This causes difficulty in representing the delay with a simple and deterministic model that can be used in feedback controller design.

The frequent and unpredictable occurrence of data losses and vacant sampling further complicate the problem. The strategy developed in this chapter simplifies the design of a feedback controller by requiring only a model of the system under control, excluding the communication network. Specifically, no prior information of the transmission characteristics of the network is required.

3.1.1 Transparency of Transmission Delays

Figure 3.1 gives a simplified schematic diagram of the networked control strategy developed here. It consists of a plant (the system under control), and the sensor and controller nodes, which are located at a remote site, and connected through a network to an MPC controller module located at a local monitoring and control station. The data transmission delays of sensor-to-controller segment and controller-to-actuator segment of the networked control system (NCS) are represented by τ_{sc} and τ_{ca}, respectively. In this approach, it is assumed that the sensor node, the actuator node, and the controller have an identical sampling period (sampling interval) h. The internal clocks of the three nodes are synchronized using the clock synchronization algorithm presented in Section 2.2.2.

The sensor node samples the outputs (responses), combines them into one data packet, and

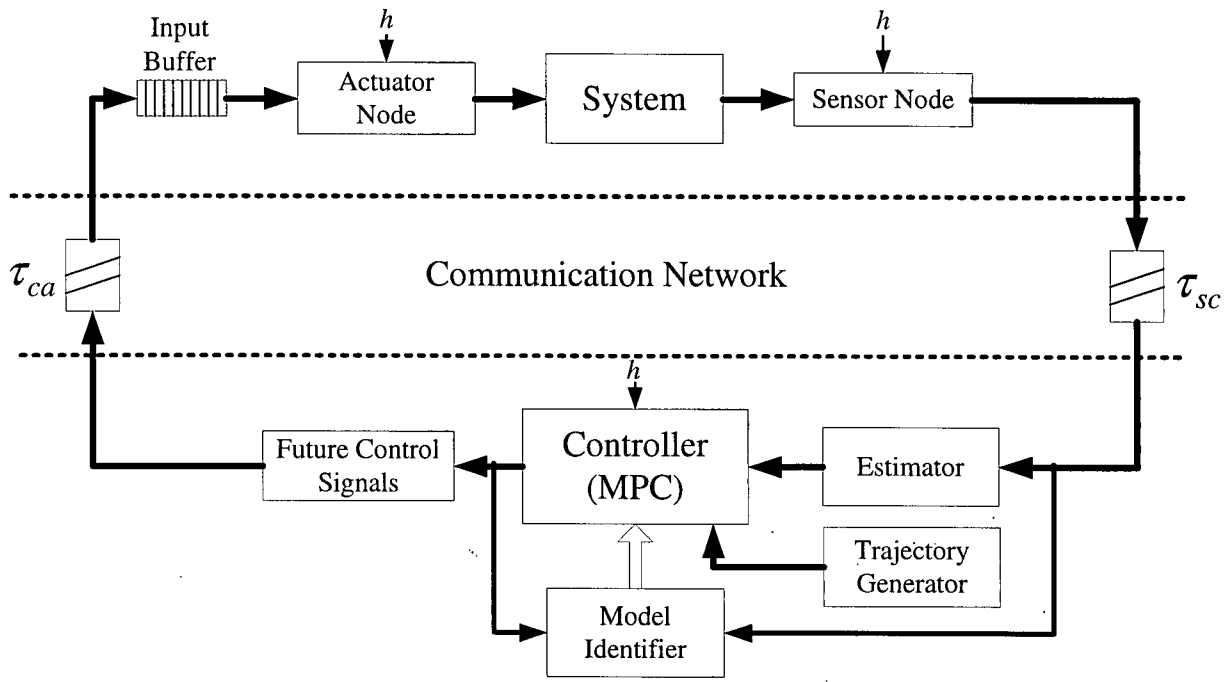


Figure 3.1: The developed networked control system (NCS).

then transmits the data packet to the controller node. Each data packet is "time-stamped" so that the delay information can be extracted at the controller node so as to indicate how "old" the received measurement is. At the controller, the received sensory data are stored in a shared memory (or, a database). This stored data history is used by the model predictive controller (MPC) and the model identifier. At each sampling interval, τ_{sc} is determined first. The number of measurements that need to be estimated by the minimum effort estimator to fill in the missing sensory data up to the current k-th sampling instant, is determined using τ_{sc} and the "age" of the latest sensor data. Then the output of the estimated system is updated into the shared memory. Besides using in the subsequent computations of control actions, the predicted output can be used in the next sample instant if there is an event of vacant sample or data loss.

For better visualization, Figure 3.2 illustrates a simplified example of the basic mechanism of the developed networked control strategy at a particular time step k. Assume that the sensor-to-controller transmission delay (discrete) is $\tau_{sc}^k = \lceil \tau_{sc} \rceil = 3h$ and the controller-to-

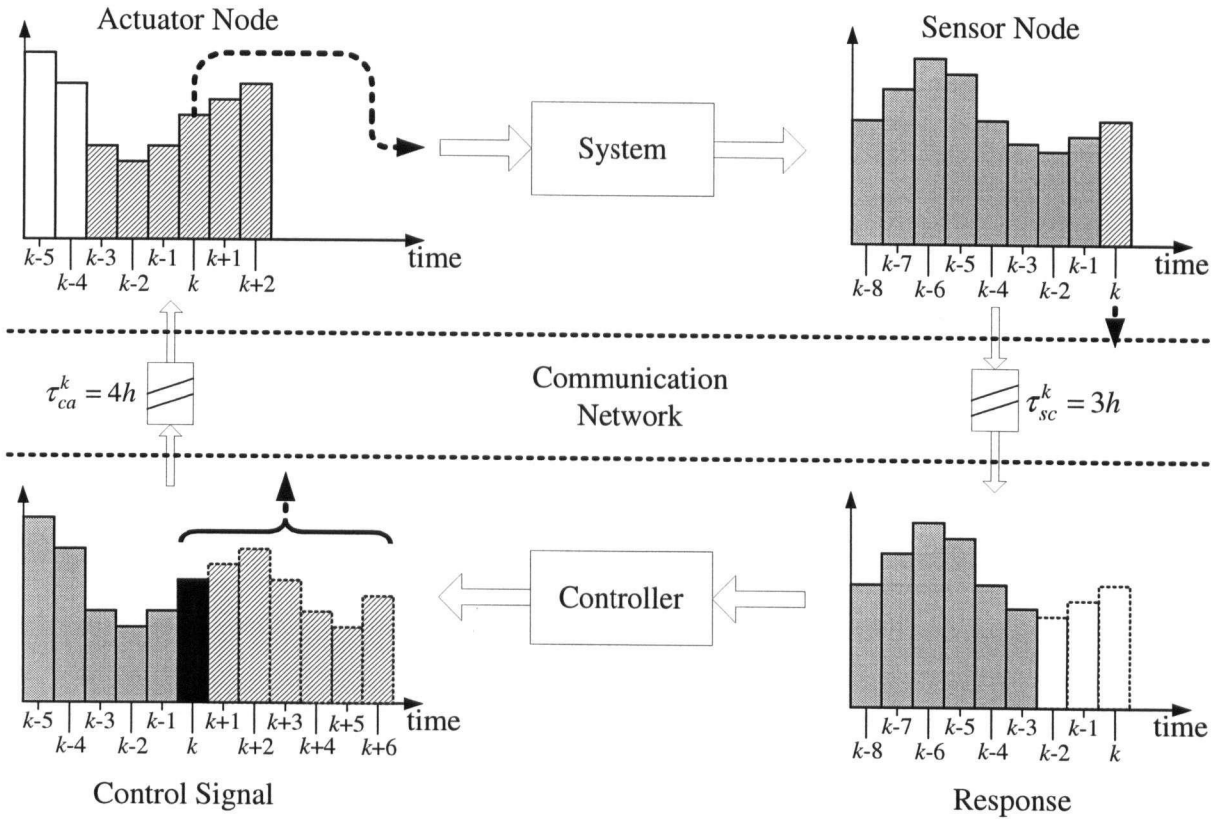


Figure 3.2: An illustrative example of the developed networked control strategy.

actuator transmission delay (discrete) is $\tau_{ca}^{k} = \lceil \tau_{ca} \rceil = 4h$. At the time instant k, the sensor node samples the system outputs, combines them into one data packet, time-stamps it, and sends it through the network. Now this data will take some time ($3h$) to reach the controller, so at the same instant k, the controller will need to predict the 3 missing sensor data values up to time k. Using the past and predicted responses, the controller will then calculate the current and future control signals. Based on the current transmission delays information, which is known from the "age" the latest received data, the required prediction horizon is deduced. With $\tau_{ca}^{k} = 4h$, the prediction horizon is set to 6 steps ahead (see Section 3.1.4 for the factors involved in deciding the lengths of prediction horizons). So, the current and the 6 future control signals will be time-stamped and transmitted to the actuator node. Meanwhile, at the same instant, the buffer before the actuator would contain a string of control signals that had arrived earlier. These 6 predicted control inputs (indicated by hatched columns in the figure) were predicted at time $k-4$. Accordingly, at this instant, the control action corresponding to the discrete time step k is selected from the string and sent to the actuator. This cycle is repeated at every sampling interval.

3.1.2 Minimum Effort Estimator

A minimum effort estimator similar to that in (Kaynak, *et al.*, 1991) is used to estimate the delayed or missing sensor data. It is a polynomial type estimator and employs a multivariable input-output model of the CARIMA (Controlled-Auto-Regressive-Integrated-Moving-Average) form (Clarke, *et al.*, 1987):

$$\boldsymbol{A}(z^{-1})\boldsymbol{y}(k) = \boldsymbol{B}(z^{-1})\boldsymbol{u}(k-1) + \frac{1}{\Delta}\boldsymbol{C}(z^{-1})\boldsymbol{e}(k) \tag{3.1}$$

where $\boldsymbol{u}(k) \in \mathbb{R}^m$ is the control input vector, $\boldsymbol{y}(k) \in \mathbb{R}^p$ is the output vector, $\boldsymbol{A}(z^{-1})$ and $\boldsymbol{C}(z^{-1})$ are $p \times p$ monic polynomial matrices, $\boldsymbol{B}(z^{-1})$ is a $p \times m$ polynomial matrix, and $\Delta = 1 - z^{-1}$ is the difference operator. Note that this model represents the plant under control, and does not include any dynamics incurred by the communication network. The CARIMA model is chosen to coincide with the prediction model used in the predictive control part (see Section 3.1.3). Hence, the online estimation of only one set of model parameters is needed in the indirect adaptive control case (see Section 3.1.4). It is assumed that $\boldsymbol{e}(k)$ is white noise

giving $\boldsymbol{C}(z^{-1}) = \boldsymbol{I}_p$, where $\boldsymbol{I}_i \in \mathbb{R}^{i\times i}$ represents an identity matrix. Let the number of delayed or missing sensor data samples at the controller, at the k-th sampling instant be $\tau_{sc}^k = \lceil \tau_{sc} \rceil$. Using (3.1), the estimated value for $\boldsymbol{y}(k-\tau_{sc}^k+1)$ can be determined as:

$$\hat{\boldsymbol{y}}(k-\tau_{sc}^k+1) = z(\boldsymbol{I}_p - \Delta\boldsymbol{A}(z^{-1}))\boldsymbol{y}(k-\tau_{sc}^k) + \boldsymbol{B}(z^{-1})\Delta\boldsymbol{u}(k-\tau_{sc}^k) \tag{3.2}$$

Define the estimation error as $\boldsymbol{\varepsilon}(k) = \boldsymbol{y}(k) - \hat{\boldsymbol{y}}(k)$. Using this estimation error as the correction term, the optimal output estimate is obtained as:

$$\tilde{\boldsymbol{y}}(k-\tau_{sc}^k+1) = \hat{\boldsymbol{y}}(k-\tau_{sc}^k+1) + \boldsymbol{\varepsilon}(k-\tau_{sc}^k) \tag{3.3}$$

Continuing the same way, the delayed output can be estimated up to the k-th instant; i.e.,

$$\tilde{\boldsymbol{y}}(k-\tau_{sc}^k+i) = \hat{\boldsymbol{y}}(k-\tau_{sc}^k+i) + \boldsymbol{\varepsilon}(k-\tau_{sc}^k) \text{ for } i = 1,2,...,\tau_{sc}^k \tag{3.4}$$

where $\hat{\boldsymbol{y}}(k-\tau_{sc}^k+i) = \tilde{\boldsymbol{y}}(k-\tau_{sc}^k+i-1)$ for $i>1$, to replace the unknown values of the output at instant $(k-\tau_{sc}^k)$.

In the current test system, which is the two-axis electro-hydraulic manipulator of the fish-processing machine (see Section 2.4), $\boldsymbol{A}(z^{-1})$ is of second degree and $\boldsymbol{B}(z^{-1})$ is of fifth degree (from the CARIMA equivalent of (2.16)), with $m=2$ and $p=2$; i.e.,

$$\boldsymbol{A}(z^{-1}) = \boldsymbol{I}_2 + \boldsymbol{A}_{1,2\times2}z^{-1} + \boldsymbol{A}_{2,2\times2}z^{-2} \tag{3.5}$$

$$\boldsymbol{B}(z^{-1}) = \boldsymbol{B}_{0,2\times2} + \boldsymbol{B}_{1,2\times2}z^{-1} + \boldsymbol{B}_{2,2\times2}z^{-2} + \boldsymbol{B}_{3,2\times2}z^{-3} + \boldsymbol{B}_{4,2\times2}z^{-4} + \boldsymbol{B}_{5,2\times2}z^{-5} \tag{3.6}$$

Application of (3.4) yields:

$$\begin{aligned}\tilde{\boldsymbol{y}}(k-\tau_{sc}^k+i) = {} & (\boldsymbol{I}_2 - \boldsymbol{A}_1)\tilde{\boldsymbol{y}}(k-\tau_{sc}^k+i-1) + (\boldsymbol{A}_1 - \boldsymbol{A}_2)\tilde{\boldsymbol{y}}(k-\tau_{sc}^k+i-2) \\ & + \boldsymbol{A}_2\tilde{\boldsymbol{y}}(k-\tau_{sc}^k+i-3) + \boldsymbol{B}_0\Delta\boldsymbol{u}(k-\tau_{sc}^k+i-1) + \boldsymbol{B}_1\Delta\boldsymbol{u}(k-\tau_{sc}^k+i-2) \\ & + \boldsymbol{B}_2\Delta\boldsymbol{u}(k-\tau_{sc}^k+i-3) + \boldsymbol{B}_3\Delta\boldsymbol{u}(k-\tau_{sc}^k+i-4) + \boldsymbol{B}_4\Delta\boldsymbol{u}(k-\tau_{sc}^k+i-5) \\ & + \boldsymbol{B}_5\Delta\boldsymbol{u}(k-\tau_{sc}^k+i-6) + \boldsymbol{\varepsilon}(k-\tau_{sc}^k)\end{aligned} \tag{3.7}$$

for $i = 1,2,...,\tau_{sc}^k$ with $\tilde{\boldsymbol{y}}(k-\tau_{sc}^k+i-n_a) = \boldsymbol{y}(k-\tau_{sc}^k+i-n_a)$ for $i \le n_a$, where $n_a = \deg(\boldsymbol{A}(z^{-1}))$. Here, $\Delta\boldsymbol{u}(k-\tau_{sc}^k+1)$, $\Delta\boldsymbol{u}(k-\tau_{sc}^k+2)$, ..., $\Delta\boldsymbol{u}(k-1)$ are calculated at previous time steps, by the predictive controller discussed in the next section.

3.1.3 Multivariable Predictive Control

Once the delayed or missing system responses are estimated, the next step of the networked control strategy that is develop here is to predict the required k-step-ahead future control signals that will drive the system to track a desired trajectory. An extension to the multivariable Generalized Predictive Control or GPC (Zelinka, 1997; Camacho and Bordons, 1998) is used for this purpose. The derivation of the GPC algorithm is outlined now to set the groundwork for the development of the variable predictive horizon and adaptive extension, which are required in the present scheme. As indicated in the previous section, the CARIMA system model (3.1) is adopted. Here an optimal set of current and future changes in control signals: $\Delta \boldsymbol{u}(k+j)$ for $j=0,1,...,H_u$ is sought to continuously minimize the quadratic cost function

$$V_k(H_1,H_2,H_u)=\sum_{j=H_1}^{H_2}\left\|\boldsymbol{y}^*(k+j\,|\,k)-\boldsymbol{r}(k+j)\right\|_{\boldsymbol{Q}}^2+\sum_{j=1}^{H_u}\left\|\Delta\boldsymbol{u}(k+j-1)\right\|_{\boldsymbol{R}}^2 \tag{3.8}$$

where $\boldsymbol{y}^*(k+j\,|\,k)$ is the j-step-ahead predicted system outputs based on the history up to the time instant k, and $\boldsymbol{r}(k+j)$ are the desired (reference) future trajectories. H_1, H_2, and H_u are, respectively, the minimum and maximum prediction horizons, and the control horizon, where $1\le H_1\le H_2$ and $H_u\le H_2$. The weighting sequence matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are diagonal and positive definite.

The j-step-ahead outputs need to be predicted for use in the control law. The Diophantine equation:

$$\boldsymbol{I}_m=\boldsymbol{E}_j^k(z^{-1})\Delta\boldsymbol{A}(z^{-1})+z^{-j}\boldsymbol{F}_j^k(z^{-1}) \tag{3.9}$$

and (3.1) yield,

$$\boldsymbol{y}(k+j)=\boldsymbol{F}_j^k(z^{-1})\boldsymbol{y}(k)+\boldsymbol{E}_j^k(z^{-1})\boldsymbol{B}(z^{-1})\Delta\boldsymbol{u}(k+j-1)+\boldsymbol{E}_j^k(z^{-1})\boldsymbol{e}(k+j) \tag{3.10}$$

Note that $\deg(\boldsymbol{E}_j^k(z^{-1}))=j-1$ and $\deg(\boldsymbol{F}_j^k(z^{-1}))=\deg(\boldsymbol{A}(z^{-1}))$. Now $\boldsymbol{y}^*(k+j\,|\,k)$ is found by taking the expected value of (3.10) and using the fact that $\boldsymbol{e}(k)$ is zero-mean white noise ($\mathrm{E}[\boldsymbol{e}(k)]=0$); thus,

$$\boldsymbol{y}^*(k+j\,|\,k)=\boldsymbol{F}_j^k(z^{-1})\boldsymbol{y}(k)+\boldsymbol{E}_j^k(z^{-1})\boldsymbol{B}(z^{-1})\Delta\boldsymbol{u}(k+j-1) \tag{3.11}$$

The second term of (3.11) consists of the past (up to time k) and future inputs $\Delta \boldsymbol{u}$. This term can be separated into two parts by introducing the new Diophantine equation:

$$\boldsymbol{E}_j^k(z^{-1})\boldsymbol{B}(z^{-1}) = \boldsymbol{G}_j^k(z^{-1}) + z^{-j}\bar{\boldsymbol{H}}_j^k(z^{-1}) \tag{3.12}$$

which yields

$$\begin{aligned} &\boldsymbol{y}^*(k+j\,|\,k) \\ &\quad = \boldsymbol{G}_j^k(z^{-1})\Delta\boldsymbol{u}(k+j-1) + \bar{\boldsymbol{H}}_j^k(z^{-1})\Delta\boldsymbol{u}(k-1) + \boldsymbol{F}_j^k(z^{-1})\boldsymbol{y}(k) \\ &\quad = \boldsymbol{G}_j^k(z^{-1})\Delta\boldsymbol{u}(k+j-1) + \bar{\boldsymbol{F}}_j^k \end{aligned} \tag{3.13}$$

where $\bar{\boldsymbol{F}}_j^k = \bar{\boldsymbol{H}}_j^k(z^{-1})\Delta\boldsymbol{u}(k-1) + \boldsymbol{F}_j^k(z^{-1})\boldsymbol{y}(k)$ is the free response term. This term can be easily computed recursively by utilizing (3.1) and (3.2) as

$$\bar{\boldsymbol{F}}_{j+1}^k = z(\boldsymbol{I} - \Delta\boldsymbol{A}(z^{-1}))\bar{\boldsymbol{F}}_j^k + \boldsymbol{B}(z^{-1})\Delta\boldsymbol{u}(k+j) \tag{3.14}$$

with $\bar{\boldsymbol{F}}_0^k = \boldsymbol{y}(k)$ and $\Delta\boldsymbol{u}(k+j) = 0$ for $j \geq 0$. Considering the three prediction horizons H_1, H_2, and H_u, where $j = H_2 - H_1$, the matrix form of the j-step-ahead prediction is obtained as

$$\boldsymbol{Y}_{H_{12}}^* = \boldsymbol{G}_{H_{12u}}\Delta\boldsymbol{U}_{H_u} + \bar{\boldsymbol{F}}_{H_{12}} \tag{3.15}$$

where

$$\boldsymbol{Y}_{H_{12}}^* = \begin{bmatrix} \boldsymbol{y}^*(k+H_1)^T & \boldsymbol{y}^*(k+H_1+1)^T & \ldots & \boldsymbol{y}^*(k+H_2)^T \end{bmatrix}^T;$$

$$\Delta\boldsymbol{U}_{H_u} = \begin{bmatrix} \Delta\boldsymbol{u}(k)^T & \Delta\boldsymbol{u}(k+1)^T & \ldots & \Delta\boldsymbol{u}(k+H_u-1)^T \end{bmatrix}^T;$$

$$\bar{\boldsymbol{F}}_{H_{12}} = \begin{bmatrix} \boldsymbol{F}_{H_1}^{k\,T} & \boldsymbol{F}_{H_1+1}^{k\,T} & \ldots & \boldsymbol{F}_{H_2}^{k\,T} \end{bmatrix}^T; \text{ and}$$

$$\boldsymbol{G}_{H_{12u}} = \begin{bmatrix} \boldsymbol{G}_{H_1-1} & \boldsymbol{G}_{H_1-2} & \cdots & \boldsymbol{G}_{H_1-H_u} \\ \boldsymbol{G}_{H_1} & \boldsymbol{G}_{H_1-1} & \cdots & \boldsymbol{G}_{H_1-H_u+1} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{G}_{H_2-1} & \boldsymbol{G}_{H_2-2} & \cdots & \boldsymbol{G}_{H_2-H_u} \end{bmatrix} \in \mathbb{R}^{p(H_2-H_1+1)\times mH_u}$$

with $\boldsymbol{G}_q = \boldsymbol{0}_{p\times m}$ for $q < 0$. Using the matrix notation of (3.15), the GPC quadratic cost function can be written as:

$$V_k = (\boldsymbol{G}_{H_{12u}} \Delta \boldsymbol{U}_{H_u} + \bar{\boldsymbol{F}}_{H_{12}} - \boldsymbol{r})^T \bar{\boldsymbol{Q}} (\boldsymbol{G}_{H_{12u}} \Delta \boldsymbol{U}_{H_u} + \bar{\boldsymbol{F}}_{H_{12}} - \boldsymbol{r}) + \Delta \boldsymbol{U}_{H_u}{}^T \bar{\boldsymbol{R}} \Delta \boldsymbol{U}_{H_u} \tag{3.16}$$

in which $\bar{\boldsymbol{Q}} = \text{diag}(\boldsymbol{Q}_1, \boldsymbol{Q}_2, ..., \boldsymbol{Q}_{H_2 - H_1 + 1})$ and $\bar{\boldsymbol{R}} = \text{diag}(\boldsymbol{R}_1, \boldsymbol{R}_2, ..., \boldsymbol{R}_{H_u})$. By performing either quadratic programming or analytical differentiation to minimize V_k with respect to $\Delta \boldsymbol{U}_{H_u}$, the optimal sequence of control actions $\Delta \boldsymbol{U}_{H_u}$ are obtained as

$$\Delta \boldsymbol{U}_{H_u} = \left[\boldsymbol{G}_{H_{12u}}{}^T \bar{\boldsymbol{Q}} \boldsymbol{G}_{H_{12u}} + \bar{\boldsymbol{R}} \right]^{-1} \boldsymbol{G}_{H_{12u}}{}^T \bar{\boldsymbol{Q}} (\boldsymbol{r} - \bar{\boldsymbol{F}}_{H_{12}}) \tag{3.17}$$

In a typical receding-horizon GPC strategy, only $\Delta \boldsymbol{u}(k)$ is extracted from $\Delta \boldsymbol{U}_{H_u}$. However, the present networked control strategy uses all the elements of $\Delta \boldsymbol{U}_{H_u}$ to compensate for delayed or missing control signals in the controller-to-actuator communication lines. These current and future control signals will be time-stamped and transmitted back to the actuator node. When the future control signals in $\Delta \boldsymbol{U}_{H_u}$ are to be used, the weighting sequences of $\bar{\boldsymbol{Q}}$ and $\bar{\boldsymbol{R}}$ cannot be set too high, in order to maintain closed-loop stability. In networked control, it is found that satisfactory performance is achieved only for a certain range of weighting sequences depending on the level of network congestion (see Section 3.4).

3.1.4 Variable Prediction Horizon

In order to reduce network traffic and the controller computation load, the number of future steps of a control signal should be as few as possible. As a result, the size of the data packets traveling from the controller back to the actuators will be reduced as well, since larger data packets consume more network bandwidth, resulting in longer transmission delays. This objective is achieved by varying (minimizing) the prediction horizon of the GPC strategy. The required number of future control steps is determined by the variable prediction horizon, which is in turn determined from the "latest" known controller-to-actuator delay τ_{ca} (Note: sensor-to-controller data packets contain the τ_{ca} information which is determined by subtracting the time the actuator received a data packet by the time stamped in that particular data packet, at the actuator node and then passed on to the sensor node to be sent back to the controller node). It should be noted that the size of the sensor-to-controller data packets is fixed since they only carry information of sensor readings, time-stamps and some required states of the system. This

is in contrast to the size of the controller-to-actuator data packets, which is variable, depending on the length of control horizon as discussed in the following.

In the variable horizon strategy introduced here, the control horizon is varied as $H_u^k = \alpha + \tau_{ca}^k + \beta$ where $\tau_{ca}^k = \lceil \tau_{ca} \rceil$, α is the minimum control horizon required for a pre-specified tracking performance, and β is set as a step ahead safety measure to compensate for vacant sampling or sudden continuous loss of data packets. Through the experimental work carried out on an Ethernet network in this study, it is found that setting $1 \le \beta \le 5$ is sufficient. The value of β can be changed accordingly to characterize a particular type of network without affecting the performance of the control strategy. This is because the performance will improve with a longer prediction horizon. The only concern is the extra computation load for the control computer. Noting that $H_u \le H_2$, what is used here is $H_2 = H_u + 1$, with H_1 set to the physical dead-time of the system. Without loss of stability, H_1 can be set to 1 (Clarke, *et al.*, 1987).

Every time the prediction horizons are varied, the free response term $\bar{\boldsymbol{F}}_{H_{12}}$ and the dynamic matrix $\boldsymbol{G}_{H_{12u}}$ have to be dimensionally restructured. The term $\bar{\boldsymbol{F}}_{H_{12}}$ can be easily restructured by recursively evaluating (3.14). However, the adaptive restructuring of $\boldsymbol{G}_{H_{12u}}$ can only be formulated after the order of the plant is determined. Let $\boldsymbol{G}_q$ be the matrix element of $\boldsymbol{G}_{H_{12u}}$ and $\boldsymbol{G}_q = \begin{bmatrix} \boldsymbol{g}_q^1 & \boldsymbol{g}_q^2 & \cdots & \boldsymbol{g}_q^m \end{bmatrix} \in \mathbb{R}^{p \times m}$. It is indicated in Section 3.1.2 that the servo system used in the current work has $n_a = \deg(\boldsymbol{A}(z^{-1})) = 2$, $n_b = \deg(\boldsymbol{B}(z^{-1})) = 5$, $m = 2$ and $p = 2$. Hence, given that the initial conditions are zero, it can be seen from (3.15) that the vectors $\boldsymbol{g}_q^m$ are the expected output sequence for $q = 1, 2, ..., H_2 - 1$ when subjected to a unit-impulse response at time k; i.e.,

$$\boldsymbol{g}_q^m = \boldsymbol{y}(k+q) = (\boldsymbol{I} - \boldsymbol{A}_1)\boldsymbol{g}_{q-1}^m + (\boldsymbol{A}_1 - \boldsymbol{A}_2)\boldsymbol{g}_{q-2}^m + \boldsymbol{A}_2\boldsymbol{g}_{q-3}^m + \boldsymbol{B}_q\Delta\boldsymbol{u}_1 \tag{3.18}$$

where for $m = 2$:

$$\Delta\boldsymbol{u}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}^T \text{ for } \boldsymbol{g}_q^1;$$

$$\Delta\boldsymbol{u}_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}^T \text{ for } \boldsymbol{g}_q^2;$$

$$\boldsymbol{g}_q^m = \boldsymbol{0} \text{ for } q < 0$$

and

$$\boldsymbol{B}_q = \boldsymbol{0} \text{ for } q < H_1 - 1 \text{ and } q > n_b = 5.$$

3.1.5 Online Parameter Adaptation

Since the generalized predictive controller is a model-based approach, an up-to-date model of the system is required to permit good prediction and control. Because the system may be nonlinear and time varying as well, an online model identifier is used to determine a model of the system. The normalized-gain version of the recursive least squares (RLS) estimator is used in the present work (Ljung, 1987). The algorithm can be written as:

$$\hat{\boldsymbol{\theta}}(k) = \hat{\boldsymbol{\theta}}(k-1) + \gamma(k)\boldsymbol{\Xi}(k)^{-1}\boldsymbol{\Psi}(k)\boldsymbol{\Lambda}(k)^{-1}\boldsymbol{\eta}(k) \tag{3.19}$$

$$\boldsymbol{\eta}(k) = \boldsymbol{y}(k) - \boldsymbol{\Psi}(k)^T\hat{\boldsymbol{\theta}}(k-1) \tag{3.20}$$

$$\boldsymbol{\Xi}(k) = \boldsymbol{\Xi}(k-1) + \gamma(k)\left[\boldsymbol{\Psi}(k)\boldsymbol{\Lambda}(k)^{-1}\boldsymbol{\Psi}(k)^T - \boldsymbol{\Xi}(k-1)\right] \tag{3.21}$$

where the estimated parameter vector $\hat{\boldsymbol{\theta}} \in \mathbb{R}^{p(pn_a+m(n_b+1))}$ is formed as

$$\hat{\boldsymbol{\theta}}(k) = \left[\left(\boldsymbol{A}_1^1\right)^T \cdots \left(\boldsymbol{A}_1^p\right)^T \middle| \left(\boldsymbol{A}_2^1\right)^T \cdots \left(\boldsymbol{A}_2^p\right)^T \middle| \cdots \middle| \left(\boldsymbol{A}_{n_a}^1\right)^T \cdots \left(\boldsymbol{A}_{n_a}^p\right)^T \middle| \left(\boldsymbol{B}_0^1\right)^T \right.$$
$$\left. \cdots \left(\boldsymbol{B}_0^p\right)^T \middle| \left(\boldsymbol{B}_1^1\right)^T \cdots \left(\boldsymbol{B}_1^p\right)^T \middle| \cdots \middle| \left(\boldsymbol{B}_{n_b}^1\right)^T \cdots \left(\boldsymbol{B}_{n_b}^p\right)^T \right]^T$$

The subscripts represent the order coefficients, and the superscripts represent the column number. The regression vector is formed as

$$\boldsymbol{\varphi}(k) = \left[-\boldsymbol{y}(k-\tau_{sc}^k-1)^T \cdots -\boldsymbol{y}(k-\tau_{sc}^k-n_a)^T \; \boldsymbol{u}(k-\tau_{sc}^k-1)^T \right.$$
$$\left. \cdots \boldsymbol{u}(k-\tau_{sc}^k-n_b-1)^T\right]^T \in \mathbb{R}^{n_a p+(n_b+1)m}$$

The regression matrix $\boldsymbol{\Psi}(k) \in \mathbb{R}^{p(pn_a+m(n_b+1))\times p}$ in (3.19)-(3.21) is constructed from the regression vector $\boldsymbol{\varphi}(k)$ through the following Kronecker product:

$$\boldsymbol{\Psi}(k) = \boldsymbol{\varphi}(k) \otimes \boldsymbol{I}_p \tag{3.22}$$

The weighting matrix $\Lambda(k) \succ \mathbf{0}$ is chosen as

$$\Lambda(k) = \text{diag}(\lambda^p, \lambda^{p-1}, ..., \lambda^2, \lambda, 1)$$

and the updating step size $\gamma(k) = 1 - \lambda$, where $0 \leq \lambda \leq 1$ is a pre-specified forgetting factor.

In the present strategy of networked control, the RLS estimator uses delayed, actual data (not the predicted values) from the sensors and the controller, which are stored in the shared memory. Thus it is assumed that the dynamic characteristics of the system under control do not change during a worst-case delay. If the nominal model of the system is unknown, a small yet sufficient excitation signal is injected into the system during startup so that the model would converge to the actual system dynamics before control is initiated.

3.1.6 Adaptive Weighting Sequences

It is found experimentally (see Section 3.3.4, Figure 3.9) that, depending on the level of network load (delay), relatively accurate trajectory tracking is achievable by the present networked control strategy, only within a certain narrow range of weighting sequences $\boldsymbol{R}$ and $\boldsymbol{Q}$ of the GPC quadratic cost function (3.8). Generally, higher weighting sequences provide better tracking at low network loads, but lower weighting provides satisfactory tracking at high network load. In order to maintain good tracking throughout a wider range of network delays, the developed control scheme is further enhanced by incorporating a weight-scheduling algorithm. The weighting sequences are continuously modified online as a function of network delay. Since the network delay can significantly fluctuate from sample to sample, the delay level is estimated using an exponential-weighted-moving-average (EWMA) algorithm; i.e., $\tau^k_{estimated} = (1-\delta)\tau^{k-1}_{estimated} + \delta\tau^k_{measured}$. The value $\delta = 0.125$ is recommended (Kurose and Ross, 2003). The crossing points for weight scheduling have to be determined through experiments. For example, in the particular system used in this study, the following criterion is used for $\boldsymbol{R} = \text{diag}(\rho_1, \rho_2)$:

$$\rho_1 = \rho_2 = \begin{cases} 0.200 & \text{for } \tau_{estimated} \leq 10\,\text{ms} \\ 0.100 & \text{for } 10\,\text{ms} < \tau_{estimated} \leq 20\,\text{ms} \\ 0.010 & \text{for } 20\,\text{ms} < \tau_{estimated} \leq 25\,\text{ms} \\ 0.005 & \text{for } \tau_{estimated} > 25\,\text{ms} \end{cases}$$

The program steps for the developed GPC strategy are summarized below:

1. Read packets of sensor data that arrived within the previous time step, if any.
2. Scale the input signals.
3. Update the history of system outputs replacing previously predicted values with actual data.
4. Determine the latest τ_{sc}^{k} from the difference in the current time k and the time-stamp of the latest-received data packet.
5. Estimate the missing or delayed system outputs up to the k-th time step using (3.7).
6. Update the reference trajectory vector $\boldsymbol{r}(k+j)$. (Quintic polynomial reference trajectory is used here).
7. Determine the currently required prediction horizons H_2 and H_u.
8. Restructure and recalculate the free response term $\bar{\boldsymbol{F}}_{H_{12}}$ using (3.14) based on H_1, H_2, H_u, and the latest estimated system model $\hat{\boldsymbol{A}}(z^{-1})$ and $\hat{\boldsymbol{B}}(z^{-1})$.
9. Restructure and re-compute the dynamic matrix $\boldsymbol{G}_{H_{12u}}$ using (3.18) based on H_1, H_2, H_u, and the latest estimated system model $\hat{\boldsymbol{A}}(z^{-1})$ and $\hat{\boldsymbol{B}}(z^{-1})$.
10. Update the average transmission delay and then schedule the correct weighting sequences $\bar{\boldsymbol{Q}}$ and $\bar{\boldsymbol{R}}$.
11. Compute the current and future changes in control actions using (3.17).
12. Scale the control action values.
13. Update the history of control actions for use in step 5.
14. Send the strings of future control actions to individual or grouped actuators.
15. Wait until the next sampling interval and repeat steps 1-12.
16. In addition, the online system model estimator (3.19)-(3.21) runs in the background.

3.1.7 Actuator Buffering

The strings of future control actions computed by the GPC strategy are sent to the actuators over the communication network. These control actions are buffered at the actuator node, in its own memory. New data will overwrite the corresponding old data, based on their individual time step. At the sampling instant k of the actuator node, the "future" control signal that is received at the previous sampling instant (which now becomes the current control signal) will

be used to actuate the system. With such a control scheme, the system would be made deterministic because its inputs and outputs are neither varying nor delayed. Such a buffering mechanism for control actions can be easily embedded on-board within the housing of the distributed actuators with built-in network connection ports.

It should be noted that with sensory data estimation and actuator buffering, the developed network control strategy treats the event of out-of-order data the same as the event of vacant sampling or packet losses. This is because at a particular time instant, older data that arrive at the controller is used to replace the data histories for use in prediction and estimation. On the other hand, older data that arrive at the actuator will be discarded if newer data are available. This is true as long as the sequential (back-to-back) occurrences of out-of-order data, vacant sampling, or packet losses, are within the worst case delay (measured in sample steps).

3.2 Implementation Issues

The predictive networked control strategy developed in this chapter is implemented on the electro-hydraulic manipulator of the fish-processing machine, using the developed control network architecture, as discussed in Section 2.1. Referring to Figure 2.2, sensor reading and filtering are done at a rate of 1 kHz. The system outputs are filtered using a 5th-order Butterworth filter with a cutoff frequency of 80 Hz according to (2.14). Sensor readings are sent to the control client at a rate of 100 Hz or in 10 ms intervals. This is a relatively low sampling rate for a servo system. Yet it is chosen because as the sampling rate gets higher, the network traffic load becomes heavier, and the possibility of more contention time or data loss increases in a bandwidth-limited network resulting in longer data transmission delays (Lian, *et al.*, 2002b). Computation at the control client is carried out at a sampling rate of 100 Hz. The computed series of future control actions is sent back to the control server where the buffering process in the actuator module also runs at a rate of 100 Hz. The size of each sensor-to-controller data packet is 34 bytes and the size of each controller-to-actuator data packet varies from a minimum of 46 bytes (with one step-ahead prediction) depending on the instantaneous length of the prediction horizon, with each additional step requiring 16 bytes. Although such an overhead is small in the current setup with a 10Mbps (i.e., a "theoretical" limit of 1.25 Mbytes/second throughput) network, it will become significant when control is scaled to high-speed servo systems with sampling and data communication rate of over 1 kHz. Note that the main purposes here is to study the effectiveness of the developed control strategy under various

levels of delay relative to the sampling time of a system.

Although the two axes of the electro-hydraulic positioning system are dynamically similar and not dynamically coupled to a considerable degree, the GPC controller is still formulated as a 2-input, 2-output multivariable system to illustrate the generality of the developed strategy. The nominal parameters of the CARIMA model (3.1) from the combined ARX model of each axis identified in Section 2.4.2 are obtained as:

$$\boldsymbol{A}_1 = \begin{bmatrix} -1.5620 & 0 \\ 0 & -1.6600 \end{bmatrix}, \boldsymbol{A}_2 = \begin{bmatrix} 0.5885 & 0 \\ 0 & 0.6796 \end{bmatrix}, \boldsymbol{B}_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \boldsymbol{B}_1 = \begin{bmatrix} 0.007691 & 0 \\ 0 & 0.01153 \end{bmatrix},$$

$$\boldsymbol{B}_2 = \begin{bmatrix} 0.01608 & 0 \\ 0 & 0.02643 \end{bmatrix}, \boldsymbol{B}_3 = \begin{bmatrix} 0.004824 & 0 \\ 0 & 0.003392 \end{bmatrix}, \boldsymbol{B}_4 = \begin{bmatrix} -0.006505 & 0 \\ 0 & -0.01742 \end{bmatrix}, \text{ and}$$

$$\boldsymbol{B}_5 = \begin{bmatrix} -0.002786 & 0 \\ 0 & -0.005612 \end{bmatrix}.$$

A well-known problem in electro-hydraulic actuators is the phenomenon of dead-band nonlinearity, mainly due to static friction causing backlash behavior in the closed-loop system, which in turn causes limit-cycling or hunting at steady-state. Specifically, the hydraulic piston falls into a static friction regime when it stops as the direction of motion reverses. In order to overcome this static friction breakaway force, the actuator control current has to be increased beyond a certain threshold (Tafazoli, *et al.*, 1998). Here, the steady-state position tracking error is reduced by adding a dead-band compensation term of the form

$$\Delta v_k^m = -I_d^m \operatorname{sgn}(y_k^m - r_k^m) \tag{3.23}$$

to the control actions given by the m-input, where $I_d^m = 2.0$ mA. The desired reference trajectory is generated using a quintic (5th-order) polynomial in order to achieve smooth trajectories in position, velocity and acceleration. The piston reference position for the i-th segment of the m-th input reference quintic trajectory is written as

$$r_i^m(k) = d_{i1}k^5 + d_{i2}k^4 + d_{i3}k^3 + d_{i4}k^2 + d_{i5}k + d_{i6} \tag{3.24}$$

where $d_{i1},...,d_{i6}$ are the quintic spline coefficients determined from the end-point conditions of the particular segment. The first and the second derivatives of (3.24) are

$$\dot{r}_i^m(k) = \tfrac{d}{dk} r_i^m(k) = 5d_{i1}k^4 + 4d_{i2}k^3 + 3d_{i3}k^2 + 2d_{i4}k + d_{i5} \tag{3.25}$$

$$\ddot{r}_i^m(k) = \frac{d^2}{dk^2} r_i^m(k) = 20d_{i1}k^3 + 12d_{i2}k^2 + 6d_{i3}k + 2d_{i4} \tag{3.26}$$

Define k_s as the starting time step and k_e as the end time step of the i-th segment. Let $r_i^m(k_s)$, $\dot{r}_i^m(k_s)$, and $\ddot{r}_i^m(k_s)$ be the imposed end-point conditions at k_s, and $r_i^m(k_e)$, $\dot{r}_i^m(k_e)$, and $\ddot{r}_i^m(k_e)$ be the imposed end-point conditions at k_e. The coefficients $d_{i1},...,d_{i6}$ are found by solving (3.24)-(3.26) using the 6 end-point conditions. This is easily done by first combining (3.24)-(3.26) into the following matrix form:

$$\begin{bmatrix} k_s^5 & k_s^4 & k_s^3 & k_s^2 & k_s & 1 \\ 5k_s^4 & 4k_s^3 & 3k_s^2 & 2k_s & 1 & 0 \\ 20k_s^3 & 12k_s^2 & 6k_s & 2 & 0 & 0 \\ k_e^5 & k_e^4 & k_e^3 & k_e^2 & k_e & 1 \\ 5k_e^4 & 4k_e^3 & 3k_e^2 & 2k_e & 1 & 0 \\ 20k_e^3 & 12k_e^2 & 6k_e & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_{i1} \\ d_{i2} \\ d_{i3} \\ d_{i4} \\ d_{i5} \\ d_{i6} \end{bmatrix} = \begin{bmatrix} r_i^m(k_s) \\ \dot{r}_i^m(k_s) \\ \ddot{r}_i^m(k_s) \\ r_i^m(k_e) \\ \dot{r}_i^m(k_e) \\ \ddot{r}_i^m(k_e) \end{bmatrix} \tag{3.27}$$

Equation (3.27) is then solved using Gauss elimination method (Kreyszig, 1999).

3.3 Real-time Experimental Evaluation

Experiments are conducted to evaluate the performance of the system under different network conditions. The key experimental results are shown in this section. Since the two axes of the hydraulic positioning system of the industrial fish-processing machine are dynamically similar, only the results from the X-axis are given here so that the key features of the results can be emphasized.

3.3.1 Nominal Network Load

Figure 3.3 shows the response of the system under a fixed network load with the number of data "juggle" between the intermediate nodes set to 15 for each direction. The weighting sequences are set as $\boldsymbol{Q} = \text{diag}(0.10, 0.10)$ and $\boldsymbol{R} = \text{diag}(0.20, 0.20)$. It is observed that the tracking is almost perfect, having a maximum tracking error of 1.0 mm (Figure 3.3(b)) with the round-trip delay fluctuating between 20 to 60 ms or 2 to 6 times the sampling interval. This relatively high transmission delay is mainly resulted from the data "juggle" between the intermediate nodes. Direct transmission between two nodes will have low delay in the order of

5 ms. However, it must be stressed that the purposed here is to test the capability of the proposed network control strategy under severe network congestion. Long as well as noisy transmission lines in industrial networks can be the factors that impose such high network load. It must also be scalable for application to high-speed servo systems. The recorded control input signal shows the actual values sequenced by the actuator buffer.

Figure 3.4 illustrates the extent of round-trip delay that can be tolerated by the networked control strategy developed here. The tracking error in terms of IAE (integral-absolute-error), which is normalized with respect to a local (non-networked) direct GPC controller, is plotted against the round-trip delay. It is observed that the position tracking is satisfactory for a "mean" round-trip delay of up to approximately 35 ms (indicated by dot-circle markers). Note that with a time delay range (horizontal lines) as high as 90 ms, the control system remains stable; i.e., within the physical limit of the hydraulic piston. As a comparison, the tracking errors of the local direct GPC controller, when used as a remote controller (without estimation or actuator buffering), are included in the figure (solid-circle markers). In this case the control system is found to be stable only up to a mean delay of 5 ms, which is smaller than one sampling interval (10 ms).

3.3.2 Variable Network Load

In order to evaluate the ability of the control system to adapt to sudden changes in network behavior, the system is subjected to a few levels of transmission delay as shown in Figure 3.5. The network traffic is varied at every 200 samples. This can be achieved by varying the juggling of the data packets between the two intermediate nodes (see Figure 2.1) or by changing the number of external data transfers in all communication nodes including the control server and client. Tracking performance is unaffected when the transmission delay level changes as long as it is within the worst-case delay that can be handled by the control strategy (approximately 90 ms round-trip delay).

Some justification regarding the recorded tracking performance is in order. A tracking accuracy of ±1.0 mm, although relatively low, is the best that can be achieved by the electro-hydraulic system under study. Similar tracking performance is obtained by previous researchers using a non-networked, observer-based friction compensating control strategy on the same machine (Tafazoli, 1998, Figure 18).

3.3.3 Packet Loss

The feasibility of the developed control strategy is demonstrated under conditions of high network congestion, where data packets can be lost. In particular, under UDP transport protocol, data packets are dropped at one of the intermediate nodes. Both sensor-to-controller and controller-to-actuator data packets are dropped according to a memoryless exponential distribution, typically used in Internet queuing models (Bertsekas and Gallager, 1991). In order to specifically evaluate the performance of the developed control strategy during packet losses, data "juggling" is not performed, resulting in the noted decrease in transmission delays reported in the following experimental results. The responses are shown in figures 3.6 and 3.7 for 7.5% and 12.5% loss probability, respectively. The vertical lines in figures 3.6(a), 3.6(c), 3.7(a), and 3.7(c) indicate the instances of feedback (sensor) data loss and the loss of control signals. Note that even though an identical loss rate is set for the data packets of both sensor-to-controller and controller-to-actuator communication, the data packets of the control signal lose more frequently. This is because these data packets are larger than those for the sensor signals. Larger data packets get discarded more easily at the intermediate buffer, and are easily corrupted since Ethernet transmission is a serial connection. Tracking performance is still good (comparable to cases with no loss) at a loss probability of 7.5% and a round-trip delay between 8 to 20 ms (Figure 3.6(b)). However, the system starts to oscillate vigorously at a loss rate of 12.5% (Figure 3.7(c)) but still manages to settle back and track the reference trajectory at some regions. Figure 3.8 shows the tracking error over different levels of loss probability as a function of the mean round-trip delay. Again, the tracking error is normalized with respect to a local direct GPC controller. Tracking is generally unaffected up to a loss rate of 5.0% when the mean round-trip delay is under 20 ms. The system becomes unstable (hitting the physical limits of the hydraulic cylinder) when the loss rate exceeds 12.5%.

3.3.4 Effect of Weighting Sequence

Figure 3.9 shows the influence of the future error weighting sequence $\boldsymbol{Q}$, which is used in the GPC cost function. The "normalized" tracking error in terms of IAE is plotted against the mean round-trip delay. Each line corresponds to a different set of values of the weighting sequence, from 0.01 to 0.25. It is found that in the low network delay region (under 15 ms mean round-trip delay), higher weighting provides better tracking. In the high network delay

region, lower weighting gives lower tracking error. This is because at low or negligible delay, a low weighting sequence causes sluggish response. Such sluggishness actually proves useful in stabilizing the system during high network loads because sensor values will only differ slightly between each sample due to the slow dynamics of the closed-loop system. As discussed in Section 3.1.6, a simple gain scheduling routine can be used to maintain close tracking throughout a wider range of network delay.

3.3.5 Driving Bandwidth of the System

The driving bandwidth of the networked control strategy developed here is evaluated under three levels of network delay with mean network transmission delay values of 3 ms, 11 ms, and 21 ms, respectively. Figure 3.10(a)-(c) shows the position response for each of the mean transmission delay values under 6 settings of maximum velocities; i.e., 6.25 mm/s, 7.50 mm/s, 9.38 mm/s, 12.50 mm/s, 18.75 mm/s, and 37.50 mm/s. Figure 3.11 is a plot of normalized tracking error (IAE) versus the maximum velocity of the reference trajectory, which summarizes the response curves in Figure 3.10. Each line corresponds to a different mean network transmission delay value (3 ms, 11 ms, and 21 ms). It is observed that the tracking error increases with delay, but the working range of the system can be increased by actively moving the velocity constraints according to the level of network congestion.

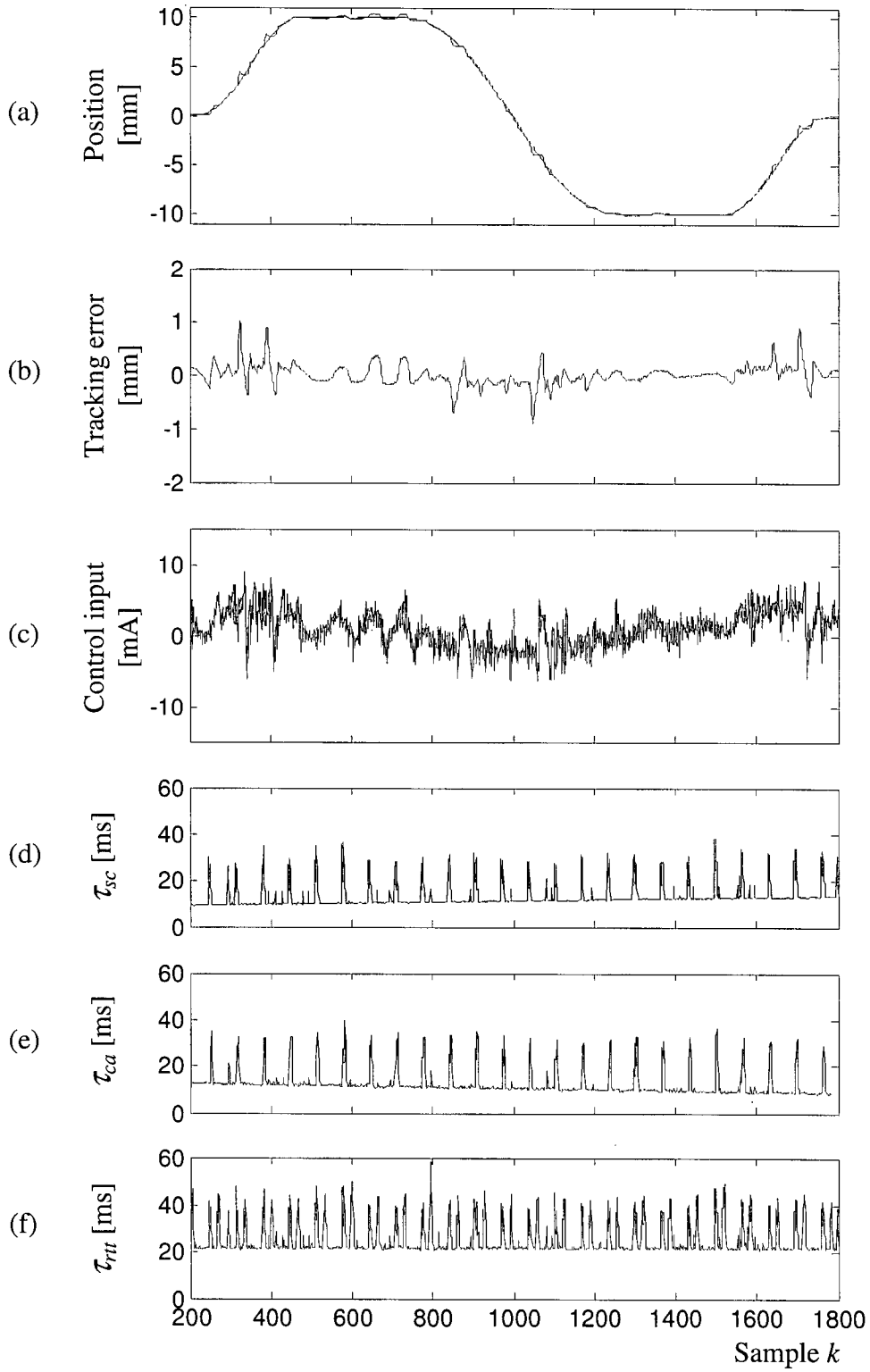


Figure 3.3: System responses under a fixed network loading. (a) Position response (solid – actual output, dashed – desired trajectory); (b) Tracking error; (c) Actual control signal sent to actuator; (d) Sensor-to-controller delay; (e) Controller-to-actuator delay; and (f) Round-trip delay.

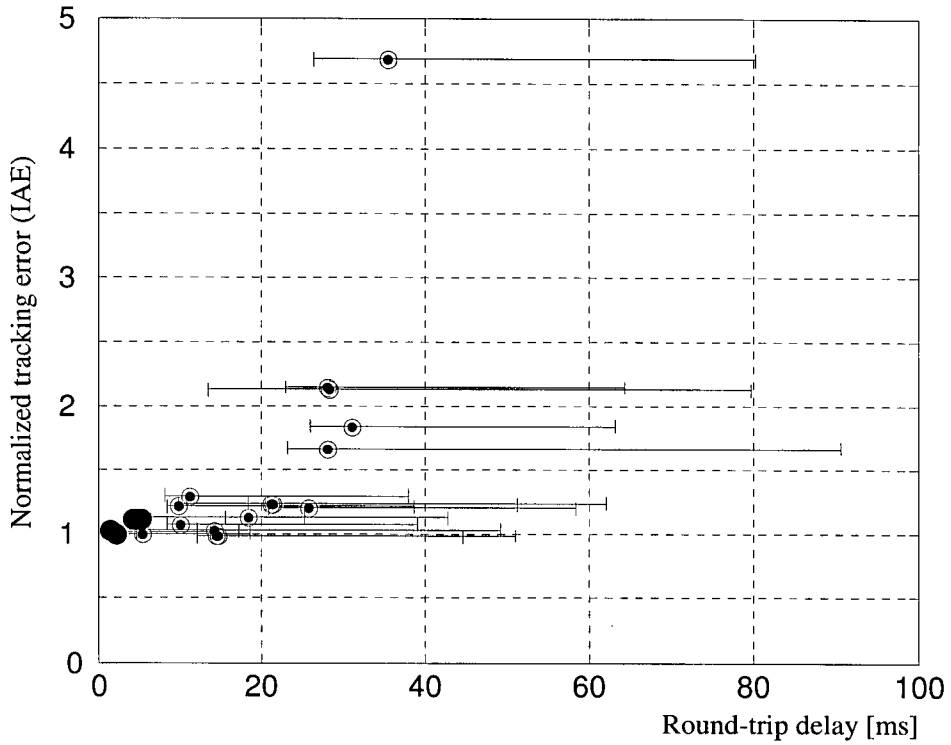


Figure 3.4: Effect of the round-trip delay on the tracking performance.

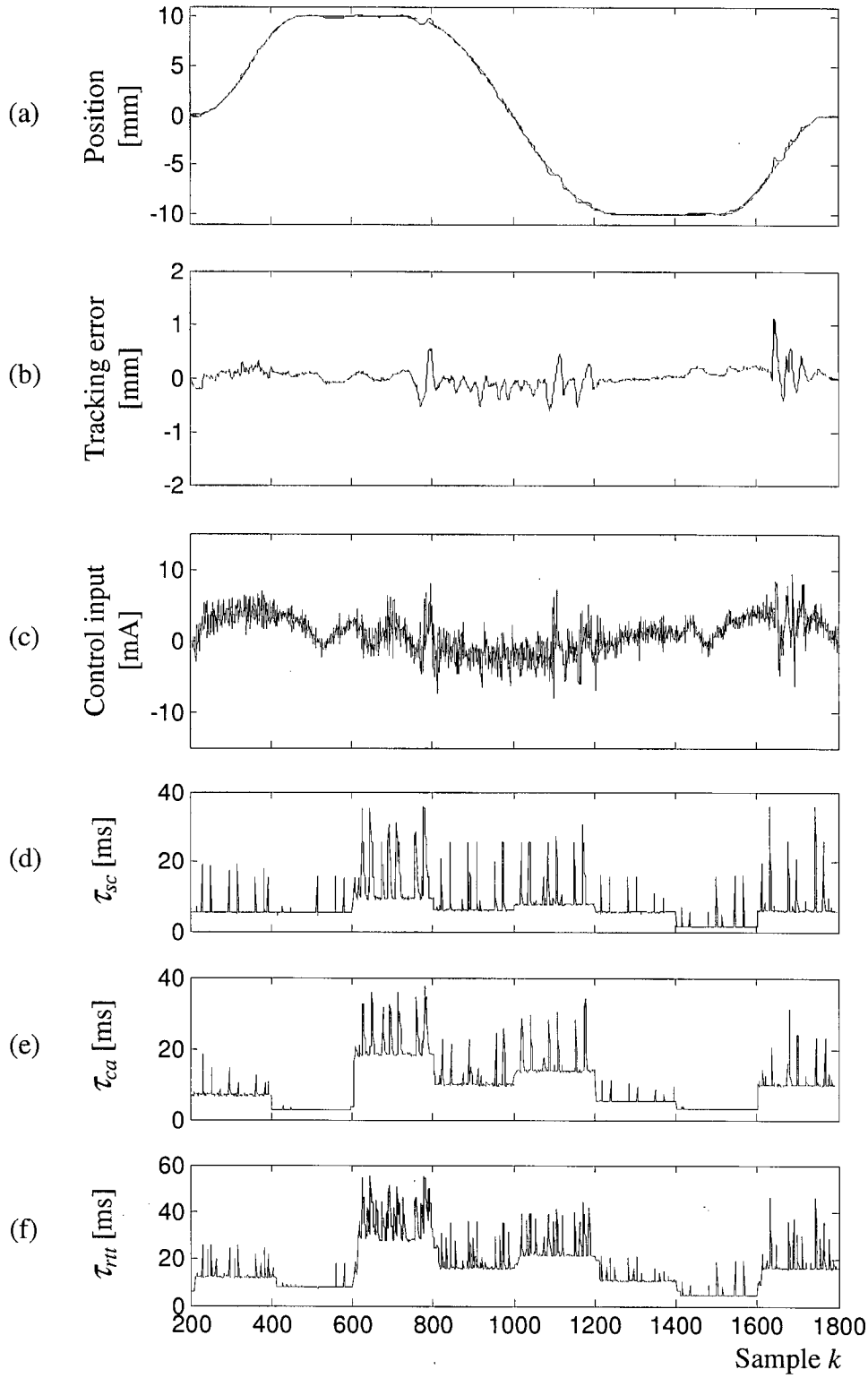


Figure 3.5: System responses under variable network loading. (a) Position response (solid – actual output, dashed – desired trajectory); (b) Tracking error; (c) Actual control signal sent to actuator; (d) Sensor-to-controller delay; (e) Controller-to-actuator delay; and (f) Round-trip delay.

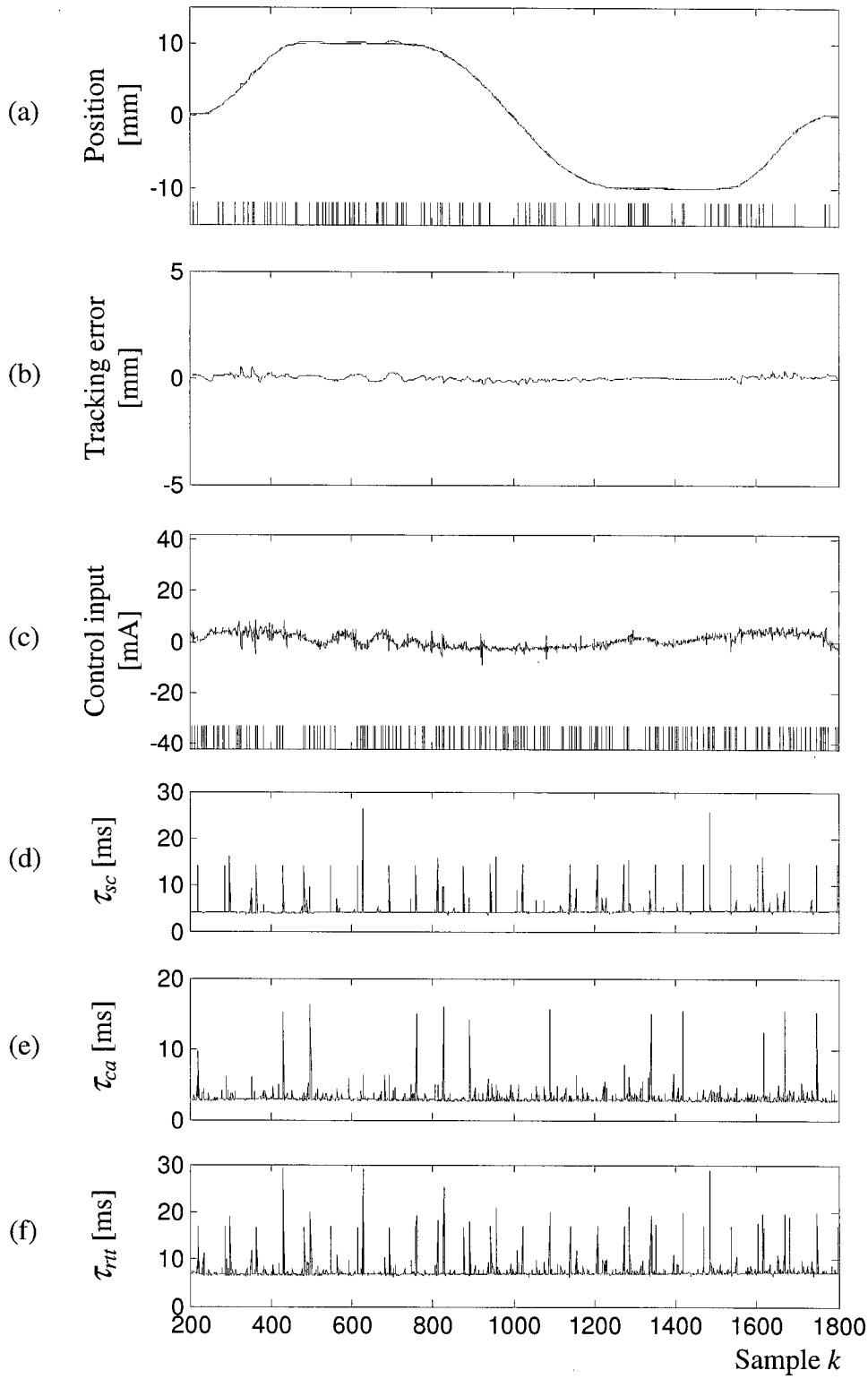


Figure 3.6: System responses under a 7.5% data loss rate. (a) Position response (solid – actual output, dashed – desired trajectory); (b) Tracking error; (c) Actual control signal sent to actuator; (d) Sensor-to-controller delay; (e) Controller-to-actuator delay; and (f) Round-trip delay.

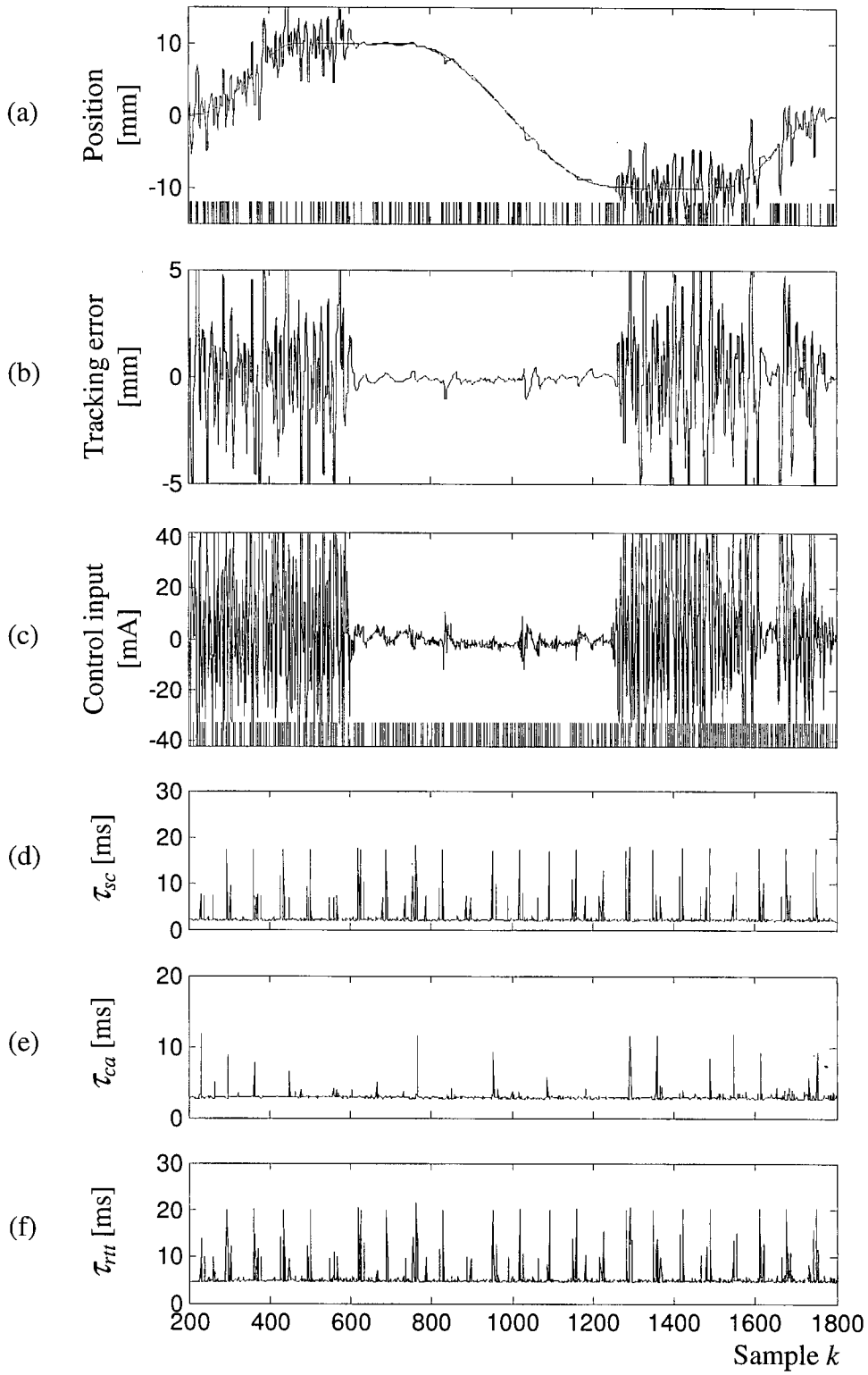


Figure 3.7: System responses under a 12.5% data loss rate. (a) Position response (solid – actual output, dashed – desired trajectory); (b) Tracking error; (c) Actual control signal sent to actuator; (d) Sensor-to-controller delay; (e) Controller-to-actuator delay; and (f) Round-trip delay.

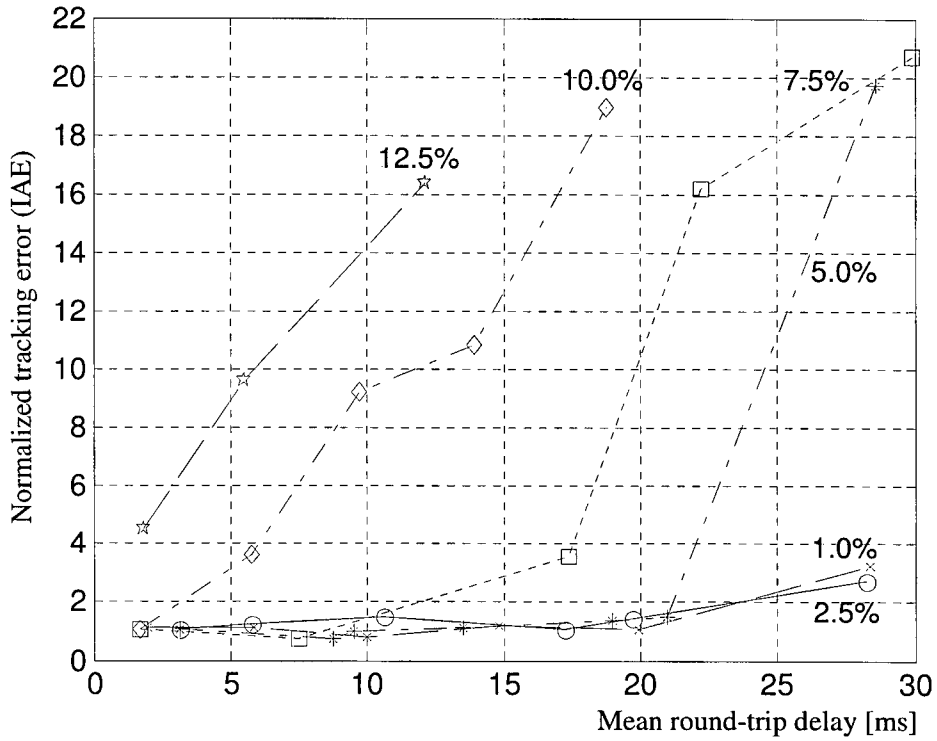


Figure 3.8: Comparison of tracking performance over different loss rates of data packets.

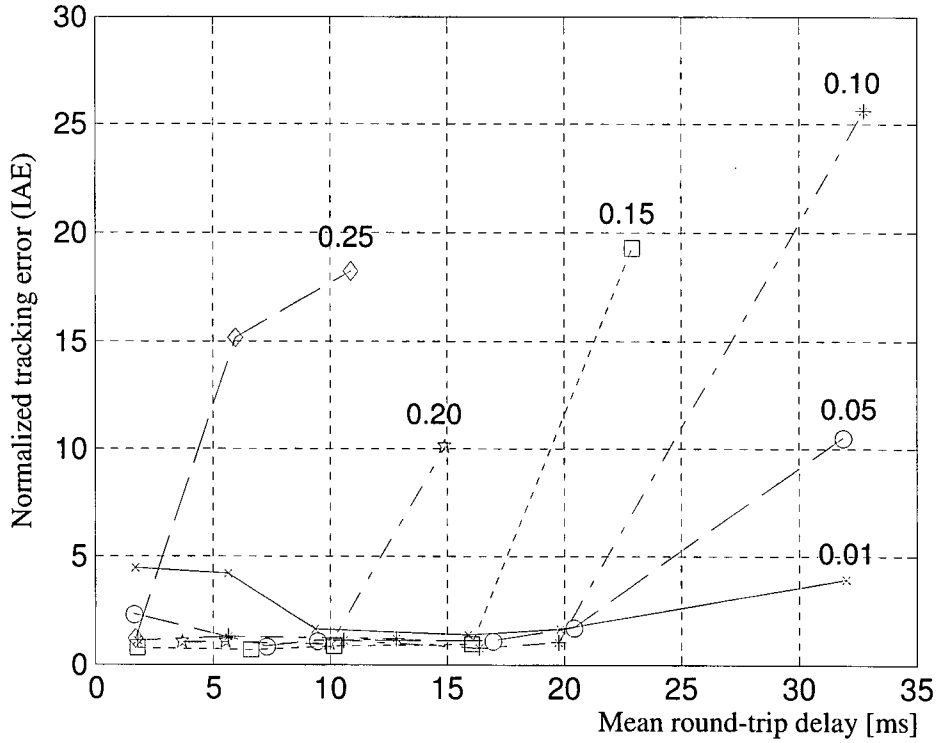


Figure 3.9: Comparison of tracking performance over different values of future error weighting ($\boldsymbol{Q}$).

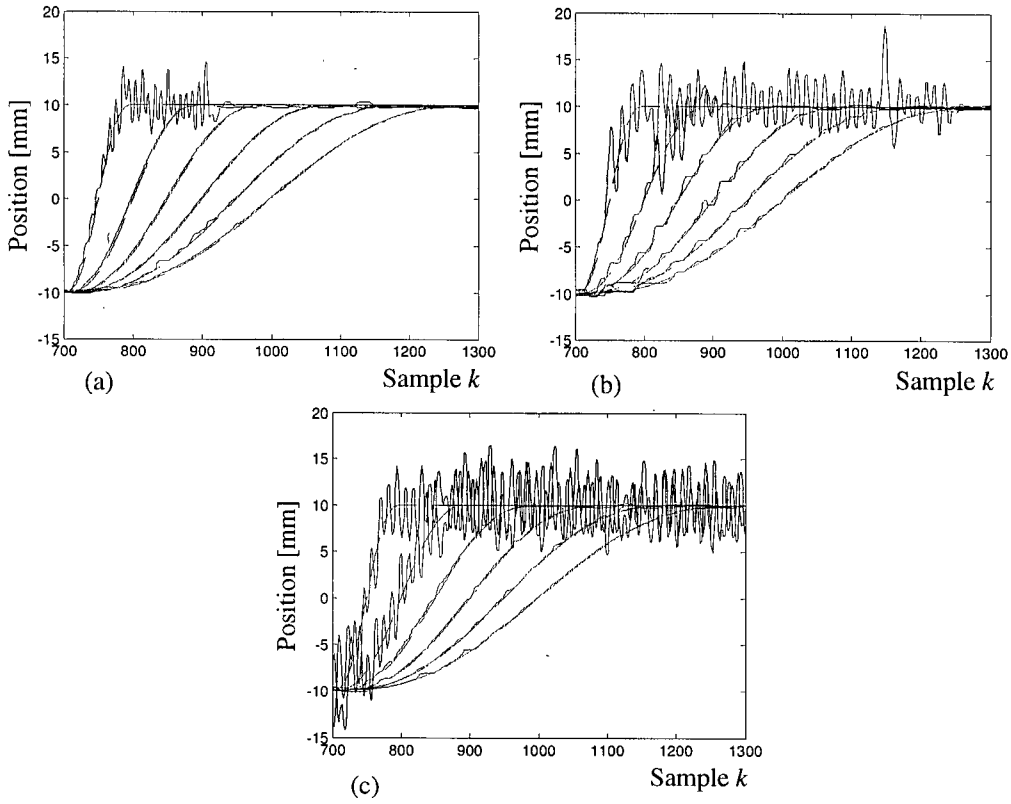


Figure 3.10: System response subjected to different speed settings under different levels of network delay. (a) $\overline{\tau}_{rtt} \cong 3$ ms; (b) $\overline{\tau}_{rtt} \cong 11$ ms; and (c) $\overline{\tau}_{rtt} \cong 21$ ms.

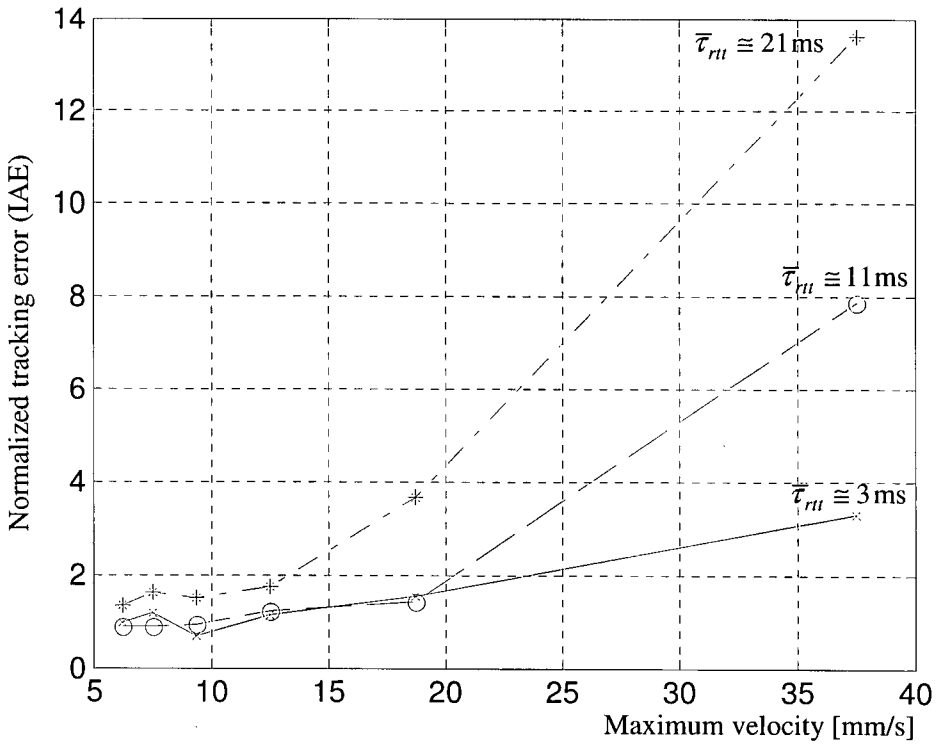


Figure 3.11: Achievable system speed under different levels of network delay.

3.4 Summary

A new feedback control strategy intended for networked control systems (NCS) was developed and implemented. Its performance was rigorously tested using an electro-hydraulic servo positioning system of an industrial fish-processing machine as the controlled plant. The control strategy uses an estimator to estimate missing or delayed sensor data, and employs a multivariable predictive controller to compute future control actions well in advance. The current work is thought to be the first investigation into the solution of network transmission problems, specifically time delay, in "control signals" between the controller and the actuators of a networked control system. As parallel contribution, the variable horizon extension to GPC has proven to be a viable improvement to the current strategy by reducing the network load as well as the computational load.

The developed control strategy for networked systems was able to handle various types of network behavior including delay levels of up to 9 times the sampling interval, vacant sampling, out-of-order data, and data loss rates up to 12.5%. In addition, the strategy was able to adapt to variable network loading while optimizing the size of data packets to minimize the network traffic. Since the developed control strategy does not require network transmission characteristics or dynamics, its use is not limited to Ethernet networks.

The next two chapters will investigate the conditions of maintaining closed-loop asymptotic stability in the developed networked control strategy, but in a border realm of constrained Model Predictive Control which is a superset of GPC.

Chapter 4
Stability of Predicted-Input Control

The present chapter investigates the stability characteristics of the predictive control strategy developed for a networked control system (NCS) in the previous chapter. The specific controller employs Generalized Predictive Control (GPC) with buffering of the future control sequence to overcome the transmission problems in the controller-to-actuator lines. The main focus of the present chapter is to determine the conditions under which the stability of the developed control strategy with input prediction is guaranteed in a broader realm of constrained model predictive control (MPC).

The next section briefly reintroduces the essential structure of the developed NCS strategy in the constrained state-space MPC setting and formulates the stability problem of specific interest in this chapter. Section 4.2 casts the MPC problem into a standard Quadratic Programming problem. Section 4.3 presents the main contribution of the chapter, specifically the development of the asymptotic stability conditions in the presence of buffered optimal input signals. The usage of the stability results in generating stability boundaries, which provide a tool for the design of the NCS-MPC controller, is investigated in Section 4.4. Some issues related to the practical implementation of the developed controller are discussed in Section 4.5. Section 4.6 verifies the developed stability results, through real-time implementation on an electro-hydraulic manipulator system of a fish-processing machine. The concepts and algorithm development of Linear Matrix Inequalities (LMI) and multi-parametric Quadratic Programming (mpQP) which are vital for the analysis and real-time implementation of the developed controller, are provided in Appendix A and Appendix B.

4.1 The Predictive Control Strategy for Networked Systems – Revisited

Since the communication delay between network nodes (sensors, actuators and controllers) is non-deterministic, as a result of random and frequent occurrence of data losses and vacant sampling, it is anticipated that the design of a feedback control solution for a networked system

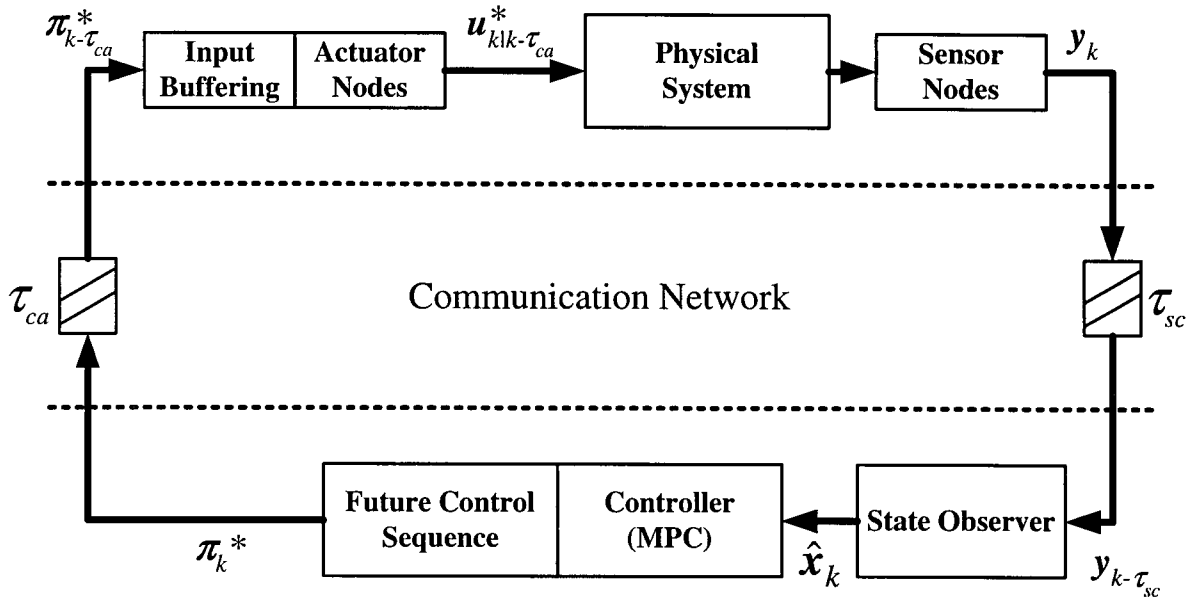


Figure 4.1: Simplified architecture of the developed NCS strategy.

(termed a "networked control strategy" in this thesis) requires no prior knowledge of the transmission dynamics. The model used here is the discrete-time representation of the system under control, given by

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{4.1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^n$ are the states of the system at time k, $\boldsymbol{u}_k \in \mathbb{R}^m$ are the control inputs at time k, and $f : \mathbb{R}^{n\times m} \rightarrow \mathbb{R}^n$. Figure 4.1 shows a simplified architecture of the networked control strategy or simply the NCS strategy as developed in this thesis. The physical system consisting of sensor nodes and actuator nodes is connected to the controller through a communication medium. All the nodes are set to a common sampling interval h with a synchronized clock (see Section 2.2). The sensor-to-controller and controller-to-actuator communication delays are represented by τ_{sc} and τ_{ca}, respectively. An extended state observer is used in order to fill the missing sensor data up to the current sampling time k. The estimated states are given by

$$\hat{\boldsymbol{x}}_{k-\tau_{sc}+i|k-\tau_{sc}} = f(\boldsymbol{x}_{k-1-\tau_{sc}+i|k-\tau_{sc}+i}, \boldsymbol{u}_{k-\tau_{sc}+i}) + \boldsymbol{L}'(\boldsymbol{y}_{k-\tau_{sc}} - \hat{\boldsymbol{y}}_{k-\tau_{sc}|k-\tau_{sc}-1}) \tag{4.2}$$

for $i = 1,\ldots,\tau_{sc}$ where $\boldsymbol{L}'$ is the observer gain matrix and $\boldsymbol{y}_k \in \mathbb{R}^p$ are the measured states (output) of the system. The sensor-to-controller delay τ_{sc} is determined from the time-stamping

information embedded in the latest sensor packet received from the sensor node.

The key to the developed strategy is the use of an MPC policy, which is a form of receding horizon optimal control law, to *pre*-compute a string of optimal future control input sequence $\boldsymbol{\pi}_k^* = \left\{\boldsymbol{u}_{k|k}^*, ..., \boldsymbol{u}_{k+H_u-1|k}^*\right\}$, where H_u is the length of the control horizon. This is done at each time step k, by optimizing a finite quadratic cost function either of the output reference tracking form given by

$$V_{H_1,H_2,H_u}(\boldsymbol{y}_k, \boldsymbol{\pi}_k) = \min_{\boldsymbol{\pi}(\cdot|k)} \left\{ \left\| \boldsymbol{y}_{k+H_2|k} - \boldsymbol{r}_{k+H_2} \right\|_{\boldsymbol{P}}^2 + \sum_{i=H_1}^{H_2-1} \left\| \boldsymbol{y}_{k+i|k} - \boldsymbol{r}_{k+i} \right\|_{\boldsymbol{Q}}^2 + \sum_{i=0}^{H_u-1} \left\| \boldsymbol{u}_{k+i|k} \right\|_{\boldsymbol{R}}^2 \right\}$$

s.t. (4.3)

$$\boldsymbol{y}_{k+i|k} \in \mathcal{Y}_i \quad \forall i = 1, ..., H_2$$

$$\boldsymbol{u}_{k+i|k} \in \mathcal{U} \quad \forall i = 0, ..., H_u - 1$$

or, the regulator form expressed as

$$V_{H_1,H_2,H_u}(\boldsymbol{x}_k, \boldsymbol{\pi}_k) = \min_{\boldsymbol{\pi}(\cdot|k)} \left\{ \left\| \boldsymbol{x}_{k+H_2|k} \right\|_{\boldsymbol{P}}^2 + \sum_{i=H_1}^{H_2-1} \left\| \boldsymbol{x}_{k+i|k} \right\|_{\boldsymbol{Q}}^2 + \sum_{i=0}^{H_u-1} \left\| \boldsymbol{u}_{k+i|k} \right\|_{\boldsymbol{R}}^2 \right\}$$

s.t. (4.4)

$$\boldsymbol{x}_{k+i|k} \in \mathcal{X}_i \quad \forall i = 1, ..., H_2$$

$$\boldsymbol{u}_{k+i|k} \in \mathcal{U} \quad \forall i = 0, ..., H_u - 1$$

where $\boldsymbol{y}_k = g(\boldsymbol{x}_k) \in \mathbb{R}^p$ is the output of the system with $g : \mathbb{R}^n \to \mathbb{R}^p$, $\boldsymbol{r}_k \in \mathbb{R}^p$ is the output set-point, H_1 is the minimum prediction horizon, H_2 is the tracking error or state prediction horizon, $\boldsymbol{P}$ is the terminal weight matrix, and $\boldsymbol{Q}$ and $\boldsymbol{R}$ are the weighting sequence matrices ($\boldsymbol{P} \succ 0$, $\boldsymbol{Q} \succ 0$ and $\boldsymbol{R} \succ 0$). In the sequel, H_1 is assumed to be zero in order to facilitate the analysis. This assumption does not pose any restriction to the generality of the analysis, however, because the term $\boldsymbol{x}_k$ (at time step k) is seen as a constant in the optimization process. Optimizing (4.3) or (4.4) yields the optimally predicted future control sequence as $\boldsymbol{\pi}_k^* = \left\{\boldsymbol{u}_{k|k}^*, ..., \boldsymbol{u}_{k+H_u-1|k}^*\right\} = \arg\min_{\boldsymbol{u}} V_{H_2,H_u}(\boldsymbol{x}_k, \boldsymbol{\pi}_k)$. In a typical application of MPC, the first control input $\boldsymbol{u}_{k|k}^*$ is used. In the developed NCS strategy, however, the subsequent control sequence is employed depending on the level of transmission delay. This is accomplished by retrofitting the actuator with a data buffering mechanism to schedule an appropriate control

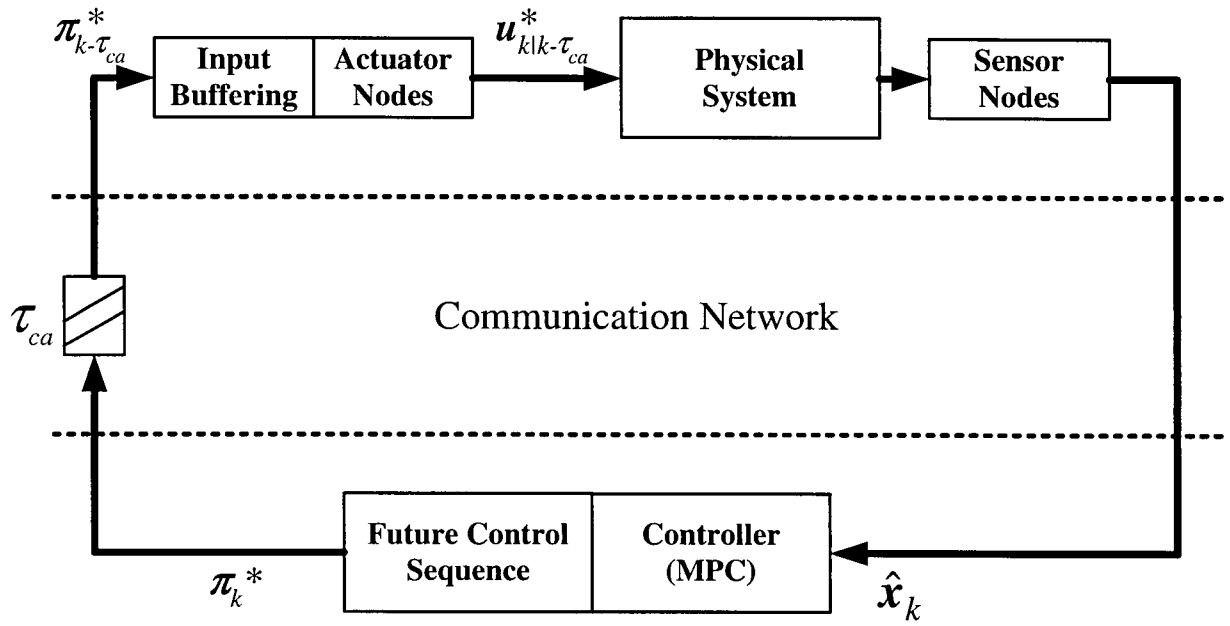


Figure 4.2: Reduced structure of the NCS for analyzing stability of the MPC policy with future input buffering.

input $u^*_{k+i|k}$ from π^*_k, based on the time-stamps of the data packets.

The unconstrained version of this NCS strategy has been benchmarked in Chapter 3 and shown to perform well, producing good results. Besides being able to cope with transmission delays, the control strategy is also able to compensate for various levels of packet losses and vacant sampling. Further modifications have been made to the developed strategy in order to enhance its performance. These include a scheme where the prediction horizon and the control horizon are variable, for reducing network traffic and computation load, and gain scheduling of the weighting matrices of the cost function to dampen the system under high network loads, thereby extending the working range of the closed-loop system. In addition, the developed strategy of control input buffering has the potential of reducing the utilized network bandwidth (or, network utilization) by lowering the frequency of data transmission, without limiting the operation bandwidth of the system under control.

As the next step of the control strategy development, conditions are established to ensure closed-loop stability of the system. According to the *Principle of Separation* (Furuta, *et al.*, 1988), the controller and the estimator (observer) of a closed-loop system may be designed and/or analyzed separately without one influencing the performance or stability of the other. Therefore, the main issue that is sought to be resolved in this chapter is whether the pre-

computed and buffered string of optimal control input sequence is still feasible and optimal at the future sampling instant of the actuator. It is assumed that $\tau_{sc}=0$; i.e., perfect state measurements are available to the MPC controller. Figure 4.2 shows the reduced structure of NCS strategy that will be analyzed. The conditions whereby the stability of the closed-loop NCS is guaranteed need to be established. This is accomplished here based on the Lyapunov stability theory. The extension of the analysis for the case of imperfect state measurements is addressed in Chapter 5.

4.2 State-space MPC Formulation

A preliminary formulation of the MPC policy in the state-space form is required for carrying out further analysis. The following two sub-sections formulate the MPC policy as a regulator problem and a reference tracking problem. These formulations are also useful in the experimental implementation of the developed control strategy. Here, the linearized discrete, time-invariant model of (4.1) is considered, which is given as

$$\boldsymbol{x}_{k+1}=\boldsymbol{A}\boldsymbol{x}_k+\boldsymbol{B}\boldsymbol{u}_k \tag{4.5}$$

with $\boldsymbol{A}\in\mathbb{R}^{n\times n}$ and $\boldsymbol{B}\in\mathbb{R}^{n\times m}$.

4.2.1 The Regulator Problem

The MPC formulation requires the recasting of (4.4) into a compact form that is solvable using either classical Quadratic Programming (QP) methods or, as adopted in this study, the method of multi-parametric Quadratic Programming (mpQP). First, the cost function in (4.4) is expanded into:

$$V_{H_2,H_u}(\boldsymbol{x}_k,\boldsymbol{\pi}_k)=\min_{\pi(\cdot|k)}\left\{\left\|\boldsymbol{x}_{k+H_2|k}\right\|_{\boldsymbol{P}}^2+\sum_{i=H_u}^{H_2-1}\left\|\boldsymbol{x}_{k+i|k}\right\|_{\boldsymbol{Q}}^2+\sum_{i=0}^{H_u-1}\left[\left\|\boldsymbol{x}_{k+i|k}\right\|_{\boldsymbol{Q}}^2+\left\|\boldsymbol{u}_{k+i|k}\right\|_{\boldsymbol{R}}^2\right]\right\} \tag{4.6}$$

Horizons H_2 and H_u in (4.6) are replaced by a single prediction horizon H by noting that $\boldsymbol{u}_{k+i}=0$ for $H_2>H_u$. Using (4.5) recursively, the first term of (4.6) is expressed as

$$\left\|\boldsymbol{x}_{k+H_2|k}\right\|_{\boldsymbol{P}}^2=\left\|\boldsymbol{x}_{k+H_2-1|k}\right\|_{\boldsymbol{A}^T\boldsymbol{P}\boldsymbol{A}}^2=\left\|\boldsymbol{x}_{k+H_u|k}\right\|_{\left(\boldsymbol{A}^{H_2-H_u}\right)^T\boldsymbol{P}\boldsymbol{A}^{H_2-H_u}}^2 \tag{4.7}$$

Similarly, the second term of (4.6) is recast into the form

$$\begin{aligned}
&\sum_{H_u}^{H_2-1} \left\| \boldsymbol{x}_{k+i|k} \right\|_{\boldsymbol{Q}}^2 \\
&\quad = \left\| \boldsymbol{x}_{k+H_u|k} \right\|_{\boldsymbol{Q}}^2 + \left\| \boldsymbol{x}_{k+H_u|k} \right\|_{\boldsymbol{A}^T\boldsymbol{Q}\boldsymbol{A}}^2 + \cdots + \left\| \boldsymbol{x}_{k+H_u|k} \right\|_{\left(\boldsymbol{A}^{H_2-H_u-1}\right)^T \boldsymbol{Q}\boldsymbol{A}^{H_2-H_u-1}}^2 \\
&\quad = \left\| \boldsymbol{x}_{k+H_u|k} \right\|_{\sum_{i=0}^{H_2-H_u-1} \left(\boldsymbol{A}^i\right)^T \boldsymbol{Q}\boldsymbol{A}^i}^2
\end{aligned} \tag{4.8}$$

Substituting (4.7) and (4.8) into (4.6), and setting $H = H_u$ yields the equivalent finite quadratic cost function:

$$\begin{aligned}
&V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k) = \min_{\boldsymbol{\pi}(\cdot|k)} \left\{ \left\| \boldsymbol{x}_{k+H|k} \right\|_{\boldsymbol{P}_0}^2 + \sum_{i=0}^{H-1} \left[\left\| \boldsymbol{x}_{k+i|k} \right\|_{\boldsymbol{Q}}^2 + \left\| \boldsymbol{u}_{k+i|k} \right\|_{\boldsymbol{R}}^2 \right] \right\} \\
&\text{s.t.} \\
&\boldsymbol{x}_{k+i|k} \in \mathcal{X}_i \qquad \forall i = 1, \ldots, H \\
&\boldsymbol{u}_{k+i|k} \in \mathcal{U} \qquad \forall i = 0, \ldots, H-1
\end{aligned} \tag{4.9}$$

which is evaluated over a prediction horizon of length H using a modified terminal weight matrix $\boldsymbol{P}_0$ given as

$$\boldsymbol{P}_0 = (\boldsymbol{A}^{H_2-H_u})^T \boldsymbol{P} (\boldsymbol{A}^{H_2-H_u}) + \sum_{i=0}^{H_2-H_u-1} (\boldsymbol{A}^i)^T \boldsymbol{Q}\boldsymbol{A}^i \succ 0 \tag{4.10}$$

The cost function (4.9) can then be transformed into the standard QP form by the continuous substitution of (4.5), which has been brought back to being pivoted at the initial state $\boldsymbol{x}_k$ in a receding horizon manner, such that

$$\begin{aligned}
\boldsymbol{x}_{k+i} &= \boldsymbol{A}\boldsymbol{x}_{k+i-1} + \boldsymbol{B}\boldsymbol{u}_{k+i-1} \\
&= \boldsymbol{A}\left[\boldsymbol{A}\boldsymbol{x}_{k+i-2} + \boldsymbol{B}\boldsymbol{u}_{k+i-2}\right] + \boldsymbol{B}\boldsymbol{u}_{k+i-1} \\
&\vdots \\
&= \boldsymbol{A}^i \boldsymbol{x}_k + \sum_{j=0}^{i-1} \boldsymbol{A}^j \boldsymbol{B}\boldsymbol{u}_{k+i-j-1}
\end{aligned} \tag{4.11}$$

for $i = 1, \ldots, H$. Define $\boldsymbol{\Pi}_H = \begin{bmatrix} \boldsymbol{u}_k^T & \boldsymbol{u}_{k+1}^T & \cdots & \boldsymbol{u}_{k+H-1}^T \end{bmatrix}^T \in \mathbb{R}^{mH}$ and

$X_H = \begin{bmatrix} x_k^T & x_{k+1}^T & \cdots & x_{k+H-1}^T \end{bmatrix}^T \in \mathbb{R}^{nH}$. For $i = H$, (4.11) is put into the compact form

$$x_{k+H} = A^H x_k + N_u \Pi_H \tag{4.12}$$

where $N_u = \begin{bmatrix} A^{H-1}B & \cdots & AB & B \end{bmatrix} \in \mathbb{R}^{n \times mH}$. Utilizing (4.11), X_H can be reformulated into the following equation as a function of only the current state x_k:

$$X_H = \underbrace{\begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^{H-1} \end{bmatrix}}_{M_x \in \mathbb{R}^{nH \times n}} x_k + \underbrace{\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ B & 0 & 0 & \cdots & 0 & 0 \\ AB & B & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ A^{H-2}B & A^{H-3}B & A^{H-4}B & \cdots & B & 0 \end{bmatrix}}_{M_u \in \mathbb{R}^{nH \times mH}} \Pi_H = M_x x_k + M_u \Pi_H \tag{4.13}$$

Considering (4.9), the first term is expressed in terms of (4.12) as

$$\|x_{k+H}\|_{P_0}^2 = x_k^T \left[\left(A^H\right)^T P_0 A^H \right] x_k + \Pi_H^T \left[N_u^T P_0 N_u \right] \Pi_H + x_k^T \left[2\left(A^H\right)^T P_0 N_u \right] \Pi_H \tag{4.14}$$

Substitution of (4.13) into the second term of (4.9) yields

$$\sum_{i=0}^{H-1} \|x_{k+i}\|_Q^2 = x_k^T \left[M_x^T Q M_x \right] x_k + \Pi_H^T \left[M_u^T Q M_u \right] \Pi_H + x_k^T \left[2 M_x^T Q M_u \right] \Pi_H \tag{4.15}$$

The third term of (4.9) can then be regrouped in terms of Π_H as

$$\sum_{i=0}^{H-1} \|u_{k+i}\|_R^2 = \Pi_H^T \bar{R} \Pi_H \tag{4.16}$$

where $\bar{R} = \text{diag}(R, \ldots, R) \in \mathbb{R}^{mH \times mH}$. By combining (4.14)-(4.16) into (4.9), the following standard QP formulation of the MPC cost function is obtained:

$$V_H(x_k, \pi_k) = \min_{\Pi_H} \left\{ \tfrac{1}{2} x_k^T Y_r x_k + \tfrac{1}{2} \Pi_H^T J_r \Pi_H + x_k^T F_r \Pi_H \right\} \tag{4.17}$$

in which

$$Y_r = 2\left(A^H\right)^T P_0 A^H + 2 M_x^T Q M_x \tag{4.18}$$

$$J_r = 2 N_u^T P_0 N_u + 2 M_u^T Q M_u + 2\bar{R} \tag{4.19}$$

$$\boldsymbol{F}_r = 2\left(\boldsymbol{A}^H\right)^T \boldsymbol{P}_0 \boldsymbol{N}_u + 2\boldsymbol{M}_x^T \boldsymbol{Q} \boldsymbol{M}_u \tag{4.20}$$

Note that since the term $\frac{1}{2}\boldsymbol{x}_k^T \boldsymbol{Y}_r \boldsymbol{x}_k$ does not depend on the variable of optimization $\boldsymbol{\Pi}_H$, it can be ignored in the optimization process. Constraints are incorporated in the MPC optimization of (4.17) through successive row augmentation into a single matrix inequality equation (not LMI) given as

$$\boldsymbol{G}_r \boldsymbol{\Pi}_H \leq \boldsymbol{W}_r + \boldsymbol{E}_r \boldsymbol{x}_k \tag{4.21}$$

where $\boldsymbol{G}_r$, $\boldsymbol{W}_r$ and $\boldsymbol{E}_r$ are matrices constructed from a particular set of constraints of interest. For instance, for a system subjected to the input constraints $-c_1 \leq u_{ij} \leq c_2$ for $i = 0,...,H-1$ and $j = 1,...,m$; $\boldsymbol{G}_r$, $\boldsymbol{W}_r$ and $\boldsymbol{E}_r$ are constructed as

$$\boldsymbol{G}_r = \begin{bmatrix} \boldsymbol{I}_{Hm} \\ -\boldsymbol{I}_{Hm} \end{bmatrix} \in \mathbb{R}^{2Hm \times Hm}$$

$$\boldsymbol{W}_r = \begin{bmatrix} c_2 & \cdots & c_2 & c_1 & \cdots & c_1 \end{bmatrix}^T \in \mathbb{R}^{2Hm}$$

$$\boldsymbol{E}_r = \boldsymbol{0} \in \mathbb{R}^{2Hm \times n}$$

An additional state constraint of $\boldsymbol{x}_{k+i} \leq \boldsymbol{x}_{\max} = \begin{bmatrix} s_1 & \cdots & s_n \end{bmatrix}^T \in \mathbb{R}^n$ can be imposed by using (4.11) and row-augmenting these $\boldsymbol{G}_r$, $\boldsymbol{W}_r$ and $\boldsymbol{E}_r$ by the following matrices:

$$\boldsymbol{G}_{\text{aug}} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{AB} & \boldsymbol{B} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \boldsymbol{A}^{H-3}\boldsymbol{B} & \boldsymbol{A}^{H-4}\boldsymbol{B} & \boldsymbol{A}^{H-5}\boldsymbol{B} & \cdots & \boldsymbol{B} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{A}^{H-2}\boldsymbol{B} & \boldsymbol{A}^{H-3}\boldsymbol{B} & \boldsymbol{A}^{H-4}\boldsymbol{B} & \cdots & \boldsymbol{AB} & \boldsymbol{B} & \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{(H-1)n \times Hm}$$

$$\boldsymbol{W}_{\text{aug}} = \begin{bmatrix} \boldsymbol{x}_{\max}^T & \cdots & \boldsymbol{x}_{\max}^T \end{bmatrix}^T \in \mathbb{R}^{(H-1)n}$$

$$\boldsymbol{E}_{\text{aug}} = -\begin{bmatrix} \boldsymbol{A}^T & \left(\boldsymbol{A}^2\right)^T & \cdots & \left(\boldsymbol{A}^{H-1}\right)^T \end{bmatrix}^T \in \mathbb{R}^{(H-1)n \times n}$$

Subsequent input and state constraints can be imposed into the optimization of (4.17) in a similar manner.

4.2.2 The Reference Tracking Problem

Referring to the MPC policy (4.3) for the reference tracking problem, it is the usual practice in MPC to ignore the terminal cost and substitute $\boldsymbol{u}_{k+i}$ by $\Delta\boldsymbol{u}_{k+i}$ $(=\boldsymbol{u}_{k+i}-\boldsymbol{u}_{k+i-1})$ in order to prioritize the tracking performance (clearly, at the cost of degraded stability as a tradeoff) of the system. The $\Delta\boldsymbol{u}_{k+i}$ formulation corresponds to including an integrator in the control loop (Bemporad, *et al.*, 2002). Assuming that $H_2 = H_u = H$, the following MPC quadratic cost function is considered now:

$$
\begin{aligned}
& V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k) = \min_{\boldsymbol{\pi}(\cdot|k)} \left\{ \sum_{i=0}^{H-1} \left\| \boldsymbol{y}_{k+i|k} - \boldsymbol{r}_{k+i} \right\|_{\boldsymbol{Q}}^2 + \sum_{i=0}^{H-1} \left\| \Delta\boldsymbol{u}_{k+i|k} \right\|_{\boldsymbol{R}}^2 \right\} \\
& \text{s.t.} \\
& \boldsymbol{y}_{k+i|k} \in \mathcal{Y}_i \quad \forall i = 1,\ldots,H \\
& \boldsymbol{u}_{k+i|k} \in \mathcal{U} \quad \forall i = 0,\ldots,H-1
\end{aligned}
\tag{4.22}
$$

Again, the purpose is to transform (4.22) into the standard QP formulation for optimization. The first term of (4.22) can be reduced into the following compact form:

$$
\sum_{i=0}^{H-1} \left\| \boldsymbol{y}_{k+i|k} - \boldsymbol{r}_{k+i} \right\|_{\boldsymbol{Q}}^2 = \boldsymbol{M}_y^T \bar{\boldsymbol{Q}} \boldsymbol{M}_y \tag{4.23}
$$

where

$$
\bar{\boldsymbol{Q}} = \operatorname{diag}(\boldsymbol{Q},\ldots,\boldsymbol{Q}) \in \mathbb{R}^{pH \times pH} \quad \text{and} \quad \boldsymbol{M}_y = \begin{bmatrix} \left(\boldsymbol{y}_{k|k} - \boldsymbol{r}_k\right)^T & \cdots & \left(\boldsymbol{y}_{k+H-1|k} - \boldsymbol{r}_{k+H-1}\right)^T \end{bmatrix}^T \in \mathbb{R}^{pH}.
$$

Define $\Delta\boldsymbol{\Pi}_H = \begin{bmatrix} \Delta\boldsymbol{u}_k^T & \Delta\boldsymbol{u}_{k+1}^T & \cdots & \Delta\boldsymbol{u}_{k+H-1}^T \end{bmatrix}^T$ and introduce a new, augmented state vector $\boldsymbol{X}_s = \begin{bmatrix} \boldsymbol{x}_k^T & \boldsymbol{u}_{k-1}^T & \boldsymbol{r}_k^T & \cdots & \boldsymbol{r}_{k+H-1}^T \end{bmatrix} \in \mathbb{R}^{n+m+pH}$. Using (4.11) and considering $\boldsymbol{y}_{k+i} = \boldsymbol{C}\boldsymbol{x}_{k+i}$ with $\boldsymbol{C} \in \mathbb{R}^{p \times n}$, $\boldsymbol{M}_y$ is reformulated as:

$$
\boldsymbol{M}_y = \boldsymbol{N}_x \boldsymbol{X}_s + \boldsymbol{N}_\Delta \Delta\boldsymbol{\Pi}_H \tag{4.24}
$$

where

$$N_x = \begin{bmatrix} C & 0 & -I_p & 0 & 0 & \cdots & 0 \\ CA & CB & 0 & -I_p & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{H-1} & \sum_{i=0}^{H-2} CA^i B & 0 & 0 & 0 & \cdots & -I_p \end{bmatrix} \in \mathbb{R}^{pH \times n+m+pH}$$

$$N_\Delta = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 0 \\ CB & 0 & \cdots & 0 & 0 & 0 \\ \sum_{i=0}^{1} CA^i B & CB & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \sum_{i=0}^{H-3} CA^i B & \sum_{i=0}^{H-4} CA^i B & \cdots & CB & 0 & 0 \\ \sum_{i=0}^{H-2} CA^i B & \sum_{i=0}^{H-3} CA^i B & \cdots & \sum_{i=0}^{1} CA^i B & CB & 0 \end{bmatrix} \in \mathbb{R}^{pH \times mH}$$

Furthermore, the second term of (4.22) is recast as

$$\sum_{i=0}^{H-1} \left\| \Delta u_{k+i} \right\|_R^2 = \Delta \Pi_H^T \bar{R} \Delta \Pi_H \tag{4.25}$$

where $\bar{R} = \text{diag}(R, \ldots, R) \in \mathbb{R}^{mH \times mH}$. By combining (4.23)-(4.25) with (4.22), the following standard QP form for the MPC optimization is obtained:

$$V_H(x_k, \pi_k) = \min_{\Delta \Pi_H} \left\{ \tfrac{1}{2} X_s^T Y_y X_s + \tfrac{1}{2} \Delta \Pi_H^T J_y \Delta \Pi_H + X_s^T F_y \Delta \Pi_H \right\} \tag{4.26}$$

in which

$$Y_y = 2 N_x^T \bar{Q} N_x \tag{4.27}$$

$$J_y = 2 N_\Delta^T \bar{Q} N_\Delta + 2\bar{R} \tag{4.28}$$

$$F_y = 2 N_x^T \bar{Q} N_\Delta \tag{4.29}$$

As in the regulator problem, the term $\frac{1}{2} X_s^T Y_y X_s$ is excluded in the optimization since it is independent of $\Delta \Pi_H$. Similarly, constraints are imposed using successive row augmentation into the same form as (4.21), replacing Π_H and x_k with $\Delta \Pi_H$ and X_s, respectively.

4.3 Stability Analysis

The approach adopted here to analyze the stability of the developed system with future control input buffering, originates from the work of Primbs and Nevistić (2000) where the terminal states $\boldsymbol{x}_{k+H}$ are related to the current states $\boldsymbol{x}_k$ by establishing a bound on the terminal cost. The levels of projected cost, which depend on the controller-to-actuator delay τ_{ca}, are essentially bounded by predetermined upper and lower bound terms. Stability is achieved without end constraints. It is known that stability particularly concerns the problem of regulation of a closed-loop system. On the other hand, the problem of reference tracking is usually analyzed in the context of performance. Hence, the stability analysis given here is focused on the MPC policy of (4.9).

4.3.1 Preliminaries

In this section, the basis of stability analysis in the sense of Lyapunov is reviewed and then formulated in the context of MPC. Next, a valid Lyapunov function is identified, in order to establish the conditions under which the closed-loop NCS is guaranteed to be asymptotically stable in the sense of Lyapunov.

Any function $f:\mathbb{R}^{\geq 0}\rightarrow\mathbb{R}^{\geq 0}$ is termed a class $\mathcal{K}$-function if it is continuous, strictly increasing, and $f(0)=0$. The same function is termed a class $\mathcal{K}_\infty$-functions if it also satisfies $f(s)\rightarrow\infty$ as $s\rightarrow\infty$.

Definition 4.1: *An equilibrium point* $\boldsymbol{x}_e$ *is stable if, there exists* $r>0$ *for every* $\bar{r}>0$, *such that if* $\|\boldsymbol{x}_k-\boldsymbol{x}_e\|<r$, *then* $\|\boldsymbol{x}_{k+i}-\boldsymbol{x}_e\|<\bar{r}$ *for* $\forall i\geq 0$.

Definition 4.2: *An equilibrium point* $\boldsymbol{x}_e$ *is asymptotically stable if it is stable, and in addition, if there exists some* $r>0$ *such that* $\|\boldsymbol{x}_k-\boldsymbol{x}_e\|<r$ *implies that* $\|\boldsymbol{x}_{k+i}-\boldsymbol{x}_e\|\rightarrow 0$ *as* $i\rightarrow\infty$.

Definition 4.3 (cf. Jiang and Wang, 2001): *A continuous function* $V(\cdot):\mathbb{R}^n\rightarrow\mathbb{R}^+$ *is a valid Lyapunov function for system (4.1) if there exist some class* $\mathcal{K}_\infty$*-functions* $\gamma_1(\cdot)$, $\gamma_2(\cdot)$ *and* $\gamma_3(\cdot)$, *and a class* $\mathcal{K}$*-function* $\sigma(\cdot)$ *such that* $\gamma_1(\|\boldsymbol{x}\|)\leq V(\boldsymbol{x})\leq\gamma_2(\|\boldsymbol{x}\|)$ *and* $V(f(\boldsymbol{x},\boldsymbol{u}))-V(\boldsymbol{x})\leq-\gamma_3(\|\boldsymbol{x}\|)+\sigma(\|\boldsymbol{u}\|)$ *for* $\forall\boldsymbol{x}\in\mathbb{R}^n$ *and* $\forall\boldsymbol{u}\in\mathbb{R}^m$.

Remark 4.1: The last condition in Definition 4.3 is equivalent to stating that there exist some

class $\mathcal{K}_\infty$-functions $\gamma_4(\cdot)$ and some class $\mathcal{K}$-function $\gamma_5(\cdot)$ such that $\|x\| \geq \gamma_5(\|u\|)$ implies $V(f(x,u)) - V(x) \leq -\gamma_4(\|x\|)$. Note that this results in an equivalent property if $\gamma_4(\cdot)$ is merely required to be continuous and positive definite.

Having provided these definitions, a theorem is given next, which states the fundamental conditions for asymptotic stability in the sense of Lyapunov's second method.

Theorem 4.1: *The system under control of (4.1) is asymptotically stable in the sense of Lyapunov if there exists a function* $V(\cdot): \mathbb{R}^n \to \mathbb{R}^+$ *which satisfies the following:*

1. *There exists a class* $\mathcal{K}_\infty$*-function* $\alpha(\cdot)$ *such that* $0 < \alpha(\|x_k\|) \leq V(x_k)$ *for* $\forall x_k \neq 0$.
2. *There exists a class* $\mathcal{K}_\infty$*-function* $\upsilon(\cdot)$ *such that* $V(x_{k+1}) - V(x_k) \leq -\upsilon(\|x_k\|)$.
3. $V(0) = 0$ *and* $V(\cdot)$ *is continuous.*

Proof:

Stability: This proof is by contradiction. Let $\bar{r}$ be any arbitrary point with $\bar{r} > 0$. Since $V(\cdot)$ is continuous, there exists $r > 0$ such that $V(x_k) \geq \alpha(\bar{r})$ for $\|x_k\| < r$. Presume that the system is unstable. Then there exists some $i \geq 0$ such that $\|x_{k+i}\| \geq \bar{r}$ and

$$V(x_k) < \alpha(\bar{r}) \leq \alpha(\|x_{k+i}\|) \leq V(x_{k+i})$$

This contradicts the hypothesis of $V(x_{k+i}) - V(x_k) \leq 0$. Therefore, the system is stable.

Convergence: Presume that the condition $V(x_{k+1}) - V(x_k) \leq -\upsilon(\|x_k\|)$ is true. With $V(x_k) > 0$, taking the sum from k to $k+i$ gives

$$-V(x_k) \leq V(x_{k+i}) - V(x_k) \leq -\sum_{j=k}^{k+i} \upsilon(\|x_j\|)$$

Since $\upsilon(\|x_j\|)$ is non-negative and $-\sum_{j=k}^{k+i} \upsilon(\|x_j\|)$ is bounded from below by $-V(x_k)$, one has $\lim_{i\to\infty} \upsilon(\|x_{j+i}\|) = 0$. Since $\upsilon(0) = 0$ and is continuous, the limit implies that $\|x_{k+i}\| \to 0$. □

Having established the basis of asymptotic stability in the sense of Lyapunov from Theorem 4.1, the next step is to establish that the system (4.1) using a finite sequence of future control actions $\pi_k = \{u_{k|k}, ..., u_{k+H-1|k}\}$ determined under the MPC control law of (4.4) is

asymptotically stable.

Definition 4.4: For $\forall r \geq 0$ and $\forall n \geq 1$, the set $\mathcal{B}_r^n$ is defined as $\mathcal{B}_r^n := \{x \in \mathbb{R}^n : \|x\| \leq r\}$.

Theorem 4.2 (cf. Sockaert, *et al.*, 1999): *The system under control of (4.1) using a finite sequence of future control actions* $\pi_k = \{u_{k|k}, ..., u_{k+H-1|k}\}$ *determined under the MPC control law of (4.4) is asymptotically stable in the sense of Lyapunov in the neighborhood* $\mathcal{X} \subseteq \mathbb{R}^n$ *if the following conditions are satisfied:*

1. *a function* $V(\cdot): \mathbb{R}^n \times \mathbb{R}^{mH} \to \mathbb{R}^+$ *with* $V(0,0) = 0$ *exists such that* $V(x,\pi) \geq \alpha_1(\|x\|)$ *where* $\alpha_1(\cdot)$ *is a* $\mathcal{K}_\infty$*-function.*
2. $\mathcal{X}$ *exists and contains an open neighborhood of the origin, such that every realization* $\{x_k, \pi_k\}$ *of the controlled system with* $x_0 \in \mathcal{X}$ *satisfies* $x_k \in \mathcal{X}$ *for* $\forall k \geq 0$ *and* $V(x_{k+1}, \pi_{k+1}) - V(x_k, \pi_k) \leq -\alpha_2(\|(x_k, u_k)\|)$ *where* $\alpha_2(\cdot)$ *is a* $\mathcal{K}_\infty$*-function.*
3. *a constant* $r > 0$ *exists for* $x_k \in \mathcal{B}_r^n$*, such that every realization* $\{x_k, \pi_k\}$ *of the controlled system with* $x_k \in \mathcal{B}_r^n$ *satisfies* $\|\pi_k\| \leq \alpha_3(\|x_k\|)$ *where* $\alpha_3(\cdot)$ *is a* $\mathcal{K}_\infty$*-function.*

Proof:

Stability: Let x_0 be an arbitrary point within $\mathcal{X}$. Since $V(0,0) = 0$ and $V(\cdot)$ is continuous at the origin, there exists a constant $r_1 > 0$ and a $\mathcal{K}$-function $\beta(\cdot)$ such that $V(x,\pi) \leq \beta(\|(x,\pi)\|)$ for $\forall x \in \mathcal{B}_{r_1}^n$ and $\forall \pi \in \mathcal{B}_{r_1}^{Hm}$. Since $\mathcal{X}$ contains the origin, there exists a constant $r_2 > 0$ such that $\mathcal{B}_{r_2}^n \subseteq \mathcal{X}$. For any $\varepsilon > 0$, there exists $\delta > 0$ such that $\delta \leq \min(r, r_1, r_2)$, $\alpha_3(\delta) \leq r_1$, and $\beta(\delta + \alpha_3(\delta)) \leq \alpha_1(\varepsilon)$. Now $\delta > 0$ exists because $\alpha_1(\varepsilon) > 0$ and $\alpha_3(\delta) \to 0$ as $\delta \to 0$, so that $\beta(\delta + \alpha_3(\delta)) \to 0$ as $\delta \to 0$. Suppose $\|x_0\| \leq \delta$; then $\|\pi_0\| \leq \alpha_3(\delta)$ and

$$V(x_0, \pi_0) \leq \beta(\|(x_0, \pi_0)\|) \leq \beta(\|x_0\| + \|\pi_0\|) \leq \beta(\delta + \alpha_3(\delta)) \leq \alpha_1(\varepsilon)$$

In addition, one has

$$V(x_k, \pi_k) \leq V(x_0, \pi_0) \leq \alpha_1(\varepsilon)$$

for $\forall k \geq 0$, since $\|x_0\| \leq \delta$. Moreover, $\alpha_1(\|x_k\|) \leq V(x_k, \pi_k)$ for $\forall k \geq 0$. Therefore, it can be established that $\alpha_1(\|x_k\|) < \alpha_1(\varepsilon)$, which implies that $\|x_k\| < \varepsilon$ for $\forall k \geq 0$ and $\forall x_0 \in \mathcal{B}_\delta^n$.

Convergence: The condition $V(\boldsymbol{x},\boldsymbol{\pi}) \geq \alpha_1(\|\boldsymbol{x}\|)$ includes $V(\boldsymbol{x},\boldsymbol{\pi}) \geq 0$ for $\forall \boldsymbol{x}$ and $\forall \boldsymbol{\pi}$. The condition $V(\boldsymbol{x}_{k+1},\boldsymbol{\pi}_{k+1}) - V(\boldsymbol{x}_k,\boldsymbol{\pi}_k) \leq -\alpha_2(\|(\boldsymbol{x}_k,\boldsymbol{u}_k)\|)$ infers that $V(\cdot)$ decreases along the trajectories of the controlled system that commenced within $\mathbb{X}$. It follows that with $\boldsymbol{x}_0 \in \mathbb{X}$, $V(\boldsymbol{x}_k,\boldsymbol{\pi}_k) \to V^*$ as $k \to \infty$, where V^* is a non-negative constant. It can be concluded that $V(\boldsymbol{x}_{k+1},\boldsymbol{\pi}_{k+1}) - V(\boldsymbol{x}_k,\boldsymbol{\pi}_k) \to 0$ as $k \to \infty$ and this implies that $\alpha_2(\|(\boldsymbol{x}_k,\boldsymbol{u}_k)\|) \to 0$. Since $\alpha_2(\cdot)$ is a class $\mathcal{K}$-function, it follows that $\|\boldsymbol{x}_k\| \to 0$ and $\|\boldsymbol{u}_k\| \to 0$ as $k \to \infty$. □

Hence, in the nominal case, if the initial optimal sequence of control input $\boldsymbol{\pi}_0^*$ is feasible, the elements $\boldsymbol{u}_{k+i|k}$ of the subsequent sequence of control input $\boldsymbol{\pi}_k^*$ as computed using the MPC policy of (4.4) or (4.9) are sufficiently stabilizing in the future as long as the time point is within H. In order to analyze the conditions in which the developed NCS-MPC control policy is asymptotically stable, a suitable candidate for the Lyapunov function needs to be identified.

Definition 4.5: *A function f is a Lipschitz function if $\|f(\boldsymbol{a}) - f(\boldsymbol{b})\| \leq L_f \|\boldsymbol{a} - \boldsymbol{b}\|$ for $\forall \boldsymbol{a} \in \mathbb{R}^c$, $\forall \boldsymbol{b} \in \mathbb{R}^c$ and $c \geq 1$; and L_f is called the corresponding Lipschitz constant.*

Lemma 4.1: *The finite quadratic cost function of the receding horizon MPC policy is a valid Lyapunov function.*

Proof:

Assume $\boldsymbol{P}_0 = \boldsymbol{Q}$ for brevity and consider the following "optimal" quadratic cost function:

$$V_H^*(\boldsymbol{x}_k,\boldsymbol{\pi}_k) = \sum_{i=0}^{H} \left[\|\boldsymbol{x}_{k+i|k}^*\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{u}_{k+i|k}^*\|_{\boldsymbol{R}}^2 \right] \tag{4.30}$$

It should be noted that $\boldsymbol{u}_{k+H|k} = 0$. Since $\boldsymbol{x}_{k|k}^* = \boldsymbol{x}_k$, one has $V_H^*(\boldsymbol{x}_k,\boldsymbol{\pi}_k) \geq \|\boldsymbol{x}_k\|_{\boldsymbol{Q}}^2$ and thus there exists a class $\mathcal{K}_\infty$-function $\gamma_1(\cdot)$ such that

$$V_H^*(\boldsymbol{x}_k,\boldsymbol{\pi}_k) \geq \gamma_1(\|\boldsymbol{x}_k\|) \tag{4.31}$$

For example, $\gamma_1(\|\boldsymbol{x}_k\|) = K_1\|\boldsymbol{x}_k\|$ where K_1 is a non-negative constant. Consider (4.1) and let $\boldsymbol{u}_{k+i} = \phi_i(\boldsymbol{x}_{k+i})$ with $\phi_i : \mathbb{R}^n \to \mathbb{R}^m$ for $i = 0,\ldots,H-1$. Let L_f, L_{ϕ_0}, L_{ϕ_1},..., $L_{\phi_{H-1}}$ be the Lipschitz constants for $f(\cdot)$, $\phi_0(\cdot)$, $\phi_1(\cdot)$,..., $\phi_{H-1}(\cdot)$, respectively. It follows that

$\left\|\boldsymbol{u}^*_{k+i|k}\right\| \le L_{\phi_j}\left\|\boldsymbol{x}_{k+i|k}\right\|$ for $i=0,\ldots,H-1$. Using this relation, $\left\|\boldsymbol{x}^*_{k+i|k}\right\|$, for $i=1,\ldots,H$, can be formed in terms of the current state $\|\boldsymbol{x}_k\|$, as follows:

$$\left\|\boldsymbol{x}^*_{k+1|k}\right\| = \left\|f(\boldsymbol{x}^*_{k|k},\boldsymbol{u}^*_{k|k})\right\| \le L_f\|\boldsymbol{x}_k\| + L_f\left\|\boldsymbol{u}^*_{k|k}\right\| \le L_f\|\boldsymbol{x}_k\| + L_f L_{\phi_0}\|\boldsymbol{x}_k\| = L_1\|\boldsymbol{x}_k\|$$

$$\left\|\boldsymbol{x}^*_{k+2|k}\right\| = \left\|f(\boldsymbol{x}^*_{k+1|k},\boldsymbol{u}^*_{k+1|k})\right\| \le L_f\left\|\boldsymbol{x}^*_{k+1|k}\right\| + L_f\left\|\boldsymbol{u}^*_{k+1|k}\right\| \le L_f L_1\|\boldsymbol{x}_k\| + L_f L_{\phi_1}\|\boldsymbol{x}_k\| = L_2\|\boldsymbol{x}_k\|$$

$$\vdots$$

$$\left\|\boldsymbol{x}^*_{k+H|k}\right\| \le L_f\left\|\boldsymbol{x}^*_{k+H-1|k}\right\| + L_f\left\|\boldsymbol{u}^*_{k+H-1|k}\right\| \le L_f L_{H-1}\|\boldsymbol{x}_k\| + L_f L_{\phi_{H-1}}\|\boldsymbol{x}_k\| = L_H\|\boldsymbol{x}_k\|$$

in which $L_1 = L_f(1+L_{\phi_0})$, $L_2 = L_f(L_1+L_{\phi_1})$,..., $L_H = L_f(L_{H-1}+L_{\phi_{H-1}})$. Hence, (4.30) can be bounded from above as follows:

$$\begin{aligned}
&V^*_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k)\\
&\quad= \left\|\boldsymbol{x}^*_{k|k}\right\|^2_{\boldsymbol{Q}} + \cdots + \left\|\boldsymbol{x}^*_{k+H|k}\right\|^2_{\boldsymbol{Q}} + \left\|\boldsymbol{u}^*_{k|k}\right\|^2_{\boldsymbol{R}} + \cdots + \left\|\boldsymbol{u}^*_{k+H-1|k}\right\|^2_{\boldsymbol{R}}\\
&\quad\le \|\boldsymbol{x}_k\|^2_{\boldsymbol{Q}} + \cdots + L_H\|\boldsymbol{x}_k\|^2_{\boldsymbol{Q}} + L_{\phi_0}\|\boldsymbol{x}_k\|^2_{\boldsymbol{R}} + \cdots + L_{\phi_{H-1}}\|\boldsymbol{x}_k\|^2_{\boldsymbol{R}}\\
&\quad\le (1+L_1+\cdots+L_H)\|\boldsymbol{x}_k\|^2_{\boldsymbol{Q}} + (L_{\phi_0}+\cdots+L_{\phi_{H-1}})\|\boldsymbol{x}_k\|^2_{\boldsymbol{R}}
\end{aligned}$$

Therefore, there exists a class $\mathcal{K}_\infty$-function $\gamma_2(\cdot)$ such that

$$V^*_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k) \ge \gamma_2(\|\boldsymbol{x}_k\|) \tag{4.32}$$

Considering the cost function one step ahead; i.e., $V_H(\boldsymbol{x}_{k+1},\boldsymbol{\pi}_{k+1})$, according to Theorem 4.2, it is feasible but not necessarily optimum to use the state and the input that are predicted in the current time step k to arrive at:

$$V_H(\boldsymbol{x}_{k+1},\boldsymbol{\pi}_{k+1}) = \sum_{i=1}^{H}\left[\left\|\boldsymbol{x}^*_{k+i|k}\right\|^2_{\boldsymbol{Q}} + \left\|\boldsymbol{u}^*_{k+i|k}\right\|^2_{\boldsymbol{R}}\right] = V^*_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k) - \left\|\boldsymbol{x}^*_{k|k}\right\|^2_{\boldsymbol{Q}} - \left\|\boldsymbol{u}^*_{k|k}\right\|^2_{\boldsymbol{R}}$$

The optimal $V^*_H(\boldsymbol{x}_{k+1},\boldsymbol{\pi}_{k+1})$ is then bounded from above as follows:

$$V^*_H(\boldsymbol{x}_{k+1},\boldsymbol{\pi}_{k+1}) \le V^*_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k) - \left\|\boldsymbol{x}^*_{k|k}\right\|^2_{\boldsymbol{Q}} - \left\|\boldsymbol{u}^*_{k|k}\right\|^2_{\boldsymbol{R}}$$

This result in

$$V_H^*(\boldsymbol{x}_{k+1},\boldsymbol{\pi}_{k+1})-V_H^*(\boldsymbol{x}_k,\boldsymbol{\pi}_k)\le-\left\|\boldsymbol{x}_{k|k}^*\right\|_{\boldsymbol{Q}}^2-\left\|\boldsymbol{u}_{k|k}^*\right\|_{\boldsymbol{R}}^2\le-\left\|\boldsymbol{x}_{k|k}^*\right\|_{\boldsymbol{Q}}^2$$

which means, there exists a class $\mathcal{K}_\infty$-function $\gamma_3(\cdot)$ such that

$$V_H^*(\boldsymbol{x}_{k+1},\boldsymbol{\pi}_{k+1})-V_H^*(\boldsymbol{x}_k,\boldsymbol{\pi}_k)\le-\gamma_3(\|\boldsymbol{x}_k\|) \tag{4.33}$$

According to Definition 4.3, the combination of (4.31)-(4.33) is sufficient to conclude that (4.30) and thus (4.4) and (4.9) are valid Lyapunov functions. □

4.3.2 Maintaining Feasibility with Suboptimal MPC

Under less than nominal conditions, Theorem 4.2 alone is not sufficient to ensure the feasibility of the control inputs $\boldsymbol{u}_{k+i|k}$ from $\boldsymbol{\pi}_k^*$ in the subsequent time steps $k+i$ for $i\le\tau_{ca}\le H-1$ as applied in the developed NCS buffering strategy. To overcome this, an additional condition is enforced into the MPC policy. Consider the states and control inputs at k and $k+\tau_{ca}$. The optimizing $\boldsymbol{\pi}_k^*$ is found at sample time k. Denote the corresponding cost by $V_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k^*)$ and the optimizing states and the control input sequence, respectively, by

$$V_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k^*)\begin{Bmatrix}\boldsymbol{u}_{k|k}^*, & \boldsymbol{u}_{k+1|k}^*, & \ldots, & \boldsymbol{u}_{k+H-1|k}^*\\ \boldsymbol{x}_{k|k}^*, & \boldsymbol{x}_{k+1|k}^*, & \ldots, & \boldsymbol{x}_{k+H-1|k}^*, & \boldsymbol{x}_{k+H|k}^*\end{Bmatrix}$$

At sample time $k+\tau_{ca}$, the corresponding control input of $\boldsymbol{\pi}_k$ as computed previously and stored in the buffer, is used. Now use the notation:

$\boldsymbol{\pi}_{pre}=\left\{\boldsymbol{u}_{k+\tau_{ca}|k},\boldsymbol{u}_{k+\tau_{ca}+1|k},\ldots,\boldsymbol{u}_{k+H-1|k},\boldsymbol{0},\ldots,\boldsymbol{0}\right\}$ with the corresponding cost $V_H(\boldsymbol{x}_k,\boldsymbol{\pi}_{pre})$, which is formed from the following sequence of states and control inputs:

$$V_H(\boldsymbol{x}_k,\boldsymbol{\pi}_{pre})\begin{Bmatrix}\boldsymbol{u}_{k+\tau_{ca}|k}, & \boldsymbol{u}_{k+\tau_{ca}+1|k}, & \ldots, & \boldsymbol{u}_{k+H-1|k}\\ \boldsymbol{x}_{k+\tau_{ca}|k+\tau_{ca}}, & \boldsymbol{x}_{k+\tau_{ca}+1|k+\tau_{ca}}, & \ldots, & \boldsymbol{x}_{k+H-1|k+\tau_{ca}}, & \boldsymbol{x}_{k+H|k+\tau_{ca}}\end{Bmatrix}$$

The following condition guarantees that the closed-loop NCS is asymptotically stabilizing within τ_{ca} by satisfying the second condition of Theorem 4.2:

$$V_H(\boldsymbol{x}_{k+\tau_{ca}},\boldsymbol{\pi}_{pre})-V_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k^*)\le-\mu\sum_{i=0}^{\tau_{ca}-1}W_k(\boldsymbol{x}_{k+i},\boldsymbol{u}_{k+i}) \tag{4.34}$$

where $\mu \in (0,1]$;

$$W_k(\boldsymbol{x}_{k+i}, \boldsymbol{u}_{k+i}) = \left\|\boldsymbol{x}_{k+i|k}\right\|_{\boldsymbol{Q}}^2 + \left\|\boldsymbol{u}_{k+i|k}\right\|_{\boldsymbol{R}}^2;$$

$$V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k^*) = \left\|\boldsymbol{x}_{k+H|k}\right\|_{\boldsymbol{P}_0}^2 + \sum_{i=0}^{H-1}\left\|\boldsymbol{x}_{k+i|k}\right\|_{\boldsymbol{Q}}^2 + \sum_{i=0}^{H-1}\left\|\boldsymbol{u}_{k+i|k}\right\|_{\boldsymbol{R}}^2; \text{ and}$$

$$V_H(\boldsymbol{x}_{k+\tau_{ca}}, \boldsymbol{\pi}_{pre}) = \left\|\boldsymbol{x}_{k+\tau_{ca}+H|k+\tau_{ca}}\right\|_{\boldsymbol{P}_0}^2 + \sum_{i=0}^{H-1}\left\|\boldsymbol{x}_{k+\tau_{ca}+i|k+\tau_{ca}}\right\|_{\boldsymbol{Q}}^2 + \sum_{i=0}^{H-\tau_{ca}-1}\left\|\boldsymbol{u}_{k+\tau_{ca}+i|k}\right\|_{\boldsymbol{R}}^2$$

Note that since $W_k(\boldsymbol{x}_{k+i}, \boldsymbol{u}_{k+i}) > 0$, by forcing the τ_{ca}-step cost difference to satisfy $V_H(\boldsymbol{x}_{k+\tau_{ca}}, \boldsymbol{\pi}_{pre}) - V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k^*) < 0$, (4.34) ensures that $V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k)$ will decrease along the trajectories of the closed-loop system. The design parameter μ is introduced to reduce the effect of model uncertainty and disturbances. Specifically, the smaller the μ, the less conservative the system, which is more desirable. The controller-to-actuator delay τ_{ca} is another design parameter, which specifies the worst-case delay the NCS strategy has to handle. Although the introduction of (4.34) into the MPC policy causes the controller to be suboptimal, feasibility is still maintained, which implies that asymptotic stability is still achieved.

4.3.3 Establishing the Bounds

Before proceeding to establish a stability condition for NCS with the developed strategy of future control input buffering, the terminal cost $\left\|\boldsymbol{x}_{k+H|k}\right\|_{\boldsymbol{P}_0}^2$ has to be bounded in order to formulate the cost reduction terms of (4.34). This follows directly from Primbs and Nevistić (2000). The procedure is outlined here for completeness. Assume that the finite horizon quadratic cost $V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k)$ is bounded from above and below by $\boldsymbol{U}_H^{\boldsymbol{P}_0}$ and $\boldsymbol{L}_H^{\boldsymbol{P}_0}$, respectively; i.e.,

$$\left\|\boldsymbol{x}_k\right\|_{\boldsymbol{L}_H^{\boldsymbol{P}_0}}^2 \le V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k) \le \left\|\boldsymbol{x}_k\right\|_{\boldsymbol{U}_H^{\boldsymbol{P}_0}}^2$$

Then, the cost at any time step ahead can be bounded from above by the following lemma.

Lemma 4.2: *The j-step ahead finite horizon quadratic cost function (4.9) of the MPC policy is bounded from above as*

$$V_{H-(j+1)}(\boldsymbol{x}_{k+(j+1)|k}, \boldsymbol{\pi}_{k+(j+1)}) \leq \|\boldsymbol{x}_k\|^2_{U_H^{P_0} - L_j^{Q}} \tag{4.35}$$

for $j = 0, \ldots, H-1$.

Proof:

Considering the lower bound of the cost, one has

$$V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k) \geq \|\boldsymbol{x}_k\|^2_{L_H^{P_0}} \geq \|\boldsymbol{x}_k\|^2_{L_j^{Q}} + V_{H-(j+1)}(\boldsymbol{x}_{k+(j+1)|k}, \boldsymbol{\pi}_{k+(j+1)}) \tag{4.36}$$

On the other hand, from the upper bound, one has $V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k) \leq \|\boldsymbol{x}_k\|^2_{U_H^{P_0}}$ giving $\|\boldsymbol{x}_k\|^2_{U_H^{P_0}} \geq \|\boldsymbol{x}_k\|^2_{L_j^{Q}} + V_{H-(j+1)}(\boldsymbol{x}_{k+(j+1)|k}, \boldsymbol{\pi}_{k+(j+1)})$ resulting in (4.35).[1] □

Lemma 4.2 can only be used when the current state $\boldsymbol{x}_k$ is known. In cases when $\boldsymbol{x}_k$ is unknown, this j-step ahead cost has to be determined from the ratio of the upper bound of the cost $V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k)$. The following lemma establishes the bound.

Lemma 4.3: *If* $V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k)$ *is bounded from above by an arbitrary constant* Ω, *then the* j*-step ahead finite horizon quadratic cost function (4.9) of the MPC policy is bounded from above as*

$$V_{H-(j+1)}(\boldsymbol{x}_{k+(j+1)|k}, \boldsymbol{\pi}_{k+(j+1)}) \leq \Omega \left(1 - \frac{1}{\bar{\lambda}\left(U_H^{P_0} \left(L_j^{Q} \right)^{-1} \right)} \right) \tag{4.37}$$

for $j = 0, \ldots, H-1$ *with* $\bar{\lambda}(\cdot)$ *as the maximum eigenvalue of a matrix.*

Proof:

The are two conditions to arrive at (4.37); i.e., whether $\|\boldsymbol{x}_k\|^2_{L_j^{Q}}$ from (4.36) is bounded from below or above by $\dfrac{1}{\bar{\lambda}\left(U_H^{P_0} \left(L_j^{Q} \right)^{-1} \right)} \Omega$. For the first condition, considering (4.36) one has

$$\frac{1}{\bar{\lambda}\left(U_H^{P_0} \left(L_j^{Q} \right)^{-1} \right)} \Omega + V_{H-(j+1)}(\boldsymbol{x}_{k+(j+1)|k}, \boldsymbol{\pi}_{k+(j+1)}) \leq V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k) \leq \Omega$$

[1] The superscripts to the lower- and upper- bounding terms indicate the terminal weight of interest.

yielding (4.37). For the second condition, one has

$$\bar{\lambda}\left(U_H^{P_0}\left(L_j^{Q}\right)^{-1}\right)\|x_k\|_{L_j^{Q}}^2 \le \Omega$$

which, by considering $V_H(x_k,\pi_k) \le \|x_k\|_{U_H^{P_0}}^2$ and (4.36), can be expanded to the following form:

$$\|x_k\|_{L_j^{Q}}^2 + V_{H-(j+1)}(x_{k+(j+1)|k},\pi_{k+(j+1)}) \le V_H(x_k,\pi_k) \le \|x_k\|_{U_H^{P_0}}^2 \le \bar{\lambda}\left(U_H^{P_0}\left(L_j^{Q}\right)^{-1}\right)\|x_k\|_{L_j^{Q}}^2 \le \Omega$$

arriving at (4.37) as well. This completes the proof. □

The bound on the terminal state $x_{k+H|k}$ can then be obtained by recursively applying Lemma 4.3 along the state evolution from $j=0$ to $j=H-1$ until the tightest bound for $x_{k+H|k}$ is achieved. However, Lemma 4.3 provides a rather conservative bound. This is elevated by modifying Lemma 4.3 into the following lemma.

Definition 4.6: *The parameter $\kappa_H(j)$ is defined as*

$$\kappa_H(j) = \begin{cases} 1 & \text{for } j=0 \\ \min\limits_{0\le i\le j-1} \kappa_H(i)\left(1-\dfrac{1}{\bar{\lambda}\left(U_{H-i}^{P_0}\left(L_{(j-i)-1}^{Q}\right)^{-1}\right)}\right) & \text{for } j>0 \end{cases} \tag{4.38}$$

Lemma 4.4: *If $V_H(x_k,\pi_k)$ is bounded from above by an arbitrary constant Ω, then the j-step ahead finite horizon quadratic cost function (4.9) of the MPC policy is bounded from above as*

$$V_{H-j}(x_{k+j|k},\pi_{k+j}) \le \kappa_{H-j}(j)\Omega \tag{4.39}$$

Proof:

This follows directly from Lemma 4.3 by direct substitution of (4.38) over $j=0,\ldots,H-1$. □

It should be noted that the value of Ω is unknown. The need for this value can be removed by combining Lemma 4.2 and Lemma 4.4 to bound the terminal cost in terms of the current state x_k.

Lemma 4.5: *If the finite horizon cost function (4.9) of the MPC policy is optimal, then the terminal cost is bounded from above as:*

$$\left\|x_{k+H|k}\right\|_{P_0}^2 \leq \min_{0\leq j\leq H-1} \kappa_{H-(j+1)}(H-(j+1))x_k^T\left(U_H^{P_0}-L_j^Q\right)x_k \tag{4.40}$$

Proof:

Combining (4.35), (4.38) and (4.39), the terminal cost is bounded as:

$$\left\|x_{k+H|k}\right\|_{P_0}^2 \leq \kappa_{H-(j+1)}(H-(j+1))x_k^T\left(U_H^{P_0}-L_j^Q\right)x_k$$

for $j=0,\ldots,H-1$. Since j is arbitrary, by using the j that gives the minimum cost, one obtains the tightest upper bound, arriving at (4.40). □

4.3.4 Computing the Lower and Upper Bounds

The global lower bounds L_j are computed by using the unconstrained MPC policy, by iteratively solving the following Riccati difference equation:

$$L_{j+1} = A^T\left[L_j - L_jB(B^TL_jB+R)^{-1}B^TL_j\right]A+Q \tag{4.41}$$

for $j=0,\ldots,H-1$ with $L_0=P_0$.

The computation of the upper bounds $U_j^{P_0}$ are more tedious. Consider the following equation:

$$\left\|x_{k+j|k}\right\|_Q^2+\left\|u_{k+j|k}\right\|_R^2+\left\|x_{k+j+1|k}\right\|_{U_{H-(j+1)}^{P_0}}^2 \leq \left\|x_{k+j|k}\right\|_{U_{H-j}^{P_0}}^2 \tag{4.42}$$

for $j=0,\ldots,H-1$ with $U_0^{P_0}=P_0$. Using a linear state feedback control law of the form $u_{k+j|k}=K_jx_{k+j|k}$ and substituting it into (4.5), one obtains

$$x_{k+j+1|k}=(A+BK_j)x_{k+j|k} \tag{4.43}$$

By substituting (4.43) into (4.42) and equating the matrix weighting of each norm, one obtains:

$$Q+K_j^TRK_j+(A+BK_j)^TU_{H-(j+1)}^{P_0}(A+BK_j)\leq U_{H-j}^{P_0} \tag{4.44}$$

It can be observed that (4.44) closely resembles the standard Linear Matrix Inequality (LMI) form through Schur complements (see Appendix A on the basic theory of LMI). The objective here is the following: given A, B, Q, and R, compute the minimum upper bounds $U_j^{P_0}$ in

order to provide the tightest upper bound for $V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k)$ through the minimization of the eigenvalue $\lambda\left(\boldsymbol{U}_H^{P_0}\left(\boldsymbol{L}_H^{P_0}\right)^{-1}\right)$ governed by (4.44). However, $\boldsymbol{K}_j$ is unknown rendering the direct optimization of (4.44) impossible, and this relation has to be restructured. Define $\tilde{\boldsymbol{U}}_j = \left(\boldsymbol{U}_{H-j}^{P_0}\right)^{-1}$ and $\tilde{\boldsymbol{K}}_j = \boldsymbol{K}_j \tilde{\boldsymbol{U}}_j$. By substituting $\tilde{\boldsymbol{U}}_j$ and $\tilde{\boldsymbol{K}}_j$ into (4.44) and then pre-multiplying and post-multiplying by $\tilde{\boldsymbol{U}}_j^T$ and $\tilde{\boldsymbol{U}}_j$, respectively, one obtains the following equivalent equation:

$$\tilde{\boldsymbol{U}}_j^T \boldsymbol{Q} \tilde{\boldsymbol{U}}_j + \tilde{\boldsymbol{K}}_j^T \boldsymbol{R} \tilde{\boldsymbol{K}}_j + (\boldsymbol{A}\tilde{\boldsymbol{U}}_j + \boldsymbol{B}\tilde{\boldsymbol{K}}_j)^T \tilde{\boldsymbol{U}}_{j+1}^{-1} (\boldsymbol{A}\tilde{\boldsymbol{U}}_j + \boldsymbol{B}\tilde{\boldsymbol{K}}_j) \le \tilde{\boldsymbol{U}}_j^T \tag{4.45}$$

or,
$$\tilde{\boldsymbol{U}}_j^T - (\boldsymbol{A}\tilde{\boldsymbol{U}}_j + \boldsymbol{B}\tilde{\boldsymbol{K}}_j)^T \tilde{\boldsymbol{U}}_{j+1}^{-1} (\boldsymbol{A}\tilde{\boldsymbol{U}}_j + \boldsymbol{B}\tilde{\boldsymbol{K}}_j) - \tilde{\boldsymbol{U}}_j^T \boldsymbol{Q} \tilde{\boldsymbol{U}}_j - \tilde{\boldsymbol{K}}_j^T \boldsymbol{R} \tilde{\boldsymbol{K}}_j \ge \boldsymbol{0} \tag{4.46}$$

or,
$$\tilde{\boldsymbol{U}}_j^T - \begin{bmatrix} (\boldsymbol{A}\tilde{\boldsymbol{U}}_j + \boldsymbol{B}\tilde{\boldsymbol{K}}_j)^T & \tilde{\boldsymbol{U}}_j^T \boldsymbol{Q}^{\frac{1}{2}} & \tilde{\boldsymbol{K}}_j^T \boldsymbol{R}^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{U}}_{j+1}^{-1} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} (\boldsymbol{A}\tilde{\boldsymbol{U}}_j + \boldsymbol{B}\tilde{\boldsymbol{K}}_j) \\ \boldsymbol{Q}^{\frac{1}{2}} \tilde{\boldsymbol{U}}_j \\ \boldsymbol{R}^{\frac{1}{2}} \tilde{\boldsymbol{K}}_j \end{bmatrix} \ge \boldsymbol{0} \tag{4.47}$$

Using Schur complements, (4.47) is put into the following LMI equivalent form:

$$\begin{bmatrix} \tilde{\boldsymbol{U}}_j^T & (\boldsymbol{A}\tilde{\boldsymbol{U}}_j + \boldsymbol{B}\tilde{\boldsymbol{K}}_j)^T & \tilde{\boldsymbol{U}}_j^T \boldsymbol{Q}^{\frac{1}{2}} & \tilde{\boldsymbol{K}}_j^T \boldsymbol{R}^{\frac{1}{2}} \\ \boldsymbol{A}\tilde{\boldsymbol{U}}_j + \boldsymbol{B}\tilde{\boldsymbol{K}}_j & \tilde{\boldsymbol{U}}_{j+1}^{-1} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{Q}^{\frac{1}{2}} \tilde{\boldsymbol{U}}_j & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{R}^{\frac{1}{2}} \tilde{\boldsymbol{K}}_j & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \ge \boldsymbol{0} \tag{4.48}$$

In order to minimize the eigenvalue given by $\lambda\left(\boldsymbol{U}_H^{P_0}\left(\boldsymbol{L}_H^{P_0}\right)^{-1}\right)$, the following eigenvalue equation is considered (see (A.7) in Appendix A):

$$\lambda \boldsymbol{I} - \boldsymbol{U}_H^{P_0}\left(\boldsymbol{L}_H^{P_0}\right)^{-1} \ge \boldsymbol{0} \tag{4.49}$$

which has the LMI equivalent of the form (through Schur complements):

$$\begin{bmatrix} \lambda \boldsymbol{L}_H^{P_0} & \boldsymbol{I} \\ \boldsymbol{I} & \tilde{\boldsymbol{U}}_0 \end{bmatrix} \ge 0 \tag{4.50}$$

As a result, the series of optimal (minimal) upper bounds $U_j^{P_0}$ for $j=0,\ldots,H-1$ is computed by optimizing the following eigenvalue problem in the context of LMI:

$$\begin{aligned} &\min_{\lambda,\tilde{U}_j,\tilde{K}_j} \lambda \\ &\text{s.t. (4.48) and (4.50)} \end{aligned} \tag{4.51}$$

In the present study, this LMI optimization is carried out using the Matlab® LMI Toolbox. The flexibility of LMI conveniently allows further incorporation of various constraints into the problem without altering the original problem structure (i.e., (4.48) and (4.50) are kept unaltered). This is further discussed in Section 4.4.1.

4.3.5 Main Stability Results

Definition 4.7 (The Principle of Optimality) (cf. Bellman, 1961): *An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*

In the context of the present study, Bellman's Principle of Optimality states that from any point on an optimal trajectory, the remaining trajectory is optimal for the corresponding problem initiated at that point. Hence, the finite horizon quadratic cost is equivalent to

$$V_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k)=V_{H-\tau_{ca}}(\boldsymbol{x}_{k+\tau_{ca}},\boldsymbol{v}_{k+\tau_{ca}})+\sum_{i=0}^{\tau_{ca}-1}W_k(\boldsymbol{x}_{k+i},\boldsymbol{u}_{k+i}) \tag{4.52}$$

where the "truncated" sequences of control inputs are $\boldsymbol{v}_{k+\tau_{ca}}=\left\{\boldsymbol{u}_{k+\tau_{ca}|k},\ldots,\boldsymbol{u}_{k+H-1|k}\right\}\in\mathbb{R}^{(H-\tau_{ca})m}$. Consequently, the τ_{ca}-step cost difference terms in (4.34) can be effectively recast so that all costs are considered from time step k, yielding

$$\begin{aligned} &V_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k)-V_H(\boldsymbol{x}_{k+\tau_{ca}},\boldsymbol{\pi}_{pre}) \\ &=\left[V_{H-\tau_{ca}}(\boldsymbol{x}_{k+\tau_{ca}},\boldsymbol{v}_{k+\tau_{ca}})-V_H(\boldsymbol{x}_{k+\tau_{ca}},\boldsymbol{\pi}_{pre})\right]+\sum_{i=0}^{\tau_{ca}-1}W_k(\boldsymbol{x}_{k+i},\boldsymbol{u}_{k+i}) \\ &=\left[V_H(\boldsymbol{x}_{k+\tau_{ca}},\boldsymbol{v}_{k+\tau_{ca}})-V_{H+\tau_{ca}}(\boldsymbol{x}_{k+\tau_{ca}},\tilde{\boldsymbol{\pi}}_k)\right]+\sum_{i=0}^{\tau_{ca}-1}W_k(\boldsymbol{x}_{k+i},\boldsymbol{u}_{k+i}) \\ &=\left[V_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k)-V_{H+\tau_{ca}}(\boldsymbol{x}_k,\tilde{\boldsymbol{\pi}}_k)\right]+\sum_{i=0}^{\tau_{ca}-1}W_k(\boldsymbol{x}_{k+i},\boldsymbol{u}_{k+i}) \end{aligned} \tag{4.53}$$

where $\tilde{\pi}_k = \{u_{k|k},...,u_{k+H-1|k},0,...,0\} \in \mathbb{R}^{(H+\tau_{ca}-1)m}$.

Theorem 4.3: *The NCS-MPC policy with horizon H utilizing the optimal control sequence π_{pre} of the previous time step up to a controller-to-actuator delay of τ_{ca}, is asymptotically stabilizing within the region of attraction $\mathcal{R}$ if there exists any j such that*

$$(1-\mu)\Lambda_H^{\tau_{ca}} - \Psi_{H,\tau_{ca}}^{j} \succ 0 \tag{4.54}$$

where $$\Lambda_H^{\tau_{ca}} = L_H^{P_0} - U_H^{P_0} + L_{\tau_{ca}}^{Q}$$

and $$\Psi_H^j = \max\left[0, \bar{\lambda}\left(U_{\tau_{ca}}^{P_0} P_0^{-1}\right) - 1\right] \kappa_{H-(j+1)}(H-(j+1)) \left[U_H^{P_0} - L_j^{Q}\right].$$

Proof:

From the stability condition (4.34) combining with (4.53), the following is obtained:

$$(1-\mu)\sum_{i=0}^{\tau_{ca}-1} W_k(x_{k+i}, u_{k+i}) - \left[V_{H+\tau_{ca}}(x_k, \tilde{\pi}_k) - V_H(x_k, \pi_k)\right] > 0 \tag{4.55}$$

Since the first left-hand-side term of (4.55) may be expressed as $\sum_{i=0}^{\tau_{ca}-1} W_k(x_{k+i}, u_{k+i}) = V_H(x_k, \pi_k) - V_{H-\tau_{ca}}(x_{k+\tau_{ca}}, \pi_{k+\tau_{ca}})$, it can be bounded from below by utilizing Lemma 4.2, to bound the costs as $V_H(x_k, \pi_k) \geq \|x_k\|_{L_H^{P_0}}^2$ and $V_{H-\tau_{ca}}(x_{k+\tau_{ca}}, \pi_{k+\tau_{ca}}) \leq \|x_k\|_{U_H^{P_0} - L_{\tau_{ca}-1}^{Q}}^2$ resulting in

$$\sum_{i=0}^{\tau_{ca}-1} W_k(x_{k+i}, u_{k+i}) > \|x_k\|_{L_H^{P_0}}^2 - \|x_k\|_{U_H^{P_0} - L_{\tau_{ca}-1}^{Q}}^2 = \|x_k\|_{L_H^{P_0} - U_H^{P_0} + L_{\tau_{ca}-1}^{Q}}^2 \tag{4.56}$$

Next, the second left-hand-side term of (4.55) has to be bounded from above. The term $\left[V_H(x_k, \pi_k) - V_{H+\tau_{ca}}(x_k, \tilde{\pi}_k)\right]$ in the right-hand-side of (4.53) is equivalent to $\left[\|x_{k+H|k}\|_{P_0}^2 - V_{\tau_{ca}}(x_{k+H|k}, \pi = 0)\right]$, which provides the means to bound it from below. Since $V_{\tau_{ca}}(x_{k+H|k}, \pi = 0) \leq V_{\tau_{ca}}(x_{k+H|k}, \pi_{k+H})$, the following bound can be enforced:

$$V_H(x_k, \pi_k) - V_{H+\tau_{ca}}(x_k, \tilde{\pi}_k) \geq \|x_{k+H|k}\|_{P_0}^2 - V_{\tau_{ca}}(x_{k+H|k}, \pi_{k+H}) \tag{4.57}$$

or, $$V_{H+\tau_{ca}}(\boldsymbol{x}_k,\tilde{\boldsymbol{\pi}}_k)-V_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k)\le V_{\tau_{ca}}(\boldsymbol{x}_{k+H|k},\boldsymbol{\pi}_{k+H})-\left\|\boldsymbol{x}_{k+H|k}\right\|_{\boldsymbol{P}_0}^2 \tag{4.58}$$

On the other hand, it can be established that $V_{\tau_{ca}}(\boldsymbol{x}_{k+H|k},\boldsymbol{\pi}_{k+H})\le\left\|\boldsymbol{x}_{k+H|k}\right\|_{\boldsymbol{U}_{\tau_{ca}}^{\boldsymbol{P}_0}}^2$ since $V_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k)\le\left\|\boldsymbol{x}_k\right\|_{\boldsymbol{U}_H^{\boldsymbol{P}_0}}^2$. Consequently, using Lemma 4.5 the upper bound for $\left[V_{H+\tau_{ca}}(\boldsymbol{x}_k,\tilde{\boldsymbol{\pi}}_k)-V_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k)\right]$ is resolved as

$$\begin{aligned}
&V_{H+\tau_{ca}}(\boldsymbol{x}_k,\tilde{\boldsymbol{\pi}}_k)-V_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k)\\
&\quad\le\left\|\boldsymbol{x}_{k+H|k}\right\|_{\boldsymbol{U}_{\tau_{ca}}^{\boldsymbol{P}_0}}^2-\left\|\boldsymbol{x}_{k+H|k}\right\|_{\boldsymbol{P}_0}^2\\
&\quad\le\left(\bar{\lambda}\left(\boldsymbol{U}_{\tau_{ca}}^{\boldsymbol{P}_0}\boldsymbol{P}_0^{-1}\right)-1\right)\left\|\boldsymbol{x}_{k+H|k}\right\|_{\boldsymbol{P}_0}^2\\
&\quad\le\max\left[0,\bar{\lambda}\left(\boldsymbol{U}_{\tau_{ca}}^{\boldsymbol{P}_0}\boldsymbol{P}_0^{-1}\right)-1\right]\kappa_{H-(j+1)}(H-(j+1))\left\|\boldsymbol{x}_k\right\|_{\boldsymbol{U}_H^{\boldsymbol{P}_0}-\boldsymbol{L}_j^{\boldsymbol{Q}}}^2\\
&\quad\le\left\|\boldsymbol{x}_k\right\|_{\boldsymbol{\psi}_{H,\tau_{ca}}^j}
\end{aligned} \tag{4.59}$$

Hence, (4.56) and (4.59) provide the bounds to establish (4.54), which guarantee asymptotic stability. □

The preceding theorem is particularly useful as a design guide for the developed NCS control strategy. The parameters involved in designing a stable controller are $\boldsymbol{Q}$, $\boldsymbol{R}$, H, μ, and $\boldsymbol{P}_0$. They will determine the worst-case delay τ_{ca} that the controller is able to handle. Other possible parameters that can influence the closed-loop stability of the control strategy developed here include various constraints for input, state and rate. As discussed in Section 4.4.1, the constraints are formulated and incorporated into the LMI (4.51), which modify the upper bounds of the finite horizon quadratic cost $\boldsymbol{U}_j^{\boldsymbol{P}_0}$ accordingly. It should be noted that the computational speed of the MPC optimization is also a factor in deciding the prediction horizon H.

4.4 Evaluation of the Stability Boundaries

This section illustrates the use of the main stability theorem, as developed in the present chapter, in the electro-hydraulic manipulator systems of the fish-processing machine (see Chapter 2). The effect of various design parameters, particularly, the desired worst-case delay,

the length of the prediction horizon, the terminal cost weighting, and the constraints, on the stability boundaries of the closed-loop system are evaluated numerically based on the linear model of the system as given in (2.17). Before proceeding further, the means of imposing constraints on the controller has to be addressed.

4.4.1 Imposing Constraints

Three common forms of constraints are state constraints, input constraints, and state minimum decay rate constraints. In terms of the stability of interest in this chapter, these constraints directly affect the tightness of the upper bounds $\boldsymbol{U}_j^{P_0}$ for $j=0,\ldots,H-1$ of the finite horizon cost function of the MPC policy. A more relaxed upper bound $\boldsymbol{U}_j^{P_0}$ and hence providing a wider stability range, is achieved by imposing constraints with a narrower range. This is true as long as the feasibility of the solution of the MPC optimization is maintained through a given state trajectory.

State constraints of the form $\|\boldsymbol{x}\|_{\boldsymbol{T}}^2 \le \eta$ are enforced by augmenting the eigenvalue minimization of (4.51) with the following LMI:

$$\begin{bmatrix} \gamma_j \boldsymbol{T} + \boldsymbol{L}_{j-1}^{\boldsymbol{Q}} & \boldsymbol{I} \\ \boldsymbol{I} & \tilde{\boldsymbol{U}}_0 \end{bmatrix} \ge \boldsymbol{0} \tag{4.60}$$

for $0 \le j \le H-1$. LMI (4.60) is obtained by first considering (4.42) which gives

$$\left\|\boldsymbol{x}_{k+j|k}\right\|_{\boldsymbol{U}_{H-j}^{P_0}}^2 \le \left\|\boldsymbol{x}_k\right\|_{\boldsymbol{U}_H^{P_0}-\boldsymbol{L}_{j-1}^{\boldsymbol{Q}}}^2 \tag{4.61}$$

The introduction of γ_j for $0 \le j \le H-1$ such that

$$\boldsymbol{U}_H^{P_0} - \boldsymbol{L}_{j-1}^{\boldsymbol{Q}} \le \gamma_j \boldsymbol{T} \tag{4.62}$$

will imply $\left\|\boldsymbol{x}_{k+j|k}\right\|_{\boldsymbol{U}_{H-j}^{P_0}}^2 \le \left\|\boldsymbol{x}_k\right\|_{\boldsymbol{U}_H^{P_0}-\boldsymbol{L}_{j-1}^{\boldsymbol{Q}}}^2 \le \gamma_j \|\boldsymbol{x}\|_{\boldsymbol{T}}^2 \le \gamma_j \eta$, and thereby guaranteeing that the intended state constraints $\boldsymbol{x}^T \boldsymbol{T} \boldsymbol{x} \le \eta$ are complied. Note that γ_j is unknown *a priori* but is a free variable in the LMI optimization of (4.51). Through Schur complements, one can easily transform (4.62) into its LMI equivalent of (4.60).

Input constraints of the form $\left|u_{i,k+j|k}\right| \le u_{\lim}$ for $i=1,...,m$ and $0 \le j \le H-1$ are included by augmenting the LMI in (4.51) with the following two LMIs:

$$\begin{bmatrix} \boldsymbol{N}_j & \tilde{\boldsymbol{K}}_j \\ \tilde{\boldsymbol{K}}_j^T & \tilde{\boldsymbol{U}}_j \end{bmatrix} \ge \boldsymbol{0} \tag{4.63}$$

$$2u_{\lim}^2 \delta_j \ge \eta \delta_j^2 N_{ii} + \gamma_j u_{\lim}^2 \tag{4.64}$$

for $0 \le j \le H-1$, where $\boldsymbol{N}_j$ are arbitrary and symmetric matrices and δ_j are arbitrary constants in the vicinity of γ_j. N_{jj} are the diagonal elements of $\boldsymbol{N}_j$. LMIs (4.63) and (4.64) are constructed by assuming the linear state feedback control law $\boldsymbol{u}_{k+j|k} = \boldsymbol{K}_j \boldsymbol{x}_{k+j|k}$ and taking the square of the input constraint, arriving at:

$$\left|u_{i,k+j|k}\right|^2 = \left|\left(\boldsymbol{K}_j \boldsymbol{x}_{k+j|k}\right)_i\right|^2 \le \left|\left(\tilde{\boldsymbol{K}}_j \tilde{\boldsymbol{U}}_j^{-1} \boldsymbol{x}\right)_i\right|^2 \le \left\|\left(\tilde{\boldsymbol{K}}_j \tilde{\boldsymbol{U}}_j^{-\frac{1}{2}}\right)_i\right\| \left\|\left(\tilde{\boldsymbol{U}}_j^{-\frac{1}{2}} \boldsymbol{x}\right)_i\right\| \le \left\|\left(\tilde{\boldsymbol{K}}_j \tilde{\boldsymbol{U}}_j^{-\frac{1}{2}}\right)_i\right\| \gamma_j \eta \le u_{\lim}^2 \tag{4.65}$$

The diagonal elements of the rightmost inequality in (4.65) is equivalent to $\left(\tilde{\boldsymbol{K}}_j \tilde{\boldsymbol{U}}_j^{-1} \tilde{\boldsymbol{K}}_j^T\right)_{ii} \gamma_j \eta \le u_{\lim}^2$ which can be transformed into an equivalent LMI form by introducing a new intermediate matrix $\boldsymbol{N}_j$ such that

$$\tilde{\boldsymbol{K}}_j \tilde{\boldsymbol{U}}_j^{-1} \tilde{\boldsymbol{K}}_j^T \le \boldsymbol{N}_j \le \frac{1}{\gamma_j \eta} \bar{\boldsymbol{u}}_{\lim} \tag{4.66}$$

where $\bar{\boldsymbol{u}}_{\lim} = \mathrm{diag}(u_{\lim}^2, \ldots, u_{\lim}^2) \in \mathbb{R}^{m \times m}$. The LMI form of the left-hand side inequality of (4.66) gives (4.63). The diagonal elements of the right-hand side inequality of (4.66) gives $N_{jj} \le \frac{u_{\lim}^2}{\gamma_j \eta}$ which is a non-convex constraint and can be approximated by linearizing $\frac{1}{\gamma_j}$ as $\frac{2}{\delta_j} - \frac{\gamma_j}{\delta_j^2} \le \frac{1}{\gamma_j}$ with $\delta_j \cong \gamma_j$. As a result, the inequality becomes $N_{jj} \le \left(\frac{2}{\delta_j} - \frac{\gamma_j}{\delta_j^2}\right) \frac{u_{\lim}^2}{\eta}$, and rearranging it gives (4.64).

The state minimum decay rate constraint used here is of the form $\left\|\boldsymbol{x}_{k+j+1|k}\right\|_{U_{H-(j+1)}^{P_0}}^2 \le \rho^2 \left\|\boldsymbol{x}_{k+j|k}\right\|_{U_{H-j}^{P_0}}^2$ for $0 \le j \le H-1$ where $\rho \in (0,1)$ is the additional tuning

parameter for specifying the speed of the closed-loop response (Kothare, *et al.*, 1996). In order to transform this constraint into its LMI equivalent, again, assume the linear state feedback law $\boldsymbol{u}_{k+j|k} = \boldsymbol{K}_j \boldsymbol{x}_{k+j|k}$ to yield

$$\left\| \boldsymbol{x}_{k+j|k} \right\|^2_{(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K}_j)^T \boldsymbol{U}^{P_0}_{H-(j+1)} (\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K}_j)} \leq \rho^2 \left\| \boldsymbol{x}_{k+j|k} \right\|^2_{\boldsymbol{U}^{P_0}_{H-j}} \tag{4.67}$$

By comparing the norm weightings of (4.67) and substituting $\tilde{\boldsymbol{U}}_j = \left(\boldsymbol{U}^{P_0}_{H-j}\right)^{-1}$ and $\tilde{\boldsymbol{K}}_j = \boldsymbol{K}_j \tilde{\boldsymbol{U}}_j$ followed by pre-multiplying and post-multiplying by $\tilde{\boldsymbol{U}}^T_j$ and $\tilde{\boldsymbol{U}}_j$, one gets

$$(\boldsymbol{A}\tilde{\boldsymbol{U}}_j + \boldsymbol{B}\tilde{\boldsymbol{K}}_j)^T \tilde{\boldsymbol{U}}_{j+1} (\boldsymbol{A}\tilde{\boldsymbol{U}}_j + \boldsymbol{B}\tilde{\boldsymbol{K}}_j) \leq \rho \tilde{\boldsymbol{U}}_j \tag{4.68}$$

which has the following LMI equivalent form:

$$\begin{bmatrix} \rho^2 \tilde{\boldsymbol{U}}_j & \left(\boldsymbol{A}\tilde{\boldsymbol{U}}_j + \boldsymbol{B}\tilde{\boldsymbol{K}}_j\right)^T \\ \boldsymbol{A}\tilde{\boldsymbol{U}}_j + \boldsymbol{B}\tilde{\boldsymbol{K}}_j & \tilde{\boldsymbol{U}}_{j+1} \end{bmatrix} \geq \boldsymbol{0} \tag{4.69}$$

for $j = 0, \ldots, H-1$, and can be incorporated into the LMI eigenvalue optimization of (4.51) for the purpose of obtaining the upper bounds $\boldsymbol{U}^{P_0}_j$ for $j = 0, \ldots, H-1$ to evaluate asymptotic stability using Theorem 4.3.

4.4.2 Stability Surfaces of the NCS

The stability boundaries of the NCS-MPC control strategy as developed in this thesis and implemented on the electro-hydraulic manipulator system, is investigated in this section. The weighting matrices are chosen as $\boldsymbol{Q} = \text{diag}(1, 10, 0.001, 0.001)$ and $\boldsymbol{R} = 100$. Here, μ is set at 0.90 in (4.54) while the stability boundaries are considered with various values of terminal weights $\boldsymbol{P}_0$ as fractions of the open-loop infinite weight $\boldsymbol{U}_\infty$ which is obtained by solving the discrete Lyapunov equation $\boldsymbol{U}_\infty = \boldsymbol{Q} + \boldsymbol{A}^T \boldsymbol{U}_\infty \boldsymbol{A}$. Although according to Theorem 4.3 stability is guaranteed for "any" $j \geq 1$ that renders (4.54) to be positive definite, a stability *metric* is introduced to determine the relative degree of stability of a particular setting. This relative degree of stability is expressed as the ratio of the number j to the length of the prediction horizon minus one.

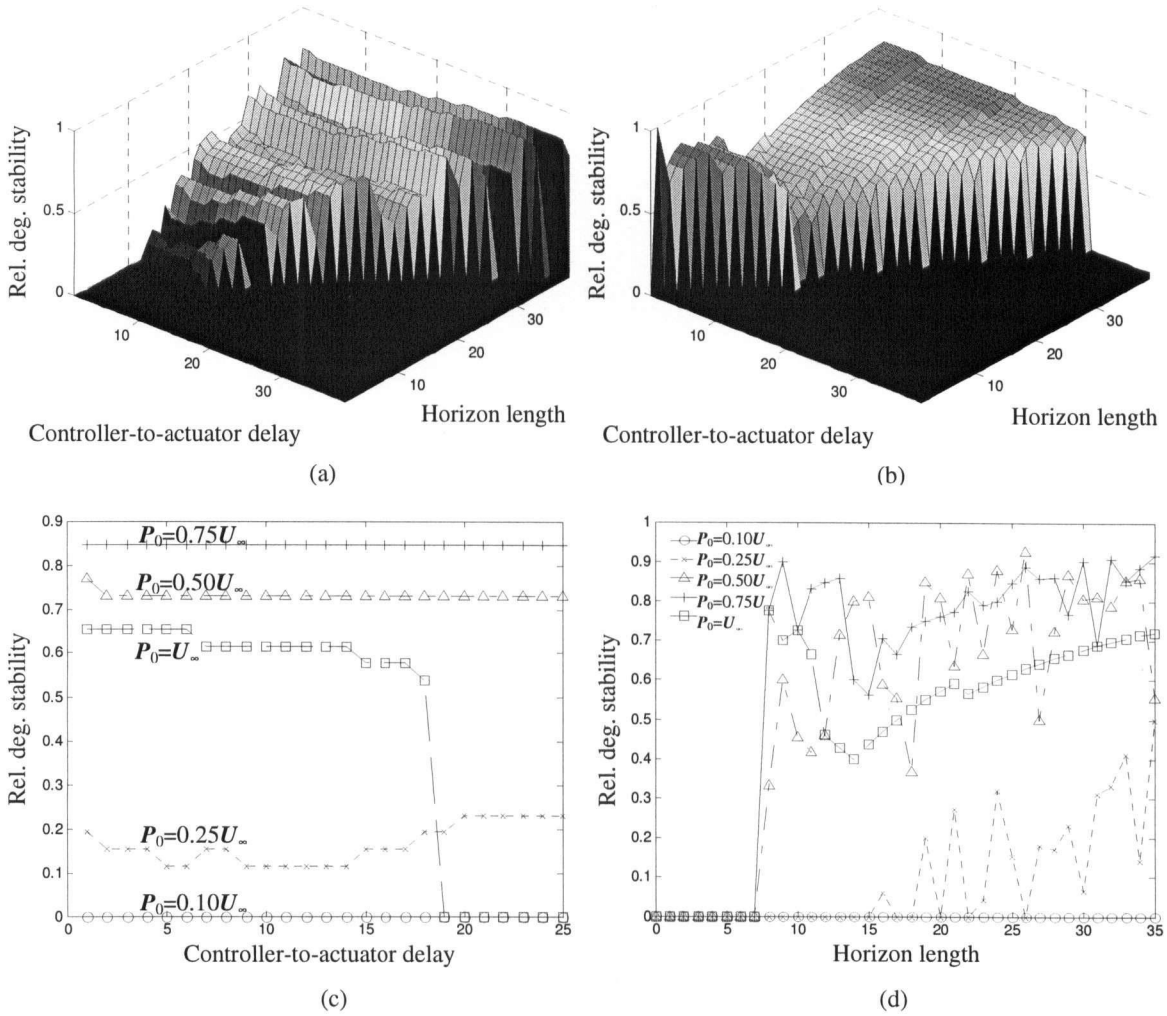


Figure 4.3: Stability boundaries with $u_{\lim} = 0.20$. (a) Stability surface for $\boldsymbol{P}_0 = 0.25\boldsymbol{U}_\infty$; (b) Stability surface for $\boldsymbol{P}_0 = \boldsymbol{U}_\infty$; (c) Comparison of relative stability over different terminal weights for a horizon length of 25; and (d) Comparison of relative stability over different terminal weights at $\tau_{ca} = 8$.

Figure 4.3(a) shows the stability surface at $\boldsymbol{P}_0 = 0.25\boldsymbol{U}_\infty$ with the input constraint $u_{\lim} = 0.20$ (scaled), and the state constraint terms set to $\boldsymbol{T} = \text{diag}(1,1,1,1)$ and $\eta = 1$. The system is unstable with a short prediction horizon. When the length of the prediction horizon is increased beyond 11, the closed-loop stability is maintained. Stability can be achieved for a shorter prediction horizon with a higher terminal weight of $\boldsymbol{P}_0 = \boldsymbol{U}_\infty$, as shown in Figure 4.3(b). However, for a particular stabilizing prediction horizon, the worst-case controller-to-actuator delay that can be tolerated is lower. Although intuitively, the stability increases as $\boldsymbol{P}_0 \to \boldsymbol{U}_\infty$, it is not the case in the presence of delay when the future control inputs in the buffered sequence

are used, as seen here. This is because, as the terminal weight $\boldsymbol{P}_0$ increases beyond a certain value, the terminal cost tends to dominate the optimization (minimization) and leads to a solution that is more suboptimal. So, in order to maintain stability of the closed-loop NCS, $\boldsymbol{P}_0$ has to be an intermittent value relative to $\boldsymbol{U}_\infty$. This effect can be reduced by introducing an additional constraint on the minimum decay rate of the states, in order to modify the speed of response of the closed-loop system. Figure 4.3(c) shows the relative stability of the closed-loop system under various levels of terminal weights at a fixed prediction horizon of 25. Again, note the lower degree of stability when $\boldsymbol{P}_0 = \boldsymbol{U}_\infty$ compared to $\boldsymbol{P}_0 = 0.50\boldsymbol{U}_\infty$ and $\boldsymbol{P}_0 = 0.75\boldsymbol{U}_\infty$. Stability can be maintained only up to a controller-to-actuator delay of 18. This phenomenon can be overcome by introducing a state minimum decay rate constraint into the control optimization. The effect of terminal weights on the closed-loop stability at a required worst-case delay of 8 is shown in Figure 4.3(d). Obviously, in order to maintain stability, a longer prediction horizon is required for lower terminal weights. The fluctuations in the plots are due to numerical errors in the computation, particularly, in the optimization of the LMIs. It should not be of any concern because Theorem 4.3 only requires $j \geq 1$ for closed-loop stability.

Figure 4.4 shows the stability boundaries with $\rho = 0.99$ and all the remaining parameters set as before. Figure 4.4(a) shows the stability surface at $\boldsymbol{P}_0 = 0.25\boldsymbol{U}_\infty$ which is the same value as used in Figure 4.3(a). Comparing figure 4.4(a) with 4.3(a) it can be seen that, the relative degree of stability has decreased as a result of introducing the state minimum decay rate constraint. However, a significant improvement is achieved when $\boldsymbol{P}_0 = \boldsymbol{U}_\infty$, as can be observed in figures 4.4(b) and 4.4(c). The drop in the level of worst-case delay due to longer prediction horizon is eliminated, and the overall stability has increased.

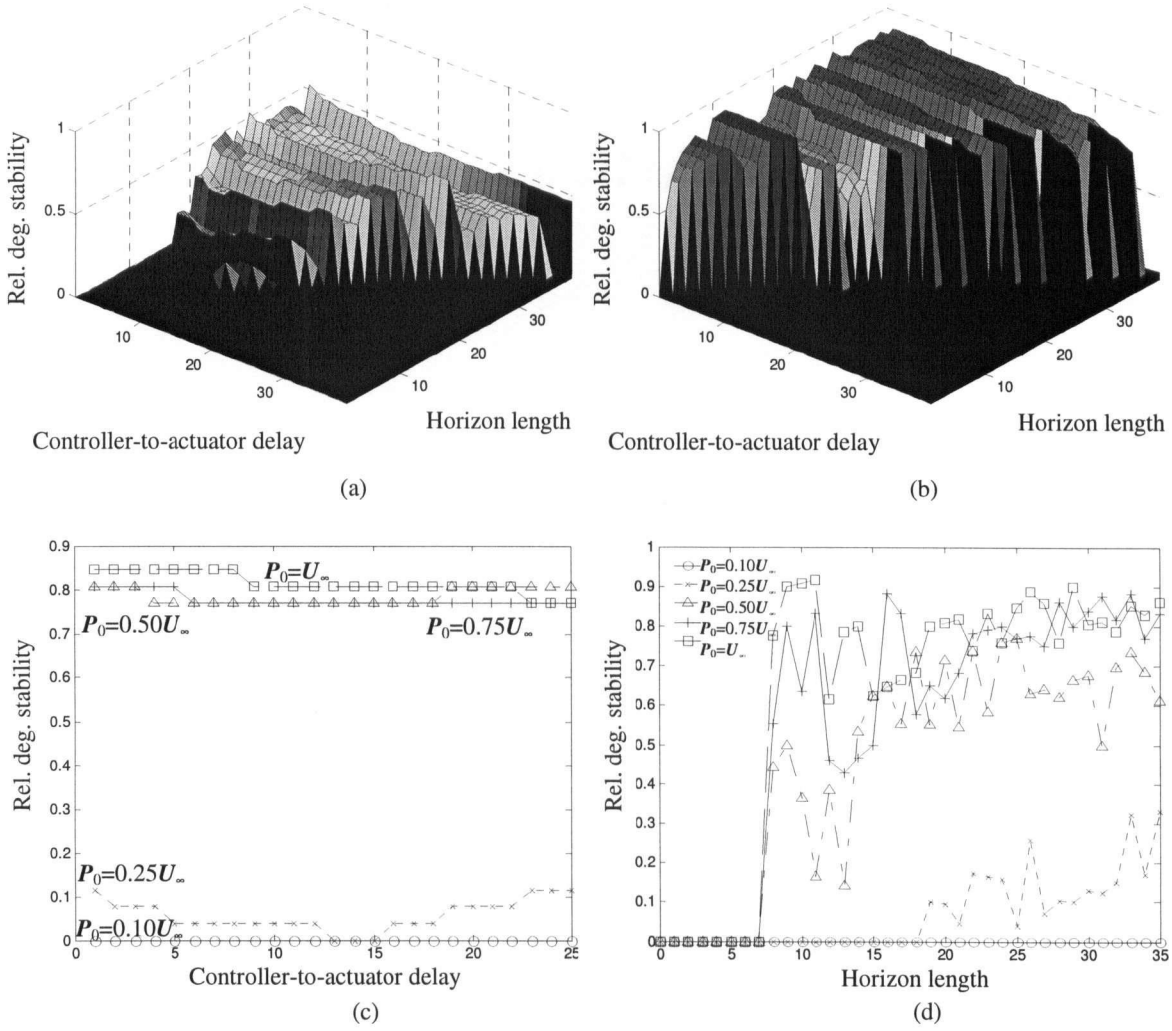


Figure 4.4: Stability boundaries with $u_{\lim} = 0.20$ and $\rho = 0.99$. (a) Stability surface for $\boldsymbol{P}_0 = 0.25\boldsymbol{U}_\infty$; (b) Stability surface for $\boldsymbol{P}_0 = \boldsymbol{U}_\infty$; (c) Comparison of relative stability over different terminal weights for a horizon length of 25; and (d) Comparison of relative stability over different terminal weights at $\tau_{ca} = 8$.

4.5 Implementation Issues

In order to evaluate the responses of the controlled system using a pre-computed control input signal, an intermediate node for data packet forwarding is incorporated into the transmission path between the controller and the actuator. The function of this intermediate node is to regulate the desired data transmission characteristics. It uses a first-in-first-out (FIFO) queuing policy to consistently simulate various levels of delay. The delay level is varied by changing the length of the queue. Hence, when the intermediate node receives a data packet

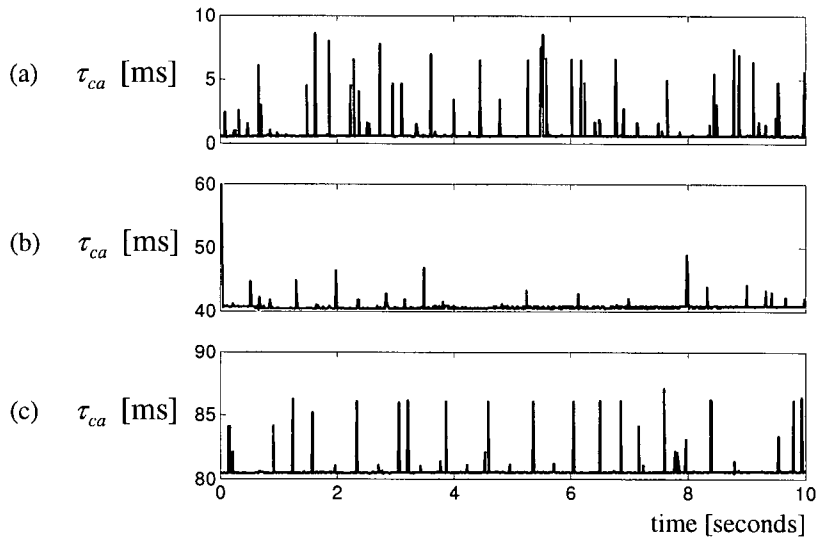


Figure 4.5: Controller-to-actuator transmission delay τ_{ca} using the intermediate FIFO packet forwarding queue. (a) 1-step delay; (b) 5-step delay; and (c) 9-step delay.

from the controller, it stacks the data packet on its queue, and then the oldest data packet is forwarded to the actuator. Figures 4.5(a), 4.5(b), and 4.5(c) show the dynamics of the controller-to-actuator data transmission of the specified 1-step, 5-step, and 9-step delay, respectively. Note that the sampling interval used here is 10 ms. The loss of data packets is simulated by dropping off entries of the queue according to a memoryless exponential distribution typically used in network queuing models (Bertsekas and Gallager, 1991). At most, only one selected data packet is dropped at each sampling interval. The transmission of sensory data (states) from the sensor to the controller is done through a direct connection with no loss and negligible delay.

The constrained MPC control scheme requires the solution of the optimal quadratic cost function (4.4) or (4.9) at every sampling interval. Since the present real-time servo system with a sample interval of 10 ms is relatively fast, solving (4.4) or (4.5) using traditional quadratic programming (QP) methods is inefficient, and feasible solutions throughout the evolution of the state trajectory are not guaranteed. Computational time is further increased, in an exponential manner, due to the introduction of additional constraints and the length of the prediction horizon. In addition, feasibility of the solution cannot be guaranteed *a priori*. Here, an offline multi-parametric Quadratic Programming (mpQP) method is adopted in order to achieve fast online optimization of the quadratic cost function of MPC. This method explores

the state-space of a given system, and partitions the space into polyhedral regions of continuous and piecewise linear control laws of the constrained MPC problem. After that, an online binary search tree is built in order to efficiently determine which region, and hence which control law, a particular state belongs. The online binary search tree is built by systematically and sequentially sorting each region according to the direction of each hyperplane (the borders of the regions) on which it resides within the state-space. In typical applications of constrained MPC, which uses the first optimal control input sequence at every sampling interval, there exist multiple regions with identical control laws. These regions can be grouped together thereby decreasing the number of unique regions, and consequently reducing the computational effort while traversing the search tree. However, this is not the case in the present application because subsequent control input signals are utilized as well, rendering each initial region unique.

It is found that since the search tree building algorithm requires a significant amount of time, a remedy for speeding up the algorithm would be to reduce the depth of the search tree by increasing the number of regions left at leaf nodes. This significantly reduces the time for search tree building. After traversing the binary search until a particular leaf node is reached, a sequential search is carried out online until the terminal region is reached, where the corresponding control law is calculated. Although in the offline stage, the algorithms for mpQP state-space partitioning and search tree building are rather tedious, complex, and consume a significant amount of computational time and memory, the speed advantage gained during online execution is essential, especially in the applications of high-speed servo systems of industrial machinery. The basic concept of mpQP and the detailed design work for the algorithms to generate off-line and traverse on-line the corresponding binary search trees are presented in Appendix B.

4.6 Real-time Experimental Evaluation

In the experiment outlined now, the length of the prediction horizon is set to 15 with a terminal weight of $\boldsymbol{P}_0 = 0.80\boldsymbol{U}_\infty$. An input constraint is imposed as $-0.20 \le u \le 0.20$. An additional constraint is imposed on the velocity state as $-0.10 \le \dot{y} \le 0.10$ in order to reduce the speed of response of the closed-loop system, thereby, increasing the stability of the system under control. This results in a total of 3890 polyhedral regions within the normalized unity state-space consisting of 11604 hyperplanes. When the number of remaining regions to be

explored at the leaf nodes is set to 100 regions, the resulting binary search tree contains 441 nodes with a tree depth of 8 layers. The partitioning of regions (using Matlab® Multi-parametric Toolbox) and the generation of the binary search tree (coded and compiled in C++ language) take approximately 180 minutes. The state minimum decay rate constraint is not imposed because it requires the introduction of additional system states in the form of the square of each of the four original system states. This result in a tremendous increase in the computational effort required to generate the mpQP regions as well as the corresponding binary search tree. With a prediction horizon of 15 steps, the size of each data packet from the controller to the actuator is 270 bytes.

4.6.1 Stability Evaluation under Transmission Delay

The state response curves and the corresponding control input signal for a positioning operation starting from an initial piston position of 5 mm are shown in Figure 4.6. Over the experiment duration of 10 s, six levels of controller-to-actuator delay expressed in terms of pre-specified sample steps, are included for comparison. The graphed input current is the actual sequenced value used at a particular sampling instant. Under delay free conditions, as shown in Figure 4.6(a), the settling time is approximately 1 s with all four controlled states converging to the origin of equilibrium. Figures 4.6(b) and 4.6(c) show that similar responses are obtained with minor increases in the settling time of the piston position state y, under 1 and 3 sample steps of controller-to-actuator delay. In Figure 4.6(d) where the controller-to-actuator delay is set to 6 sample steps, y converges to the origin very slowly, settling at approximately 7.8 s, and a low level of continuous ringing follows. Constant cyclic responses are also observed in the remaining three states. At a controller-to-actuator delay of 10 sample steps (Figure 4.6(e)), y does not converge asymptotically and the pressure states exhibit saw-tooth cyclic responses. Severe jumps are observed for the corresponding input current. Since the selected system is open-loop stable, only asymptotic convergence to the origin of one or more states can be used to represent stability. Hence, due to the non-convergence of the piston position state, the responses as obtained in figures 4.6(e) and 4.6(f) are categorized as asymptotically unstable. Under 14 sample steps of controller-to-actuator delay, as shown in Figure 4.6(f), the piston position state diverges slowly with time up to the 4 s mark, as P_h increases and P_r decreases, which is opposite of the experience in the previous test run. After the 4s mark, P_h oscillates

above its steady-state value and P_r oscillates below its steady-state value.

Figure 4.7 shows the overall effect of the controller-to-actuator delay using our MPC scheme with future control input buffering on the integrated absolute state regulation error. The curve marked with "×" indicates that the piston position state ceases to converge asymptotically after 10 sample steps of delay. The integrated absolute state regulation error for the two pressure states (plots indicated by a "square" and a "triangle") as well as the average error (the curve indicated by a "+" sign) increases with the controller-to-actuator delay. A drastic increase, which indicates divergence, is observed at 14 sample steps of delay.

The ability of the developed networked control scheme to maintain asymptotic stability subjected to input disturbances, is tested and shown in Figure 4.8. The electro-hydraulic manipulator is set at an initial position of 5 mm. At every 6 s interval starting from the 8 s mark, an input pulse of 10 mA magnitude and 0.1 s duration is applied to the system in alternating directions. Figures 4.8(a) and 4.8(b) show the resulting state response curves under 3 sample steps and 5 sample steps of controller-to-actuator delay, respectively. In both cases, asymptotic convergence is maintained under input disturbances, although the system responses become more sluggish as the controller-to-actuator delay increases.

Figure 4.9 shows that asymptotic stability is maintained with various initial conditions for the piston position under 5 sample steps of controller-to-actuator delay. With an initial position of 5 mm, the system settles after 5 s while it settles at 1.5 s with an initial position of 2.5 mm. The piston settles back at the origin after 2 s under an initial position of –2 mm, and an initial position of –4 mm requires a duration of approximately 7 s for the system to settle. It is noted from the input current plots that a greater control effort is required to move the manipulator in the positive direction than in the negative direction.

4.6.2 Stability Evaluation under Packet Loss

Now the asymptotic state convergence with the developed networked controller is experimentally analyzed in the presence of packet losses in the communication path. The packet sequence of a pre-computed future control input is dropped at the intermediate queue, as discussed in Section 4.5. Figure 4.10 shows the state response curves and the input signal of the system at a fixed 3 sample steps of controller-to-actuator delay and various levels of data packet loss rate at 10% increments. The vertical lines below each input current plot indicate the

instances where data packets are lost. Generally, the settling time of the piston position state increases as the data packet loss rate increases. It is also observed that the fluctuation in the input current increases as the data packet loss rate increases. However, asymptotic convergence is maintained until a data loss rate of 90% is reached. This seems somewhat unreal but the "effective" mean delay that corresponds to the 90% data loss rate is only approximately 10.5 sample steps, as shown in Figure 4.11. Furthermore, Figure 4.11 shows the mean effective delay of corresponding data packet loss rate for a pre-set queuing delay of 1 to 6 sample steps. The numbers in parentheses on top of each bar represent the maximum delay experienced at the actuator. As intuitively clear, the higher the data packet loss rate, the further will the mean effective delay deviate from the pre-set queuing delay.

The combined effect of data packet loss and controller-to-actuator transmission delay on the asymptotic convergence, and hence stability of the system, is shown in Figure 4.12. Here, only the integrated absolute regulating error of the piston position state is used as an indicator of the state convergence. It is observed that stability is maintained for all data loss rates up to a controller-to-actuator delay of 4 sample steps. At 5 sample steps of delay, asymptotic convergence is guaranteed only up to a data loss rate of 60%. This limit on the data loss rate for asymptotic convergence reduces to 50% at 6 sample steps of delay. This verifies that as long as the prediction horizon is sufficiently long and with an appropriate terminal weight $\boldsymbol{P}_0$, asymptotic stability is guaranteed using a pre-computed control input sequence as long as the "effective" delay is within the pre-designed sample step value, regardless of the dynamics of packet loss, vacant sampling, and out-of-order data arrival.

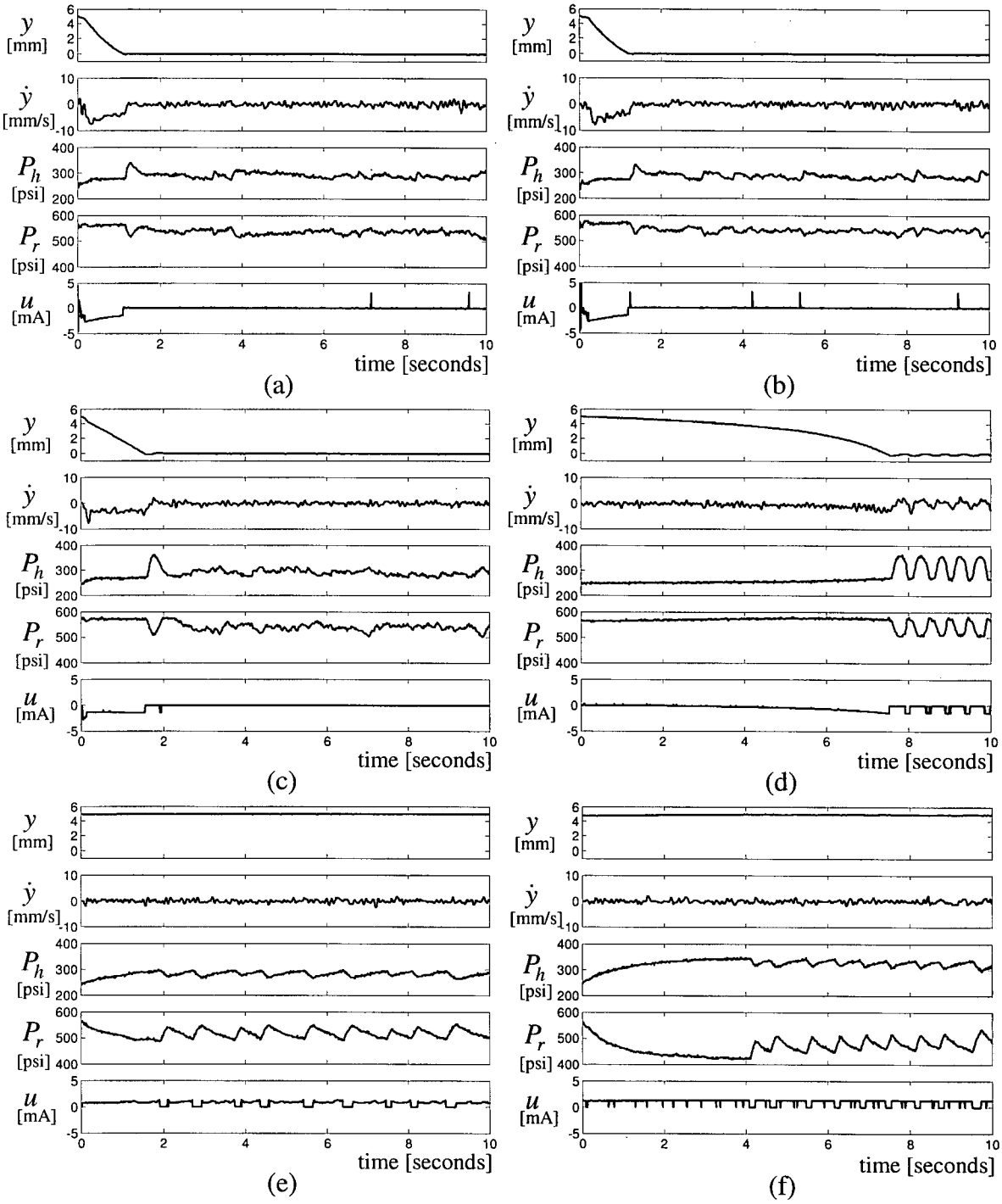


Figure 4.6: State responses and input under 6 levels of controller-to-actuator delay. (From top to bottom: piston position y [mm], piston velocity $\dot{y}$ [mm/s], pressure at the head-side of the cylinder P_h [psi], pressure at the rod-side of the cylinder P_r [psi], and input current u [mA]). (a) Delay free; (b) 1-step delay; (c) 3-step delay; (d) 6-step delay; (e) 10-step delay; and (f) 14-step delay.

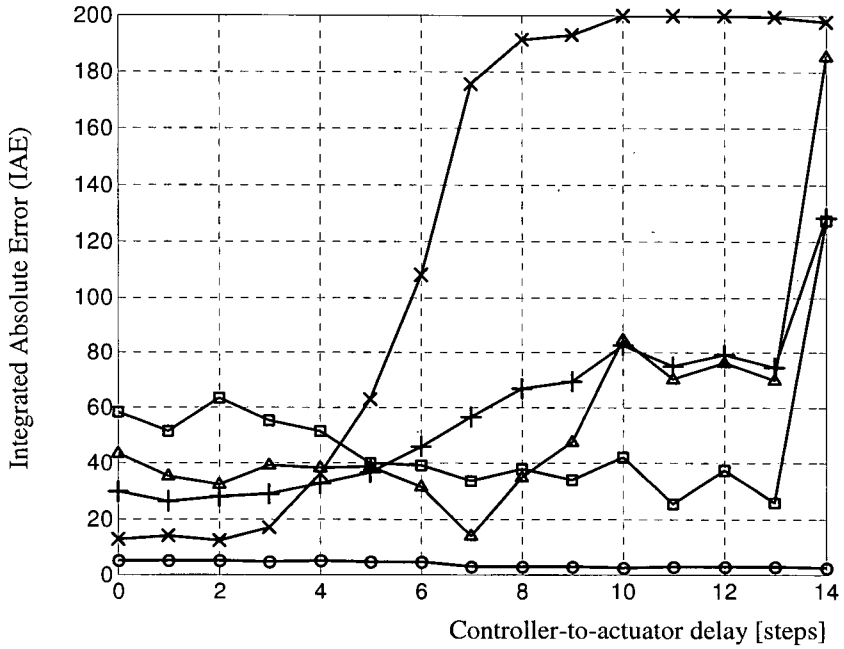


Figure 4.7: Effect of controller-to-actuator delay on the integrated absolute state regulation error. (cross – position; circle – velocity; square – head pressure; triangle – rod pressure; and plus – average)

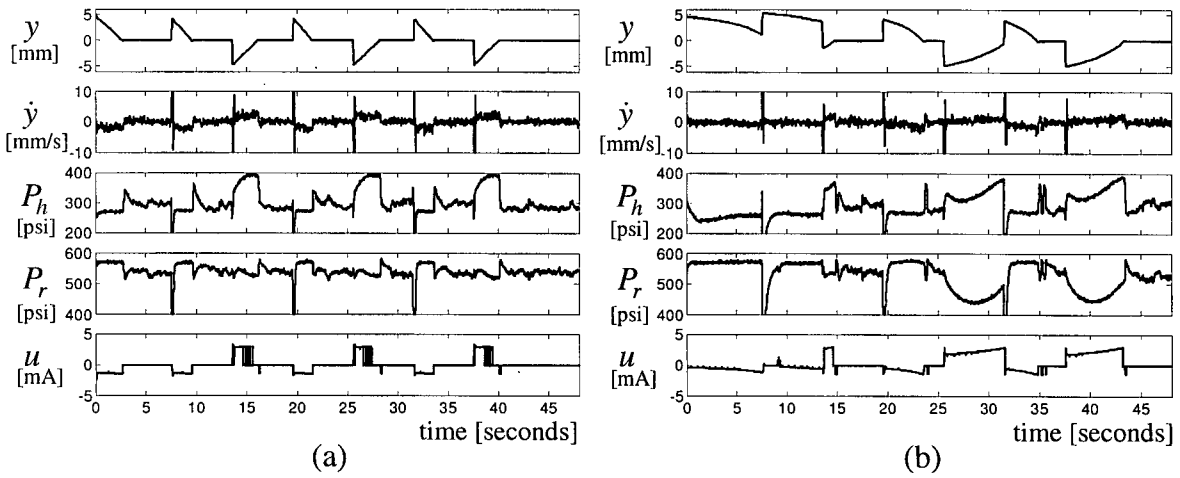


Figure 4.8: State response curves and input under input disturbances. (From top to bottom: piston position y [mm], piston velocity $\dot{y}$ [mm/s], pressure at the head-side of the cylinder P_h [psi], pressure at the rod-side of the cylinder P_r [psi], and input current u [mA]). (a) 3 steps of controller-to-actuator delay; and (b) 5 steps of controller-to-actuator delay.

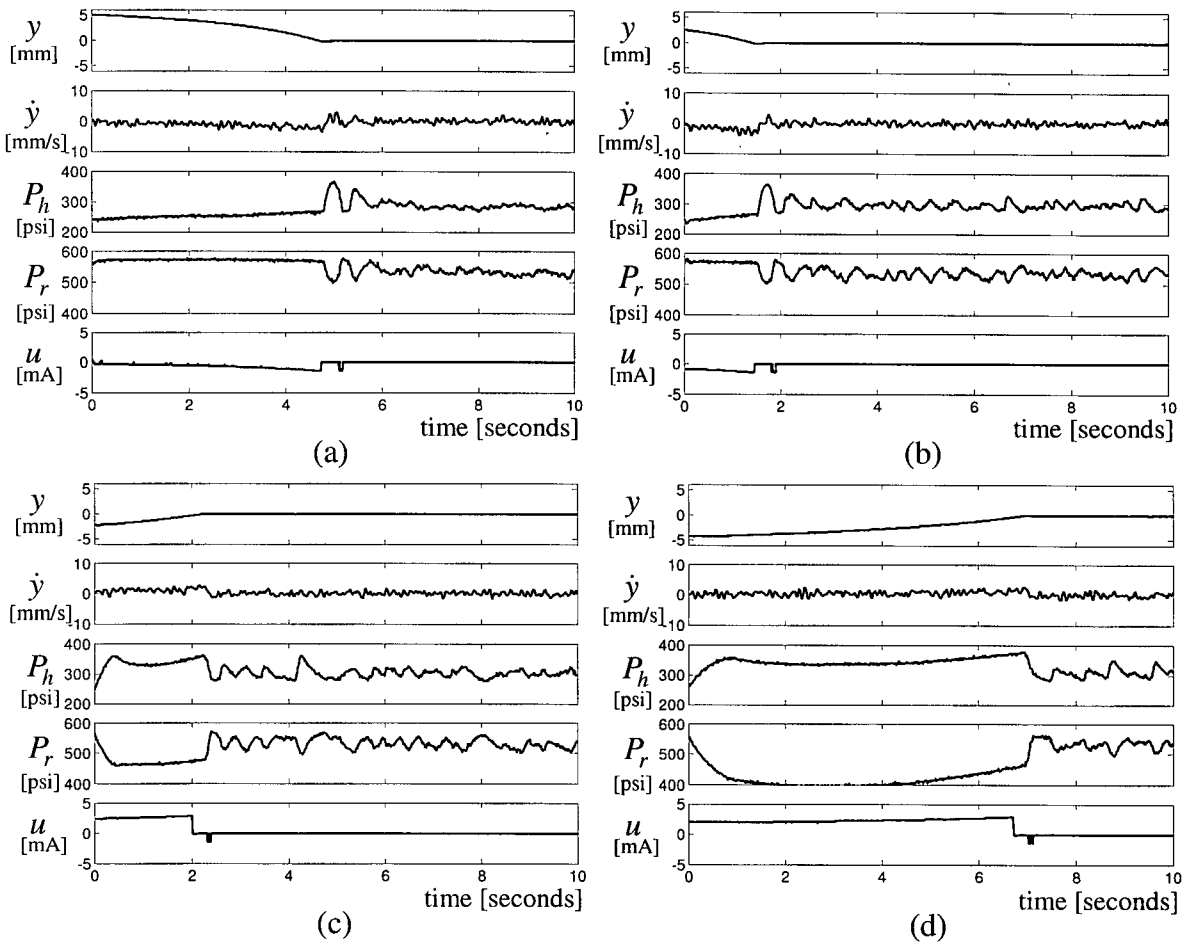


Figure 4.9: Asymptotic stability under 5 steps of controller-to-actuator delay and different initial conditions. (From top to bottom: piston position y [mm], piston velocity $\dot{y}$ [mm/s], pressure at the head-side of the cylinder P_h [psi], pressure at the rod-side of the cylinder P_r [psi], and input current u [mA]). (a) Initial piston position = 5 mm; (b) Initial piston position = 2.5 mm; (c) Initial piston position = –2 mm; and (d) Initial piston position = –4 mm.

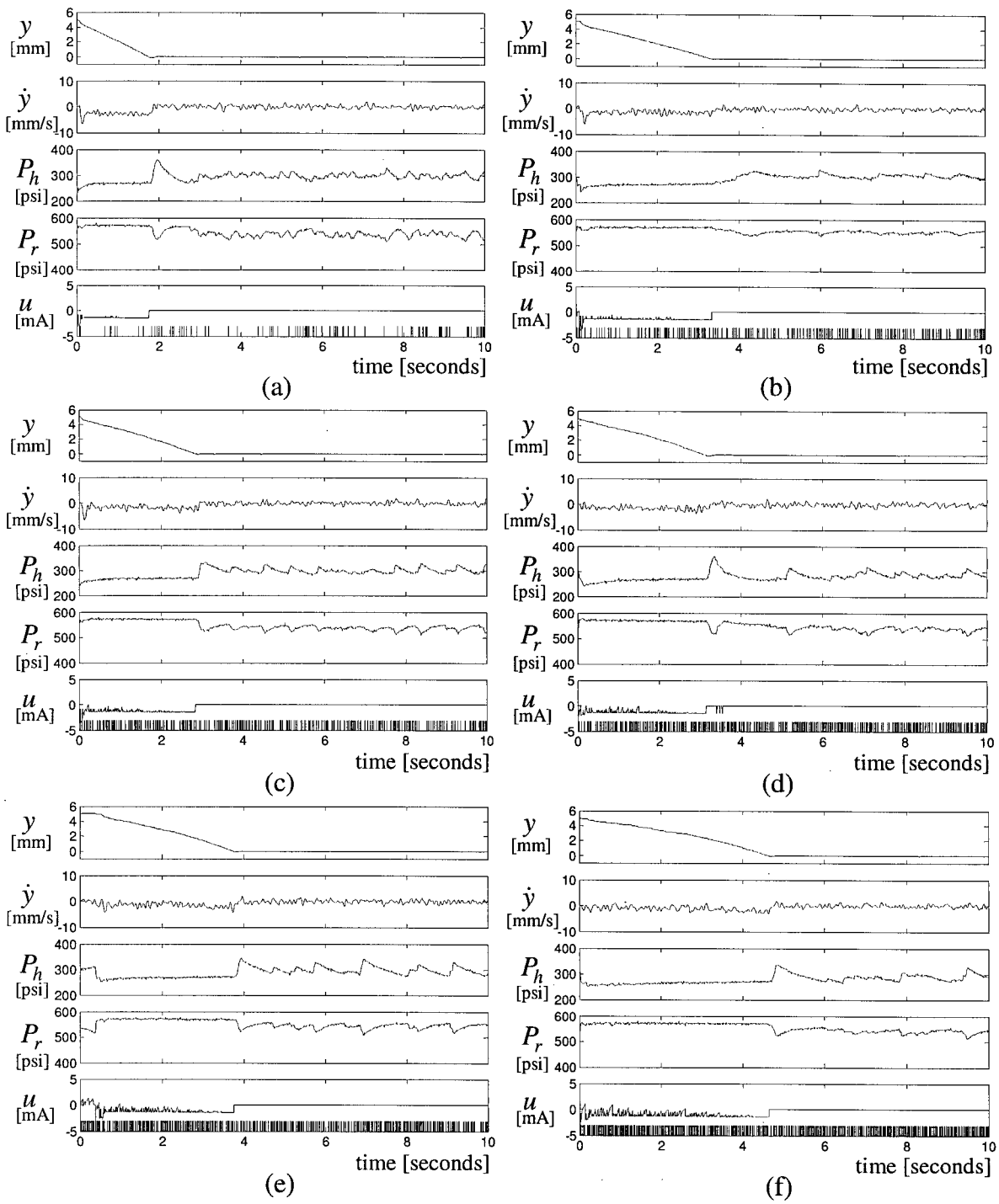


Figure 4.10: State response curves and input at 3 steps of controller-to-actuator delay under 9 levels of data packet loss rate. (From top to bottom: piston position y [mm], piston velocity $\dot{y}$ [mm/s], pressure at the head-side of the cylinder P_h [psi], pressure at the rod-side of the cylinder P_r [psi], and input current u [mA]). (a) 10% loss rate; (b) 20% loss rate; (c) 30% loss rate; (d) 40% loss rate; (e) 50% loss rate; (f) 60% loss rate (g) 70% loss rate; (h) 80% loss rate; and (i) 90% loss rate.

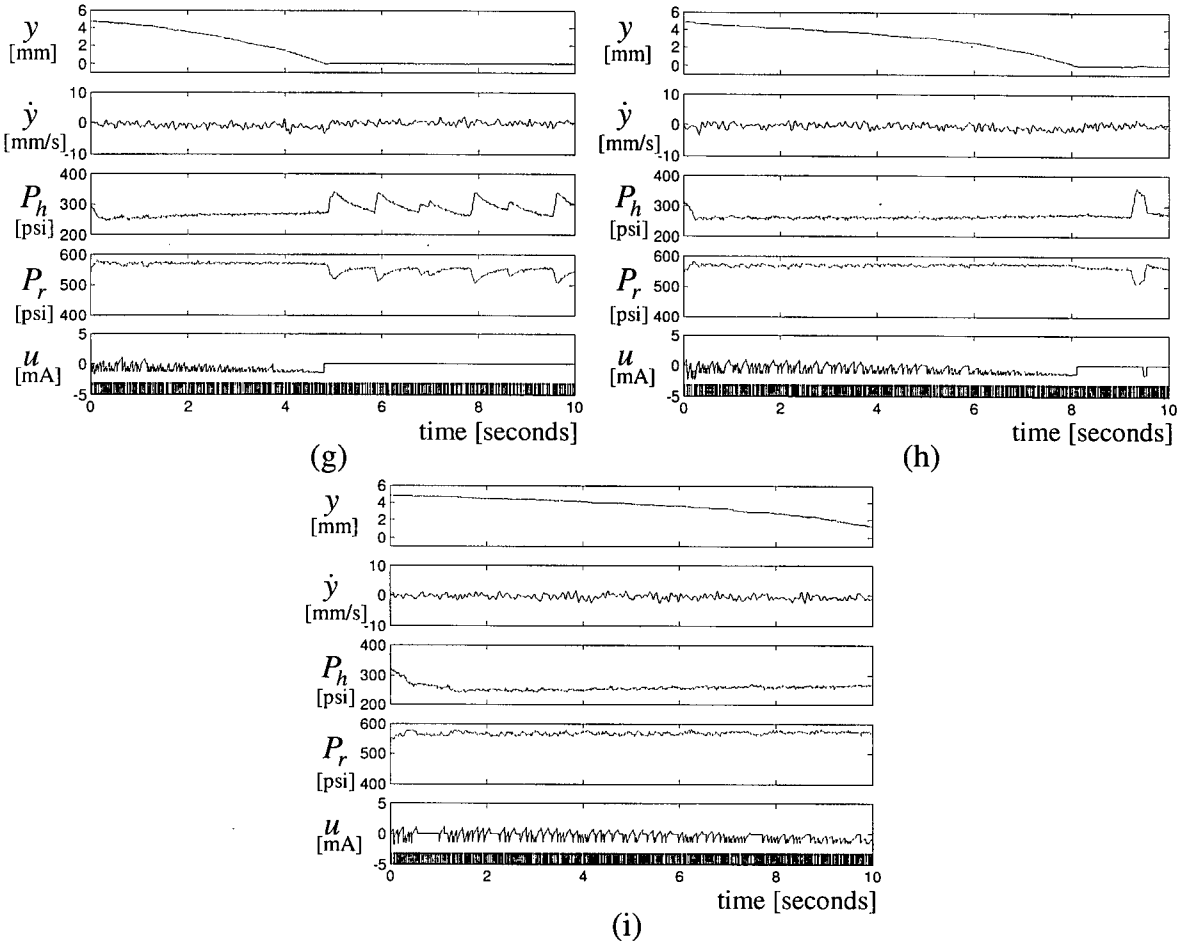


Figure 4.10 cont.: State response curves and input at 3 steps of controller-to-actuator delay under 9 levels of data packet loss rate. (From top to bottom: piston position y [mm], piston velocity $\dot{y}$ [mm/s], pressure at the head-side of the cylinder P_h [psi], pressure at the rod-side of the cylinder P_r [psi], and input current u [mA]). (a) 10% loss rate; (b) 20% loss rate; (c) 30% loss rate; (d) 40% loss rate; (e) 50% loss rate; (f) 60% loss rate (g) 70% loss rate; (h) 80% loss rate; and (i) 90% loss rate.

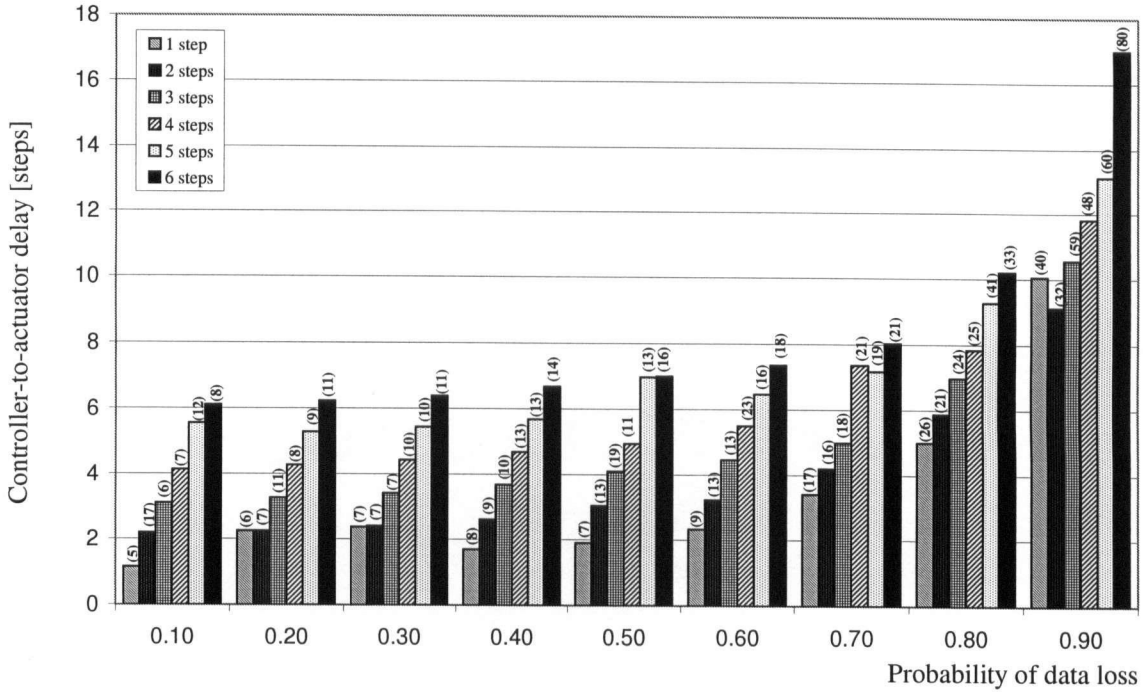


Figure 4.11: Effective mean delay of data packet loss. (Each set corresponds to the pre-set queuing delay. The numbers on top of each bar in parenthesis indicate the maximum instantaneous delay recorded during experiment).

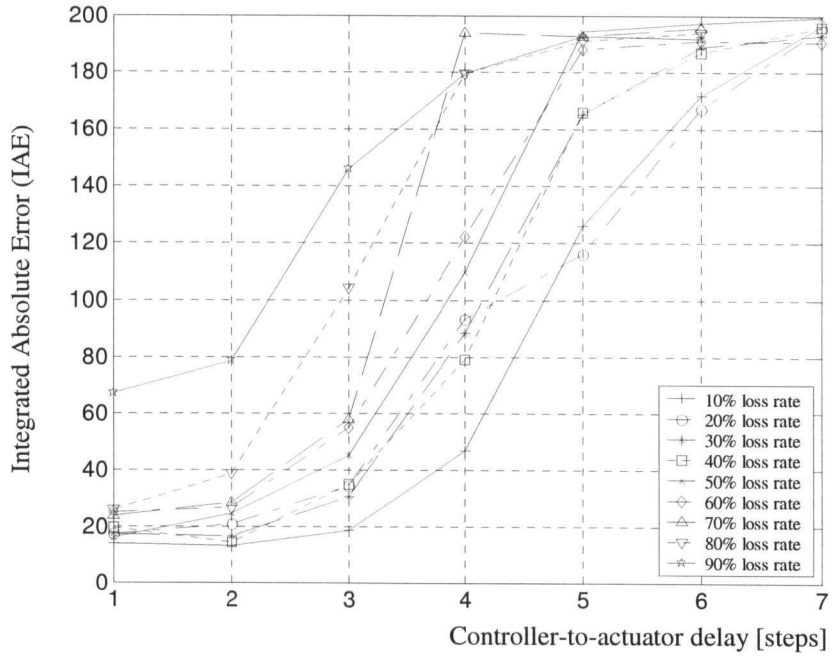


Figure 4.12: Effect of data packet loss rate and controller-to-actuator delay on the asymptotic convergence of the position state of the system.

4.7 Summary

This chapter studied the stability of a newly-developed and optimal control strategy for Networked Control Systems (NCS). The control strategy hones the potential of constrained Model Predictive Control (MPC) by buffering the predicted control sequence at the actuator in anticipation of typical data transmission errors associated with NCS. Utilizing the second method of Lyapunov, the controller was shown to be feasible and asymptotically stable. A stability theorem was developed, which provided a suboptimal measure for the controller in real-time, and was adequate to estimate the worst-case transmission delay that can be handled by the developed control buffering strategy. Although, under non-nominal conditions, the pre-computed control actions are sub-optimal, they are still feasible, and therefore stabilizing. The stability conditions, as governed by the theorem, were validated through real-time implementation on an electro-hydraulic servo system of an industrial processing machine, through an Ethernet network. The developed theorem was provided as a design criterion to set the parameters of the controller. Besides the transmission delay, the developed strategy has the capability of compensating for data losses, vacant sampling, and out-of-order data, without modification.

Chapter 5
Stability Under Imperfect State Measurements

Pre-computing and buffering of future optimal control input sequences are fundamental to the developed networked feedback control strategy, which is based on Model Predictive Control (MPC). The conditions for maintaining closed-loop asymptotic stability of this control system have been established in the sense of Lyapunov, in the previous chapter. By relying on the *Principle of Separation* (Furuta, *et al.*, 1988), perfect state measurement under conditions of zero sensor-to-controller delay has been assumed in the previous analysis. The present chapter extends the stability analysis to include imperfect state measurements from a state observer (estimator), which is incorporated before the MPC controller to estimate delayed or missing sensor data (see Figure 4.1). The level of state estimation error (difference between estimated and actual states of the system) that can be sustained under the conditions set for stable future control input buffering without losing global closed-loop stability, is investigated. The analysis given here is an extended version of the work by Santos and Biegler (1999), which analyzed robust stability of an MPC controller with model mismatch by imposing terminal state constraints. In accordance with the analysis of the previous chapter, a terminal weight is incorporated into the MPC cost function without any end constraints, in order to determine the degree of state estimation error that can be sustained, with guaranteed closed-loop asymptotic stability in the sense of the second method of Lyapunov. The established stability bound for the state estimation errors will provide a key reference to the design of robust state observers for NCS.

The next section introduces the preliminary concepts of the stability problem that is under study and establishes an upper-bound gain for the state estimation error. Section 5.2 describes the basis of Lyapunov stability in the context of the analysis presented in Chapter 4 and identifies the conditions to guarantee closed-loop asymptotic stability in the presence of state mismatch. The sensitivity of the finite-horizon quadratic optimization of the MPC policy, to state estimation errors, is investigated in Section 5.3. Section 5.4 presents the development

which leads to the main stability results of the present analysis, and synthesizes the overall algorithm for determining the upper bound of the state estimation error. The developed stability theorems are then used to determine the state error bound of the NCS-MPC control strategy as implemented in Section 5.5 on the electro-hydraulic servo manipulator. Section 5.6 describes the possible improvements to the MPC policy in order to increase the degree of state estimation error while maintaining global asymptotic stability.

5.1 Problem Description

Consider the discrete-time, state-space model of the system under control, as given by

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{5.1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^n$ are the states of the system at time k, $\boldsymbol{u}_k \in \mathbb{R}^m$ are the control inputs at time k, and $f : \mathbb{R}^{n\times m} \rightarrow \mathbb{R}^n$. The finite receding-horizon cost function of the MPC control policy which is employed in both previous and present chapters is of the following quadratic form (see Section 4.2.1, (4.9)):

$$\begin{aligned} V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k) = & \min_{\boldsymbol{\pi}(\cdot|k)} \left\{ \left\| \boldsymbol{x}_{k+H|k} \right\|_{\boldsymbol{P}_0}^2 + \sum_{i=0}^{H-1} \left[\left\| \boldsymbol{x}_{k+i|k} \right\|_{\boldsymbol{Q}}^2 + \left\| \boldsymbol{u}_{k+i|k} \right\|_{\boldsymbol{R}}^2 \right] \right\} \\ & \text{s.t.} \\ & \boldsymbol{x}_{k+i|k} \in \mathcal{X}_i \quad \forall i = 1,\ldots,H \\ & \boldsymbol{u}_{k+i|k} \in \mathcal{U} \quad \forall i = 0,\ldots,H-1 \end{aligned} \tag{5.2}$$

in which H is the length of the prediction horizon, $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ is the state weighting matrix, $\boldsymbol{R} \in \mathbb{R}^{m\times m}$ is the input weighting matrix, and $\boldsymbol{P}_0 \in \mathbb{R}^{n\times n}$ is the terminal weight matrix of the state. The series of optimal control input is given by $\boldsymbol{\pi}_k^* = \left\{ \boldsymbol{u}_{k|k}^*, \ldots, \boldsymbol{u}_{k+H-1|k}^* \right\} = \arg\min_{\boldsymbol{u}} V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k)$ with the measured instantaneous state $\boldsymbol{x}_k$ as the starting state of the quadratic programming problem. It has been established in the previous chapter that buffering the sequences of $\boldsymbol{\pi}_k^*$ for use at future time steps to compensate for the transmission errors between the controller and the actuator is stabilizing when the instantaneous state $\boldsymbol{x}_k$ is measured perfectly and directly from the sensor without network transmission between the sensor and the controller. In the case where the sensor readings are transmitted over a communication network and when the actual

state is delayed or lost, $\boldsymbol{x}_k$ has to be estimated at every time step k using a state observer. Let the estimated state be denoted by $\hat{\boldsymbol{x}}_k$. Depending on the speed of convergence and accuracy of the adopted state observer, the deviation of $\hat{\boldsymbol{x}}_k$ from the actual $\boldsymbol{x}_k$ may render the solution to (5.2) infeasible or suboptimal, which may eventually lead to instability of the developed NCS-MPC control strategy. The main objective here is to determine an upper bound for the magnitude of deviation between the estimated state $\hat{\boldsymbol{x}}_k$ and the actual measured state $\boldsymbol{x}_k$ for maintaining closed-loop asymptotic stability.

The following analysis assumes that the state observer employs the model (5.1) of the system in its estimation, and the evolution of this model between the estimated state and the actual state is Lipschitz (see Definition 4.5), such that

$$\left\| f(\boldsymbol{x}_k, \boldsymbol{u}_k) - f(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k) \right\| \leq \hat{L}_f \left\| \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k \right\| \tag{5.3}$$

where $\hat{L}_f$ is the bounding Lipschitz constant. Also, assume that the difference between the evolution of the measured actual state and the state trajectory obtained from the model (5.1) of the system, is bounded by a class $\mathcal{K}_\infty$-function (see Section 4.3.1) $\gamma_x(\cdot)$ such that

$$\left\| \boldsymbol{x}_{k+1} - f(\boldsymbol{x}_k, \boldsymbol{u}_k) \right\| \leq \gamma_x(\left\| \boldsymbol{x}_k \right\|) \tag{5.4}$$

for $k > 0$. The difference in the future evolution of the actual and the estimated states can be expressed as follows:

$$\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1} = \left(\boldsymbol{x}_{k+1} - f(\boldsymbol{x}_k, \boldsymbol{u}_k)\right) + \left(f(\boldsymbol{x}_k, \boldsymbol{u}_k) - f(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k)\right) \tag{5.5}$$

Taking the norm of (5.5) and substituting (5.3) and (5.4), the norm of the future evolution of the state estimation error can be bounded as:

$$\begin{aligned} &\left\| \boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1} \right\| \\ &\quad \leq \left\| \boldsymbol{x}_{k+1} - f(\boldsymbol{x}_k, \boldsymbol{u}_k) \right\| + \left\| f(\boldsymbol{x}_k, \boldsymbol{u}_k) - f(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k) \right\| \\ &\quad \leq \gamma_x(\left\| \boldsymbol{x}_k \right\|) + \hat{L}_f \left\| \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k \right\| \\ &\quad \leq \gamma_e(\left\| \boldsymbol{x}_k \right\|) \end{aligned} \tag{5.6}$$

for $k > 0$ in which $\gamma_e(\cdot)$ is a class $\mathcal{K}_\infty$-function. Define $\gamma_e(\cdot) = K_e \left\| \boldsymbol{x}_k \right\|$ where $K_e > 0$, yielding

$$\left\| \boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1} \right\| \leq K_e \left\| \boldsymbol{x}_k \right\| \tag{5.7}$$

It follows that (5.7) governs the future evolution of the state estimation error through an upper bounding gain K_e and can be used as a measure of error tolerance for maintaining closed-loop asymptotic stability. Note here that a larger K_e is more desirable.

5.2 Stability Basis

Stability analysis using Lyapunov's second method is based on the existence of a class $\mathcal{K}_\infty$-function $V(\boldsymbol{x}_k)$ that satisfies the condition $V(\boldsymbol{x}_k)-V(\boldsymbol{x}_{k+1})\geq \upsilon(\|\boldsymbol{x}_k\|)$ as $k\to\infty$, where $\upsilon(\|\boldsymbol{x}_k\|)$ is another class $\mathcal{K}_\infty$-function (Theorem 4.1). In other words, in order to ensure asymptotic stability, $V(\boldsymbol{x}_k)$ should be decreasing with time along the trajectories of the closed-loop system. In the context of receding horizon MPC, the finite quadratic cost function (5.2) is a candidate Lyapunov function (Theorem 4.3). It has been established in the second condition of Theorem 4.2 that, with perfect state measurement, the system represented by the model (5.1), when subjected to a finite sequence of future control actions $\boldsymbol{\pi}_k=\{\boldsymbol{u}_{k|k},...,\boldsymbol{u}_{k+H|k}\}$ as determined from the MPC control law (5.2), is asymptotically stable if $V_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k)-V_H(\boldsymbol{x}_{k+1},\boldsymbol{\pi}_{k+1})>0$. With the integration of the state observer, the actual measured state $\boldsymbol{x}_k$ is not directly available to the MPC controller, thereby making $V_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k)$ an unknown for $k>0$. Instead, the time evolution of the estimated cost $V_H(\hat{\boldsymbol{x}}_k,\hat{\boldsymbol{\pi}}_k)$ (i.e., the receding horizon cost computed with the estimated state $\hat{\boldsymbol{x}}_k$ as its initial optimization condition) has to be considered, along the closed-loop system trajectories. The following lemma establishes the bound for the main stability criteria.

Lemma 5.1: *The system under control described by model* (5.1) *using a finite sequence of future control actions from the MPC control policy* (5.2) *with prediction horizon* H *and estimated state* $\hat{\boldsymbol{x}}_k$ *is asymptotically stable in the sense of Lyapunov if there exists a neighborhood* $\mathbb{X}\subseteq\mathbb{R}^n$ *containing an open neighborhood of the origin such that*

$$V_H(\hat{\boldsymbol{x}}_k,\hat{\boldsymbol{\pi}}_k)-V_H(\hat{\boldsymbol{x}}_{k+1},\hat{\boldsymbol{\pi}}_{k+1})\geq\|\boldsymbol{x}_k\|_Q^2-\left[V_H(\hat{\boldsymbol{x}}_{k+1},\hat{\boldsymbol{\pi}}_{k+1})-V_H^*(\boldsymbol{x}_{k+1},\boldsymbol{\pi}_{k+1})\right]>0 \qquad (5.8)$$

Proof:

Adding and subtracting $V_H^*(\boldsymbol{x}_k,\boldsymbol{\pi}_k)$ to the difference in the estimated costs at the time steps k

and $(k+1)$, gives

$$\begin{aligned} &V_H(\hat{\boldsymbol{x}}_k,\hat{\boldsymbol{\pi}}_k)-V_H(\hat{\boldsymbol{x}}_{k+1},\hat{\boldsymbol{\pi}}_{k+1}) \\ &\quad=\left[V_H(\hat{\boldsymbol{x}}_k,\hat{\boldsymbol{\pi}}_k)-V_H^*(\boldsymbol{x}_{k+1},\boldsymbol{\pi}_{k+1})\right]-\left[V_H(\hat{\boldsymbol{x}}_{k+1},\hat{\boldsymbol{\pi}}_{k+1})-V_H^*(\boldsymbol{x}_{k+1},\boldsymbol{\pi}_{k+1})\right] \end{aligned} \tag{5.9}$$

This estimated cost difference can be bounded from below by bounding the first and the second terms on the right-hand side of (5.9) from below and above, respectively. Consider the first term of (5.9) and $V_H^*(\boldsymbol{x}_k,\boldsymbol{\pi}_k)-V_H^*(\boldsymbol{x}_{k+1},\boldsymbol{\pi}_{k+1})$ which can be bounded from below as $V_H^*(\boldsymbol{x}_k,\boldsymbol{\pi}_k)-V_H^*(\boldsymbol{x}_{k+1},\boldsymbol{\pi}_{k+1})\geq\|\boldsymbol{x}_k\|_{\boldsymbol{Q}}^2$. Since, optimality dictates $V_H^*(\boldsymbol{x}_k,\boldsymbol{\pi}_k)\leq V_H(\hat{\boldsymbol{x}}_k,\hat{\boldsymbol{\pi}}_k)$ for $\hat{\boldsymbol{x}}_k\neq\boldsymbol{0}$, and assuming that $\hat{\boldsymbol{x}}_k\approx\boldsymbol{x}_k$, the first term of (5.9) can also be bounded from below as $V_H(\hat{\boldsymbol{x}}_k,\hat{\boldsymbol{\pi}}_k)-V_H^*(\boldsymbol{x}_{k+1},\boldsymbol{\pi}_{k+1})\geq\|\boldsymbol{x}_k\|_{\boldsymbol{Q}}^2$, resulting in (5.8). □

Before proceeding to establish an upper bound on the second term of (5.8), the sensitivity of the MPC optimization problem (5.2) at time step $(k+1)$, to a deviation $(\boldsymbol{x}_{k+1}-\hat{\boldsymbol{x}}_{k+1})$ in the initial state should be characterized. This characterization is required to facilitate the incorporation of the state estimation error bounding gain K_e into (5.8) through (5.7). This is accomplished in the next section.

5.3 Sensitivity of State Estimation Errors

In order to characterize the sensitivity of the MPC optimization of (5.2), it is necessary to reformulate (5.2) into an equivalent multi-parametric form whereby both inputs and state trajectories for $\forall i=0,...,H$ become decision variables. Here, only equality constraints are considered. Possible extension to include inequality constraints is addressed in Section 5.6. Define $\boldsymbol{\chi}_{k+i}=\left[\boldsymbol{x}_{k+i}^T \quad \boldsymbol{u}_{k+i}^T\right]^T\in\mathbb{R}^{m+n}$ for $i=0,...,H$ and $\boldsymbol{z}_k=\left[\boldsymbol{\chi}_k^T \quad \boldsymbol{\chi}_{k+1}^T \quad \cdots \quad \boldsymbol{\chi}_{k+H}^T\right]^T\in\mathbb{R}^{(H+1)(m+n)}$ with $\boldsymbol{u}_{k+H}=\boldsymbol{0}$. Consider the linearized version of the discrete, time-invariant model (5.1), as given by

$$\boldsymbol{x}_{k+1}=\boldsymbol{A}\boldsymbol{x}_k+\boldsymbol{B}\boldsymbol{u}_k \tag{5.10}$$

where $\boldsymbol{A}\in\mathbb{R}^{n\times n}$ and $\boldsymbol{B}\in\mathbb{R}^{n\times m}$. The finite horizon quadratic cost function of (5.2) can be transformed into the following equivalent MPC optimization problem:

$$V_H(\boldsymbol{x}_k, \boldsymbol{\pi}_k) = V_H(z_k) = \min_{z_k} \tfrac{1}{2} z_k^T \boldsymbol{J}_e z_k$$
$$\text{s.t. } \boldsymbol{c}(z_k) = [\boldsymbol{A_I} - \boldsymbol{A_B}] z_k - \boldsymbol{v}_x = \boldsymbol{0} \tag{5.11}$$

where

$$\boldsymbol{J}_e = \text{diag}(2\boldsymbol{Q}, 2\boldsymbol{R}, 2\boldsymbol{Q}, 2\boldsymbol{R}, \ldots, 2\boldsymbol{Q}, 2\boldsymbol{R}, 2\boldsymbol{P}_0, \boldsymbol{0}) \in \mathbb{R}^{(H+1)(m+n)\times(H+1)(m+n)}$$

$$\boldsymbol{A_I} = \text{diag}(\boldsymbol{I}, \boldsymbol{0}, \ldots, \boldsymbol{I}, \boldsymbol{0}) \in \mathbb{R}^{(H+1)(m+n)\times(H+1)(m+n)}$$

$$\boldsymbol{A_B} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{(H+1)(m+n)\times(H+1)(m+n)}$$

$$\boldsymbol{v}_x = \begin{bmatrix} \boldsymbol{x}_k^T & \boldsymbol{0}^T & \cdots & \boldsymbol{0}^T \end{bmatrix}^T \in \mathbb{R}^{(H+1)(m+n)}$$

The first order necessary and sufficient condition for a local minimum of a general convex constrained optimization problem (including (5.2) and (5.11)) is given in the following theorem.

Theorem 5.1 (Karush-Kuhn-Tucker optimality conditions): *Let $\mathcal{X}$ be a vector space and $\mathcal{Z}$ a normal space. Let $f(\cdot)$ be a Gateaux differentiable real-valued functional on $\mathcal{X}$, and $\mathcal{C}$ a Gateaux differential mapping from $\mathcal{X}$ into $\mathcal{Z}$. Assume that the Gateaux differentials are linear in their increments. If $\boldsymbol{x}^*$ minimizes $f(\cdot)$ subject to a constraint vector function $\boldsymbol{c}(\boldsymbol{x}) \geq \boldsymbol{0}$, then there exist Lagrange multipliers $\boldsymbol{\lambda}^*$ such that $\boldsymbol{x}^*$ and $\boldsymbol{\lambda}^*$ satisfy the following system:*

$$\begin{aligned}
&\nabla_x \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) = \boldsymbol{0} \\
&c_i(\boldsymbol{x}^*) = 0 \text{ for } i \in \mathcal{E} \\
&c_i(\boldsymbol{x}^*) \geq 0 \text{ for } i \in \mathcal{I} \\
&\lambda_i^* \geq 0 \text{ for } i \in \mathcal{I} \\
&\lambda_i^* c_i(\boldsymbol{x}^*) = 0 \text{ for } \forall i
\end{aligned} \tag{5.12}$$

where the Lagrangian $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_i \lambda_i c_i(\boldsymbol{x})$, $c_i(\boldsymbol{x})$ *is the* i*-th function of* $\boldsymbol{c}(\boldsymbol{x})$, λ_i *is the* i*-th element of vector* $\boldsymbol{\lambda}$, $\mathcal{E}$ *is the set of equality constraints, and* $\mathcal{I}$ *is the set of inequality constraints.*

Proof: see Luenberger (1969), pp. 249-250. □

The variations of the optimality conditions (5.12) to a deviation in $\boldsymbol{x}$ can be approximated using the mean value theorem given below.

Theorem 5.2 (Mean Value Theorem): *Suppose that* a *and* b *are numbers and that the continuous- time function* $f(\cdot)$ *has continuous first partial derivatives at* $[a,b]$. *Then*

$$f(a) - f(b) = \int_0^1 \nabla f(b + \tau(a-b))^T (a-b)\, \mathrm{d}\tau \tag{5.13}$$

Proof: see Kreyszig (1999), pg. 445. □

Let $\hat{\boldsymbol{\chi}}_{k+i} = \begin{bmatrix} \hat{\boldsymbol{x}}_{k+i}^T & \hat{\boldsymbol{u}}_{k+i}^T \end{bmatrix}^T \in \mathbb{R}^{m+n}$ for $i = 0, \ldots, H$ and $\hat{z}_k = \begin{bmatrix} \hat{\boldsymbol{\chi}}_k^T & \hat{\boldsymbol{\chi}}_{k+1}^T & \cdots & \hat{\boldsymbol{\chi}}_{k+H}^T \end{bmatrix}^T \in \mathbb{R}^{(H+1)(m+n)}$ be the multi-parametric vectors corresponding to the estimated initial state $\hat{\boldsymbol{x}}_k$. Utilizing theorems 5.1 and 5.2, the following lemma, which characterizes the sensitivity of the MPC policy (5.11) to the state estimation error at $(k+1)$ in the form of a bound, can be deduced.

Lemma 5.2 (cf. Santos and Biegler, 1999): *There exists a bound, governed by the deviation on the initial state conditions, on the solution to the convex optimization problem of* (5.11) *such that*

$$\left\| \hat{z}_{k+1} - z_{k+1} \right\| \leq Z_x Z_z \left\| \boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1} \right\| \tag{5.14}$$

where
$$Z_x \geq \left\| \int_0^1 \left[\begin{matrix} \nabla_{x_{k+1}} \left(\boldsymbol{\eta}^T \nabla_{z_{k+1}} V_H(z_{k+1}) \right) \\ \nabla_{x_{k+1}} \boldsymbol{c}(z_{k+1}) \end{matrix} \right]_{\xi_\tau} \mathrm{d}\tau \right\|$$

and
$$Z_z \geq \left\| \left(\int_0^1 \left[\begin{matrix} \nabla_{z_{k+1}} \left(\boldsymbol{\eta}^T \nabla_{z_{k+1}} V_H(z_{k+1}) \right) \\ \nabla_{z_{k+1}} \boldsymbol{c}(z_{k+1}) \end{matrix} \right]_{\xi_\tau} \mathrm{d}\tau \right)^{-1} \right\|$$

in which $\xi_\tau = \begin{bmatrix} \hat{x}_{k+1}^T & \hat{z}_{k+1}^T \end{bmatrix}^T + \tau \left(\begin{bmatrix} x_{k+1}^T & z_{k+1}^T \end{bmatrix}^T - \begin{bmatrix} \hat{x}_{k+1}^T & \hat{z}_{k+1}^T \end{bmatrix}^T \right)$ *and* $\boldsymbol{\eta} \in \mathbb{R}^{(H+1)(m+n)}$ *is the basis for the null space of the gradient of the constraint matrix function* $\nabla_{z_{k+1}} \boldsymbol{c}(z_{k+1})$.

Proof:

Consider the first two Karush-Kuhn-Tucker (KKT) optimality conditions of (5.12) in Theorem 5.1 for the equality constrained MPC optimization of (5.11) at $(k+1)$, combined in a vector form as

$$\begin{bmatrix} \nabla_{z_{k+1}} \left(V_H(z_{k+1}) - \boldsymbol{\lambda} \boldsymbol{c}(z_{k+1}) \right) \\ \boldsymbol{c}(z_{k+1}) \end{bmatrix} = \boldsymbol{0} \tag{5.15}$$

Assuming that $\nabla_{z_{k+1}} \boldsymbol{c}(z_{k+1}) \in \mathbb{R}^{(H+1)(m+n) \times (H+1)(m+n)}$ has a full row rank, the basis $\boldsymbol{\eta} \in \mathbb{R}^{(H+1)(m+n)}$ for the null space of $\nabla_{z_{k+1}} \boldsymbol{c}(z_{k+1})$ can be defined such that $\nabla_{z_{k+1}} \boldsymbol{c}(z_{k+1}) \boldsymbol{\eta} = \boldsymbol{0}$. By projecting the gradient of the Lagrangian on the null space of $\nabla_{z_{k+1}} \boldsymbol{c}(z_{k+1})$, (5.15) is obtained as

$$\begin{bmatrix} \boldsymbol{\eta}^T \nabla_{z_{k+1}} V_H(z_{k+1}) \\ \boldsymbol{c}(z_{k+1}) \end{bmatrix} = \boldsymbol{0} \tag{5.16}$$

Let the left-hand-side of (5.16) be equal to the vector function $\boldsymbol{p}(z_{k+1}) = \begin{bmatrix} \boldsymbol{\eta}^T \nabla_{z_{k+1}} V_H(z_{k+1}) \\ \boldsymbol{c}(z_{k+1}) \end{bmatrix}$.

Define $\xi = \begin{bmatrix} x_{k+1}^T & z_{k+1}^T \end{bmatrix}^T$ and $\hat{\xi} = \begin{bmatrix} \hat{x}_{k+1}^T & \hat{z}_{k+1}^T \end{bmatrix}^T$. Using Theorem 5.2, the affine approximation of $\boldsymbol{p}(\cdot)$ for variations of $\hat{\xi}$ is realized as

$$p(\xi)-p(\hat{\xi})=\int_0^1 \nabla p(\hat{\xi}+\tau(\xi-\hat{\xi}))^T(\xi-\hat{\xi})\,\mathrm{d}\tau=\boldsymbol{0} \tag{5.17}$$

Let $\xi_\tau=\hat{\xi}+\tau(\xi-\hat{\xi})$. Partial differentiation of $p(\xi_\tau)$ in (5.17) with respect to x_{k+1} and z_{k+1}, respectively, yields

$$\int_0^1 \begin{bmatrix} \nabla_{x_{k+1}}\left(\eta^T\nabla_{z_{k+1}}V_H(z_{k+1})\right) & \nabla_{z_{k+1}}\left(\eta^T\nabla_{z_{k+1}}V_H(z_{k+1})\right) \\ \nabla_{x_{k+1}}c(z_{k+1}) & \nabla_{z_{k+1}}c(z_{k+1}) \end{bmatrix}\Bigg|_{\xi_\tau} \begin{bmatrix} x_{k+1}-\hat{x}_{k+1} \\ z_{k+1}-\hat{z}_{k+1} \end{bmatrix} \mathrm{d}\tau=\boldsymbol{0} \tag{5.18}$$

Expand (5.18) and then rearrange to get

$$\begin{aligned} \hat{z}_{k+1}-z_{k+1}=&\left(\int_0^1\begin{bmatrix}\nabla_{z_{k+1}}\left(\eta^T\nabla_{z_{k+1}}V_H(z_{k+1})\right)\\ \nabla_{z_{k+1}}c(z_{k+1})\end{bmatrix}\Bigg|_{\xi_\tau}\mathrm{d}\tau\right)^{-1} \\ &\cdot\left(\int_0^1\begin{bmatrix}\nabla_{x_{k+1}}\left(\eta^T\nabla_{z_{k+1}}V_H(z_{k+1})\right)\\ \nabla_{x_{k+1}}c(z_{k+1})\end{bmatrix}\Bigg|_{\xi_\tau}\mathrm{d}\tau\right)\left[x_{k+1}-\hat{x}_{k+1}\right] \end{aligned} \tag{5.19}$$

By taking the Euclidean norms on both sides of (5.19), and defining Z_z and Z_x as the upper bounds to the norms of the first and the second integrands of (5.19), respectively, one obtains (5.14). □

When the linearized model of the system (5.10) is used and the MPC cost function is quadratic and subjected to $c(z_{k+1})=[A_I\ -A_B]z_{k+1}-v_{x_{k+1}}$ as in (5.11), the gradient $\nabla_{z_{k+1}}c(z_{k+1})$ becomes a constant matrix, hence, the basis η for the null space of $\nabla_{z_{k+1}}c(z_{k+1})$. In addition, the other partial differentials in Z_x and Z_z; namely, $\nabla_{x_{k+1}}\nabla_{z_{k+1}}V_H(z_{k+1})\in\mathbb{R}^{(H+1)(m+n)\times n}$, $\nabla_{x_{k+1}}c(z_{k+1})\in\mathbb{R}^{(H+1)(m+n)\times n}$, and $\nabla_{z_{k+1}}\nabla_{z_{k+1}}V_H(z_{k+1})\in\mathbb{R}^{(H+1)(m+n)\times(H+1)(m+n)}$, are also constant matrices. As a result, the integrands are constant, rendering Z_x and Z_z to be constants as given by $Z_x=\left\|\begin{bmatrix}\eta^T\nabla_{x_{k+1}}\nabla_{z_{k+1}}V_H(z_{k+1})\\ \nabla_{x_{k+1}}c(z_{k+1})\end{bmatrix}\right\|$ and $Z_z=\left\|\begin{bmatrix}\eta^T\nabla_{z_{k+1}}\nabla_{z_{k+1}}V_H(z_{k+1})\\ \nabla_{z_{k+1}}c(z_{k+1})\end{bmatrix}^{-1}\right\|$. For a particular system, there can be several different bases η that satisfy $\nabla_{z_{k+1}}c(z_{k+1})\eta=\boldsymbol{0}$, but any basis η results in the same Z_x and Z_z. The four partial differentials within Z_x and Z_z are obtained as

follows:

$$\nabla_{x_{k+1}}\nabla_{z_{k+1}}V_H(z_{k+1}) = 2\cdot\begin{bmatrix}Q_{diag}{}^T & 0^T & \cdots & 0^T\end{bmatrix}^T$$

$$\nabla_{x_{k+1}}c(z_{k+1}) = \begin{bmatrix}I_n^T & 0_{m\times n}^T & -A^T & 0^T & \cdots & 0^T\end{bmatrix}^T$$

$$\nabla_{z_{k+1}}\nabla_{z_{k+1}}V_H(z_{k+1}) = 2\cdot\mathrm{diag}((Q)_{11},\ldots,(Q)_{nn},(R)_{11},\ldots,(R)_{mm},\ldots,(P_0)_{11},\ldots,(P_0)_{nn},0_m)$$

$$\nabla_{z_{k+1}}c(z_{k+1}) = \begin{bmatrix}
I_n & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0\\
-A & -B & I_n & 0 & \cdots & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0\\
0 & 0 & -A & -B & \cdots & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0\\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots\\
0 & 0 & 0 & 0 & \cdots & -A & -B & I_n & 0\\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0
\end{bmatrix}$$

where Q_{diag} is a matrix with only the diagonal entries of Q, $(Q)_{jj}$ are the diagonal elements of Q, $(R)_{jj}$ are the diagonal elements of R, and $(P_0)_{jj}$ are the diagonal elements of P_0.

5.4 Main Stability Results

Referring to Lemma 5.1, the second term $\left[V_H(\hat{x}_{k+1},\hat{\pi}_{k+1}) - V_H^*(x_{k+1},\pi_{k+1})\right]$ in (5.8) can be formulated in terms of the estimated states and their corresponding estimation errors from k to $k+H$ by first recasting the finite horizon quadratic cost function of the MPC policy (5.2) in terms of $\chi_{k+i} = \begin{bmatrix}x_{k+i}^T & u_{k+i}^T\end{bmatrix}^T \in \mathbb{R}^{m+n}$ as follows:

$$V_H(x_k,\pi_k) = V_H(\chi_k) = \chi_{k+H}^T M_{P_0}\chi_{k+H} + \sum_{i=0}^{H-1}\chi_{k+i}^T M_{QR}\chi_{k+i}$$

where $M_{P_0} = \mathrm{diag}(P_0,0) \in \mathbb{R}^{(m+n)\times(m+n)}$ and $M_{QR} = \mathrm{diag}(Q,R) \in \mathbb{R}^{(m+n)\times(m+n)}$. Then, the difference between the costs from the initial estimated and actual states at $(k+1)$ is obtained as:

$$V_H(\hat{\chi}_{k+1}) - V_H^*(\chi_{k+1})$$
$$= \hat{\chi}_{k+H+1}^T \boldsymbol{M}_{P_0} \hat{\chi}_{k+H+1} - \chi_{k+H+1}^T \boldsymbol{M}_{P_0} \chi_{k+H+1} + \sum_{i=1}^{H} \left(\hat{\chi}_{k+i}^T \boldsymbol{M}_{QR} \hat{\chi}_{k+i} - \chi_{k+i}^T \boldsymbol{M}_{QR} \chi_{k+i} \right) \quad (5.20)$$

By defining the state-input estimation error vector as $\varepsilon_{k+i} = \hat{\chi}_{k+i} - \chi_{k+i}$ for $i = 0,...,H$, the actual state-input vectors χ_{k+i} (optimal but unknown) are removed from (5.20) to yield

$$V_H(\hat{\chi}_{k+1}) - V_H^*(\chi_{k+1})$$
$$= \left[\varepsilon_{k+H+1}^T \boldsymbol{M}_{P_0} \hat{\chi}_{k+H+1} + \hat{\chi}_{k+H+1}^T \boldsymbol{M}_{P_0} \varepsilon_{k+H+1} + \sum_{i=1}^{H} \left(\varepsilon_{k+i}^T \boldsymbol{M}_{QR} \hat{\chi}_{k+i} + \hat{\chi}_{k+i}^T \boldsymbol{M}_{QR} \varepsilon_{k+i} \right) \right] \quad (5.21)$$
$$- \left[\varepsilon_{k+H+1}^T \boldsymbol{M}_{P_0} \varepsilon_{k+H+1} + \sum_{i=1}^{H} \varepsilon_{k+i}^T \boldsymbol{M}_{QR} \varepsilon_{k+i} \right]$$

In order to bound $V_H(\hat{\chi}_{k+1}) - V_H^*(\chi_{k+1})$ or equivalently $V_H(\hat{x}_{k+1}, \hat{\pi}_{k+1}) - V_H^*(x_{k+1}, \pi_{k+1})$ in (5.8) from above, $\hat{\chi}_{k+i}$ and ε_{k+i} have to be bounded first. The following two lemmas formalize the required bounds relative to x_k.

Lemma 5.3: *If $x_k \subseteq \mathcal{X}$ and $\mathcal{X}$ is a bounded space for $\forall k > 0$, then the estimated state-input vectors are bounded by the state x_k as*

$$\|\hat{\chi}_{k+i}\| \leq K_z \|x_k\| \quad (5.22)$$

for $\forall i > 0$ where K_z is non-negative.

Proof:

Assume that the estimated state $\hat{x}_k$ remains within a bounded space $\mathcal{X}$ for $\forall k > 0$. Since both x_k and $\hat{x}_k$ are governed by the model (5.1) of the system, it may be concluded that (5.22) always exists. □

Lemma 5.4: *The state-input estimation error vectors ε_{k+i} are bounded by the state x_k according to*

$$\|\varepsilon_{k+i}\| \leq Z_x Z_z K_e \|x_k\| \quad (5.23)$$

for $\forall i > 0$.

Proof:

By taking the Euclidean norm of $\varepsilon_{k+i} = \hat{\chi}_{k+i} - \chi_{k+i}$ one obtains $\|\varepsilon_{k+i}\| = \|\hat{\chi}_{k+i} - \chi_{k+i}\|$. Since $[\hat{\chi}_{k+i} - \chi_{k+i}]$ is within $[\hat{z}_{k+i} - z_{k+i}]$, one obtains $\|\hat{\chi}_{k+i} - \chi_{k+i}\| \le \|\hat{z}_{k+i} - z_{k+i}\|$ and hence, $\|\varepsilon_{k+i}\| \le \|\hat{z}_{k+i} - z_{k+i}\|$. From Lemma 5.2, it is established that $\|\hat{z}_{k+1} - z_{k+1}\| \le Z_x Z_z \|x_{k+1} - \hat{x}_{k+1}\|$, giving the bound $\|\varepsilon_{k+i}\| \le Z_x Z_z \|x_{k+1} - \hat{x}_{k+1}\|$. It is characterized in (5.7) that $\|x_{k+1} - \hat{x}_{k+1}\| \le K_e \|x_k\|$ and this completes the proof. □

The (asymptotic) stabilizing properties of the closed-loop system that arise by implementing a state observer to estimate the states of the system are established below.

Theorem 5.3: *The system under control as described by model* (5.1) *using a finite sequence of future control actions from the MPC control policy* (5.2) *with a prediction horizon of* H *and estimated state* $\hat{x}_k$, *is asymptotically stable in the sense of Lyapunov if the state estimation error bounding gain* $K_e \ge 0$ *of* (5.7) *is bounded as*

$$K_e \le \frac{K_B}{2\left(\|M_{P_0}\| + 2H\|M_{QR}\|\right) Z_x Z_z K_z} \tag{5.24}$$

and

$$\|x_k\|_Q^2 - K_B\left(\|x_k\|\right)^2 > 0 \tag{5.25}$$

Proof:

Proceed from the stability bound of (5.8) given in Lemma 5.1. The second term on the right-hand side of (5.8) is equivalent to (5.21). In order to establish an upper bound for this second term, (5.21) has to be bounded from above. The following bound is obtained by taking the Euclidean norm on both sides of (5.21) and dropping the second term of (5.21):

$$\begin{aligned} & V_H(\hat{\chi}_{k+1}) - V_H^*(\chi_{k+1}) \\ & \le \left\|\varepsilon_{k+H+1}^T M_{P_0} \hat{\chi}_{k+H+1}\right\| + \left\|\hat{\chi}_{k+H+1}^T M_{P_0} \varepsilon_{k+H+1}\right\| \\ & \quad + \sum_{i=1}^{H}\left(\left\|\varepsilon_{k+i}^T M_{QR} \hat{\chi}_{k+i}\right\| + \left\|\hat{\chi}_{k+i}^T M_{QR} \varepsilon_{k+i}\right\|\right) \\ & \le 2\|\varepsilon_{k+H+1}\|\|M_{P_0}\|\|\hat{\chi}_{k+H+1}\| + 2\sum_{i=1}^{H}\|\varepsilon_{k+i}\|\|M_{QR}\|\|\hat{\chi}_{k+i}\| \end{aligned} \tag{5.26}$$

Substitute (5.22) from Lemma 5.3 and (5.23) from Lemma 5.4 into (5.26). One obtains

$$\begin{aligned} & V_H(\hat{\chi}_{k+1}) - V_H^*(\chi_{k+1}) \\ & \quad \le 2\left\|\boldsymbol{M}_{P_0}\right\| Z_x Z_z K_e K_z \left(\|\boldsymbol{x}_k\|\right)^2 + 2H\left\|\boldsymbol{M}_{QR}\right\| Z_x Z_z K_e K_z \left(\|\boldsymbol{x}_k\|\right)^2 \\ & \quad \le 2\left(\left\|\boldsymbol{M}_{P_0}\right\| + 2H\left\|\boldsymbol{M}_{QR}\right\|\right) Z_x Z_z K_e K_z \left(\|\boldsymbol{x}_k\|\right)^2 \\ & \quad \le K_B \left(\|\boldsymbol{x}_k\|\right)^2 \end{aligned} \tag{5.27}$$

where $$K_B = 2\left(\left\|\boldsymbol{M}_{P_0}\right\| + 2H\left\|\boldsymbol{M}_{QR}\right\|\right) Z_x Z_z K_e K_z \tag{5.28}$$

thus arriving at (5.24). By combining (5.27) and (5.8), the asymptotic stability condition of (5.25) is established. □

The cost evolution bounding gain K_B is determined by searching and comparing the spaces of the actual and the estimated states (except the origin) that satisfy (5.25) and (5.27). The rationale is that each pair of initial states $\boldsymbol{x}_k$ and $\hat{\boldsymbol{x}}_k$ (to the optimization problem of (5.11)) has an associated value of K_B which is calculated from:

$$K_B = \frac{\left|V_H^*(\hat{\boldsymbol{x}}_{k+1}, \hat{\boldsymbol{\pi}}_{k+1}) - V_H^*(\boldsymbol{x}_{k+1}, \boldsymbol{\pi}_{k+1})\right|}{\left(\|\boldsymbol{x}_k\|\right)^2} \tag{5.29}$$

in accordance with (5.27). Hence, the maximum upper bound to condition (5.27) (less conservative) is achieved by determining the maximum value of K_B that is possible within the combination spaces of $\boldsymbol{x}_k$ and $\hat{\boldsymbol{x}}_k$ while satisfying the asymptotic stability condition of (5.25).

Similarly, the input-state parametric evolution bounding gain K_z is determined by searching the space of the estimated state $\hat{\boldsymbol{x}}_k$ (except the origin) that satisfies (5.22). The optimal decision variables $\left\|\hat{\chi}_{k+i}^*\right\|$ for $i = 0, \ldots, H$ from the solution of (5.11) with the initial condition $\hat{\boldsymbol{x}}_k$, are used to calculate each corresponding K_z using the following equation:

$$K_z = \frac{\left\|\hat{\chi}_{k+i}\right\|}{\left\|\hat{\boldsymbol{x}}_k\right\|} \tag{5.30}$$

The maximum possible value of K_z is sought. Note that if the states and the input are unconstrained in the optimization of (5.11), K_z is unbounded and may assume a substantially large value depending on the configuration of the MPC control policy (see Section 5.5).

Since $\boldsymbol{Q} \succ 0$, it is clear from (5.25) in Theorem 5.3 that if $K_B = 0$, $V_H(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{\pi}}_k)$ is guaranteed to be decreasing with time along the trajectory of the closed-loop system, thereby ensuring asymptotic stability. It is observed from (5.24) that the state estimation error bounding gain K_e is *poorly* governed. According to the established results for stability of MPC (Keerthi and Gilbert, 1988, Rawlings and Muske, 1993, Mayne, *et al.*, 2000, Tang and de Silva, 2005b, 2005c), closed-loop stability can be improved by increasing the prediction horizon H and the terminal weight matrix $\boldsymbol{P}_0$. Absolute stability can be achieved with an infinite prediction horizon. However, in the presence of imperfect state feedback, counter intuitively, increasing H or $\boldsymbol{P}_0$ reduces K_e, thereby reducing the stability tolerance to state estimation errors. A trade-off in the design of the MPC controller is required. The severity of this effect is further explored numerically in the following section.

The overall off-line algorithm for determining the stable upper bound of the state estimation error for a given system and parameters of the developed NCS-MPC control strategy is synthesized in Table 5.1. The multiple optimizations of (5.11) to search for $K_{z,\max}$ and $K_{B,\max}$ can be rather computationally intensive. The computing load can be decreased by reducing on the resolution of the search space.

Table 5.1: The algorithm for determining the stable upper bound of the state estimation error of the developed NCS-MPC control strategy.

1.	For a given system model, set $\boldsymbol{Q}$, $\boldsymbol{R}$, $\boldsymbol{P}_0$, and H.
2.	Calculate $\boldsymbol{J}_e$, $\boldsymbol{A}_I$ and $\boldsymbol{A}_B$ of the optimization problem (5.11).
3.	Compute the gradients and Hessians within (5.14); i.e., $\nabla_{z_{k+1}} \boldsymbol{c}(z_{k+1})$, $\nabla_{x_{k+1}} \nabla_{z_{k+1}} V_H(z_{k+1})$, $\nabla_{x_{k+1}} \boldsymbol{c}(z_{k+1})$, and $\nabla_{z_{k+1}} \nabla_{z_{k+1}} V_H(z_{k+1})$.
4.	Determine the null space basis $\boldsymbol{\eta}$ and find Z_x and Z_z.
5.	Initialize $K_{z,\max} = 0$ and $K_{B,\max} = 0$.
6.	Select a state $\hat{\boldsymbol{x}}_k$ within the bounded state-space $\mathcal{X}$.
7.	Optimize (5.11) using $\hat{\boldsymbol{x}}_k$ to find $\hat{\boldsymbol{\chi}}^*_{k+i}$ for $i = 0,\ldots,H$.
8.	For each $\hat{\boldsymbol{\chi}}^*_{k+i}$, determine $K_z = \left\|\hat{\boldsymbol{\chi}}^*_{k+i}\right\| / \left\|\hat{\boldsymbol{x}}_k\right\|$, and if $K_z > K_{z,\max}$, then set $K_{z,\max} = K_z$.
9.	Repeat the steps 6-8 with a new state $\hat{\boldsymbol{x}}_k$ until the entire $\mathcal{X}$ is explored.
10.	Select a state $\boldsymbol{x}_k$ within the bounded state-space $\mathcal{X}$.
11.	Optimize (5.11) using $\boldsymbol{x}_k$ to find $\boldsymbol{x}^*_{k+1}$ from the solution.
12.	Optimize (5.11) using $\boldsymbol{x}^*_{k+1}$ from step 11 to find $V^*_H(\boldsymbol{x}_{k+1}, \boldsymbol{\pi}_{k+1})$.
13.	Select a state $\hat{\boldsymbol{x}}_k$ within the bounded state-space $\mathcal{X}$.
14.	Optimize (5.11) using $\hat{\boldsymbol{x}}_k$ to find $\hat{\boldsymbol{x}}^*_{k+1}$ from the solution.
15.	Optimize (5.11) using $\boldsymbol{x}^*_{k+1}$ from step 14 to find $V^*_H(\hat{\boldsymbol{x}}_{k+1}, \hat{\boldsymbol{\pi}}_{k+1})$.
16.	Compute $K_B = \left\|V^*_H(\hat{\boldsymbol{x}}_{k+1}, \hat{\boldsymbol{\pi}}_{k+1}) - V^*_H(\boldsymbol{x}_{k+1}, \boldsymbol{\pi}_{k+1})\right\| / \left(\left\|\boldsymbol{x}_k\right\|\right)^2$.
17.	If $K_B > K_{B,\max}$ and $\left\|\boldsymbol{x}_k\right\|^2_{\boldsymbol{Q}} - K_B\left(\left\|\boldsymbol{x}_k\right\|\right)^2 > 0$, then set $K_{B,\max} = K_B$.
18.	Repeat the steps 13-17 with a new state $\hat{\boldsymbol{x}}_k$ until the entire $\mathcal{X}$ is explored.
19.	Repeat the steps 10-18 with a new state $\boldsymbol{x}_k$ until the entire $\mathcal{X}$ is explored.
20.	Compute $K_{e,\max}$ with (5.24) using $K_{z,\max}$ and $K_{B,\max}$.

5.5 Evaluation of Stability Boundaries

The stability boundaries of the NCS-MPC control strategy with imperfect state estimation as applied to the electro-hydraulic manipulator system of the fish-processing machine (see Chapter 2) is investigated in this section. The effect of various design parameters, especially the terminal weight matrix and the prediction horizon, on the level of state estimate error that can be tolerated to maintain closed-loop asymptotic stability is evaluated based on the 4-state-1-

input model of the electro-hydraulic manipulator as given in (2.17). As in Section 4.4.2, the weighting matrices are chosen as $\boldsymbol{Q} = \text{diag}(1,10,0.001,0.001)$ and $\boldsymbol{R} = 100$. The upper stable bound on state estimation error is evaluated using the developed algorithm (Table 5.1) with various lengths of prediction horizon H and various values of terminal weights $\boldsymbol{P}_0$ as fractions of the open-loop infinite weight $\boldsymbol{U}_\infty$. Note that $\boldsymbol{U}_\infty$ is attained by solving the discrete Lyapunov equation $\boldsymbol{U}_\infty = \boldsymbol{Q} + \boldsymbol{A}^T \boldsymbol{U}_\infty \boldsymbol{A}$.

As discussed in the previous section, the maximum attainable cost evolution bounding gain K_B (the higher the better) and the input-state parametric evolution bounding gain K_z (the lower the better) depends on the configuration of a particular system and the MPC controller. As these gains are nonlinearly related to the states, the location where the maximum occurs also varies and has to be searched exhaustively. In addition, the maximum K_B must also satisfy the stability condition (5.25). Figure 5.1 shows the variation of the maximum K_z on the normalized state plane of the first two estimated states of the system (i.e., position and velocity of the hydraulic piston) while the pressure states are kept at the origin (equilibrium). The length of the prediction horizon is set to $H = 5$ and the terminal weight matrix is set to $\boldsymbol{P}_0 = 0.5\boldsymbol{U}_\infty$. It is observed that although the surface of K_z is not fully symmetric, the low regions of K_z are concentrated approximately along the $\hat{x}_2$-axis while the high regions are along the $\hat{x}_1$-axis. This is because the system is quite sensitive to variations in $\hat{x}_1$ but less sensitive to variations in $\hat{x}_2$. Figure 5.2 shows the variation of the maximum and stable K_B on the normalized state plane of the first two states of the system with the pressure states fixed at the origin. The settings are identical to those in Figure 5.1. It is observed that the maximum and stable K_B does not vary with the location of the states but rather depends on the parameters of the MPC controller; i.e., $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{Q}$, $\boldsymbol{R}$, $\boldsymbol{P}_0$, and H.

Figure 5.3(a) shows the maximum tolerable state estimation error (norm of the state vector) K_e in order to maintain closed-loop asymptotic stability under different settings of $\boldsymbol{P}_0$ and prediction horizon values up to 10 steps. The values of Z_x and Z_z in (5.14) are 3.34 and 10.77, respectively. At $\boldsymbol{P}_0 = 0.01\boldsymbol{U}_\infty$ and $H = 1$, K_e is approximately 0.012, which means that in order for the system to remain stable, the norm of the state estimation error $\|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1}\|$ must

not exceed 1.2% of the norm of the current state $\|\boldsymbol{x}_k\|$. Note that this percentage is given relative to a normalized system and the norm of the combined states. If the velocity and the two pressures of the cylinder are measured perfectly, then the maximum tolerable error in the position of the piston is 0.6 mm. The error tolerance decreases with the increase in $\boldsymbol{P}_0$ and more rapidly with the length of the prediction horizon, which is mainly due to the exponential increase in K_z, as shown in Figure 5.3(c). The surge in K_z can be effectively controlled by imposing state and input constraints on the MPC control law, which will also result in tighter bounds for Z_x and Z_z, thereby increasing K_e. Figure 5.3(b) shows that K_B remains approximately at 1, independent of $\boldsymbol{P}_0$, at shorter prediction horizon. Fluctuations of K_B are observed for longer prediction horizons because future state predictions within the MPC policy are governed by the system model (see (4.11) in Section 4.2.1). When the prediction is moved farther into the future, the state evolution diverges further from the initial state $\boldsymbol{x}_k$ causing the optimization of the MPC problem to be more sensitive.

It is found that the 4-state model of the electro-hydraulic manipulator system is itself too "sensitive" to state variations. This is mainly due to the fact that the states corresponding to the pressures of the cylinder are not directly controllable. By reducing the order to a 2-state model with position and velocity state feedback; i.e., $\boldsymbol{A} = \begin{bmatrix} 1.06730 & -0.10423 \\ -0.02484 & 0.89404 \end{bmatrix}$ and $\boldsymbol{B} = \begin{bmatrix} 0.00138 \\ 0.01027 \end{bmatrix}$, and keeping the controller settings as before, a higher stability tolerance of the state estimation error is achieved. This is shown in Figure 5.4(a). As can be observed from Figure 5.4(b), K_B is reduced only by a small margin to approximately 0.52, with less fluctuations at a longer prediction horizon. More importantly, K_z is substantially reduced with a desirable reverse effect from $\boldsymbol{P}_0$ as the prediction horizon is increased, as shown in Figure 5.4(c). The reverse effect from $\boldsymbol{P}_0$ is desirable because for asymptotic stability, the developed future input buffering NCS technique (see Chapter 4) requires a sufficiently large $\boldsymbol{P}_0$ and a sufficiently long prediction horizon.

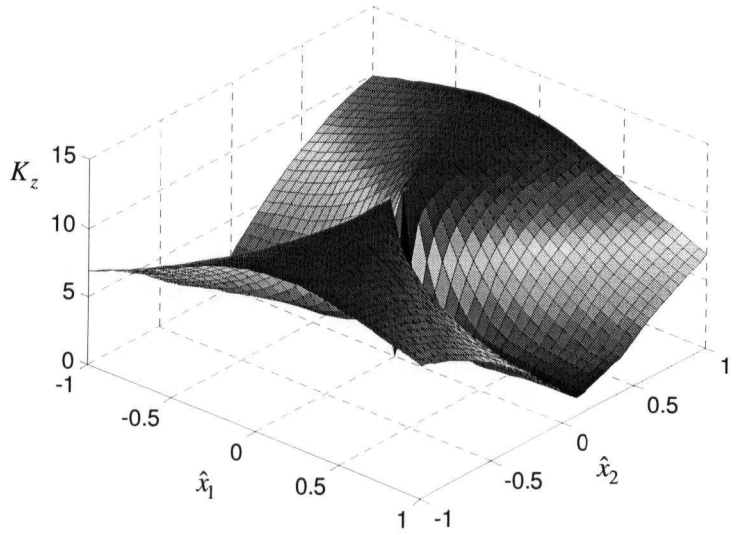


Figure 5.1: Variation of the maximum K_z on the estimated state plane of $\hat{x}_1$-$\hat{x}_2$ with $\hat{x}_3 = \hat{x}_4 = 0$ under a prediction horizon $H = 5$ and a terminal weight matrix $\boldsymbol{P}_0 = 0.5\boldsymbol{U}_\infty$.

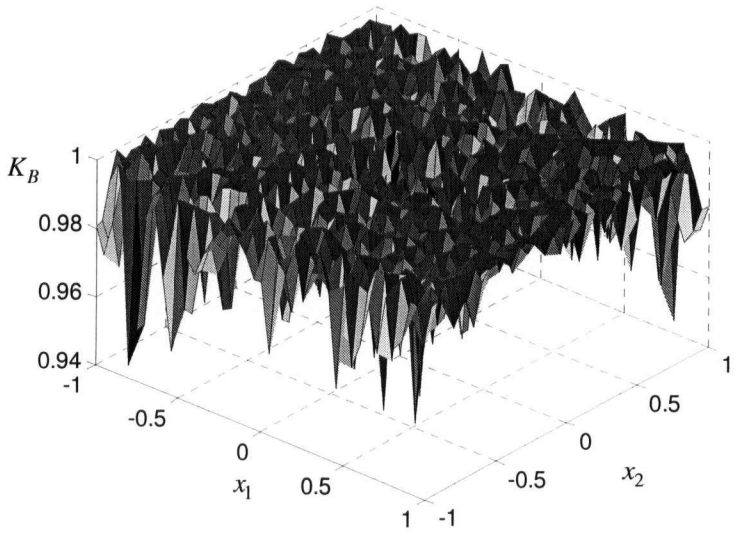


Figure 5.2: Variation of the maximum K_B on the state plane of x_1-x_2 with $x_3 = x_4 = 0$ under a prediction horizon $H = 5$ and a terminal weight matrix $\boldsymbol{P}_0 = 0.5\boldsymbol{U}_\infty$.

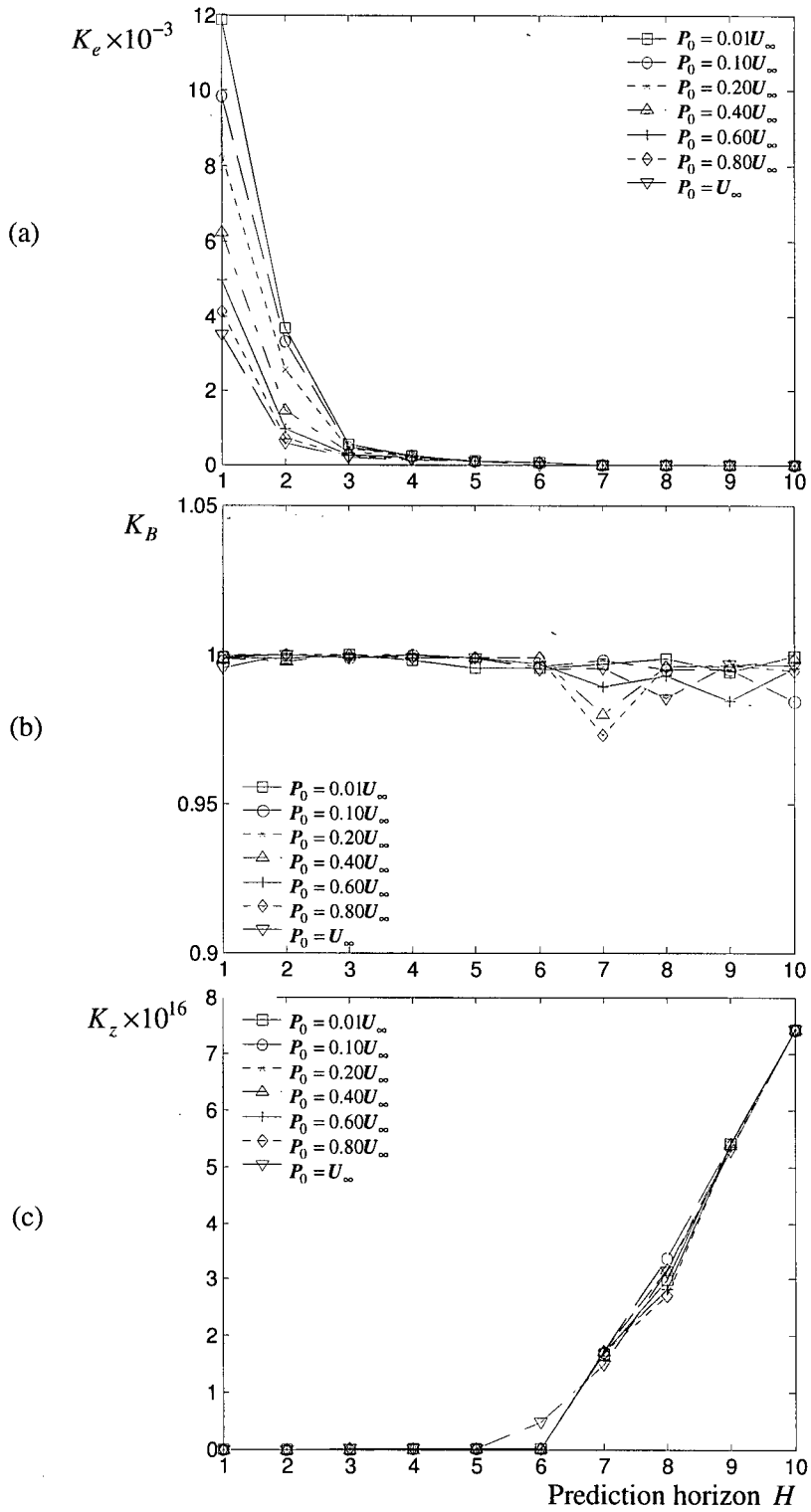


Figure 5.3: The effect of P_0 and H on stability based on the original model. (a) Maximum bound of K_e; (b) Maximum bound of K_B; and (c) Maximum bound of K_z.

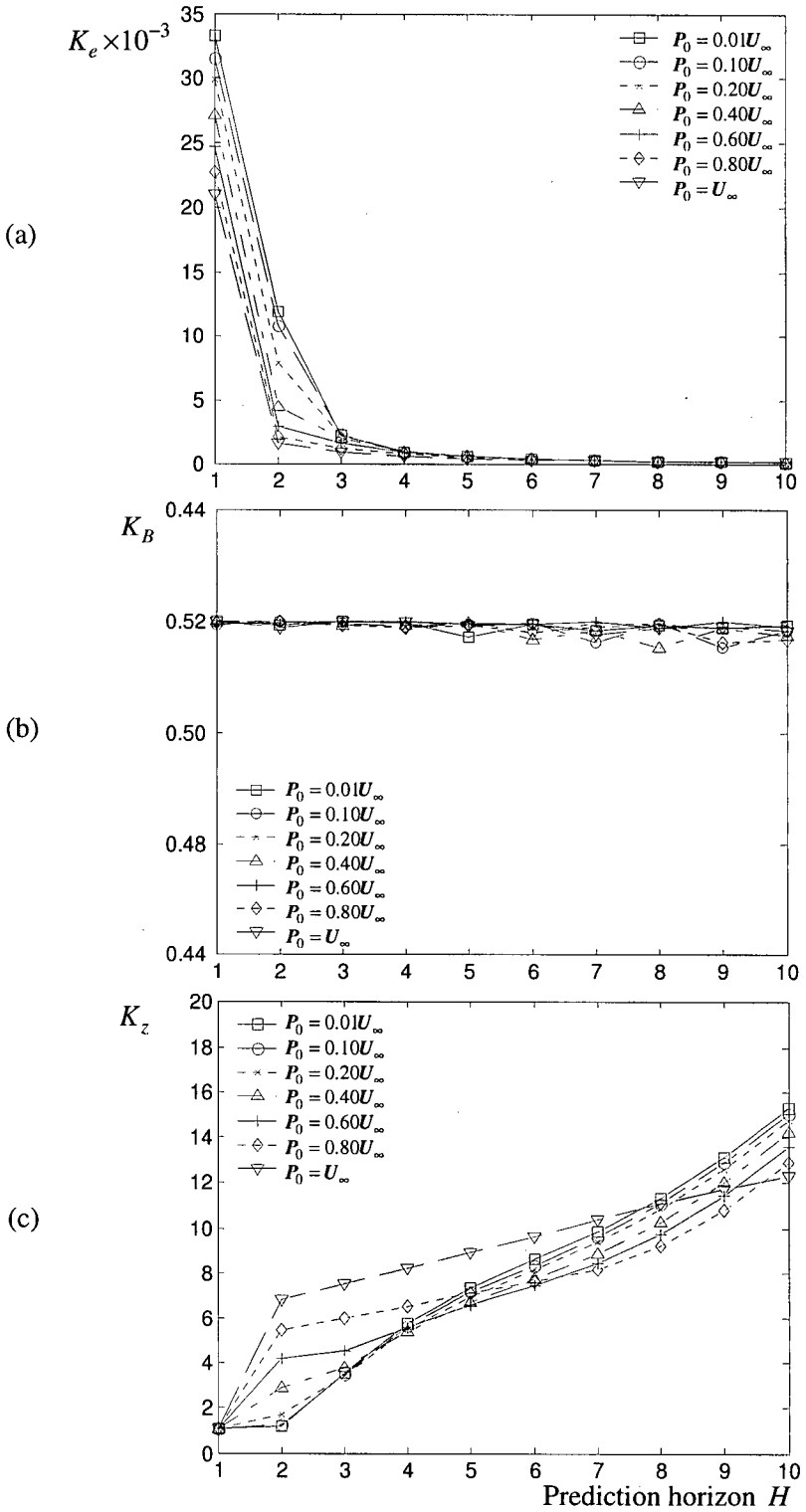


Figure 5.4: The effects of P_0 and H on stability based on the reduced model of the system. (a) Maximum bound of K_e; (b) Maximum bound of K_B; and (c) Maximum bound of K_z.

5.6 Extension to MPC with Inequality Constraints

The upper bound of the state estimation error that can be tolerated in order to maintain closed-loop asymptotic stability of the developed NCS-MPC control strategy, may be improved by imposing state and input inequality constraints to the MPC optimization problem of (5.2) or equivalently (5.11). The analysis of the sensitivity of the state estimation errors, as carried out in Section 5.3 is only feasible for cases with equality constraints. A difficulty arises when *inequality* constraints have to be imposed in the sensitivity analysis. An iterative, multi-shooting optimization method is required to continuously adapt to conditions of active and inactive constraints. Alternatively, it is possible to convert all the inequality constraints into equivalent equality constraints and carry out the sensitivity analysis as before.

One approach for converting inequality constraints into equality constraints is to introduce slack variables (Evtushenko and Zhadan, 1994). Consider the general constrained MPC optimization problem with i equality constraints and j inequality constraints, as given by:

$$\begin{aligned} &\text{minimize } V_H(z_k) \\ &\text{s.t. } \boldsymbol{c}_{\mathcal{E}}(z_k) = \boldsymbol{0}_i \text{ and } \boldsymbol{c}_{\mathcal{I}}(z_k) \le \boldsymbol{0}_j \end{aligned} \tag{5.31}$$

In order to convert $\boldsymbol{c}_{\mathcal{I}}(z_k) \le \boldsymbol{0}_j$ to equality constraints, an additional (slack) variable vector $\boldsymbol{\rho} \in \mathbb{R}^j$ is introduced first and then all the constraints are combined to transform (5.31) into the following equivalent problem:

$$\begin{aligned} &\text{minimize } V_H(z_k, \boldsymbol{\rho}) \\ &\text{s.t. } \boldsymbol{\Phi}(z_k, \boldsymbol{\rho}) = \begin{bmatrix} \boldsymbol{c}_{\mathcal{E}}(z_k) \\ \boldsymbol{c}_{\mathcal{I}}(z_k) + \boldsymbol{\rho} \end{bmatrix} = \boldsymbol{0}_{i+j} \text{ and } \boldsymbol{\rho} > 0_j \end{aligned} \tag{5.32}$$

In order to take into account the constraint $\boldsymbol{\rho} > \boldsymbol{0}_j$, use a differential mapping $\mathcal{M} : \mathbb{R}^j \to \mathbb{R}^j_+$ and make the space transformation $\boldsymbol{\rho} = \mathcal{M}(\bar{\boldsymbol{\rho}})$ where $\bar{\boldsymbol{\rho}} \in \mathbb{R}^j$. Let $\mathcal{M}_\rho$ be the square Jacobian matrix of the mapping $\mathcal{M}(\bar{\boldsymbol{\rho}})$ with respect to $\bar{\boldsymbol{\rho}}$. Assuming that the inverse $\bar{\boldsymbol{\rho}} = \mathcal{M}(\boldsymbol{\rho})$ is defined, the Jacobian and Gram matrices are obtained, respectively, as $\boldsymbol{J}_a(\boldsymbol{\rho}) = \mathcal{M}(\bar{\boldsymbol{\rho}})\big|_{\bar{\boldsymbol{\rho}} = \mathcal{M}(\boldsymbol{\rho})} \in \mathbb{R}^{j \times j}$ and $\boldsymbol{G}_m(\boldsymbol{\rho}) = \boldsymbol{J}_a(\boldsymbol{\rho}) \boldsymbol{J}_a^T(\boldsymbol{\rho}) \in \mathbb{R}^{j \times j}$. Combining the variables and the constraints for the reduced problem of (5.32), the following equivalent problem with equality constraints only is obtained:

$$\begin{aligned} &\text{minimize } V_H(z_k, \boldsymbol{\rho}, \bar{\boldsymbol{\rho}}) \\ &\text{s.t. } \bar{\boldsymbol{\Phi}}(z_k, \boldsymbol{\rho}, \bar{\boldsymbol{\rho}}) = \begin{bmatrix} \boldsymbol{c}_{\mathcal{E}}(z_k) \\ \boldsymbol{c}_{\mathcal{I}}(z_k) + \mathcal{M}(\bar{\boldsymbol{\rho}}) \end{bmatrix} = \boldsymbol{0}_{i+j} \end{aligned} \tag{5.33}$$

Another way of introducing slack variables is to let that variable be squared; i.e., $\boldsymbol{\rho}^2 \in \mathbb{R}^j$ to remove the above-mentioned non-negativeness condition, so that the equivalent optimization problem is given as:

$$\begin{aligned} &\text{minimize } V_H(z_k) \\ &\text{s.t. } \boldsymbol{c}_{\mathcal{E}}(z_k) = \boldsymbol{0}_i \text{ and } \boldsymbol{c}_{\mathcal{I}}(z_k) + \boldsymbol{\rho}^2 = \boldsymbol{0}_j \end{aligned} \tag{5.34}$$

However, additional Karush-Kuhn-Tucker optimality conditions have to be considered since the new Lagrangian becomes $\mathcal{L}(z, \lambda) = V_H(z) - \lambda \boldsymbol{c}_{\bar{\mathcal{E}}}(z)$ where $\boldsymbol{c}_{\bar{\mathcal{E}}}(z) = \begin{bmatrix} \boldsymbol{c}_{\mathcal{E}}(z) \\ \boldsymbol{c}_{\mathcal{I}}(z) + \boldsymbol{\rho}^2 \end{bmatrix}$. They are $\nabla_z \mathcal{L}(z, \lambda) = \boldsymbol{0}$, $\nabla_\rho \mathcal{L}(z, \lambda) = \boldsymbol{0}$, $\nabla_\lambda \mathcal{L}(z, \lambda) = \boldsymbol{0}$ and $\boldsymbol{c}_{\bar{\mathcal{E}}}(z) = \boldsymbol{0}$. Another possible alternative of transforming inequality constraints into equality constraints without the use of slack variables is to replace $\boldsymbol{c}_{\mathcal{I}}(z_k) \leq \boldsymbol{0}_j$ with $\min\{\boldsymbol{c}_{\mathcal{I}}(z_k), 0\}^2 = \boldsymbol{0}_j$, assuming $\boldsymbol{c}_{\mathcal{I}}(z_k)$ is continuously differentiable (Gockenbach, 2003).

5.7 Summary

This chapter investigated the effect of imperfect state measurements on the closed-loop asymptotical stability of the NCS-MPC strategy as developed in the previous chapters. The maximum deviation of the estimated state from the actual state that can be tolerated by the system to maintain stablity was derived by utilizing Lyapunov's second method and the concept of sensitivity analysis of the underlying optimization problem of the MPC control policy. The stability boundaries subjected to different configurations of the NCS-MPC control strategy was evaluated numerically on the electro-hydraulic manipulator of the fish-processing machine. The developed stability theorem provides a useful and binding design criterion for further development of robust state observers to compensate for network transmission problems between sensors and the controller.

Chapter 6
Infrastructure for Web-based Remote System Monitoring

In the previous chapters, various issues of data transmission for feedback control of a networked system have been resolved thorough analysis, computer simulation, and experimentation. In this context a novel control strategy based on constrained Model Predictive Control with future input buffering and long-range estimators has been developed and implemented. The functionality of a large-scale networked system can be further enhanced by incorporating higher layers having monitoring, diagnostic and supervisory capabilities. As an important contribution in this direction, the present chapter gives a framework for developing a universal and scalable network infrastructure for web-based monitoring and supervisory control of dynamic systems. Practical implementation of this framework is detailed, using a low cost and flexible two-tier client-server architecture, where a user is able to remotely carry out a variety of tasks including system operation, experimentation, system monitoring and supervision, task scheduling, system reconfiguration, control, and safety/emergency routines, through a web-browser interface. A single web-server provides smooth information flow using a robust and intelligent scheduling scheme. Techniques of web-based monitoring and supervision will be subjected by and large to the same critical data transmission problems as for feedback control, which degrade the performance of the system. However, these issues will not be addressed in the present thesis but left for possible future work.

Section 6.1 gives an overview of the intended application of the developed web-based remote monitoring infrastructure. The interconnection and implementation of networking hardware are discussed in Section 6.2. Section 6.3 details the software development for the system architecture, which includes the various servers, data flow management, user scheduling and authentication, and remote graphical interface. Section 6.4 demonstrates the applicability of the approach using the industrial fish-processing machine (see Chapter 2).

6.1 Infrastructure Overview

The application framework of the research presented in this thesis focuses on the development of a universal network architecture, including both hardware and software, which can be used for web-based monitoring and supervisory control of industrial production facilities and for use in research and academic environments. In particular, the developed technology is implemented to establish a web-based research infrastructure between several collaborating research institutions under the umbrella of the NUS/UBC Applied Science Research Centre (www.researchcentre.apsc.ubc.ca), which will create a working model to carry out full scale trail runs. The initial collaborators are the Faculty of Applied Science at the University of British Columbia and the faculties of Engineering and Science at the National University of Singapore, particularly in conjunction with the departments of Electrical and Computer Engineering and Mechanical Engineering, and their respective industrial partners and research institutes. A conceptual structure of collaboration is depicted in Figure 6.1. This arrangement

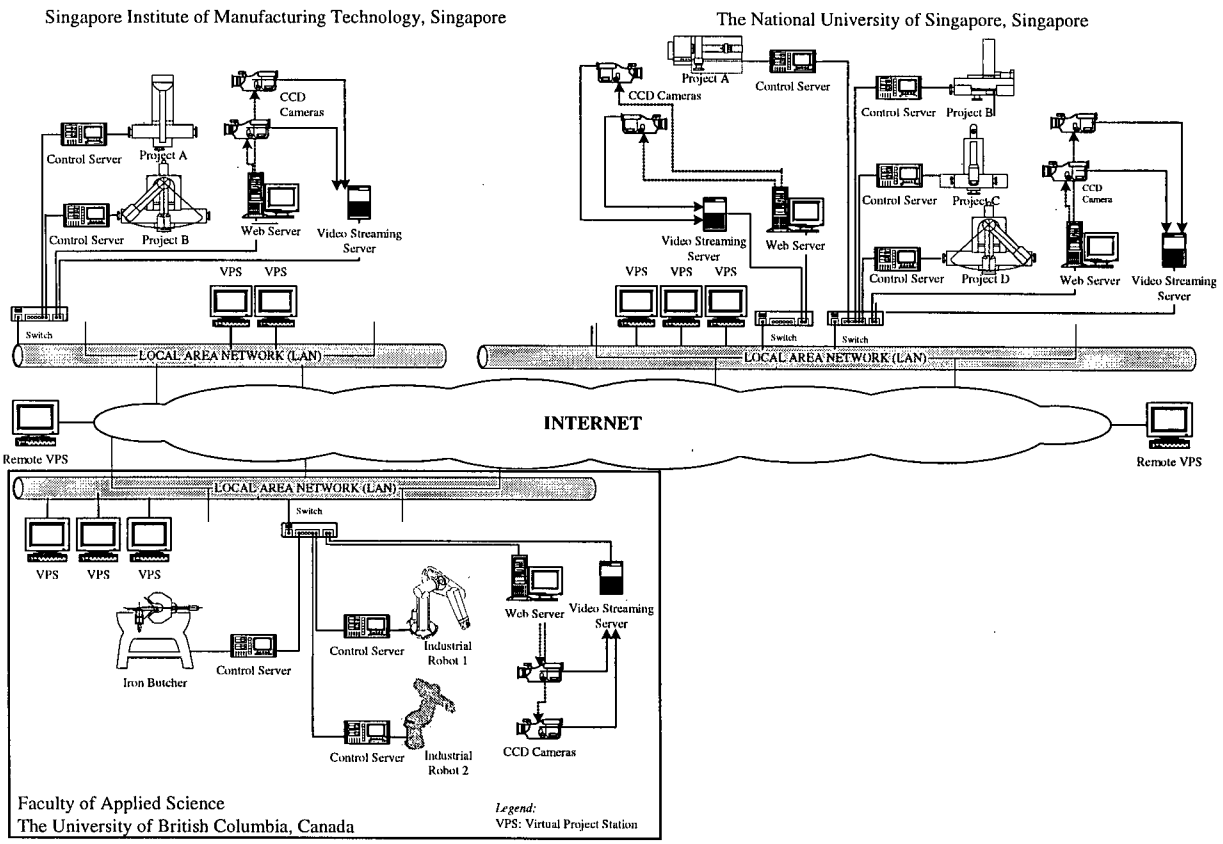


Figure 6.1: The general infrastructure for collaboration among different research institutions.

will allow authorized researchers and students to access laboratory equipment located at any of the university departments or organizations, through Virtual Project Stations (VPS). A VPS is basically a workstation with network (or modem) connection and a web-browser installed in it. It can access an experimental setup (or, industrial machine) for a variety of purposes such as performance monitoring, supervisory control, execution of pre-programmed experiments, and manually changing the system states (i.e., virtual pushing of system buttons). When low-level control laws and a higher supervisory level of a system have to be executed from a VPS, it will be necessary to download, compile and run a set of dedicated program codes on the VPS. For example, in (Overstreet and Tzes, 1999) the controller is compiled as a DLL (Dynamic Linked Library).

6.2 Hardware Networking

An objective of the current application is to utilize existing computer network technologies, which are cost effective and widely available in most corporate and educational institutions, to implement the developed infrastructure (Figure 6.1). The infrastructure is designed to perform optimally with a Fast Ethernet (100Base-T) backbone where each network

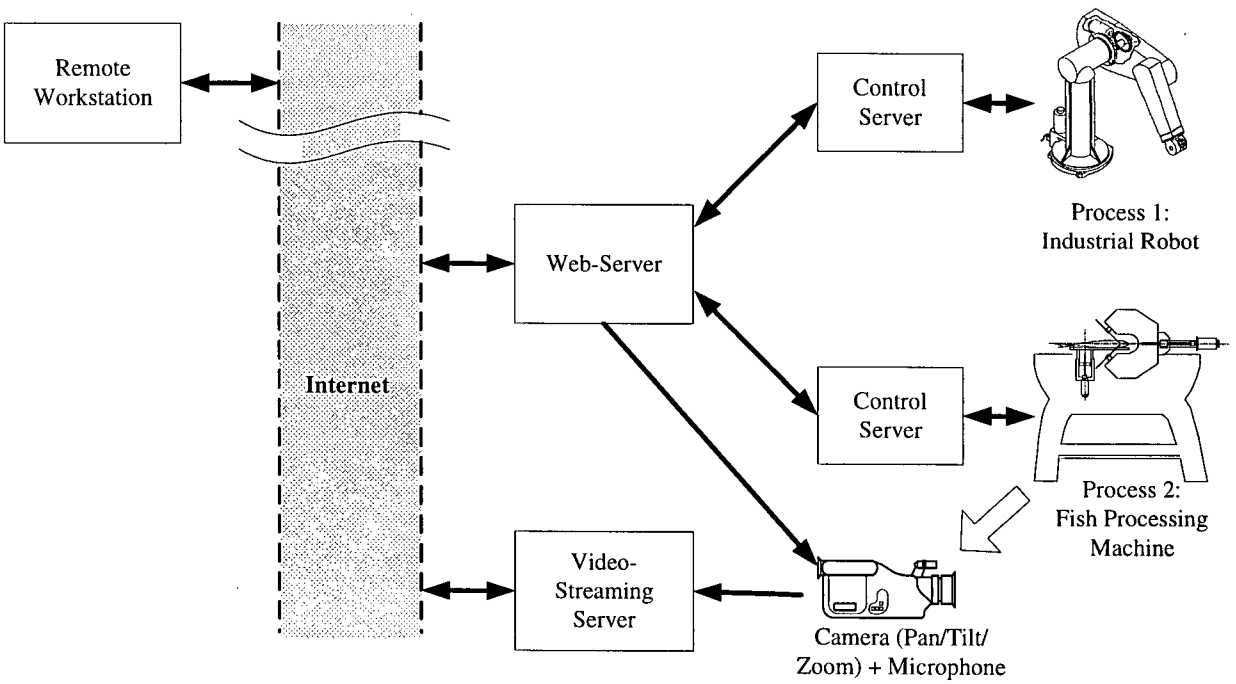


Figure 6.2: Simplified hardware architecture for web-based remote monitoring and supervisory control of a system.

device only needs a low cost Network Interface Card (NIC). Figure 6.2 shows a simplified hardware architecture, which is connected to two industrial systems or experimental setups (a fish-processing machine and an industrial robot). Each setup is directly connected to its individual control server, which handles basic networked communication between the system and the web-server, data acquisition, sending of control signals to the system, and the execution of low level control laws. Each control server contains one or more data acquisition boards, which have analog-to-digital conversion (ADC), digital-to-analog conversion (DAC), digital I/O capabilities, and frame grabbers for image processing. In this two-tier client-server architecture, the web-server provides secured single point access to all the systems within an establishment. All the control servers are setup such that they can communicate with only the web-server. Any data received by a control server, which are not transmitted from the web-server, will be ignored.

Video cameras and microphones are located at strategic locations to capture live audio and video signals allowing the remote user to view and listen to a system facility, and to communicate with local research personnel. The camera selected in the present application is the Panasonic Model KXDP702 color camera with built-in pan, tilt and 21x zoom, which can be controlled through a standard RS-232C communication protocol. Multiple cameras can be daisy-chained to the video-streaming server. For capturing and encoding the audio-video (AV) feed from a camera, the Winnov Videum 1000 PCI board is installed in the video-streaming server. It can capture video signals at a maximum resolution of 640×480 at 30 fps, with a hardware compression that significantly reduces the computational overheads of the video-streaming server. Each AV capture board can support only one AV input; hence, multiple boards have to be installed.

6.3 Client-Server Software Architecture

Figure 6.3 shows the interaction of the system components and associated information flow. The control servers may run different types of operating systems. In particular, the control server of the fish-processing machine (Figure 6.2) runs Microsoft Windows NT 4.0 with Service Pack 6. In order to facilitate real-time processing and control in this NT-based system, Venturcom's RTX (Real-Time eXtension) software is installed on the control server, allowing direct access to legacy data acquisition boards. This provides high-performance, deterministic, real-time and non-real-time processing within the control server. The web-server runs on

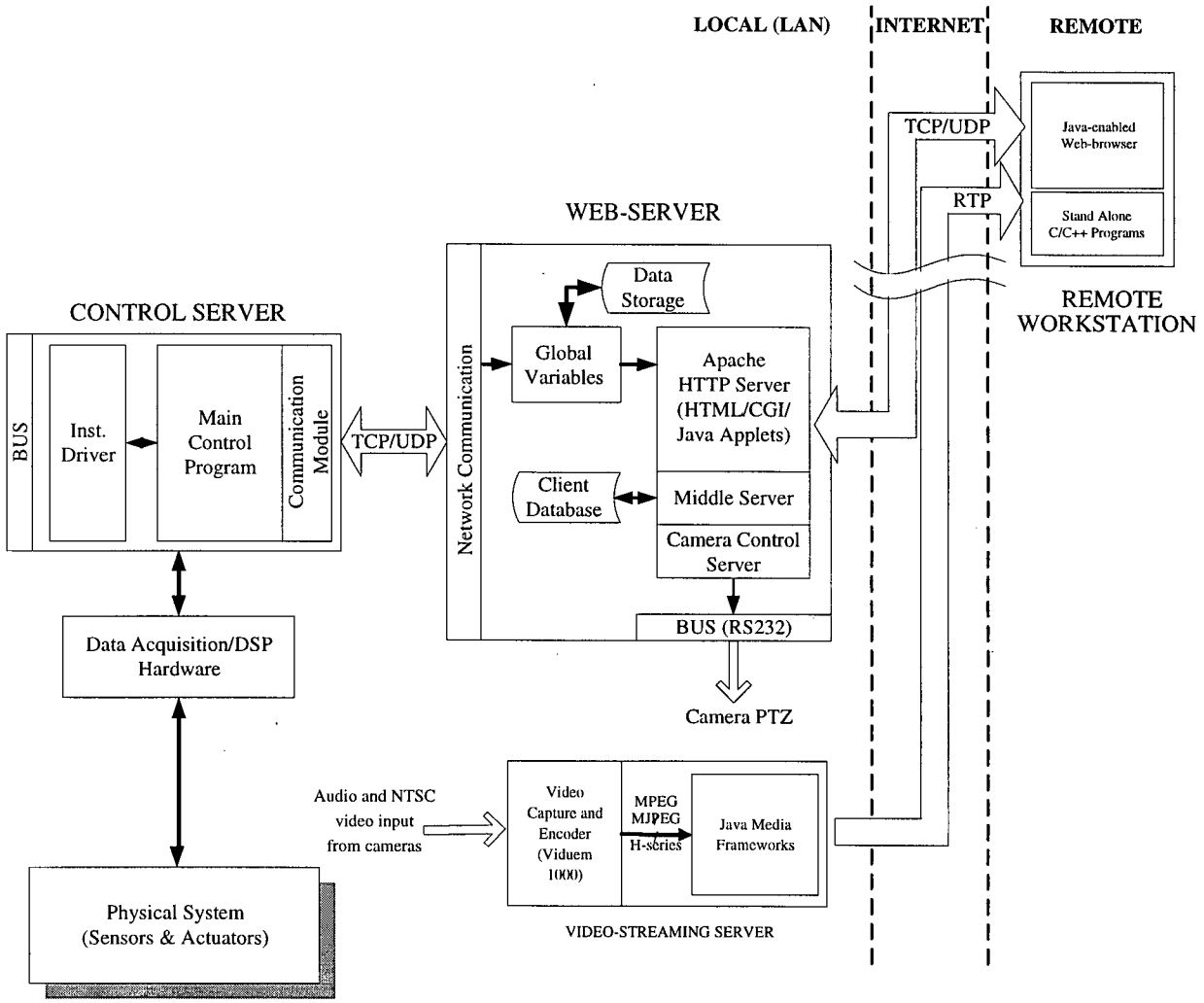


Figure 6.3: System component interaction and information flow.

Microsoft Windows 2000 Professional operating system with open-source Apache HTTP Server supporting Perl/CGI (Common Gateway Interface) and Java™ Applets (Deitel and Deitel, 1999). Microsoft Windows 2000 Professional is used as well on the video-streaming server in view of specific requirements of the AV capture device.

The control server handles data acquisition and low-level control. It sends data and receives commands from the web-server through an Ethernet network. Up-to-date system responses and states are transmitted in real time to the web-server. All external communications with a remote VPS are handled by the web-server. This two-tier structure is similar to the Double Client-Server Structure proposed in (Ko, *et al.*, 2001). Many advantages can be gained by using such a configuration. In particular, since individual web servers running on each

system (on the control server) are not required, this will free up processing power for high bandwidth control computations, thereby increasing the performance of the control server. On the other hand, the web-server requires a significant amount of processor overhead. As indicated earlier, it is possible to run a different operating system on each control server depending on the particular application. The overall security and user accesses can be better managed by using a single web-server allowing only single point access.

As needed depending on the type of service that is requested, both UDP (User Datagram Protocol) and TCP (Transmission Control Protocol) are implemented for client-server communication. TCP guarantees a reliable bi-directional transfer of data. If a data packet is delayed or damaged, it will effectively stop traffic until either the original data packets or the backup data packets arrive. Hence, TCP is used during monitoring and when experiments are performed. However, when real-time controllers are executed remotely, a high rate of data transfer is required without creating extra overheads in the transmission. This is achieved using UDP, which ensures a more consistent sampling rate with reduced fluctuations (Munir and Brook, 2001).

Referring to Figure 6.3, the web-server contains three main programs; specifically, the HTTP server with embedded CGI modules and Java™ Applets; the middle server, which is mainly responsible for scheduling the communication data between the users and the control servers; and the camera control server, which moves the physical orientation of the cameras according the user commands.

6.3.1 The HTTP Server and User Authentication

The HTTP server is the main access portal to all the systems in a particular facility. It provides the main webpage for the user to log on to the facility and then to the particular system of interest. User authentication is of utmost importance in a typical establishment in order to ensure that only authorized personnel can access a specific equipment or system. Different levels of access have to be available because different modes of operation are required. The lowest level of access would correspond to a guest who is only allowed to view the live AV streams without access to the functionality of any laboratory/system equipment. The second level of access is for normal users who are allowed to conduct manually preprogrammed experiments. Only one user is allowed to conduct an experiment at a given time, and the other users who are logged on to the same experiment will only be able to monitor

the system data, and initiate discussions on the white board. A higher level of security access is required for implementing new control laws either locally or remotely. In the present setup, user access is controlled by utilizing CGI user authentication scripts (uncompiled programs) written in Perl scripting language. A user is required to log on with a pre-registered username and password. This is cross-checked with the user database and the proper level of access is granted to the requesting user. The corresponding user interface in the form of Java™ applets (see Section 6.3.4) is then transferred to the user's web-browser (VPS) for display and subsequent interaction. The entire history of user access is logged for purposes of activity tracking. Information on active users, identified through their unique Internet Protocol ID, with their corresponding levels of access, is particularly stored for usage scheduling and data flow management by the middle server, as discussed in the next section.

CGI scripting is also used for allowing the users to upload specific files to the server, which will then be forwarded to the intended control server for execution. These files will include such information as, details of the system configuration, desired trajectories that have be tracked by the system, and control laws such as the rule base of a fuzzy controller. Clearly, these files have to be in a pre-specified format or protocol so that the intended system will not be damaged due to incorrect reading of the details in the files. In order to safeguard against improper file generation by the user, a file evaluation parser is incorporated to validate incoming files.

6.3.2 User Scheduling and Data Flow Management

The middle server is implemented in modules with each module dedicated to a specific system. It is programmed in C++ programming language using a standard networking API (Application Program Interface). The main purpose of the middle server is to act as a strict "mediator" between the user (Java™ applet interface) and a system (control server) unifying

Table 6.1: The types of messages for communication between the middle server and the control server.

Message ID	Description
11	Middle server requests parameter changes
12	Middle server requests main state changes
13	Middle server requests control server to start sending data

access control and keeping unnecessary computational load from the control server. A list of active users is kept in the middle server. Since only one authorized user is allowed to change the parameters of a system at a given time, the middle server is responsible for granting this access and keeping track of the duration for which the particular user is allowed to be in control. The middle server also manages the data flow between the user and the systems. Table 6.1 gives the three basic message types with their corresponding message identification string between the middle server and the control server. In the communication between the middle server and the control server, the middle server acts as a client. No authentication is required for the message exchange between these two servers. This is true because both servers are fully under the control of the designer, and the control server only accepts data from the middle server, which resides on a unique IP and port number. In addition, the message structure is kept confidential (see Section 6.4 for the custom message structure used in the fish-processing machine.) The details of the message types of Table 6.1 are given below:

- ***Middle server requests parameter changes (ID-11)***. When the message receiver at the control server receives this request from the middle server, it updates with these new values the local variables that are stored in a shared memory area. These parameters include the various parameters/gains of the implemented controllers, reference trajectories, and the instantaneous set points, should the system be manipulated manually by the user.
- ***Middle server requests main state changes (ID-12)***. Upon receiving this request from the middle server, the control server is triggered to change the main states of the system. Some of the more common states among different systems are the main power to devices such as motors, pumps and conveyors (switched through a digital power relay circuitry); start or stop trigger for running some tasks; triggering a reset in counters or axis homing; and triggering of different controller modes.
- ***Middle server requests control server to start sending data (ID-13)***. This request is made as the first user is logged on, when the particular system is first in the idle state. Then the control server starts sending system monitoring data to the middle server at the requested time interval. The data updates will in turn be sent to the appropriate VPS.

The different types of message exchange between the user interface applet (as a client) and the middle server are given in Table 6.2. As the behavior of a user is unpredictable, it is important to impose additional precautions in mediating the communication between a user and

Table 6.2: The types of messages for communication between the middle server and the user interface applet.

Message ID	Description
101	Client requests connection
102	Client acknowledges connection request approval
103	Client requests parameter changes
104	Client requests disconnection
105	Client requests main state changes
106	Client sends keep-alive flag
201	Middle server approves connection request
202	Middle server confirms parameter change
203	Middle server forwards system updates
204	Middle server confirms disconnection from client
205	Middle server assigns *superuser* status to client
401	Middle server denies connection request
402	Middle server denies parameter change request

the system of interest. The main message types implemented in the present work are discussed below:

- ***Client requests connection (ID-101).*** When this request is received by the middle server, it first determines the level of access which the requesting VPS client is granted, by cross checking with the active user list from the CGI user authentication scripts. If the user has the highest level of access and if there is no other active user of the same access level for the particular system, the user is granted the *superuser* status permitting full control of the system. The user is sent a unique identification string for use in subsequent communications and then is put on a temporary list, waiting for acknowledgement. Similar action is taken if there is already a *superuser* connected to the system or if the requesting client has a lower level of access, except a randomly generated unique identification string is sent to the VPS. If the total number of preset clients is already logged on to the system, the requesting client will be rejected with a warning message appearing on the client's VPS.
- ***Client acknowledges connection request approval (ID-102).*** When a client VPS sends

an acknowledgement of its connection request, the middle server first checks the client's identification string and IP with the temporary list to avoid any ill-intentioned client from trying to bypass the initial user authentication step. When the authenticity of a client is established, the client will be taken off the temporary list and added to the active client database. Then, the middle server will request the control server to start sending system monitoring data (with message ID-13) if the control server is not already doing so. The frequency in which the data is updated to the client can be set by the requesting client or be made adaptive according to the current Internet congestion level.

- ***Client requests parameter changes (ID-103).*** Upon receiving this request from the client, the middle server determines if the requesting client has the *superuser* status. If affirmative, the request will be forwarded to the control server with message type ID-11. A similar request from any other user type will be rejected and a request denial message will be sent to the client with the message type ID-401.
- ***Client requests disconnection (ID-104).*** A client disconnecting from the system has to be removed from the client database and its place is made available to a new client. If the disconnecting client is a *superuser* and there exist other user(s) with the same level of access, the "oldest" user will be promoted to the *superuser* status and sent the unique *superuser* identification string triggering the user's VPS to enable system manipulation interfaces. The client database in the middle server is then updated accordingly. If the requesting client is the only client for the particular system, the middle server will notify the control server to stop sending data updates.
- ***Client requests main state changes (ID-105).*** This request precedes the message type ID-12 given above. The middle server verifies the user status and if the user has the highest level of access it forwards the request to the control server.
- ***Client sends keep-alive flag (ID-106).*** In order to prevent cases where a client connection is unintentionally lost, for example, due to crashing of the VPS or an unforeseen fault in the client's communication network, the VPS of each client automatically sends a message with a keep-alive flag, to the middle server at a pre-specified interval. A countdown timer is maintained for each active client connection in the middle server. Each countdown timer will be reset to a preset value (1 minute in the present implementation) upon receiving a keep-alive message from the corresponding client. The client will be removed from the client database if the corresponding timer lapses. There is

also the possibility of a user becoming idle over a long period of time. This hogs the available connection space as well as the data transmission bandwidth. This can be avoided by implementing a countdown timer at the VPS and requiring the user to depress a refresh button to maintain an uninterrupted session.

- ***Middle server approves connection request (ID-201)***. This message is sent by the middle server upon receiving a positive client request (through message ID-101). In return, the client acknowledges the approval in this tried-and-true three way handshake. The system manipulation interface on the client's VPS is enabled if it is granted the *superuser* status.
- ***Middle server confirms parameter changes (ID-202)***. An authorized user invoking changes in the system parameters is given confirmation, once accomplished, by the middle server in the form of visual feedback on the status window in the user's VPS.
- ***Middle server forwards system updates (ID-203)***. Upon receiving periodic data updates from a particular system, the middle server forwards the data to the corresponding set of active clients. When the client's VPS receives this message, the textual and graphical interfaces are updated accordingly.
- ***Middle server confirms disconnection from client (ID-204)***. Whenever a client "properly" disconnects from a system, whether manually or automatically through the Java™ applet destruction sequence, a disconnection request is sent to the middle server. After receiving the confirmation, the user's VPS applet will be safely terminated, particularly freeing up memory and destroying all opened network sockets.
- ***Middle server assigns superuser status to client (ID-205)***. When a client is notified that its usage status has been upgraded to *superuser*, the client's VPS will automatically store the newly received identification string for subsequent communication and will enable the system manipulation interface.
- ***Middle server denies connection request (ID-401)***. This message notifies a requesting user to try again later because the server is busy or the user quota has been filled.
- ***Middle server denies parameter change request (ID-402)***. Users without the appropriate access level are denied their request to change the parameters of a system.

The message types mentioned above are embedded within two fixed message structures for communication between the users' VPS and the middle server, and between the middle server and the control server of a particular system. These message structures are formed differently with different packet sizes depending on the configuration of a particular system. The message

structures implemented for the fish-processing machine are discussed in Section 6.4.

6.3.3 Camera Control Server

For security purposes, Java™ applet technology prohibits a client applet from communicating with any network device other than the network device (web-server) which the applet is retrieved from. Consequently, in order to manipulate the video cameras (pan, tilt and zoom) from the VPS sites of the users, the camera control server has to be implemented on the web-server computer, as indicated in Figure 6.3. Coded in the C++ programming language, the camera control server manages camera motion commands as requested by authorized users. As the video cameras are physically connected to the computer through RS-232C communication links (serial ports), the camera control server interprets a client's request and sends the appropriate motion control signal to the intended camera. Table 6.3 summarizes the available commands for camera motion.

Table 6.3: The command strings for camera manipulation.

Command	Description
0x07	Auto focus mode
0x08	Manual focus mode
0x20	Pan camera to the left
0x21	Pan camera to the right
0x22	Tilt camera up
0x23	Tilt camera down
0x24	Move zooming lens to wide
0x25	Move zooming lens to telephoto
0x26	Move focusing lens to far
0x27	Move focusing lens to near
0x39	Move the camera to home position

6.3.4 The User Interface Client

The scalability, portability, and platform independence of Java™ allow remote monitoring and control applications, in the form of applets, to run on any web-browser, eliminating the need to develop custom communicating software (Weaver and Zhang, 1999). The

communication between the web-server and a remote VPS is achieved by using Java™, whose programs are compiled to platform-independent codes, and dispersed on the web-server as Java™ applets. When a user is logged on to the web-server through a VPS, the Java™ applets are automatically loaded into the VPS (client), and a temporary communication socket is created between the VPS and the web-server. This client-server communication is established for the entire session until the user logs out or the network connection is terminated. User interface applets are designed based on the intended operation of a particular system and varies from one system to another. Section 6.4 describes the user interface applet that is implemented on the fish-processing machine.

Stand alone C/C++ programs are also used in some highly specific operation modes of a particular system. Examples include the implementation of remote, time-critical, low-level control algorithm, or custom supervisory controllers. These programs have very specific functions and can only be implemented by administrators of the system who possess a detailed internal knowledge of the developed infrastructure, particularly, the communication protocols, the electrical hardware (sensors, actuators, etc.), and the functional bandwidth of the system.

6.3.5 Audio and Video Feedback

Section 6.2 introduced the use of color cameras and microphones, which are connected to audio-visual (AV) capturing boards. These input devices are installed in the video-streaming server to provide live audio and video feedback to the VPS (see Figure 6.3). Two popular commercial solutions are RealSystem Server Professional by RealNetworks (RealNetworks, Inc.) and Windows Media Services by Microsoft Windows Media Technologies (Microsoft Corp.). However, these two solutions incur a significant delay (approximately 20 seconds) from live to view due to their buffering and intelligent transmission features. Hence, in line with the objective of platform independency, the current implementation has adopted Java™ Media Framework by Sun (Sun Microsystem, Inc.) to provide AV streaming to the Internet. It uses Real-Time Protocol (RTP) to transmit live AV feeds, resulting in minimal latency (live to view in less than 1 second) and low utilization of the transmission bandwidth. Java™ Media Framework can be easily embedded in Java™ applets, and it provides the option to distribute the AV streams through multicasting, unicasting or video conferencing modes. Java™ Media Framework can also work together with Darwin Streaming Server, an open source streaming solution from Apple Computer, Inc., to broadcast AV streams to the Internet.

6.4 A Practical Demonstration

The remote web-based monitoring technology as well, which is developed in the present work, is implemented and tested on the industrial fish-processing machine (see Chapter 2). For this particular 2-axis test system, the communication message structures between the control server and the middle layer are formed using `(unsigned) short` (2 bytes), `(unsigned) long` (4 bytes) and `float` (4 bytes) variable types. The message structure that is transmitted from the control server to the middle server has a size of 120 bytes and is constructed as follows:

```
typedef struct {
  unsigned long  k_sample;      // sample time step
  short          endAll;        // terminate execution flag
  short          controlMode;   // current control mode
  short          powerOn;       // power on-off state
  short          conveyorOn;    // conveyor on-off state
  short          startState;    // task run state
  short          padding;       // data packet byte padding
  float          KpX, KiX, KdX;// PID gains for the X-axis
  float          KpY, KiY, KdY;// PID gains for the Y-axis
  float          ykX, ykY;      // measured position outputs
  float          ymkX, ymkY;    // reference model position outputs
  float          rkX, rkY;      // interpolated quintic trajectory points
  float          rspX, rspY;    // position set-points for each axis
  float          ukX, ukY;      // control input currents
  // Gain settings for a model-referenced fuzzy controller for
  // each axis (see Chapter 7):
  float          MRFCgeX, MRFCgdeX, MRFCguX;
  float          MRFCgeY, MRFCgdeY, MRFCguY;
  float          paX, paY;      // head-side cylinder pressures
  float          pbX, pbY;      // rod-side cylinder pressures
} MESSAGE_FROM_CONTROL_SERVER_TO_MIDDLE_SERVER;
```

The message request structure from the middle server to the control server is 76 bytes in size and is composed of:

```
typedef struct {
  unsigned short msgType;       // message type identifier
  short          startState;    // task run trigger
  short          endALL;        // halt operation trigger
```

```
    short           reset;          // reset trigger
    short           controlMode;    // set control mode
    short           powerOn;        // power on-off trigger
    short           conveyorOn;     // conveyor on-off trigger
    short           manualMode;     // manual manipulation trigger
    short           remoteRequest;// trigger for staring data stream
    short           remoteRequestInterval;// data streaming rate
    float           KpX, KiX, KdX;// PID gains for the X-axis
    float           KpY, KiY, KdY;// PID gains for the Y-axis
    // Gain settings for a model-referenced fuzzy controller for
    // each axis (see Chapter 7):
    float           MRFCgeX, MRFCgdeX, MRFCguX;
    float           MRFCgeY, MRFCgdeY, MRFCguY;
    float           rspX, rspY;     // position set-points for each axis
} MESSAGE_FROM_MIDDLE_SERVER_TO_CONTROL_SERVER;
```

For the communication between the middle server and the VPS units of the clients, the message structures contain variables of type `short` and `long` (`int` in Java). Variables from the control server with significant decimal values are scaled accordingly to type `short` to compensate for the different byte ordering methods used in Java™ and C/C++ languages. The size of the message structure for sending information to a remote VPS is 74 bytes. The message structure is formed as follows:

```
typedef struct {
    short           msgType;        // message type identifier
    short           clientID;       // client identification string
    short           powerOn;        // power on-off state
    short           conveyorOn;     // conveyor on-off state
    short           startState;     // task run state
    short           controlMode;    // current control mode
    short           KpX, KiX, KdX;// PID gains for the X-axis
    short           KpY, KiY, KdY;// PID gains for the Y-axis
    short           rspX, rspY;     // position set points
    short           rkX, rkY;       // current trajectory points
    short           ykX, ykY;       // measured position response
    short           ukX, ukY;       // control input currents
    short           paX, paY;       // head-side cylinder pressures
    short           pbX, pbY;       // rod-side cylinder pressures
    // Gain settings for a model-referenced fuzzy controller for
```

```
    // each axis (see Chapter 7):
    short           MRFCgeX, MRFCgdeX, MRFCguX;
    short           MRFCgeY, MRFCgdeY, MRFCguY;
    // performance attributes (rise-time, settling-time,
    // overshoot, and steady-state error)
    short           perf_rt, perf_st, perf_os, perf_of;
    long            k_sample;     // current sample time step
} MESSAGE_FROM_MIDDLE_SERVER_TO_REMOTE_VPS;
```

The performance attributes for the model-referenced adaptive controllers as will be discussed in the next chapter are computed at the middle server. The message structure that is formed by the VPS for requesting changes through the middle server having 44 bytes each is as follows:

```
typedef struct {
    short           msgType;      // message type identifier
    short           clientID;     // client identification string
    short           powerOn;      // power on-off trigger
    short           conveyorOn;   // conveyor on-off trigger
    short           startState;   // task run trigger
    short           controlMode;  // control mode trigger
    short           manualMode;   // manual manipulation trigger
    short           padding;      // data packet byte padding
    short           rspX, rspY;   // position set-points for each axis
    short           KpX, KiX, KdX;// PID parameters for the X-axis
    short           KpY, KiY, KdY;// PID parameters for the Y-axis
    // Gain settings for a model-referenced fuzzy controller for
    // each axis (see Chapter 7)
    short           MRFCgeX, MRFCgdeX, MRFCguX;
    short           MRFCgeY, MRFCgdeY, MRFCguY;
} MESSAGE_FROM_REMOTE_VPS_TO_MIDDLE_SERVER;
```

Figure 6.4 shows the developed Java™ interface applet on the VPS of the Intelligent Iron Butcher. The buttons for connection to the system and changing the main states of the system; specifically, the power trigger, the conveyor trigger, the task run trigger, and the task stop trigger, are located in the upper region. The left pane displays the performance attributes of the system and the radio buttons for changing the controller modes. Five controller modes are implemented; namely, PID servo, PID under supervision of Model-Referenced Fuzzy Control (MRFC), PID under supervision of Model-Referenced Adaptive Control (MRAFC), intelligent auto-switching of adaptive controllers, and Generalized Predictive Control (GPC). The

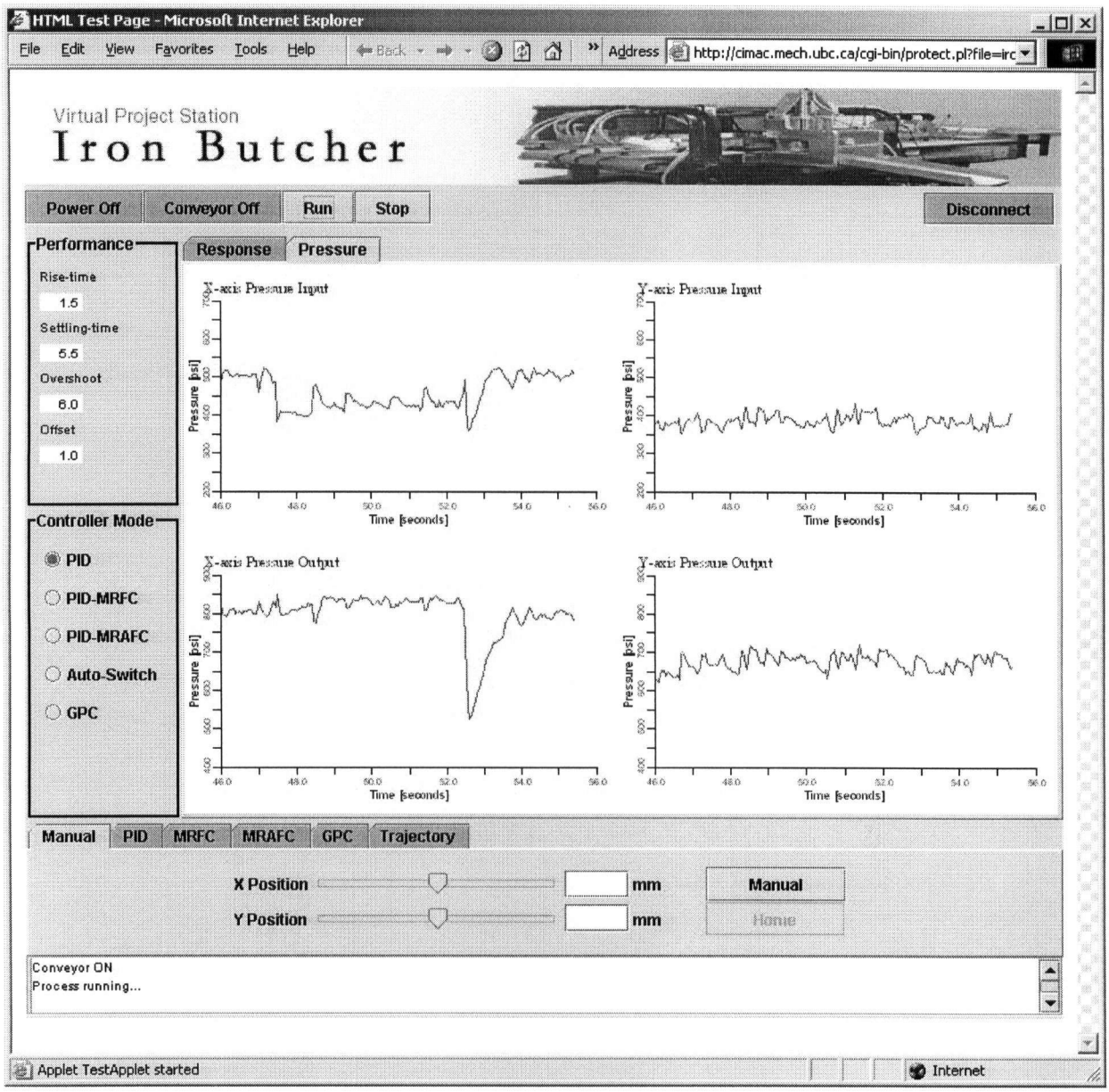


Figure 6.4: User interface applet for the Intelligent Iron Butcher.

development of the remote supervisory controllers is presented in Chapter 7. The center-right region of the applet is reserved for graphical displays. The user will be able to tab between different views such as the implemented position responses and pressure responses. Additional graphical information; for example, the planar motion of the manipulator, or the block diagrams, can be easily added to this region. The lower region contains tabbed panes for manipulating the parameters of the system. They include manually moving the manipulator, changing the parameters of the PID, MRFC, MRAFC, and GPC controllers, and uploading and

running custom trajectories. The lowest region of the interface applets is reserved for providing text response for the user.

6.5 Summary

Practical methodology for developing a network infrastructure for remote web-based monitoring and intelligent supervisory control of a physical system was presented in this chapter. The procedure included the identification of the line of hardware components as required to establish the interconnectivity between a web-server, a video-streaming server, cameras, microphones, control servers, and the physical machineries or systems. The application software, which has been either adopted or developed in house for the multi-level client-server communication structure between various local hardware components and remote virtual project stations, was described. The industrial fish-processing machine described in Chapter 2 was used as an application example, representing the system to be monitored and controlled. Such implementations are equally useful in industrial facilities, research facilities, and academic laboratories for remote testing and experimentation.

Chapter 7
Remote Supervisory Control Systems

The flexibility and modularity of the network architecture, which was developed in the previous chapter, form the rationale for incorporation of a multi-layered intelligent supervisory control structure with the objective of on-line improvement of the performance of a remote plant. The scheme, as developed in the present chapter, integrates a supervisor into the remote plant, and the supervisor employs knowledge-based decision making to continuously monitor the performance of the plant. The performance metrics deduced from observation of the plant response under controlled conditions are then used to infer the best adaptive controller for the plant under an existing condition. A knowledge-based system that incorporates both human expertise and analytical knowledge regarding the plant and the controllers is developed. A client-server supervisory control architecture for networked-assisted controller switching for the remote plant is developed. Switching has to be done in such a manner that the transition from one controller to another takes place in a smooth manner. Proper design of the intelligent switching system is a key to achieving this objective. A full-scale implementation of the developed approach is made on the industrial fish-processing machine (see Chapter 2), which is the experimental platform for this thesis, and is used to demonstrate the application of the developed system in an industrial environment for system monitoring and supervisory control.

In this chapter, first the development of the distributed client-server supervisory control architecture is presented. Specifically, Section 7.1 introduces the intelligent hierarchical control architecture of the industrial fish-processing machine. The newly developed distributed client-server supervisory control architecture of the machine is described in Section 7.2. The subsequent two sections discuss the design procedure used to establish two knowledge-based model-referenced adaptive controllers. An intelligent switching strategy to maintain optimal on-line performance of the system is developed in Section 7.5. Finally, experimental case studies are presented to evaluate the performance of the developed strategy.

7.1 Hierarchical Control Architecture

A networked control configuration can improve the performance, flexibility, and efficiency of an industrial plant or system. The capability of web-based monitoring will allow remote adjustment of the basic operation conditions and parameters (e.g., machine throughput, on-off switching, controller selection, and operation timing) of the system. In view of the network connectivity of the developed web-based monitoring infrastructure (see Chapter 6), particularly for use with a production facility like the fish-processing machine, an intelligent hierarchical structure (de Silva, 1995) as shown in Figure 7.1 can be employed for its monitoring and control. Each level of the system structure can be implemented on one or more networked

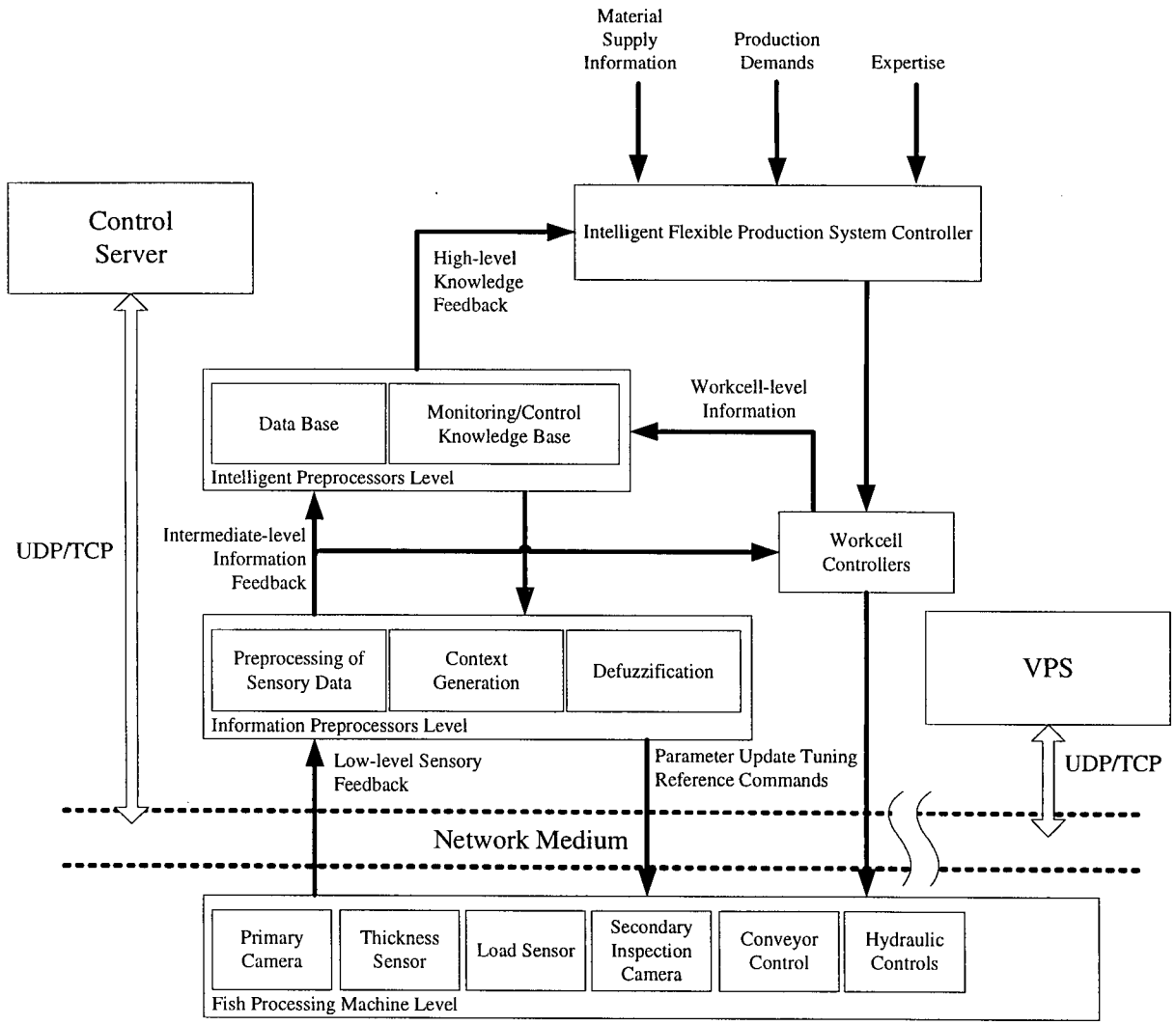


Figure 7.1: An intelligent hierarchical structure for monitoring and control of a plant.

computers, and the communication between different layers may be achieved through an Ethernet backbone. In the present context, each layer is implemented as the server to its succeeding lower layer, and as a client to its preceding upper layer. Referring to Figure 7.1, the component layer of the fish-processing machine, which is the lowest layer, consists of various sensory and actuation components. It also contains low-level feedback controllers for direct or local control using information from the sensory components. The high-resolution information from the sensors is processed at the information preprocessor layer, which is the intermediate layer, to generate context information that can be used by the upper layer (intelligent preprocessor layer). This upper layer includes the workcell controller, which performs higher-level control tasks than those of the component-level controllers; in particular, component coordination, generation of reference signals to drive the components, downloading of component-task programs, and monitoring the workcell components. As well, some intelligent control activity will prove useful in this intermediate layer. The context information for the knowledge system of this layer, for both workcell and the intelligent preprocessor, may come from preprocessed sensory information and also from upper layers and the operator interfaces. The intelligent preprocessor is mainly used for presenting compatible low-resolution information (high-level knowledge) for the upper layers of the control hierarchy. The top layer; i.e., the intelligent flexible production system controller, handles such activities like task planning, procedural decomposition, subtask allocation, and overall system monitoring. Its knowledge system may have to deal with qualitative, incomplete, imprecise, or vague information and may encounter unplanned and unfamiliar situations.

The fish-processing machine layer, the information preprocessor layer, the intelligent preprocessor layer, and the workcell controllers are shown as control servers in figures 6.1 and 6.2. They can be implemented in a single computer or distributed over several computers. The flexible production system controller is represented as the web-server where it has access to all the information gathered from other lower level systems. Using a remote VPS, an authorized operator or a plant manager can gain access to all the information in the production facility, and will be able to input material-supply information, production demands, supplementary expertise for the intelligent preprocessor, and so on, thereby facilitating improved decision making.

In order to take advantage of computationally powerful VPS units for implementing complex control laws, thereby relaxing the computation load of the local control server and also providing the capability of remote intervention by experts, the direct controllers of the

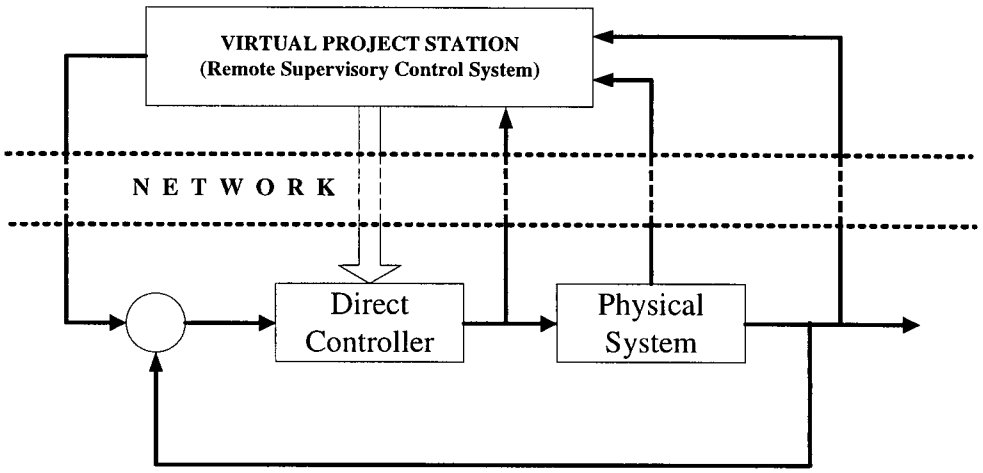


Figure 7.2: Adaptive control of a remote plant through a communication network.

components of the fish processing machine can be executed remotely from a VPS, thus closing the feedback loop through a communication network. The local control server and the web-server are required only to transmit real-time sampled sensory data to the VPS, and receive control signals from the VPS to be passed to the system actuators. This approach has been thoroughly investigated in chapters 2 to 5.

An alternative control approach that may be implemented on the developed network infrastructure consists of high-level adaptive controllers running on the VPS units to supervise or optimize the low-level controllers of the components on line, as shown in Fig. 7.2. A distributed client-server supervisory control architecture has been developed and is presented next.

7.2 Distributed Client-Server Supervisory Control Architecture

The developed system architecture takes advantage of the common availability of an Ethernet backbone in modern industrial settings to implement a remote supervisory control system for real-time monitoring and control of a process such as the fish-processing machine. Figure 7.3 shows the architecture of the developed networked intelligent client-server supervisory control system. Two discrete-time proportional-integral-derivative (PID) controllers are used for servo-level feedback control. The PID controllers implemented here are of the velocity-feedback form (Ogata, 1987), as given by the z-transform relation

$$U(z^{-1}) = -K_P Y_s(z^{-1}) + K_I \left(\frac{R(z^{-1}) - Y_s(z^{-1})}{1 - z^{-1}} \right) - K_D (1 - z^{-1}) Y_s(z^{-1}) \tag{7.1}$$

where, $Y_s(z^{-1})$ is the system output, $U(z^{-1})$ is the control input, $R(z^{-1})$ is the command reference, and K_P, K_I, and K_D are the controller-gain parameters. In order to reduce the

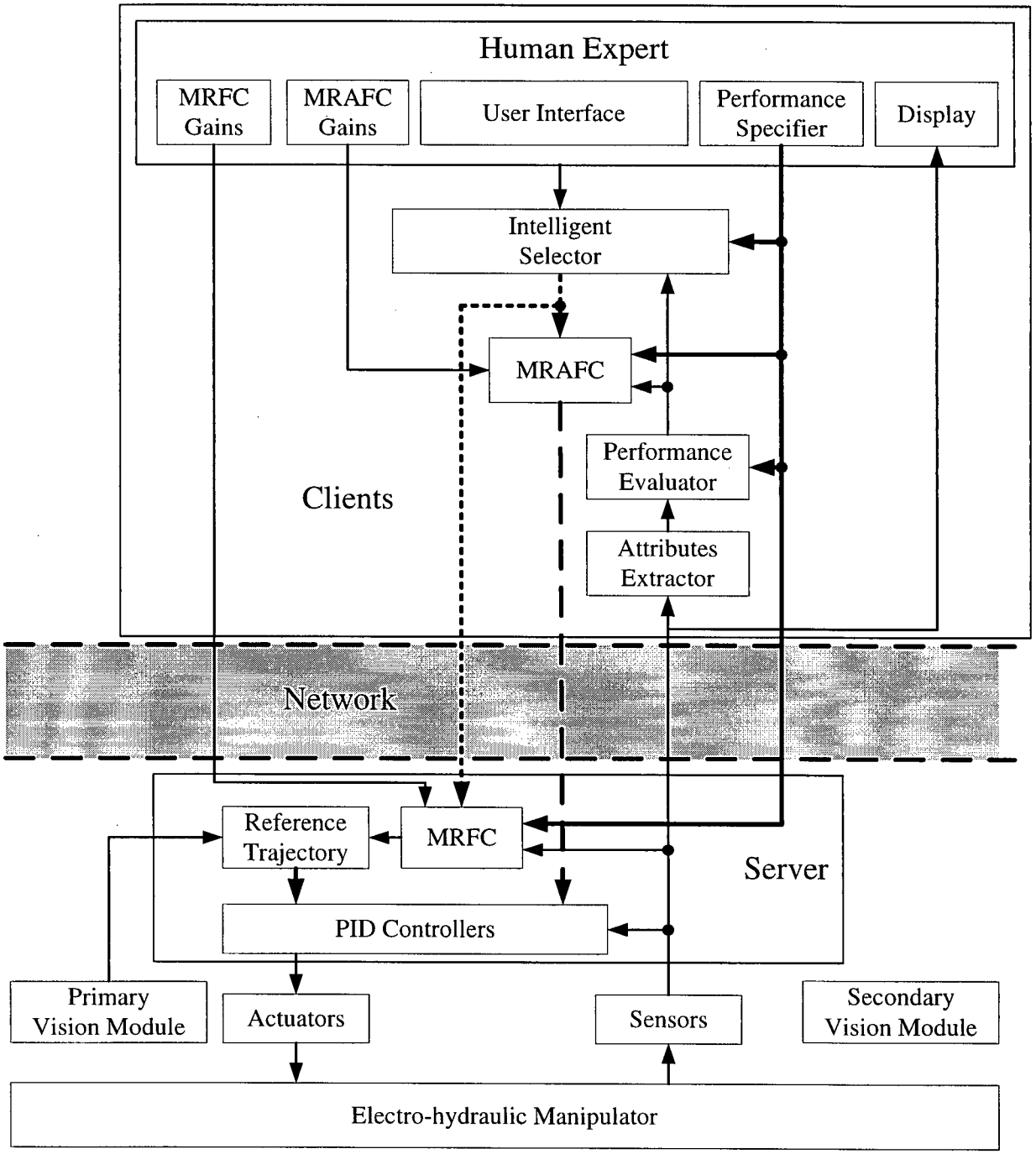


Figure 7.3: The developed architecture for networked intelligent supervisory control.

steady-state hunting effect caused by stick-slip friction, which is typically present in hydraulic actuators, a dead-band compensation term is added to (7.1). It is of the form $-I_a \operatorname{sgn}(y_s - r)$, where $I_a = 2.0$ mA, y_s is the position response, and r is the reference input. The servo control bandwidth is set at 200 Hz, which is sufficient for the relatively slow manipulations that are carried out.

The dynamics of the electro-hydraulic manipulator frequently change due to the nonlinear friction in the sliding carriages and wear and tear of the fluid seals. This causes the servo controllers to run out of tune, thereby degrading the system performance. In order to compensate for such unacceptable dynamics, the servo controllers have to be constantly tuned. This may be achieved by either tuning the reference trajectories that are fed into the servo controller (signal adaptation), or tuning the parameters of the servo controllers (parameter adaptation). Referring to Figure 7.3, in the signal adaptive scheme, a Model-Referenced Fuzzy Controller (MRFC) is implemented locally outside the servo loop. In order to force the response of the manipulator to a desired reference trajectory, the MRFC uses both the instantaneous tracking error and the change in tracking error, to determine the necessary change in the reference command, which is the input to the PID controller. This determination is made by using a fuzzy rule base, which is further elaborated in Section 7.3.

In the knowledge-based parameter adaptive scheme, the Model-Referenced Adaptive Fuzzy Controller (MRAFC), is used to modify the parameters of the PID controller; i.e., K_P, K_I, and K_D. Since MRAFC is situated remotely over an Ethernet network, the computational requirement of the local controllers is eased, and some flexibility is provided to introduce or remove additional features to or from the architecture without having to shutdown the plant. The information used by MRAFC is preprocessed by the "attribute extractor" and the "performance evaluator." The attribute extractor extracts the performance attributes from the position response. Six commonly used step input attributes are adopted here. They are rise-time (95%), settling-time (±2%), maximum overshoot, steady-state offset, damped natural frequency, and damping ratio. In the performance evaluator, these attributes are compared with the operator-specified attributes (specified as a "desired" reference model in the performance specifier) to derive the corresponding six deviation indices, which are then used by MRAFC to deduce the required levels of tuning in the PID parameters. The MRAFC algorithm is further investigated in Section 7.4.

It is found that although MRFC is capable of achieving near perfect trajectory tracking, it works well when the low-level servo controller is only mildly out-of-tune from the desired dynamics (Tang, *et al.*, 2002a, 2002b). On the other hand, while MRAFC is more appropriate for fast and coarse tuning of servo controllers, with the objective of bringing a severely off-tuned controller to converge to a set of pre-specified performance characteristics, it is unable to achieve perfect trajectory tracking. A compromise would be to make use of both MRFC and MRAFC in the supervisory architecture, switching them on and off under appropriate conditions. In the present work, an intelligent selector is developed to switch between MRFC and MRAFC, or to turn both of them off, depending the performance of the servo loop. This intelligent selector uses similar information as MRAFC in its decision making scheme, as discussed in Section 7.5.

The intelligent supervisory control scheme is implemented as a client-server architecture. The servo controller, the trajectory generator, and the MRFC are implemented as the server with listening ports for connecting incoming remote clients. The intelligent selector, the MRAFC, the attribute extractor, the performance evaluator, and the human expert interface are implemented as remote client modules. In normal operation, the automated server can run servo controllers without intervention from any remote clients. The plant operator has the option of engaging the remote clients at anytime. System responses as well as critical system states; e.g., performance metrics and production throughput, are readily accessible though an on-line display interface. Basic operator interventions include manually switching between MRFC, MRAFC, and all-off modes, or engaging the intelligent selector back to automatic operation.

As discussed in the following sections, the inputs and outputs of both MRFC and MRAFC must be normalized or scaled appropriately using pre-specified gains. Additional autonomous supervisors can be incorporated to adaptively modify these gains under real-time operation. However, in the present study, the gains are selected intuitively by the human expert and conditionally scheduled, in order to keep the focus on remote intelligent switching of the adaptive schemes. The performance specifier module allows the operator to set the desired performance attributes and the parameters of the reference model. Here, the reference model is a damped simple oscillator with damping ratio ζ and natural frequency ω_n, as given by the transfer function:

$$\frac{Y_m(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{7.2}$$

where $Y_m(\cdot)$ is the output of the reference model. For implementation, (7.2) can easily be transformed into the equivalent difference equation. The sampling rate for MRFC is selected to be equal to the servo loop rate, while data are sent to the clients at a rate of 50 Hz. Under the run state, the MRAFC updates the PID parameters once for every full step response that is detected. Similarly, the intelligent selector runs whenever a new step input is detected.

When using the UDP transmission protocol, the issues that may affect the system performance include the data transmission delay, mis-synchronization of data, and data losses. Due to the nature of the current supervisory scheme, these issues are not critical, particularly since data are streamed at a relatively low frequency (50 Hz), and the attribute extractor needs only to acquire information from a series of buffered data (response history) before making that information available to the MRAFC, the intelligent selector, or the human expert. Switching commands and parameter updating are done at even a slower rate. In this manner, the stability of the servo loops can be maintained.

7.3 Model-Referenced Fuzzy Control

The Model-Referenced Fuzzy Control (MRFC) is an extension to the technique proposed in (Lian, 1997). This is a signal adaptation scheme, which works by modifying the reference command into a servo controller according to the tracking error and the change in the tracking error. The tracking error is taken as the difference between the response of the reference model and the actual position response of the system; i.e.,

$$\varepsilon(k) = y_m(k) - y_s(k) \tag{7.3}$$

The change in the tracking error is the difference in the tracking error between the current and the previous time steps, as given by:

$$\Delta\varepsilon(k) = \frac{\varepsilon(k) - \varepsilon(k-1)}{T_s} \tag{7.4}$$

The basic structure of the MRFC control scheme is illustrated in Figure 7.4. It should be noted that this MRFC technique is independent of the type of direct digital controller that is used in controlling the system, as no parameter adaptation is needed.

The two measurements $\varepsilon(k)$ and $\Delta\varepsilon(k)$ are used as the antecedent variables for the fuzzy inference engine. The antecedents are multiplied by appropriate gains so as to fall in the range between −1 and 1. This normalization is done in order to facilitate the design of the fuzzy inference engine. The gain values of the tracking error and the change in the tracking error are chosen as 1.5 and 0.1, respectively. The correction term for the reference input signal, or the change in the reference command, forms the fuzzy consequent variable.

The fuzzy rule base is designed by carefully evaluating a step response trajectory in the case when the MRFC is expected to behave properly. A unique pattern can be established by

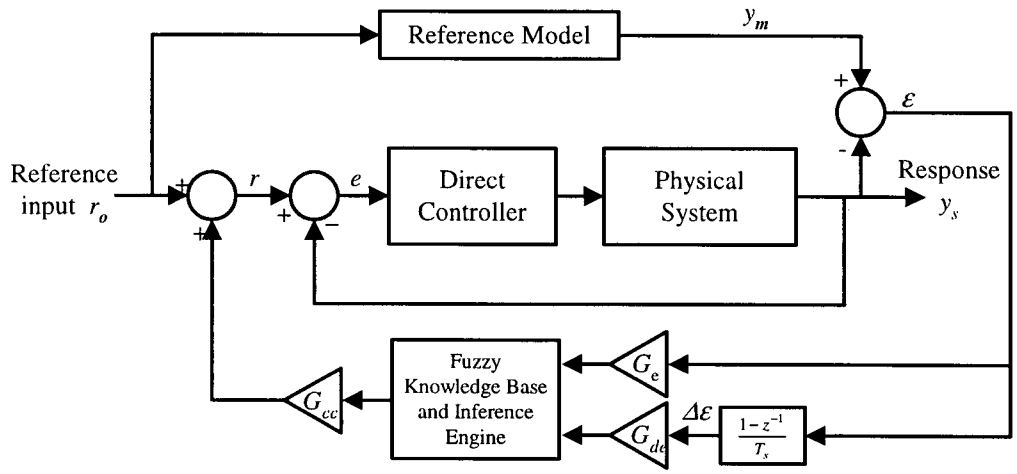


Figure 7.4: The basic control structure of the MRFC.

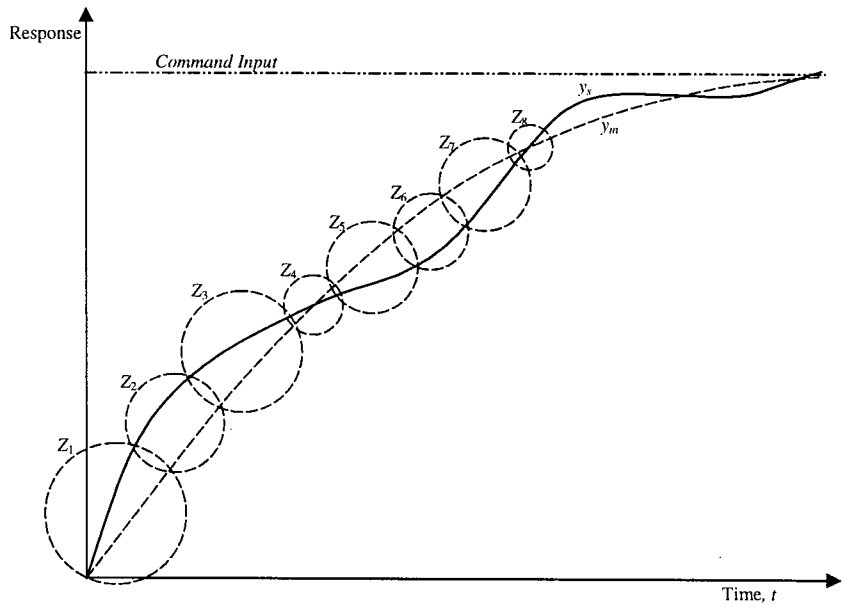


Figure 7.5: The typical response profile of reference model tracking.

examining how the polarities and the magnitudes of the tracking error and the change in the tracking error vary along the transient response of the system under control. A typical response characteristic for a step input is sketched in Figure 7.5. The response profile in Figure 7.5 is separated into eight zones which are denoted by Z_1, Z_2, Z_3, ... and Z_8. Note that these zones follow a cyclic pattern with decreasing area (between the response profiles of the system and the reference model) until the steady-state is reached. The polarities of the tracking error and its rate of change are found to vary from zone to zone. They are categorized in Table 7.1. The control objective is to ideally reach the null value for both fuzzy antecedent variables. In Table 7.1, the system response approaches perfect tracking of the reference model in clockwise cycles ($Z_1 \rightarrow Z_2 \rightarrow Z_3 \rightarrow \ldots \rightarrow Z_8$). From experience and control knowledge, one can determine proper corrective actions for various reference zones. A simple corrective set of rules is given in Table 7.2.

The fuzzy resolution (i.e., the number of fuzzy states used) of the corrective action can be improved to generate a more detailed fuzzy rule base, resulting in a smoother control action.

Table 7.1: Zone polarities of the model tracking response.

	Change in Tracking Error, $\Delta\varepsilon$		
Error, ε	Negative	Near Zero	Positive
Negative	Z_1	Z_2	Z_3
Near Zero	Z_8	Target	Z_4
Positive	Z_7	Z_6	Z_5

Table 7.2: Anticipated corrective action corresponding to each zone.

Reference Zone	**Corrective Action**
Z_1	Reduce control signal by a large value
Z_2	Reduce control signal moderately
Z_3	Reduce control signal slightly
Z_4	Increase control signal slightly
Z_5	Increase control signal moderately
Z_6	Increase control signal slightly
Z_7	No change in control signal
Z_8	Reduce control signal slightly

Each of the two antecedent variables assumes 7 fuzzy states: NH (Negative High), NM (Negative Medium), NL (Negative Low), ZE (Zero), PL (Positive Low), PM (Positive Medium), and PH (Positive High), represented by triangular membership functions symmetrically spaced along the normalized universe of discourse. Figure 7.6 shows the membership functions of the antecedent variables as used in the present study, where the universe of discourse is normalized between −1 and 1. The fuzzy consequent variable also assumes 7 fuzzy states with triangular membership functions, and is also normalized between −1 and 1, as illustrated in Figure 7.7. This fuzzy output is scaled appropriately so that it is smooth and sufficiently significant with the ability to alter the course of the actuators while avoiding any chattering in the response. A scaling constant of 12 provides a good overall performance in the current application. Since both antecedent variables have 7 fuzzy states each, there will be a maximum total of 49 fuzzy rules. These rules are given in Table 7.3. In order to reduce the real-time computational load, the MRFC rule base is transformed into a look-up table. The antecedent variables are divided into a resolution of 0.05. A fast bi-section algorithm is used to search the table entries, and a bilinear interpolation is used to yield the fuzzy output.

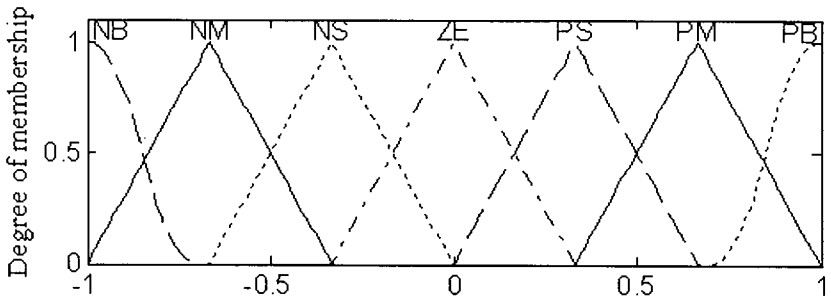


Figure 7.6: Membership functions of the antecedent variables for MRFC.

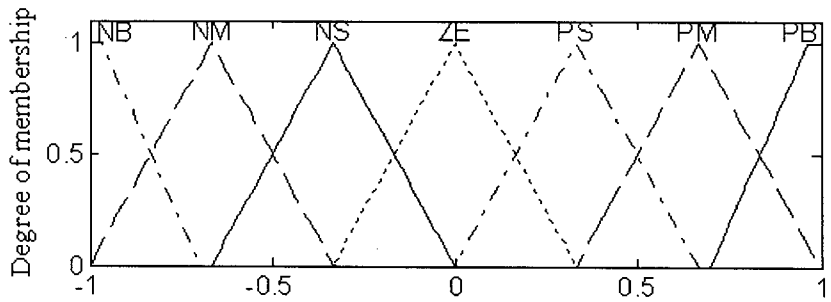


Figure 7.7: Membership functions of the consequent variable for MRFC.

Table 7.3: The MRFC rule base.

	Change in Tracking Error						
Error	NH	NM	NL	ZE	PL	PM	PH
NH	*NH*	*NH*	*NH*	*NH*	*NM*	*NL*	*ZE*
NM	*NH*	*NH*	*NH*	*NM*	*NL*	*ZE*	*PL*
NL	*NH*	*NH*	*NM*	*NL*	*ZE*	*PL*	*PM*
ZE	*NH*	*NM*	*NL*	*ZE*	*PL*	*PM*	*PH*
PL	*NM*	*NL*	*ZE*	*PL*	*PM*	*PH*	*PH*
PM	*NL*	*ZE*	*PL*	*PM*	*PH*	*PH*	*PH*
PH	*ZE*	*PL*	*PM*	*PH*	*PH*	*PH*	*PH*

7.4 Model-Referenced Adaptive Fuzzy Control

The remote Model-Referenced Adaptive Fuzzy Control scheme uses the indices of deviation from the performance evaluator and the reference model from the performance specifier to make tuning decision on the parameters of the local servo controllers. The attribute extractor (see Figure 7.3) is used for extracting the performance attributes from the responses of both the system and the reference model, when subjected to the same input. Six basic control engineering performance attributes are extracted using simple numerical search algorithms, which involve finding the points of zero crossing of the first derivative of a step response. These attributes are rise time (95%), settling time (±2%), maximum overshoot, steady-state offset, damped natural frequency, and damping ratio. It is convenient to choose these performance attributes not only because they make the rule base for adaptation easy to develop, but also because of the fact that they are well known to control engineers. The rise time (t_r) is taken to be the time it takes for the response to reach 95% of the final steady state value, for the very first time. The setting time (t_s) is taken to be the time required for the step response to settle within ±2% of its final value. The maximum overshoot (M_p) is computed as the difference between first peak value of the step response (at the point of first zero crossing of the first derivative of the step response) and the steady-state value, expressed as a fraction of the latter. The offset is the difference between the response at steady-state and the desired value, non-dimensionalized with respect to the desired value. The damped natural frequency (ω_d) is

computed using $\omega_d = \frac{\pi}{t_p}$ where t_p is the time at the first peak of the response. The damping ratio (ζ) is determined according to the formula $\zeta = \sqrt{\frac{\ln^2 M_p}{\pi^2 + \ln^2 M_p}}$, which uses the well-known relationship between the damping ratio and the fractional overshoot, based on the step response of a simple oscillator. It should be noted that the process of extracting the performance attributes involves locating the maxima and minima of a response profile, which is susceptible to measurement noise and local disturbances. This can be overcome by filtering the stream of feedback data.

The performance evaluator, in Figure 7.3, is used for computing the index of deviation (degree of deviation) of the performance of the system with respect to the reference model. The index is computed by taking the ratio of the features of the two responses as follows. If the adaptation goal is to make the attribute value of the system small, then we use: Deviation_Index = 1 – (Model_Feature/System_Feature), and if the goal is to make the feature value large, we use: Deviation_Index = 1 – (System_Feature/Model_Feature). This gives a normalized range between 0 and 1 for the index. Close agreement between the reference model and the system corresponds to a small positive index of deviation. The larger the value of the index of deviation, the poorer the tracking performance. If the index value is negative, this corresponds to better than desired performance, and is "better than the specification" and denoted as an "over-specification." For some performance attributes, a check has to be made to determine if the system value is zero, in order to avoid division by zero. If the check is positive, then an "over-specification" is assigned with a default negative value for the particular index. Table 7.4 gives the six indices of deviation. These can be taken to represent the experience of an expert.

The fuzzy rule base that is employed here has the indices of deviation as its antecedent variables. The numerical values of the indices of deviation are fuzzified into membership functions of five different fuzzy states, and are represented by OV (Over-specified), IN (In-specification), MG (Marginal), PR (Poor), and VP (Very Poor). Membership functions of IN, MG, and PR are triangular shaped while OV and VP are asymmetric polynomial curves. Figure 7.8 illustrates an example of the membership functions used in the present application.

Table 7.4: The six indices of deviation.

Performance attributes	Notation used in rule base	Index of deviation
Rise-time (95%)	RT	$RT = 1 - \frac{RT_m}{RT_s}$
Settling-time (±2% criterion)	ST	$ST = 1 - \frac{ST_m}{ST_s}$
Maximum overshoot	OS	$OS = 1 - \frac{OS_m}{OS_s}$
Steady-state offset	OF	$OF = 1 - \frac{OF_m}{OF_s}$
Damped natural frequency	WD	$WD = 1 - \frac{WD_s}{WD_m}$
Damping ratio	DR	$DR = 1 - \frac{DR_s}{DR_m}$

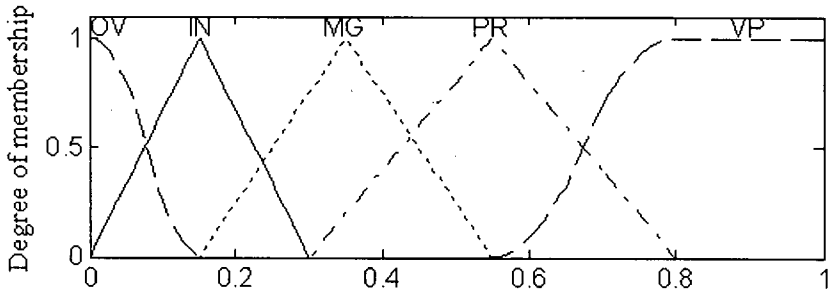


Figure 7.8: Typical membership functions used to represent the index of deviation for MRAFC.

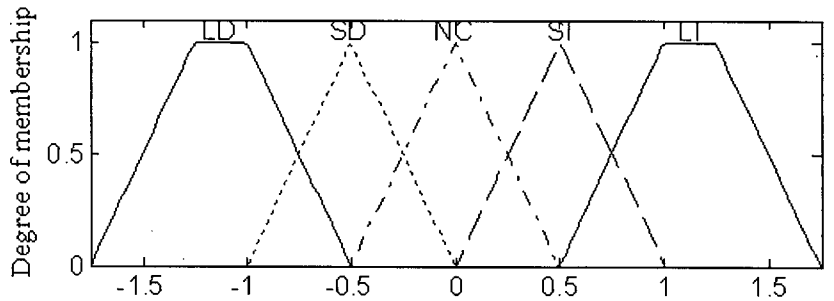


Figure 7.9: Typical membership functions used for the consequent variables for MRAFC.

The incremental change in the adjustable parameters K_P, K_I, and K_D of the discrete PID controller are used as the consequent (action) variables of the fuzzy inference engine. Each action variable has 5 possible fuzzy states over a normalized universe of discourse between –1 and 1. They are symbolically represented by LD (Large Decrement), SD (Small Decrement), NC (No Change), SI (Small Increment), and LI (Large Increment). Figure 7.9 shows the five

membership functions that are used for the consequent variables. The three intermediate membership functions are of triangular shape, while the two end membership functions are of trapezoidal shape. Trapezoidal shaped membership functions are used to provide a stronger tuning action when the deviation between the desired and the actual responses is high. The parameter ranges of the PID controller are not identical. Specifically, K_P ranges from 0 to 10, K_I ranges from 0 to 1, and K_D ranges from 0 to 80. Hence, the normalized outputs of the fuzzy inference engine should be scaled accordingly. Depending on the required sensitivity or the rate of adaptation, each consequent variable can be multiplied by a constant gain before adding to the actual value. In the present study, the gains are made to be linearly proportional to the Integral Time Absolute Error (ITAE) between the desired and the actual step responses. This will allow smoother and rapid convergence of the parameters of the PID controller. In other words, a large tuning action is required when the controller is severely out-of-tune, causing a large deviation in the actual response of the system from the desired one. On the other hand, when the controller is moderately out-of-tune, then just minor adjustments of the tuning parameters would suffice. In order to avoid "over-tuning," a neutral zone is specified, where the deviation between the desired and the actual responses of the system is negligible, from the viewpoint of the type of operation of the system.

The fuzzy rule base for MRAFC is derived by observing the characteristic mapping between the antecedent and consequent variables. The trend of each antecedent variable (performance attribute) is observed by systematically varying the particular consequent variable while keeping the others fixed. Figure 7.10 shows the response profiles for various levels of K_P while K_I and K_D are kept constant. When K_P is increased, there occur distinct increments in rise-time and damping ratio, and decrements in maximum overshoot and damped natural frequency. However, settling-time and steady-state offset do not have a polarized behavior in this case. Hence, their effect is ignored in the rule-base. For the other two consequent variables, the trends are obtained in a similar manner. With six indices of deviation, a total of 30 linguistic rules are used in remote MRAFC tuning of the servo controllers. The resulting rule base is given in Table 7.5. Note that the suffix "D" in each parameter of the controller denotes an "incremental change." The composition rule of inference is utilized for decision making (inference) in the present study (de Silva, 1995). The developed rule base is realistic, accurate and uncoupled, and these characteristics can significantly improve the

computational efficiency of the scheme (de Silva, 1991).

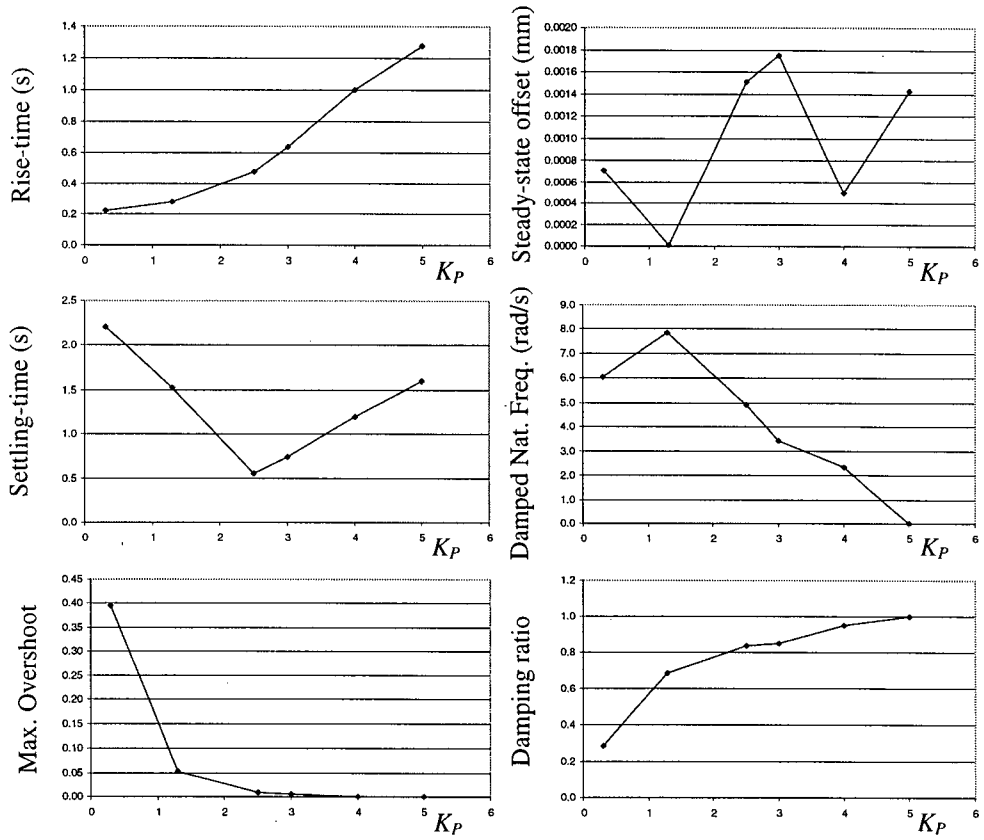


Figure 7.10: Response profiles for various levels of K_P.

Table 7.5: The rule base for the MRAFC.

	Change in controller parameters		
	DKP	**DKI**	**DKD**
RT = VP	LD	LI	LI
RT = PR	SD	LI	SI
RT = MG	NC	SI	SI
RT = IN	NC	NC	NC
RT = OV	SI	LD	LD
ST = VP	-	LI	LI
ST = PR	-	SI	SI
ST = MG	-	SI	SI
ST = IN	-	NC	NC
ST = OV	-	SD	LD
OS = VP	LI	SD	LD
OS = PR	SI	NC	LD
OS = MG	NC	NC	SD
OS = IN	NC	NC	NC
OS = OV	SD	SI	SI
OF = VP	SI	LI	-
OF = PR	NC	SI	-
OF = MG	NC	NC	-
OF = IN	NC	NC	-
OF = OV	NC	SD	-
WD = VP	LD	SD	SI
WD = PR	LD	SD	SI
WD = MG	SD	SD	SI
WD = IN	NC	NC	NC
WD = OV	LI	SI	LD
DR = VP	LI	LD	LI
DR = PR	SI	SD	SI
DR = MG	NC	SD	SI
DR = IN	NC	NC	NC
DR = OV	LD	SI	LD

7.5 Intelligent Switching of Adaptive Controllers

An intelligent selector is incorporated into the distributed supervisory control structure to autonomously choose the best adaptive controller under a given performance condition. Its objective is to ensure that an acceptable performance of the system is maintained throughout the entire course of operation of the electro-hydraulic manipulator, particularly resulting in low tracking error and reduced control energy.

Fuzzy logic is used for generating the decisions of intelligent switching. With the option of switching between MRFC, MRAFC, or keeping both off, the action variables of the rule base are selected to have the three states: ALL-OFF, MRFC, and MRAFC. Here, the Takagi-

Sugeno-Kang (Sugeno, 1985) method of fuzzy inference is used where the implication method is multiplication. The action variables have a constant membership function with 0 for ALL-OFF, 1 for MRFC, and 2 for MRAFC. This allows smooth switching between the three states.

The antecedent variables for the rule base are selected to be the six indices of deviation: RT, ST, OS, OF, WD, and DR, computed by the performance evaluator (see Figure 7.3). Similar to their representation in MRAFC, the indices of deviation are fuzzified into membership functions of five primary fuzzy states: OV (Over-specified), IN (In-specification), MG (Marginal), PR (Poor), and VP (Very Poor). The membership functions are arranged as in Figure 7.11.

In the development of the rule base, priorities have to be established for each index of deviation and it should be known under what circumstances a particular adaptive controller is best suited. When the indices of deviation are in-specification, neither the MRFC nor the MRAFC is required. When RT, OF, WD or DR is over-specified, MRAFC is switched on to tune the servo controller. MRAFC is not required in the case of over-specification of the maximum overshoot index; i.e., with an overdamped response, which can be corrected by MRFC. It should be noted that steady-state offset cannot be corrected by MRFC. Then, retuning of the servo controllers would be required. The 30 linguistic rules developed for switching the adaptive controllers in the present application are given in Table 7.6.

Using the weighted average defuzzification technique, the crisp output of the intelligent selector is set to lie between 0 and 2. All adaptive controllers are switched off when the fuzzy output is lower than 0.8. MRFC is switched on between 0.8 and 1.3. MRAFC is switched on when the fuzzy output value is above 1.3. It is cautioned that the system responses under MRFC should not be used for MRAFC tuning, because in that case the performance features cannot be correctly extracted.

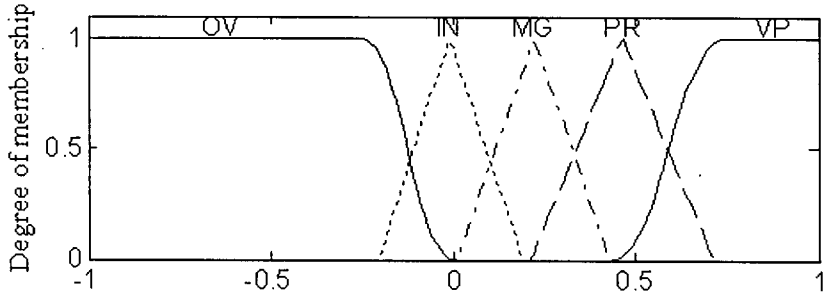


Figure 7.11: The antecedent membership functions for the intelligent adaptive control selector.

Table 7.6: The intelligent selector rule base for adaptive controller switching.

If RT=OV	*then* SWITCH=MRAFC
If RT=IN	*then* SWITCH=ALL-OFF
If RT=MG	*then* SWITCH=MRFC
If RT=PR	*then* SWITCH=MRFC
If RT=VP	*then* SWITCH=MRAFC
If ST=OV	*then* SWITCH=MRFC
If ST=IN	*then* SWITCH=ALL-OFF
If ST=MG	*then* SWITCH=MRFC
If ST=PR	*then* SWITCH=MRFC
If ST=VP	*then* SWITCH=MRAFC
If OS=OV	*then* SWITCH=MRFC
If OS=IN	*then* SWITCH=ALL-OFF
If OS=MG	*then* SWITCH=MRFC
If OS=PR	*then* SWITCH=MRAFC
If OS=VP	*then* SWITCH=MRAFC
If OF=OV	*then* SWITCH=MRAFC
If OF=IN	*then* SWITCH=ALL-OFF
If OF=MG	*then* SWITCH=MRAFC
If OF=PR	*then* SWITCH=MRAFC
If OF=VP	*then* SWITCH=MRAFC
If WD=OV	*then* SWITCH=MRAFC
If WD=IN	*then* SWITCH=ALL-OFF
If WD=MG	*then* SWITCH=MRFC
If WD=PR	*then* SWITCH=MRAFC
If WD=VP	*then* SWITCH=MRAFC
If DR=OV	*then* SWITCH=MRAFC
If DR=IN	*then* SWITCH=ALL-OFF
If DR=MG	*then* SWITCH=MRFC
If DR=PR	*then* SWITCH=MRAFC
If DR=VP	*then* SWITCH=MRAFC

7.6 Experimental Case Studies

An experimental investigation is carried out on the electro-hydraulic manipulator of the Intelligent Iron Butcher (see Chapter 2) in order to evaluate the performance under controller switching as provided by the intelligent supervisory system. Four representative results are given here. In the present case, the parameters of the reference model are set such that the desired performance attributes are approximately given as in Table 7.7, corresponding to $\omega_n = 5.0$ rad/s and $\zeta = 0.91$. In order to simulate the change in system dynamics, the parameters of the PID servo controller are deliberately changed. The case studies are conducted with initial PID settings of $K_P = 4.3$, $K_I = 0.077$ and $K_D = 2.2$, which provide good tracking.

Table 7.7: Desired performance attributes.

Performance Attributes	Desired Value
Rise time [s]	0.60
Settling time [s]	0.80
Maximum overshoot [%]	0.20
Steady-state offset [mm]	0.0005
Damped natural frequency [rad/s]	4.0
Damping ratio	0.90

At the 7,500th time step the parameters are off tuned at $K_P = 0.1$, $K_I = 0.08$ and $K_D = 30.0$, to generate an oscillatory response. Finally, at the 16,500th time step, the parameters are changed to $K_P = 4.0$, $K_I = 0.02$ and $K_D = 10.0$, which results in an over-damped response. Since the dynamics of the two axes of the electro-hydraulic manipulator are similar, only the responses of the X-axis are reported here. Each axis is subjected to a series of square wave input of amplitude 5.0 mm and a cyclic period of 10 seconds. This closely resembles the reference input provided by the primary vision module during the normal operation of the machine, as indicated in Section 2.4.1. It has been shown in (Tang, *et al.*, 2002b) that MRFC and MRAFC are able to cope with any random form of step commands, and not necessarily limited to a square wave excitation, during the tuning cycle.

Figure 7.12 shows the system response when both adaptive controllers are switched off with only the local PID servo controller running. Figure 7.12(b) shows the position response of the reference model and the actual measured position of the manipulator. The tracking error between the responses of the reference model and the system is illustrated in Figure 7.12(c). The control input generated by the servo controller is shown in Figure 7.12(d). Note that the steady-state fluctuations are caused by the deadband compensator, which is incorporated to overcome the stick-slip friction in the hydraulic slip rings and in the slider guideways. These fluctuations are not noticeable in Figure 7.12(b). The response curves serve as a reference for benchmarking of the adaptive controllers.

The system response under constant MRAFC adaptation is shown in Figure 7.13. It is observed that the adaptive controller is able to quickly tune the PID parameters under dynamic variations, and bring the system to closely track the reference model. Figures 7.13(e)-7.13(g)

show that the three PID parameters converge to the correct values under both under-damped and over-damped offset conditions.

Figure 7.14 illustrates the response of the system when the MRFC reference command adaptation is "always on." No tuning is carried out on the parameters of the local PID controllers. Figure 7.14(b) shows that very good tracking is achieved throughout the experiment and is unaffected by the off-tuning of the PID parameters (see Figures 7.14(e)-7.14(g)). This is clearly reflected in Figure 7.14(c). However, such perfect tracking is only achieved with a drastic increase in control energy, as noted from Figure 7.14(d). Table 7.8 compares the difference in sum square change in control input and tracking error obtained for the four case studies conducted. The total control energy with MRFC is found to be over 6 times larger than that without adaptation (Figure 7.12(d)), and nearly 3 times larger than with MRAFC tuning.

Figure 7.15 shows the response of the system under automatic switching of adaptive controllers. The switching states are indicated in Figure 7.15(a). The intelligent selector is switched on at the end of the first half-cycle. In the middle of the 3rd-cycle, the MRFC is switched on upon detecting degraded tracking, and as a result the tracking error is quickly reduced to almost zero. The intelligent selector turns on the MRAFC upon detecting a large change in the system response dynamics. PID tuning is then carried out to quickly reduce the tracking error up to the middle of 9th-cycle where the intelligent selector decides to switch to MRFC, thereby providing near perfect tracking. This tracking is much closer than what is achievable under fixed MRFC or fixed MRAFC. This is because the MRFC works very accurately only when the servo controller is not significantly off-tuned. The control energy in the present case is nearly 2 times lower than that for constant MRFC.

Table 7.8: Performance comparison among the four different adaptive states.

	$\sum(\Delta u)^2 \times 10^4$ [mA^2]	$\sum \varepsilon^2 \times 10^3$ [mm^2]
PID only	4.638	28.674
PID with MRAFC	8.073	12.024
PID with MRFC	26.854	0.691
PID with Auto-Switching	15.060	9.245

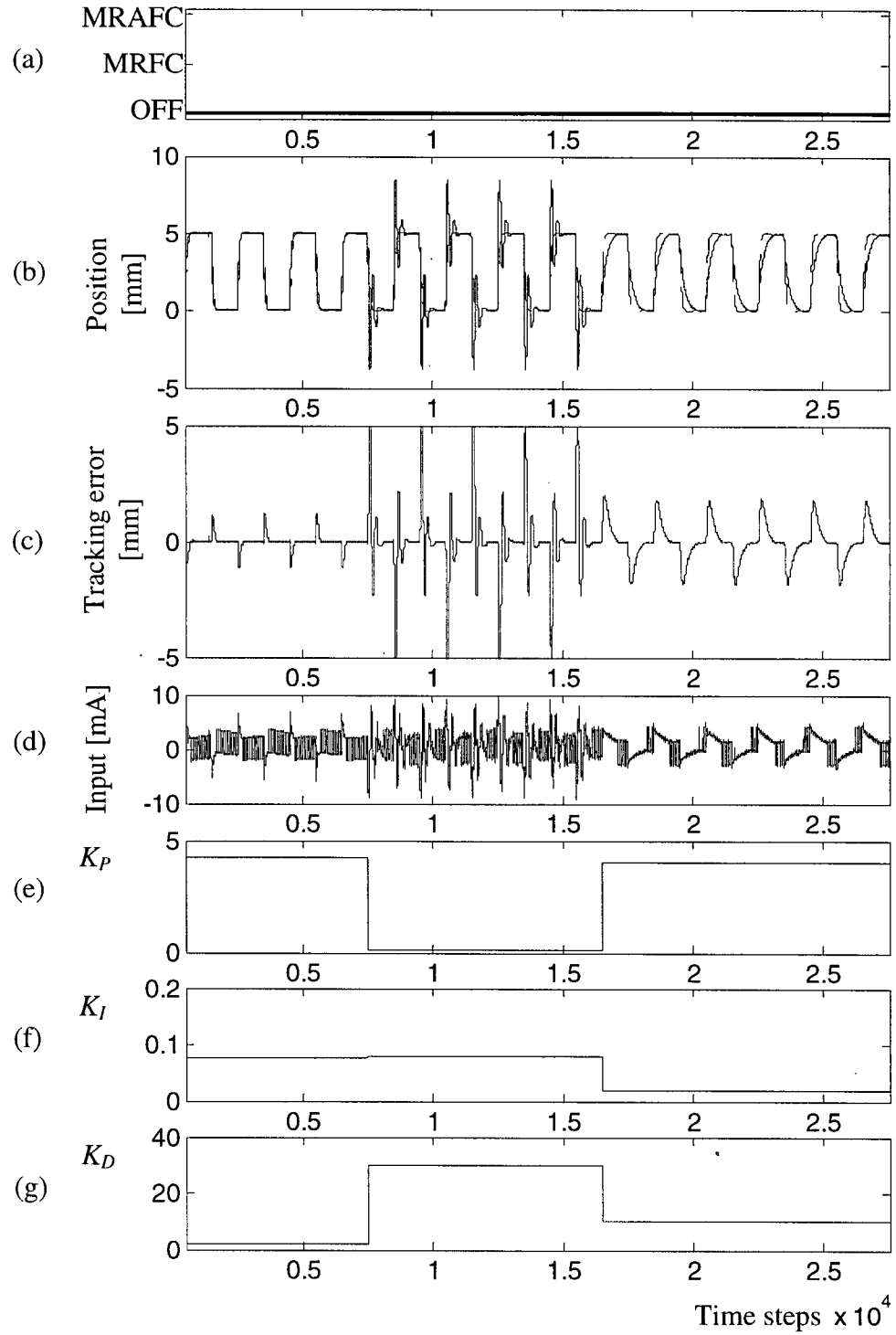


Figure 7.12: System response without adaptation. (a) Adaptation state; (b) Position response (solid-line: actual response; dashed-line: reference model response); (c) Tracking error; (d) Control input; (e) K_P; (f) K_I; and (g) K_D.

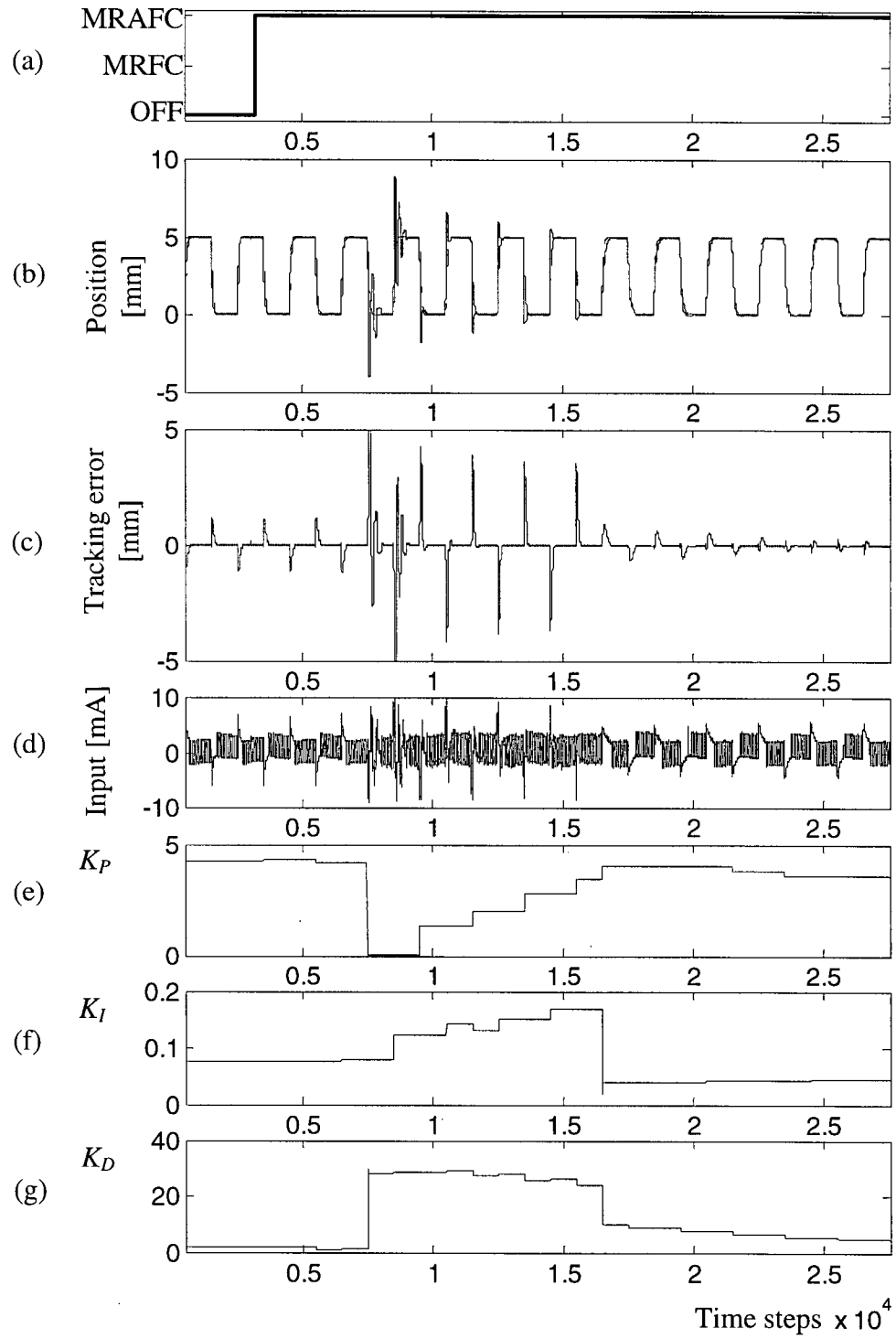


Figure 7.13: System response with MRAFC. (a) Adaptation state; (b) Position response (solid-line: actual response; dashed-line: reference model response); (c) Tracking error; (d) Control input; (e) K_P; (f) K_I; and (g) K_D.

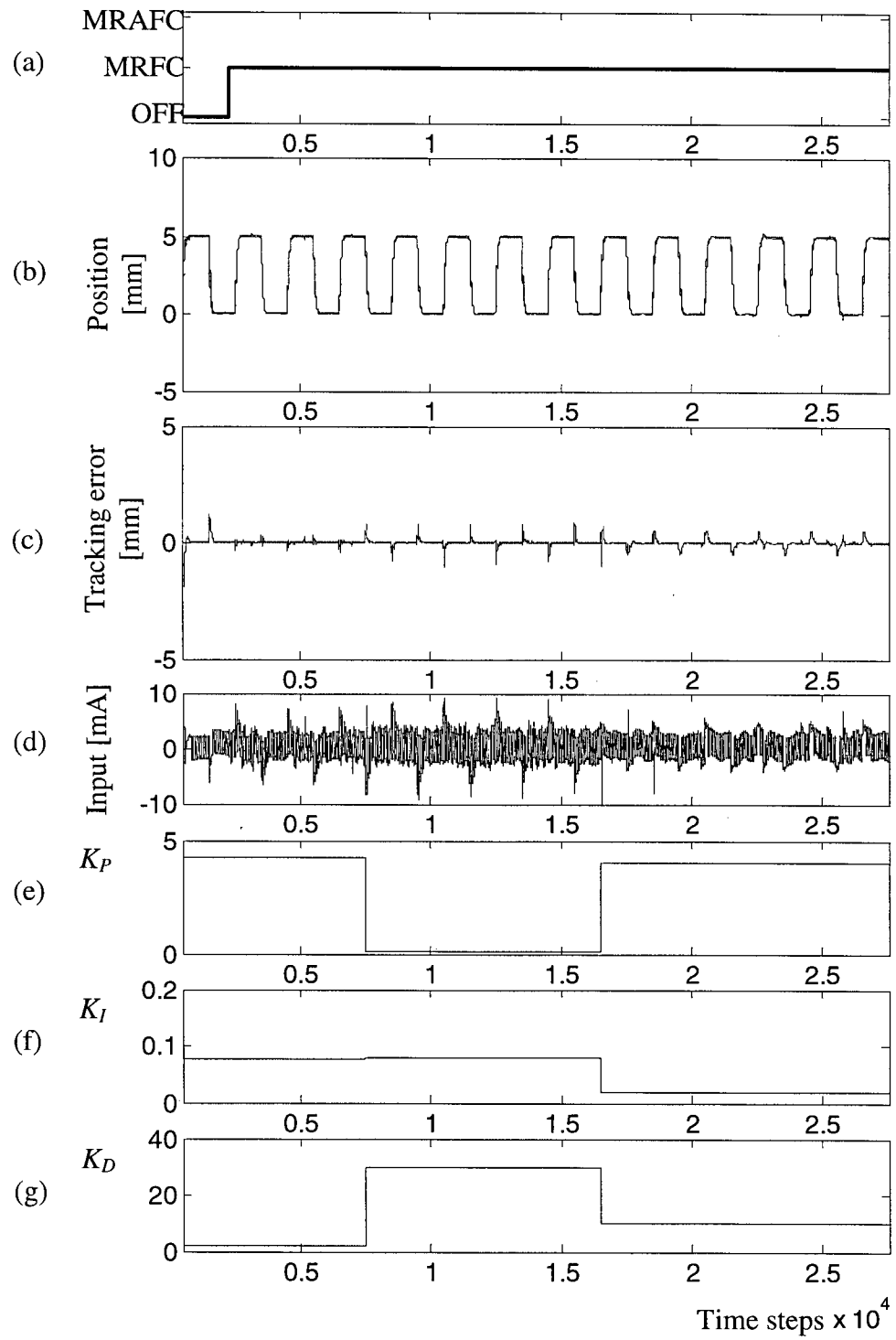


Figure 7.14: System response with MRFC. (a) Adaptation state; (b) Position response (solid-line: actual response; dashed-line: reference model response); (c) Tracking error; (d) Control input; (e) K_P; (f) K_I; and (g) K_D.

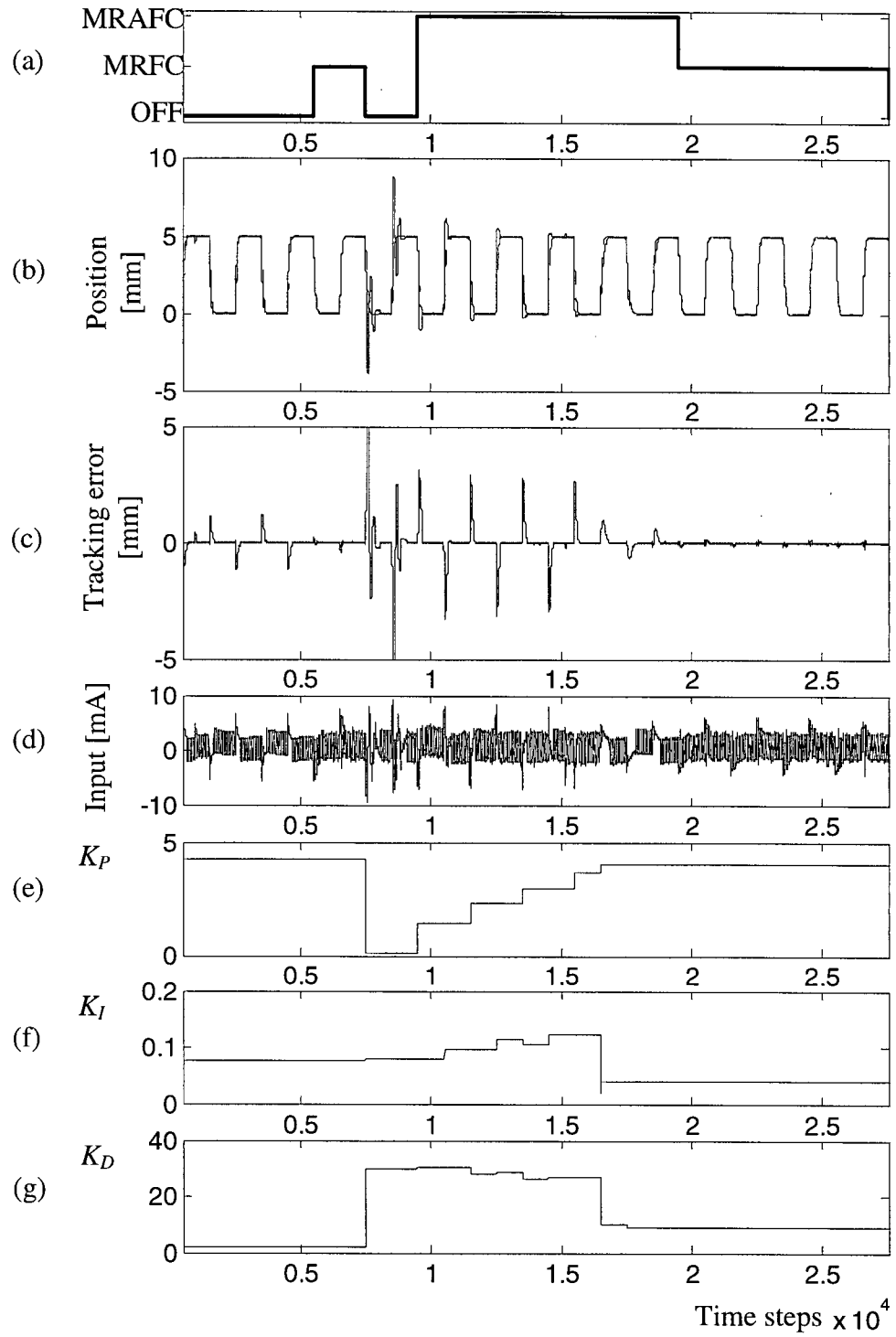


Figure 7.15: System response with automatic intelligent switching. (a) Adaptation state; (b) Position response (solid-line: actual response; dashed-line: reference model response); (c) Tracking error; (d) Control input; (e) K_P; (f) K_I; and (g) K_D.

7.7 Summary

A distributed networked intelligent supervisory control strategy was developed by employing a client-server architecture, which made the strategy modular. It provided high flexibility for the control structure with the ability to switch on any module as desired. The Ethernet protocol used in the present work also enabled easy real-time monitoring of the system performance in addition to on-line remote intervention by a human expert, to affect the machine states and configuration. The developed system was implemented for use with an industrial fish-cutting machine, which has an integral electro-hydraulic manipulator. The incorporation of an intelligent selector for adaptive control proved vital in improving the performance of the electro-hydraulic manipulator over a period of time, as shown through comprehensive experimentation. The selector was able to carry out autonomous switching of a locally implemented signal adaptive controller and a remotely implemented parameter adaptive controller.

Chapter 8
Conclusion

This thesis has investigated the analysis, development, and implementation of stable feedback control systems and remote monitoring and supervisory control systems for plants that are linked and accessed through a communication network. In a framework of networked control systems (NCS), where the plant, controllers, sensors, actuators, and human interface points may all be distributed and networked, various solutions have been developed and tested under realistic practical conditions and on an industrial machine. All aspects of theory, analysis, system architecture, software, hardware, system development, and practical implementation have been covered in the thesis. The primary research contributions made in this thesis are summarized in the next section. The subsequent section identifies the limitations of the current work, and presents some possible improvements and further research directions within the theme of network-based control.

8.1 Primary Contributions

One original contribution of this thesis is the adoption of the theory of Model Predictive Control (MPC) in the realm of Networked Control System (NCS) to address the shortcomings of other possible networked feedback control strategies. A new technique called NCS-MPC strategy has been developed, which predicts erroneous sensor data and, more importantly, predicts, buffers and schedules future control actions in advance. This approach has been demonstrated to be a viable solution for Networked Control Systems in the pursuit of concurrently overcoming the four key problems associated with unreliable network data communication: nondeterministic delay, packet losses, vacant sampling, and mis-synchronized data arrival. A main advantage of the developed strategy is the inherent transparency of the complicated dynamics of data transmission to the control synthesis. In fact, the developed strategy is documented as the first practical and successful attempt in handling data communication problems in the control action signals between the controller and the actuators of a networked plant.

The "fixed" optimal control structure of the basic unconstrained model predictive control policy has been enhanced in this research, for adapting to dynamic changes within a physical system and data transmission. An online routine of linear model identification, which utilizes the delayed system responses, has been incorporated into the control system, for catering to time-varying dynamics of the plant. Both the prediction horizon and the size of the data packets transmitted over the network have been made variable, and this flexibility has shown to be able to reduce the network traffic and the computational load of the control algorithm. In addition, active scheduling of the penalty functions of the control law, as incorporated in the developed technique, is a practical scheme to improve the working range of the controller in terms of the level of data transmission delay. The feasibility, high performance, and effectiveness of the developed NCS-MPC control strategy have been demonstrated in the present work through implementation on an industrial fish-processing machine and rigorous real-time experimentation.

In order to further investigate and substantiate the viability of the developed NCS-MPC strategy, asymptotic stability analysis in the sense of Lyapunov was conducted by including critical details and in the broader domain of constrained optimal predictive control. The analysis was divided into two major parts. The first part dealt with the case of perfect data feedback and focused on the stability, optimality and feasibility of utilizing pre-computed future control actions. The approach introduces a terminal cost, which is governed by a terminal weight to the finite horizon cost function with a lower and an upper bound. The careful design of the terminal weight is the key to guaranteeing asymptotic stability of the closed-loop system. Stability theorems were developed and analytically proved in this context, and were experimentally validated. In this manner, the developed theorems were demonstrated also as a practical tool for the proper design of a stable NCS-MPC strategy.

As a sequel to the stable control system configurations as developed in the first part of the analysis, the second part of the stability analysis concerned the determination of the maximum allowable deviation between the actual and the estimated states to maintain closed-loop asymptotic stability. This has been achieved in the thesis by utilizing the concept of sensitivity, particularly perturbation analysis of nonlinear programming, on the developed equality constrained NCS-MPC control policy. In contrast to the results obtained in the first part, it was established that the same conditions that are desirable in sustaining stability with buffered future control actions, have an undesirable effect in the presence of state estimation errors. Consequently, a design trade-off is required in the overall implementation of the developed

networked control solution. The results established from this analysis are particularly useful as a design criterion for further development of robust state observers.

In various stages of the present development of the unconstrained and constrained stable NCS-MPC control strategy, experimental validation was conducted on a newly developed control network architecture within an Ethernet network. An algorithm was developed for process scheduling with accurate clock synchronization, and was integrated into this architecture, to make it more flexible and practical, as demonstrated. The practical usage of multi-parametric Quadratic Programming has also been a key to the successful implementation of constrained optimal predictive control on high-speed servo systems since these systems allow insufficient time for the operation of traditional iterative optimization methods.

On a parallel front, a universal, web-based remote monitoring and supervisory control architecture for a distributed multi-systems domain has been established in this thesis. The two-tier client-server model as developed is low-cost, scalable and easily reconfigurable, which caters to variable domain size and interaction. The full functionally of the developed architecture was demonstrated through its implementation on an industrial fish-processing machine. The developed architecture is useful in providing remote world-wide access to equipment of scarce availability, remote management of production facilities, and process tuning and configuration by remotely located human experts. The hardware simplicity and software flexibility have made the architecture open for further development and advancement.

The remote and distributed intelligent supervisory control architecture, as developed in this thesis, is a novel design for systems with time-varying dynamics. A local and fast signal-adaptive model-referenced fuzzy control technique was adopted to assist the servo-loop controllers in providing almost error free tracking capability under nominal situations. In the event of change in the system dynamics, a remote knowledge-based model-referenced adaptive control technique, as implemented in the system, is capable of tuning the servo-loop controllers based on a set performance metrics. A high-level remote intelligent supervisor was designed and implemented to actively switch between the modes of operation of the local and remote adaptive controllers, and the servo-loop controllers. Experimental investigation has shown that the developed system has the capability to minimize the control effort while maintaining adequate tracking accuracy.

In summary, the present thesis represents a useful contribution, not only towards the advancement of the challenging and practical research in networked control systems and remote monitoring and supervisory control, but also to fundamental research in stable model

predictive control. Although, the communication network of implementations in this thesis is the commonly available Ethernet network, the developed feedback and supervisory control strategies are easily applicable to other control networks.

The contributions made in this thesis are by no means exhaustive. The next section identifies some limitations of the developments of the thesis and suggests possible enhancements and alternative approaches that may be investigated.

8.2 Limitations and Suggested Future Research

The control strategy for networked systems, as analyzed, developed, and implemented in this thesis assumes that a linear model of the plant is sufficient, which is an inherent assumption in model predictive control. A natural extension to the research is to investigate the use of a nonlinear model for the plant. Conditions for maintaining closed-loop asymptotic stability has been analyzed but they are limited to a linear, nominal model without taking into account modeling errors in the form of model uncertainty and robustness to external or internal disturbances. Robustness to modeling errors and disturbances may be considered in a future investigation. Furthermore, an analytical performance measure has to be established, which will provide further guidelines for practical design and implementation of the developed control strategy. There is also the subject of model identification using delayed and distributed system responses, which has not been investigated in the present work.

Since pre-computation of future control actions, although sub-optimal, was proven to be feasible with guaranteed asymptotic stability, the strategy has the potential for further enhancement to reduce the data transmission frequency to some level below the sampling frequency of the system. This will clearly reduce the network utilization and traffic, also leading to lower transmission delay.

Although the conditions for maintaining stability in the presence of state estimation errors in the sensory feedback data was studied in the thesis, important issues related to the design of practical and robust state observers to compensate for network transmission problems between a sensor and a controller was not the main focus. State observers that are robust, high precision, able to accommodate rapidly changing system dynamics, and with long-range estimation capability to compensate for successive loss or delay of sensory data, are crucial for wide practical implementation of the developed NCS technologies.

With regard to general applicability, the networked control strategy that was developed in this thesis is limited to systems with a single, centralized controller which requires an enormous

communication bandwidth to transmit process parameters and other data to the control point. However, many existing industrial systems; e.g., power systems, are decentralized which partition control algorithms into multi-node processing. Investigation of the adaptability of the developed strategy, along with the required modifications, to decentralized systems would be an appealing future endeavor. This may include the development of a robust scheduling and timing algorithms to provide seamless execution of the control algorithms.

In the present thesis, all the communicating nodes within the control loop; i.e., sensors, controllers, and actuators, were assumed to be simultaneously running at the same sampling interval, in a synchronous manner. As the complexity of the considered system grows, the proper synchronization and scheduling of events between all the nodes may be hard to achieve. Advantages may be gained by investigating into event-driven or asynchronous networked control systems. In the present work, only a single data packet is assumed to be transmitted from a node in a given sampling interval. It is useful to look into solutions when multiple packets of sensor and actuator data are transmitted at each sampling interval. These prospective investigations will provide means to develop more sophisticated networked control systems with multi-rate, asynchronous sampling.

Practical implementation and thorough testing have to be conducted in order to demonstrate that the developed networked control strategy is applicable to communication networks other than the Ethernet. Wireless Ethernet has been gaining much attention recently due to the introduction of smart sensor networks. The utilization of wireless communication in the feedback control loop will further complicate already problematic transmission of existing wired networks. Control solutions are particularly needed to overcome noisy and low-energy wireless transmission.

The present research on web-based remote monitoring and supervisory control may be broadened to incorporate further functionality to the developed architecture. Some useful functionalities that may be explored are: incorporating a unified method for defining the desired task that needs to be carried out by a particular system; incorporating a remote interface to allow transparent tele-operation of the system; and developing a graphical design tool for quick setup of an interface and communication sockets for additional systems. Message compression and encoding should be implemented as well to reduce the transmission bandwidth.

Bibliography

Abdullah, H.A. and C.R. Chatwin, "Distributed C3 Environment for Small to Medium-sized Enterprises," *Integrated Manufacturing Systems*, Vol. 5, No. 3, pp. 20-28, 1994.

Al-Ghazzawi, A., E. Ali, A. Nouh, and E. Zafiriou, "On-line Tuning Strategy for Model Predictive Controllers," *Journal of Process Control*, Vol. 11, pp. 265-284, 2001.

Almeida, L., P. Pedreiras, and J.A.G. Fonseca, "The FTT-CAN Protocol: Why and How," *IEEE Trans. Industrial Electronics*, Vol. 49, No. 6, pp. 1189-1201, 2002.

Anderson, R.T. and P.Y. Li, "Mathematical Modeling of a Two Spool Flow Control Servovalve using a Pressure Control Pilot," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 124, pp. 420-427, 2002.

Apple Computer, Inc., *Darwin Streaming Server*, Available at: http://www.opensource.apple.com/projects/streaming/.

Åström, K.J. and B. Wittenmark, *Adaptive Control*, 2nd Edition, Addison-Wesley Publishing, Inc., Reading, MA, 1995.

Azimi-Sadjadi, B., "Stability of networked control systems in the presence of packet losses," *Proc. IEEE Conf. on Decision and Control*, Maui, HI, pp. 676-681, 2003.

Beldiman, O., G.C. Walsh, and L. Bushnell, "Predictors for Networked Control Systems," *Proc. American Control Conference*, Chicago, IL, pp. 2347-2351, 2000.

Bellman, R.E., *Adaptive Control Processes – A Guided Tour*, Princeton University Press, Princeton, NJ, 1961.

Bemporad, A., F. Borrelli, and M. Morari, "Optimal Controllers for Hybrid Systems: Stability and Piecewise Linear Explicit Form," *Proc. IEEE Conf. on Decision and Control*, Sydney, Australia, pp. 1810-1815, 2000.

Bemporad, A., K. Fukuda, F.D. Torrisi, "Convexity Recognition of the Union Polyhedra," *Computational Geometry – Theory and Application*, Vol. 18, pp. 141-154, 2001.

Bemporad, A., M. Morari, V. Dua, and E.F. Pistikopoulos, "The Explicit Linear Quadratic Regulator for Constrained Systems," *Automatica*, Vol. 38, pp. 3-20, 2002.

Bertsekas, D.P. and R. Gallager, *Data Networks*, 2nd Edition, Prentice-Hall, Inc., Englewood-Cliffs, NJ, 1991.

Bertsekas, D.P., *Dynamic Programming and Optimal Control*, 2nd Edition, Athena Scientific, Belmont, MA, 2000.

Bhandari, A. and M.H. Shor, "Access to an Instructional Control Laboratory Experiment through the World Wide Web," *Proc. American Control Conference*, Philadelphia, PA, pp. 1319-1325, 1998.

Bitmead, R.R., M. Gevers, and V. Wertz, *Adaptive Optimal Control – The Thinking Man's GPC*, Prentice-Hall, Inc., Englewood-Cliffs, NJ, 1990.

Bordons, C. and E.F. Camacho, "A Generalized Predictive Controller for Wide Class of Industrial Processes," *IEEE Trans. Control Systems Technology*, Vol. 6, No. 3, pp. 372-387, 1998.

Boyd, S., L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1994.

Camacho, E.F. and C. Bordons, *Model Predictive Control*, Springer-Verlag, London, UK, 1998.

Cena, G. and A. Valenzano, "Achieving Round-Robin Access in Controller Area Networks," *IEEE Trans. Industrial Electronics*, Vol. 49, No. 6, pp. 1202-1213, 2002.

Chan, H. and Ü. Özgüner, "Closed-loop Control of Systems over a Communication Network with Queues," *International Journal of Control*, Vol. 62, No. 3, pp. 493-510, 1995.

Chen, H. and F. Allgöwer, "A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability," *Automatica*, Vol. 34, No. 10, pp. 1205-1217, 1998.

Chen, T.M. and R.C. Luo, "Remote Supervisory Control of An Autonomous Mobile Robot via World Wide Web," *Proc. IEEE Int. Symp. Industrial Electronics*, Guimarães, Portugal, Vol. 1, pp. 60-64, 1997.

Chisci, L., A. Garulli, and G. Zappa, "Fast Parallel Algorithms for Predictive Control," *Proc. Int. Conf. Control*, Coventry, UK, Vol. 2, pp. 1248-1253, 1994.

Clarke, D.W., C. Mohtadi, and P.S. Tuffs, "Generalized Predictive Control – Part I and II," *Automatica*, Vol. 23, No. 2, pp. 137-160, 1987.

Conti, M., L. Donatiello, and M. Furini, "Design and Analysis of RT-Ring: A Protocol for Supporting Real-time Communications," *IEEE Trans. Industrial Electronics*, Vol. 49, No. 6, pp. 1214-1226, 2002.

Costa, E.F. and J.B.R. do Val, "Stability of Receding Horizon Control of Nonlinear Systems," *Proc. IEEE Conf. on Decision and Control*, Maui, HI, pp. 2077-2081, 2003.

Cristian, F., "Probabilistic Clock Synchronization," *Distributed Computing*, Vol. 3, pp. 146-158, 1989.

Cutler, C.R. and B.C. Ramaker, "Dynamic Matrix Control – A Computer Control Algorithm," *Proc. Automatic Control Conference*, San Francisco, CA, 1980.

De Keyser, R.M.C. and A.R. Van Cuawenberghe, "Extended Prediction Self Adaptive Control," *Proc. IFAC Symp. Identification and System Parameter Estimation*, York, UK, pp. 1317-1322, 1985.

de Silva, C.W., "An Analytical Framework for Knowledge-Based Tuning of Servo Controllers," *Engineering Applications of Artificial Intelligence*, Vol. 4, No. 4, pp. 177-189, 1991.

de Silva, C.W., *Intelligent Control – Fuzzy Logic Applications*, CRC Press, Boca Raton, FL, 1995.

de Silva, C.W., *Mechatronics – An Integrated Approach*, CRC Press, Taylor and Francis, Boca Raton, FL, 2005.

Deitel, H.M. and P.J. Deitel, *Java™ – How To Program*, 3rd Edition, Prentice-Hall, Inc., Upper Saddle River, NJ, 1999.

Evtushenko, Y.G. and Zhadan, V.G., "Stable Barrier-Projection and Barrier-Newton Methods in Nonlinear Programming," *Optimization Methods and Software*, Vol. 3, No. 1, pp. 237-256, 1994.

Farsi, M. and K. Ratcliff, "Controlling with CANopen [Industrial Control Networks]," *IEE Review*, Vol. 44, No. 5, pp. 229-231, 1998.

Fiacco, A.V., "Sensitivity Analysis for Nonlinear Programming using Penalty Methods," *Mathematical Programming*, Vol. 10, pp. 287-311, 1976.

Findeisen, R., L. Imland, F. Allgöwer, and B.A. Foss, "State and Output Feedback Nonlinear Model Predictive Control – An Overview," *European Journal of Control*, Vol. 9, pp. 190-206, 2003.

Fletcher, R., *Practical Methods of Optimization*, 2nd Edition, John Wiley & Sons, Ltd., New York, NY, 1987.

Fukuda, K., *C-library cddlib Reference Manual*, Version 0.93c, 2003, Available at: http://www.ifor.math.ethz.ch/~fukuda/cdd_home/cdd.html.

Furuta, K., A. Sano, and D. Atherton, *State Variable Methods in Automatic Control*, John Wiley & Sons, Ltd., New York, NY, 1988.

Gockenbach, M.S., "Infeasible Point Methods for Inequality-constrained Nonlinear Programming," *Unpublished*, 2003.

Gu, J.H. and C.W. de Silva, "Client-Server Architecture for Implementing Real-time Robot Control in Industrial Processes," *Proc. 2nd Asian Control Conference*, Seoul, South Korea, Vol. 3, pp. 259-262, 1997.

Gusella, R. and S. Zatti, "The Accuracy of Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD," *IEEE Trans. Software Engineering*, Vol. 15, pp. 847-853, 1989.

Halevi, Y. and A. Ray, "Integrated Communication and Control Systems: Part I - Analysis," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 110, pp. 367-373, 1988.

Huzmezan, M. and G.A. Dumont, "Direct Adaptive Predictive Control using Subspace Identification in Laguerre Domain in the Presence of Constraints," *Proc. IEEE Mediterranean Conference*, Patras, Greece, 2000.

IEEE Computer Society, *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, IEEE Standard 802.3™, 2002.

Jiang, Z.-P. and Y. Wang, "Input-to-State Stability for Discrete-time Nonlinear Systems," *Automatica*, Vol. 37, pp. 857-869, 2001.

Kaynak, O., H. Hashimoto, and P. Lewis, "Minimum Effort Control of a Servo System," *Proc. IEEE Int. Conf. Industrial Electronics, Control, and Instrumentation*, Kobe, Japan, Vol. 1, pp. 755-759, 1991.

Keerthi, S.S. and E.G. Gilbert, "Optimal Infinite-Horizon Feedback Laws for a General Class of Constrained Discrete-time Systems: Stability and Moving-Horizon Approximations," *Journal of Optimization Theory and Applications*, Vol. 57, No. 2, pp. 265-293, 1988.

Kerrigan, E.C., *Robust Constraint Satisfaction: Invariant Sets and Predictive Control*, Ph.D. Thesis, Department of Engineering, University of Cambridge, Cambridge, UK, 2000.

Khalil, H.K., *Nonlinear Systems*, 3rd Edition, Prentice-Hall, Inc., Upper Saddle River, NJ, 2002.

Kim, Y.H., W.H. Kwon, and Y.I. Lee, "Min-Max Generalized Predictive Control with Stability," *Computers and Chemical Engineering*, Vol. 22, No. 12, pp. 1854-1858, 1998.

Ko, C.C., B.M. Chen, S.H. Chen, V. Ramakrishnan, R. Chen, S.Y. Hu, and Y. Zhuang, "A Large-Scale Web-based Virtual Oscilloscope Laboratory Experiment," *IEE Engineering Science and Education Journal*, Vol. 9, No. 2, pp. 69-76, 2000.

Ko, C.C., B.M. Chen, S.Y. Hu, V. Ramakrishnan, C.D. Cheng, Y. Zhuang, and J.P. Chen, "A Web-based Virtual Laboratory on a Frequency Modulation Experiment," *IEEE Trans. Systems, Man, and Cybernetics – Part C: Applications and Reviews*, Vol. 31, No. 3, pp. 295-303, 2001.

Kothare, M.V., V. Balakrishnan, and M. Morari, "Robust Constrained Model Predictive Control using Linear Matrix Inequalities," *Automatica*, Vol. 32, No. 10, pp. 1361-1379, 1996.

Kouvaritakis, B., M. Cannon, and J.A. Rossiter, "Who needs QP for Linear MPC anyway?" *Automatica*, Vol. 38, pp. 879-884, 2002.

Kreyszig, E., *Advanced Engineering Mathematics*, 8th Edition, John Wiley & Sons, Ltd., New York, NY, 1999.

Kurose, J.F. and K.W. Ross, *Computer Networking – A Top-Down Approach Featuring the Internet*, 2nd Edition, Pearson Addison-Wesley, Boston, MA, 2003.

Lian, F.-L., J. Moyne, and D. Tilbury, "Analysis and Modeling of Networked Control Systems: MIMO Case With Multiple Time Delays," *Proc. American Control Conference*, Arlington, VA, pp. 4306-4312, 2001.

Lian, F.-L., J. Moyne, and D. Tilbury, "Optimal Controller Design and Evaluation for a Class of Networked Control Systems with Distributed Control Delays," *Proc. American Control Conference*, Anchorage, AK, Vol. 4, pp. 3009-3014, 2002a.

Lian, F.-L., J. Moyne, and D. Tilbury, "Network Design Consideration for Distributed Control Systems," *IEEE Trans. Control Systems Technology*, Vol. 10, No. 2, pp. 297-307, 2002b.

Limón Marruedo, D., T. Álamo, and E.F. Camacho, "Enlarging the Domain of Attraction of MPC Controller using Invariant Sets," *Proc. IFAC 15th Triennial World Congress*, Barcelona, Spain, pp. 1-10, 2002a.

Limón Marruedo, D.L., T. Álamo, and E.F. Camacho, "Stability Analysis of Systems with Bounded Additive Uncertainties Based on Invariant Sets: Stability and Feasibility of MPC," *Proc. American Control Conference*, Anchorage, AK, pp. 364-369, 2002b.

Lin, F.-L., "Fuzzy Adaptive Model-Following Position Control for Ultrasonic Motor," *IEEE Trans. Power Electronics*, Vol. 12, No. 2, pp. 261-268, 1997.

Ljung, L., *System Identification – Theory for the User*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987.

Luck, R. and A. Ray, "An Observer-based Compensator for Distributed Delays," *Automatica*, Vol. 26, No. 5, pp. 903-908, 1990.

Luenberger, D.G., *Optimization by Vector Space Methods*, John Wiley & Sons, Inc., New York, NY, 1969.

Lundelius, J. and N. Lynch, "An Upper and Lower Bound for Clock Synchronization," *Information and Control*, Vol. 62, No. 2-3, pp. 190-204, 1984.

Mayne, D.Q., J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert, "Constrained Model Predictive Control: Stability and Optimality," *Automatica*, Vol. 36, pp. 789-814, 2000.

Meadows, Jr., E.S., *Stability and Continuity of Nonlinear Model Predictive Control*, Ph.D. Thesis, The University of Austin Texas, Austin, TX, 1994.

Microsoft Corp., *Microsoft Windows Media Technologies*, Available at: http://www.microsoft.com/windows/windowsmedia/distribute.asp.

Mills, D., "Improved Algorithms for Synchronizing Computer Network Clocks," *IEEE/ACM Trans. Networking*, Vol. 3, No. 3, pp. 245-254, 1995.

Mischalska, H. and D.Q. Mayne, "Robust Receding Horizon Control of Constained Nonlinear Systems," *IEEE Trans. Automatic Control*, Vol. 38, No. 11, pp. 1623-1633, 1993.

Munir, S. and W.J. Brook, "Internet Based Teleoperation using Wave Variables with Prediction," *Proc. IEEE/ASME Int. Conf. Advanced Intelligent Mechatronics*, Como, Italy, pp. 43-51, 2001.

National Instruments Corp., *LabVIEW™*, Available at: http://www.ni.com/labview/.

Nevistić, V. and J.A. Primbs, *Constrained Nonlinear Optimal Control – A Converse HJB Approach*, Technical Memorandum, No. CIT-CDS 96-021, California Institute of Technology, Pasadena, CA, 1996.

Nilsson, J., *Real-time Control Systems with Delays*, Ph.D. Dissertation, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1998a.

Nilsson, J., B. Bernhardsson, and B. Wittenmark, "Stochastic Analysis and Control of Real-time Systems with Random Time Delays," *Automatica*, Vol. 34, No. 1, pp. 57-64, 1998b.

Nunes, G.C., S. Kincal, and O.D. Crisalle, "A Polynomial Perspective on the Stability of Multivariable Predictive Controllers," *Computers and Chemical Engineering*, Vol. 27, pp. 1097-1111, 2003.

Ogata, K., *Discrete-time Control Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ, pp. 315-218, 1987.

Otanez, P.G., J.R. Moyne, and D.M. Tilbury, "Using Deadbands to Reduce Communication in Networked Control Systems," *Proc. American Control Conference*, Anchorage, AK, Vol. 4, pp. 3015-3020, 2002a.

Otanez, P.G., J.T. Parrott, J.R. Moyne, and D.M. Tilbury, "The Implications of Ethernet as a Control Network," *Proc. Global Powertrain Conference*, Ann Arbor, MI, pp. 1-9, 2002b.

Overstreet, J.W. and A. Tzes, "An Internet-based Real-time Control Engineering Laboratory," *IEEE Control Systems Magazine*, Vol. 19, No. 5, pp. 19-34, 1999.

Pistikopoulos, E.N., V. Dua, N.A. Bozinis, A. Bemporad, and M. Morari, "On-line Optimization via Off-line Parametric Optimization Tools," *Computers and Chemical Engineering*, Vol. 26, pp. 175-185, 2002.

Primbs, J.A. and V. Nevistić, "A New Approach to Stability Analysis for Constrained Finite Receding Horizon without End Constraints," *IEEE Trans. Automatic Control*, Vol. 45, No. 8, pp. 1507-1512, 2000.

Quanser, Inc., *WebLab*, Available at: http://www.controlab.com/.

Quevedo, D.E., G.C. Goodwin, and J.S. Welsh, "Minimizing Down-Link Traffic in Networked Control Systems via Optimal Control Techniques," *Proc. IEEE Conf. on Decision and Control*, Maui, HI, pp. 1200-1205, 2003.

Rawlings, J.B. and K.R. Muske, "The Stability of Constrained Receding Horizon Control," *IEEE Trans. Automatic Control*, Vol. 38, No. 10, pp. 1512-1516, 1993.

RealNetworks, Inc., Available at: http://www.realnetworks.com/index.html.

Röhrig, C. and A. Jochheim, "The Virtual Lab for Controlling Real Experiments via Internet," *Proc. IEEE Int. Symp. Computer Aided Control System Design*, Kohala, HI, pp. 279-584, 1999.

Santos, L.O. and L.T. Biegler, "A Tool to Analyze Robust Stability for Model Predictive Controllers," *Journal of Process Control*, Vol. 9, pp. 233-246, 1999.

Scokaert, P.O.M., D.Q. Mayne, and J.B. Rawlings, "Suboptimal Model Predictive Control (Feasibility Implies Stability)," *IEEE Trans. Automatic Control*, Vol. 44, No. 3, pp. 648-654, 1999.

Sinopoli, B., L. Schenato, M. Franceschetti, K. Poolla, and S.S. Sastry, "Time Varying Optimal Control with Packet Losses," *Proc. IEEE Conf. on Decision and Control*, Atlantis, Paradise Island, Bahamas, Vol. 2, pp. 1938-1943, 2004a.

Sinopoli, B., L. Schenato, M. Franceschetti, K. Poolla, M.I. Jordan, and S.S. Sastry, "Kalman Filtering with Intermitted Observation," *IEEE Trans. Automatic Control*, Vol. 49, No. 9, pp. 1453-1464, 2004b.

Skogestad, S. and I. Postlethwaite, *Multivariable Feedback Control – Analysis and Design*, John Wiley & Sons, Ltd., New York, NY, 1996.

Sugeno, M., *Industrial Applications of Fuzzy Control*, Elsevier Science, New York, NY, 1985.

Sun Microsystem, Inc., *Java™ Media Framework*, Available at: http://java.sun.com/products/java-media/jmf/

Tafazoli, S., C.W. de Silva, and P.D. Lawrence, "Tracking Control of an Electro-hydraulic Manipulator in the Presence of Friction," *IEEE Trans. Control Systems Technology*, Vol. 6, No. 3, pp. 401-411, 1998.

Tang, K.Z., K.K. Tan, C.W. de Silva, T.H. Lee, and S.J. Chin, "Monitoring and Suppression of Vibration in Precision Machines," *Journal of Intelligent and Fuzzy Systems*, Vol. 11, pp. 33-52, 2002.

Tang, P.L., A.N. Poo, and C.W. de Silva, "Knowledge-based Extension to Model-Referenced Adaptive Control with Application to an Industrial Process," *Journal of Intelligent and Fuzzy Systems*, Vol. 10, pp. 159-183, 2002a.

Tang, P.L., C.W. de Silva, and A.N. Poo, "Intelligent Adaptive Control in an Industrial Fish Cutting Machine," *Trans. IEE Control Theory and Applications*, Vol. 93, No. 2, pp. 60-72, 2002b.

Tang, P.L., C.W. de Silva, and X.G. Wang, "Monitoring and Control of a Wood-Drying Kiln via Virtual Project Stations," *Proc. IEEE Int. Symp. on Intelligent Control*, Vancouver, Canada, pp. 568-573, 2002c.

Tang, P.L., C.W. de Silva, and X.G. Wang, "Intelligent Monitoring and Control of Industrial Processes through the Internet," *Proc. ASME IMECE Symp. on Intelligent Systems*, New Orleans, LA, Vol. 2, 2002d.

Tang, P.L. and C.W. de Silva, "Ethernet-based Predictive Control of an Industrial Hydraulic Machine," *Proc. IEEE Conf. on Decision and Control*, Maui, HI, pp. 695-700, 2003a.

Tang, P.L. and C.W. de Silva, "Ethernet-based Intelligent Switching of Controllers for Performance Improvement in an Industrial Plant," *Proc. ASME IMECE Symp. on Intelligent Systems*, Washington, DC, Vol. 2, 2003b.

Tang, P.L. and C.W. de Silva, "Compensation for Transmission Delays in an Ethernet-based Control Network using Variable-Horizon Predictive Control," *IEEE Trans. Control Systems Technology*, 2005a. (In Press)

Tang, P.L. and C.W. de Silva, "Stability and Optimality of Constrained Model Predictive Control with Future Input Buffering in Networked Control Systems," *Proc. American Control Conference*, Portland, OR, 2005b.

Tang, P.L. and C.W. de Silva, "Stability Validation of a Constrained Model Predictive Networked Control System with Future Input Buffering," *ASME J. Dynamic Systems, Measurement, and Control*, 2005c. (Under review)

Tang, P.L. and C.W. de Silva, "A Reconfigurable System Architecture for Web-based Monitoring and Intelligent Supervisory Control of Dynamic Processes," *IFAC J. Control Engineering Practice*, 2005d. (Under review)

Tipsuwan, Y. and M.-Y. Chow, "On the Gain Scheduling for Networked PI Controller Over IP Network," *Proc. IEEE/ASME Int. Conf. Advanced Intelligent Mechatronics*, Kobe, Japan, pp. 640-645, 2003.

Tøndel, P., *Constrained Optimal Control via Multiparametric Quadratic Programming*, Ph.D. Thesis, Department of Engineering Cybernetics, Norwegian University of Science and Technology, Tronheim, Norway, 2003.

Tovar, E. and F. Vasques, "Real-time Fieldbus Communications using Profibus Networks," *IEEE Trans. Industrial Electronics*, Vol. 46, No. 6, pp. 1241-1251, 1999.

Walsh, G.C., H. Ye, and L. Bushnell, "Stability Analysis of Networked Control Systems," *Proc. American Control Conference*, San Diego, CA, pp. 2876-2880, 1999.

Walsh, G.C., H. Ye, and L. Bushnell, "Stability Analysis of Networked Control Systems," *IEEE Trans. Control Systems Technology*, Vol. 10, No. 3, pp. 438-446, 2002.

Weaver, A.C. and X. Zhang, "Monitoring and Control using the Internet and Java™," *Proc. IEEE Industrial Electronics Society*, San Jose, CA, pp. 1152-1158, 1999.

Xiao, L., A. Hassibi, and J.P. How, "Control with Random Communication Delays via a Discrete-time Jump System Approach," *Proc. American Control Conference*, Chicago, IL, Vol. 3, pp. 2199-2204, 2000.

Xiao, L., M. Johansson, H. Hindi, S. Boyd, and A. Goldsmith, *Joint Optimization of Communication Rates and Linear Systems*, Technical Report, Information System Laboratory, Stanford University, 2001.

Yook, J.K., D.M. Tilbury, H.S. Wong, and N.R. Soparkar, "Trading Computation for Bandwidth: State Estimators for Reduced Communication in Distributed Control Systems," *Proc. Japan-USA Symp. Flexible Automation*, Ann Arbor, MI, 2000.

Zelinka, P., "Multivariable Predictive Control," *Proc. IFAC New Trends in Design of Control Systems*, Smolenice, Slovak Republic, pp. 175-181, 1997.

Zhang, W., M.S. Branicky, and S.M. Phillips, "Stability of Networked Control Systems," *IEEE Control System Magazine*, Vol. 21, No. 1, pp. 84-99, 2001.

Zheng, A., "Robust Stability Analysis of Constrained Model Predictive Control," *Journal of Process Control*, Vol. 9, pp. 271-278, 1999.

Ziegler, G.M., *Lectures on Polytopes*, Springer-Verlag, New York, NY, 1995.

Appendix A
Linear Matrix Inequalities

The basics of Linear Matrix Inequalities (LMIs) and some standard optimization problems based on LMIs are briefly presented here. An elaborate treatment of the theory and application of LMIs can be found, for example, in Boyd, *et al.* (1994). Note that the symbols used in the appendices do not exactly correspond to the ones used in the body of this thesis.

A Linear Matrix Inequality (LMI) is a matrix inequality of the following form:

$$\boldsymbol{M}(\boldsymbol{x}) = \boldsymbol{M}_0 + \sum_{i=1}^{n} x_i \boldsymbol{M}_i \geq \boldsymbol{0} \tag{A.1}$$

where $\boldsymbol{x} = [x_1 \quad \cdots \quad x_n] \in \mathbb{R}^n$ and $\boldsymbol{M}_i$ are symmetric positive definite matrices, which are given. Multiple LMIs can be assembled diagonally into a single LMI. The main advantage of the LMI representation and hence its wide acceptance stems from the ability to represent nonlinear convex inequalities in an equivalent LMI form. Specifically, if $\boldsymbol{J}(\boldsymbol{x}) = \boldsymbol{J}(\boldsymbol{x})^T$, $\boldsymbol{K}(\boldsymbol{x}) = \boldsymbol{K}(\boldsymbol{x})^T$ and $\boldsymbol{L}(\boldsymbol{x})$ are matrices affine in $\boldsymbol{x}$, and governed by the following pair of quadratic matrix inequalities:

$$\begin{aligned} &\boldsymbol{K}(\boldsymbol{x}) > \boldsymbol{0} \\ &\boldsymbol{J}(\boldsymbol{x}) - \boldsymbol{L}(\boldsymbol{x})\boldsymbol{K}(\boldsymbol{x})^{-1}\boldsymbol{L}(\boldsymbol{x}) \geq \boldsymbol{0} \end{aligned} \tag{A.2}$$

then, by Schur complements, (A.2) is equivalent to the LMI:

$$\begin{bmatrix} \boldsymbol{J}(\boldsymbol{x}) & \boldsymbol{L}(\boldsymbol{x}) \\ \boldsymbol{L}(\boldsymbol{x})^T & \boldsymbol{K}(\boldsymbol{x}) \end{bmatrix} \geq \boldsymbol{0} \tag{A.3}$$

For example, the well-known Riccati equation of the form

$$\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} + \boldsymbol{Q} \leq \boldsymbol{0} \tag{A.4}$$

with $\boldsymbol{Q} = \boldsymbol{Q}^T$ and $\boldsymbol{R} = \boldsymbol{R}^T$, and $\boldsymbol{P} = \boldsymbol{P}^T$ is the unknown variable matrix, can be expressed by

the following LMI:

$$\begin{bmatrix} -\boldsymbol{A}^T\boldsymbol{P} - \boldsymbol{P}\boldsymbol{A} - \boldsymbol{Q} & \boldsymbol{P}\boldsymbol{B} \\ \boldsymbol{B}^T\boldsymbol{P} & \boldsymbol{R} \end{bmatrix} \geq \boldsymbol{0} \tag{A.5}$$

The standard LMI-based problems can be classified into four categories; namely, the LMI feasibility problems, the eigenvalue problems, the generalized eigenvalue problems, and the convex problems.

An LMI feasibility problem involves, given the LMI $\boldsymbol{M}(\boldsymbol{x}) > \boldsymbol{0}$, determining if a point $\boldsymbol{x}^*$ satisfies $\boldsymbol{M}(\boldsymbol{x}^*) > \boldsymbol{0}$. For example, from the basic inequality of Lyapunov stability:

$$\begin{aligned} &\boldsymbol{P} > \boldsymbol{0} \\ &\boldsymbol{A}_i^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A}_i < 0 \text{ for } i = 1,\ldots m \end{aligned} \tag{A.6}$$

find if $\boldsymbol{P}$ exists for a given $\boldsymbol{A}_i \in \mathbb{R}^{n\times n}$.

An eigenvalue problem is a convex optimization of the maximum eigenvalue λ of a matrix subjected to constraints of the LMI form, specifically:

$$\begin{aligned} &\min_{\boldsymbol{x},\lambda} \ \lambda \\ &\text{s.t. } \lambda\boldsymbol{I} - \boldsymbol{E}(\boldsymbol{x}) \geq \boldsymbol{0} \text{ and } \boldsymbol{F}(\boldsymbol{x}) \geq \boldsymbol{0} \end{aligned} \tag{A.7}$$

where $\boldsymbol{E}(\boldsymbol{x})$ and $\boldsymbol{F}(\boldsymbol{x})$ are affine in $\boldsymbol{x}$. Equivalently, (A.7) can be expressed as an optimization of a linear function subject to an LMI constraint given as:

$$\begin{aligned} &\min_{\boldsymbol{x}} \ \boldsymbol{c}^T\boldsymbol{x} \\ &\text{s.t. } \boldsymbol{M}(\boldsymbol{x}) \geq \boldsymbol{0} \end{aligned} \tag{A.8}$$

Here, $\boldsymbol{M}(\boldsymbol{x})$ is a symmetric matrix that depends affinely on the optimization of variable $\boldsymbol{x}$, and $\boldsymbol{c}$ is the coefficient vector of the appropriate length.

The generalized eigenvalue problems have the general form:

$$\begin{aligned} &\min_{\boldsymbol{x},\lambda} \ \lambda \\ &\text{s.t. } \lambda\boldsymbol{F}(\boldsymbol{x}) - \boldsymbol{E}(\boldsymbol{x}) \geq \boldsymbol{0},\ \boldsymbol{F}(\boldsymbol{x}) \geq 0 \text{ and } \boldsymbol{G}(\boldsymbol{x}) \geq \boldsymbol{0} \end{aligned} \tag{A.9}$$

such that the maximum generalized eigenvalue of the pair of matrices $\boldsymbol{E}(\boldsymbol{x})$ and $\boldsymbol{F}(\boldsymbol{x})$ is optimized. Here, $\boldsymbol{E}(\boldsymbol{x}) = \boldsymbol{E}(\boldsymbol{x})^T$, $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{x})^T$ and $\boldsymbol{G}(\boldsymbol{x}) = \boldsymbol{G}(\boldsymbol{x})^T$ are affine functions of $\boldsymbol{x}$.

The problem of (4.9) can be written in the equivalent form:

$$\begin{aligned} &\min_{x,\lambda} \ \bar{\lambda}\big(\boldsymbol{F}(\boldsymbol{x}),\boldsymbol{E}(\boldsymbol{x})\big) \\ &\text{s.t. } \boldsymbol{F}(\boldsymbol{x}) \geq 0 \text{ and } \boldsymbol{G}(\boldsymbol{x}) \geq \boldsymbol{0} \end{aligned} \tag{A.10}$$

Note that the constraints are convex and the objective function $\bar{\lambda}\big(\boldsymbol{F}(\boldsymbol{x}),\boldsymbol{E}(\boldsymbol{x})\big)$ is quasi-convex, rendering (A.10) a quasi-convex optimization problem.

The last class of LMI-based problems, although less common, is the convex problems which have the form

$$\begin{aligned} &\min_{x} \ \log\det \boldsymbol{E}(\boldsymbol{x})^{-1} \\ &\text{s.t. } \boldsymbol{E}(\boldsymbol{x}) \geq 0 \text{ and } \boldsymbol{F}(\boldsymbol{x}) \geq \boldsymbol{0} \end{aligned} \tag{A.11}$$

in which $\boldsymbol{E}(\boldsymbol{x}) = \boldsymbol{E}(\boldsymbol{x})^T$ and $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{x})^T$ are affine in $\boldsymbol{x}$.

The above LMI-based problems can be readily and efficiently solved in polynomial time using methods similar to those used in related optimal programming; e.g., the ellipsoid algorithm, and the interior point method. Reducing a general problem to an LMI-based optimization problem can be considered equivalent to "solving" that original problem, even though a so-called "analytic solution" to the problem may not exist. Standard control engineering problems that can utilize the advantages of LMI include: multi-criterion Linear Quadratic Gaussian control law, Model Predictive Control, robust identification, system realization, interpolation problems involving scaling, synthesizing $\mathcal{H}_2$ and $\mathcal{H}_\infty$ output feedback control laws, and multi-objective output feedback control law.

Appendix B
Multi-parametric Quadratic Programming

The key to successful real-time implementation of the constrained state-space MPC policies as developed in the present thesis is the ability to formulate the MPC problems as a multi-parametric Quadratic Programming (mpQP) problem. This allows the computationally intensive operation of MPC optimization to be brought off-line, leaving simple and fast arithmetic computations during real-time operation. This appendix provides a brief introduction to mpQP and the succeeding off-line and on-line algorithms for building and evaluating binary search trees from the prior results generated by the mpQP algorithm. Some enhancements that improve the computational efficiency of the algorithm for building the binary search trees are carried out. The off-line computation requirement for solving the mpQP problem and building of the corresponding binary search tree are also discussed.

B.1 Preliminaries

As an integral part, mpQP depends on the concept of polyhedral sets (Ziegler, 1995). Some fundamental definitions and concepts of polyhedral sets are given here.

Definition B.1: *A polyhedron is a convex set* $\mathcal{Q} \subseteq \mathbb{R}^n$ *given as an intersection of a finite number of closed half-spaces* $\mathcal{Q} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{Q}_x \boldsymbol{x} \leq \boldsymbol{Q}_c\}$.

Definition B.2: *A polytope is a bounded polyhedron* $\mathcal{P} \subseteq \mathbb{R}^n$ *given as* $\mathcal{P} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{P}_x \boldsymbol{x} \leq \boldsymbol{P}_c\}$.

A polytope $\mathcal{P} \subseteq \mathbb{R}^n$, $\mathcal{P} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{P}_x \boldsymbol{x} \leq \boldsymbol{P}_c\}$ is full dimensional if $\boldsymbol{P}_x \boldsymbol{x} < \boldsymbol{P}_c$ for $\exists \boldsymbol{x} \in \mathbb{R}^n$. The same polytope $\mathcal{P}$ is normalized if $\left\| (\boldsymbol{P}_x)_i \right\| = 1$ where $(\boldsymbol{P}_x)_i$ represents the i-th row of the matrix $\boldsymbol{P}_x$. Fundamentally, a polytope $\mathcal{P}$ can be described by its bounding vertices (vertex representation) as

$$\mathcal{P}=\left\{\boldsymbol{x}\in\mathbb{R}^n \,\middle|\, \boldsymbol{x}=\sum_{j=1}^{n_v}\alpha_j\boldsymbol{v}_j, 0\le\alpha_j\le 1, \sum_{j=1}^{n_v}\alpha_j=1\right\} \tag{B.1}$$

where $\boldsymbol{v}_j$ is the j-th vertex of $\mathcal{P}$, and n_v is the total number of vertices of $\mathcal{P}$.

Definition B.3: *If a linear inequality $\boldsymbol{a}^T\boldsymbol{x}\le\boldsymbol{b}$ holds for $\forall\boldsymbol{x}\in\mathcal{P}$, it is said to be valid for the polyhedron $\mathcal{P}$. A subset of a polyhedron is called a face of $\mathcal{P}$ if it is represented by $\mathcal{F}=\mathcal{P}\cap\{\boldsymbol{x}\in\mathbb{R}^n \mid \boldsymbol{a}^T\boldsymbol{x}=\boldsymbol{b}\}$, for some valid inequality $\boldsymbol{a}^T\boldsymbol{x}\le\boldsymbol{b}$. The faces of dimension 0, 1, $n-2$, and $n-1$ are called vertices, edges, ridges, and facets, respectively.*

Definition B.4: *The subspace $\boldsymbol{a}^T\boldsymbol{x}=\boldsymbol{b}$ in $\mathbb{R}^n$ is called a hyperplane. A hyperplane $\mathcal{H}\in\mathbb{R}^n$ is said to support a polyhedron $\mathcal{P}$ if one of the two closed half-space of $\mathcal{H}$ contains $\mathcal{P}$.*

B.2 The MpQP Algorithm

The potential of reformulating the MPC optimization problem with quadratic cost function and linear constraints as an mpQP problem was recognized by Bemporad, *et al.* (2002). As given in (4.17), the standard QP formulation to the MPC cost function is

$$\begin{aligned} V_H(\boldsymbol{x}_k,\boldsymbol{\pi}_k) = &\min_{\boldsymbol{\Pi}_H} \left\{\tfrac{1}{2}\boldsymbol{x}_k^T\boldsymbol{Y}_r\boldsymbol{x}_k+\tfrac{1}{2}\boldsymbol{\Pi}_H^T\boldsymbol{J}_r\boldsymbol{\Pi}_H+\boldsymbol{x}_k^T\boldsymbol{F}_r\boldsymbol{\Pi}_H\right\} \\ &\text{s.t. } \boldsymbol{G}_r\boldsymbol{\Pi}_H\le\boldsymbol{W}_r+\boldsymbol{E}_r\boldsymbol{x}_k \end{aligned} \tag{B.2}$$

where the $\boldsymbol{\Pi}_H=\begin{bmatrix}\boldsymbol{u}_k^T & \boldsymbol{u}_{k+1}^T & \cdots & \boldsymbol{u}_{k+H-1}^T\end{bmatrix}^T\in\mathbb{R}^{mH}$ is the control input vector to be optimized based on the current state $\boldsymbol{x}_k$. The purpose of reformulating (B.2) into the equivalent mpQP form, where both the input and state variables are treated as optimization variables, is to establish an algorithm to express the optimal control input vector $\boldsymbol{\Pi}_H^*$ as an explicit, continuous piecewise affine linear function of $\boldsymbol{x}_k$; i.e., $\boldsymbol{\Pi}_H^*=g(\boldsymbol{x})$ where $g:\mathcal{X}\to\mathbb{R}^{mH}$ within a polyhedral set (state-space) of $\mathcal{X}\in\mathbb{R}^n$. Specifically, the polyhedral set $\mathcal{X}$ can be partitioned into i convex polyhedral regions $\mathcal{X}_i$ such that $g(\boldsymbol{x})=\boldsymbol{H}_g^i\boldsymbol{x}+\boldsymbol{k}_g^i$ for $\forall\boldsymbol{x}\in\mathcal{X}_i$. Before proceeding further, define

$$\boldsymbol{s}=\boldsymbol{\Pi}_H+\boldsymbol{J}_r^{-1}\boldsymbol{F}_r^T\boldsymbol{x}_k \tag{B.3}$$

to obtain the equivalent mpQP form of (B.2) as:

$$V_H^s(\boldsymbol{x}_k, \boldsymbol{\pi}_k) = \min_s \left\{ \tfrac{1}{2} \boldsymbol{s}^T \boldsymbol{Y}_s \boldsymbol{s} \right\}$$
$$\text{s.t.} \quad \boldsymbol{G}_r \boldsymbol{s} \le \boldsymbol{W}_r + \boldsymbol{E}_s \boldsymbol{x}_k \tag{B.4}$$

in which $\boldsymbol{E}_s = \boldsymbol{E}_r + \boldsymbol{G}_r \boldsymbol{J}_r^{-1} \boldsymbol{F}_r^T$.

Theorem B.1 (cf. Bemporad, *et al.*, 2002): *Assume* $\boldsymbol{Y}_s \succ 0$. *Consider a combination of active constraints* $\boldsymbol{G}_r^A$, $\boldsymbol{W}_r^A$ *and* $\boldsymbol{E}_s^A$, *and assume that the rows of* $\boldsymbol{G}_r^A$ *are linearly independent. Let* $\mathcal{K}_0$ *be a set of all vectors* $\boldsymbol{x}$, *for which such a combination is active at the optimum. Then,* $\boldsymbol{s}$ *and the associated vector of nonnegative Lagrange multipliers* $\boldsymbol{\lambda}$ *are uniquely defined affine functions of* $\boldsymbol{x}$ *given as:*

$$\boldsymbol{s} = \boldsymbol{Y}_s^{-1} \left(\boldsymbol{G}_r^A \right)^T \left(\boldsymbol{G}_r^A \boldsymbol{Y}_s^{-1} \left(\boldsymbol{G}_r^A \right)^T \right)^{-1} \left(\boldsymbol{W}_r^A + \boldsymbol{E}_s \boldsymbol{x} \right) \tag{B.5}$$

$$\boldsymbol{\lambda} = -\left(\boldsymbol{G}_r^A \boldsymbol{Y}_s^{-1} \left(\boldsymbol{G}_r^A \right)^T \right)^{-1} \left(\boldsymbol{W}_r^A + \boldsymbol{E}_s \boldsymbol{x} \right) \tag{B.6}$$

over the critical region $\mathcal{K}_0$.

Proof:

This follows directly from the first order Karush-Kuhn-Tucker (KKT) optimality conditions (see Bemporad, *et al.*, 2002). □

A compact representation of $\mathcal{K}_0 = \{ x \in \mathcal{K} \mid \boldsymbol{R}_0 \boldsymbol{x} \le \boldsymbol{q}_0 \}$ can be obtained by substituting (B.5) into the inequality constraints of (B.4) and enforcing nonnegativeness of (B.6), arriving at the following theorem.

Theorem B.2 (cf. Fiacco, 1976): *For the problem (B.4), in the neighborhood of the vector of parameter values* $\boldsymbol{x}_0$, *there exists a unique and once continuously differentiable function of the KKT pair* $\left[\boldsymbol{s}(\boldsymbol{x})^T \quad \boldsymbol{\lambda}(\boldsymbol{x})^T \right]^T$ *where* $\boldsymbol{s}(\boldsymbol{x})$ *is a unique isolated minimizer for (B.4), and*

$$\begin{bmatrix} \frac{d}{dx} \boldsymbol{s}(\boldsymbol{x}_0) \\ \frac{d}{dx} \boldsymbol{\lambda}(\boldsymbol{x}_0) \end{bmatrix} = -\boldsymbol{M}_0^{-1} \boldsymbol{N}_0 \tag{B.7}$$

where

$$\boldsymbol{M}_0 = \begin{bmatrix} \boldsymbol{Y}_s & (\boldsymbol{G}_r)_1^T & \cdots & (\boldsymbol{G}_r)_{n_c}^T \\ -\lambda_1 (\boldsymbol{G}_r)_1 & -\boldsymbol{V}_1 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ -\lambda_{n_c} (\boldsymbol{G}_r)_{n_c} & \boldsymbol{0} & \cdots & -\boldsymbol{V}_{n_c} \end{bmatrix}$$

$$\boldsymbol{N}_0 = \begin{bmatrix} (\boldsymbol{B}_s)^T & \left(\lambda_1 (\boldsymbol{E}_s)_1\right)^T & \cdots & \left(\lambda_{n_c} (\boldsymbol{E}_s)_{n_c}\right)^T \end{bmatrix}^T$$

in which $(\boldsymbol{G}_r)_i$ *denotes the* i*-th row of* $\boldsymbol{G}_r$, $(\boldsymbol{E}_s)_i$ *denotes the* i*-th row of* $\boldsymbol{E}_s$, $\boldsymbol{V}_i = (\boldsymbol{G}_r)_i \boldsymbol{s}(\boldsymbol{x}_0) - (\boldsymbol{W}_r)_i - (\boldsymbol{E}_s)_i$, $(\boldsymbol{W}_r)_i$ *denotes the* i*-th row of* $\boldsymbol{W}_r$, *and* $\boldsymbol{B}_s \in \mathbb{R}^{mH \times n}$ *is a null matrix.*

Proof: see Fiacco (1976). □

The KKT pair of (B.7) is obtained as:

$$\begin{bmatrix} \boldsymbol{s}(\boldsymbol{x}) \\ \boldsymbol{\lambda}(\boldsymbol{x}) \end{bmatrix} = -\boldsymbol{M}_0^{-1} \boldsymbol{N}_0 \left[\boldsymbol{x} - \boldsymbol{x}_0\right] + \begin{bmatrix} \boldsymbol{s}(\boldsymbol{x}_0) \\ \boldsymbol{\lambda}(\boldsymbol{x}_0) \end{bmatrix} \tag{B.8}$$

Hence, the critical region $\mathcal{X}_0 = \{x \in \mathcal{X} \mid \boldsymbol{R}_0 \boldsymbol{x} \le \boldsymbol{q}_0\}$ is defined as the set of $\boldsymbol{x}$ where the solution to (B.8) remains optimal. $\mathcal{X}_0$ is obtained by removing the redundant constraints from a set of inequalities $\mathcal{X}_R$ given by

$$\mathcal{X}_R = \left\{ \boldsymbol{x} \subseteq \mathcal{X} \middle| \boldsymbol{G}_r^{\bar{A}} \boldsymbol{s}(\boldsymbol{x}) \le \boldsymbol{W}_r^{\bar{A}} + \boldsymbol{E}_s^{\bar{A}} \boldsymbol{x}_k,\ \boldsymbol{\lambda}^A(\boldsymbol{x}) \ge 0 \right\} \tag{B.9}$$

The set $\mathcal{X}_R$ itself is established by substituting $\boldsymbol{s}(\boldsymbol{x})$ into the inactive constraints ($^{\bar{A}}$) in (B.4) while maintaining the positivity of $\boldsymbol{\lambda}(\boldsymbol{x})$ corresponding to the active constraints (^{A}).

Having defined $\mathcal{X}_0$, the remainder of the state-space $\mathcal{X}_{rest} = \mathcal{X} - \mathcal{X}_0$ can be partitioned by utilizing the following theorem.

Theorem B.3 (cf. Bemporad, *et al.*, 2002): *Let* $\mathcal{X} \in \mathbb{R}^n$ *be a polyhedron, and* $\mathcal{X}_0$ *a polyhedron subset of* $\mathcal{X}$ *where* $\mathcal{X}_0 \neq \varnothing$. *Also let the polyhedral* $\mathcal{X}_i$ *be represented as:*

$$\mathcal{X}_i = \left\{ x \in \mathcal{X} \middle| \begin{matrix} \boldsymbol{R}_i \boldsymbol{x} > \boldsymbol{q}_i \\ \boldsymbol{R}_j \boldsymbol{x} \le \boldsymbol{q}_j \text{ for } \forall j < i \end{matrix} \right\} \text{for } i = 1, \ldots, \dim(\boldsymbol{q}) \tag{B.10}$$

and let $\mathcal{X}_{rest} = \bigcup_{i=1}^{\dim(q)} \mathcal{X}_i$. *Then, the following conditions apply: (i)* $\mathcal{X}_{rest} \cup \mathcal{X}_0 = \mathcal{X}$, *and (ii)* $\mathcal{X}_0 \cap \mathcal{X}_i = \varnothing$ and $\mathcal{X}_i \cap \mathcal{X}_j = \varnothing$ for $\forall i \neq j$; *i.e., the partitions* $\{\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_{\dim(q)}\}$ *form the entire polyhedron* $\mathcal{X}$.

Proof: see Bemporad, *et al.* (2002). □

Note that each polyhedral region $\mathcal{X}_i$ is basically bounded by l number of individual hyperplanes expressed in the linear equation representation $\boldsymbol{r}_l^T \boldsymbol{x} = q_l$ where $\boldsymbol{r}_l^T$ is the l-th row of $\boldsymbol{R}_j$ and q_l is the l-th element of $\boldsymbol{q}_j$ of (B.10). Neighboring and possibly non-neighboring regions may share the same hyperplanes. The optimal control law $\boldsymbol{\Pi}_H^* = g(\boldsymbol{x})$ corresponding to each partitioned region $\mathcal{X}_i$ can be directly determined from (B.3) and (B.8). The mpQP algorithm is then given as follows:

1. Obtain $\boldsymbol{x}_0$ by minimizing (B.4) with $\boldsymbol{x}$ as a free variable within a given space $\boldsymbol{x} \in \mathcal{X}$.
2. Compute $s(\boldsymbol{x}_0)$ and $\lambda(\boldsymbol{x}_0)$ from the solution of (B.4) with $\boldsymbol{x} = \boldsymbol{x}_0$.
3. Determine $s(\boldsymbol{x})$ and $\lambda(\boldsymbol{x})$ from (B.8).
4. Define $\mathcal{X}_R$ according to (B.9).
5. Obtain $\mathcal{X}_0$ by removing the redundant constraints from $\mathcal{X}_R$.
6. Set $\mathcal{X}_i = \mathcal{X}_0$ and define the corresponding control input function $g_i(\boldsymbol{x})$.
7. Define the remaining unexplored region $\mathcal{X}_{rest}$ according to Theorem B.3.
8. If $\mathcal{X}_{rest} \neq \varnothing$ repeat 1-7 for the remaining unexplored space $\mathcal{X}_{rest}$; else exit.

After partitioning a predefined polyhedral state-space into separate regions where each region corresponds to a linear control input function, a fast and efficient algorithm to pinpoint the region to which a particular state $\boldsymbol{x}_k$ belongs (during real-time controller execution) is required. This is achieved through a binary search tree algorithm and is derived in the next section.

B.3 Off-line Binary Search Tree Algorithm

Due to the convexity of the partitioned polyhedral regions, an off-line binary search tree can be built so that during run-time, a given particular state $\boldsymbol{x}_k$ can be efficiently associated

with its corresponding region and hence the control input function by just traversing the branches of the search tree. Each branching node will only require evaluating one linear inequality associated with one of the unique hyperplanes that define the region. The algorithm for building the binary search tree in this thesis is based on the algorithm proposed in Tøndel (2003). Enhancements have been made to the original algorithm mainly to improve the computational speed, better memory management, and lower memory requirements for both off-line and on-line execution.

B.3.1 Methodology

Consider n_R number of partitioned regions $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_{n_R}\} \in \mathcal{X}$, having their corresponding control input functions $g_1(\boldsymbol{x})$, $g_2(\boldsymbol{x}), \ldots, g_{n_R}(\boldsymbol{x})$. In the conventional MPC policy when the first optimal input sequence is used, there may be more than one region having the same control function. However, subsequent future control input sequences are used in the MPC strategies developed in this thesis rendering a unique control input function to be associated with each partitioned region. Let the n_h unique hyperplanes that define the region partitioning be designated as $h_l : \boldsymbol{r}_l^T \boldsymbol{x} = q_l$ for $l = 1, \ldots, n_h$. Define $d_l(\boldsymbol{x}) = \boldsymbol{r}_l^T \boldsymbol{x} - q_l$. Let the index representation $\mathcal{J}$ of a polyhedral sub-space $\mathcal{P}(\mathcal{J})$ of $\mathcal{X}$ (containing and/or intersecting one or more partitioned regions) indicate a combination of hyperplanes with the sign of $d_l(\boldsymbol{x})$. For example, $\mathcal{J} = \{h_2^+, h_5^-, h_6^-\}$ means a polyhedral sub-space is located at $d_2(\boldsymbol{x}) \geq 0$, $d_5(\boldsymbol{x}) \leq 0$, and $d_6(\boldsymbol{x}) \leq 0$. Let the set of polyhedral regions within $\mathcal{P}(\mathcal{J})$ be indexed as the set $\mathcal{I}(\mathcal{J}) = \{i \,|\, \mathcal{X}_i \cap \mathcal{P}(\mathcal{J}) \text{ is full dimensional}\}$. The intended mechanism of the binary search tree for the MPC controller is to evaluate $d_l(\boldsymbol{x}) = \boldsymbol{r}_l^T \boldsymbol{x} - q_l$ at each node to determine its sign for a given $\boldsymbol{x} \in \mathcal{X}$. Based on the signs of each node, the tree is traversed from the *root* node to an eventual *leaf* node which is associated to a particular region $\mathcal{X}_i$. Figure B.1 shows an illustrative example of an $\mathbb{R}^2$ state-space with 6 partitioned regions. The corresponding desired binary search tree is also shown with a total of 11 tree nodes and a tree depth of 3 layers.

Designate N_k as a node of the tree where k is the node number (not the sample time). A

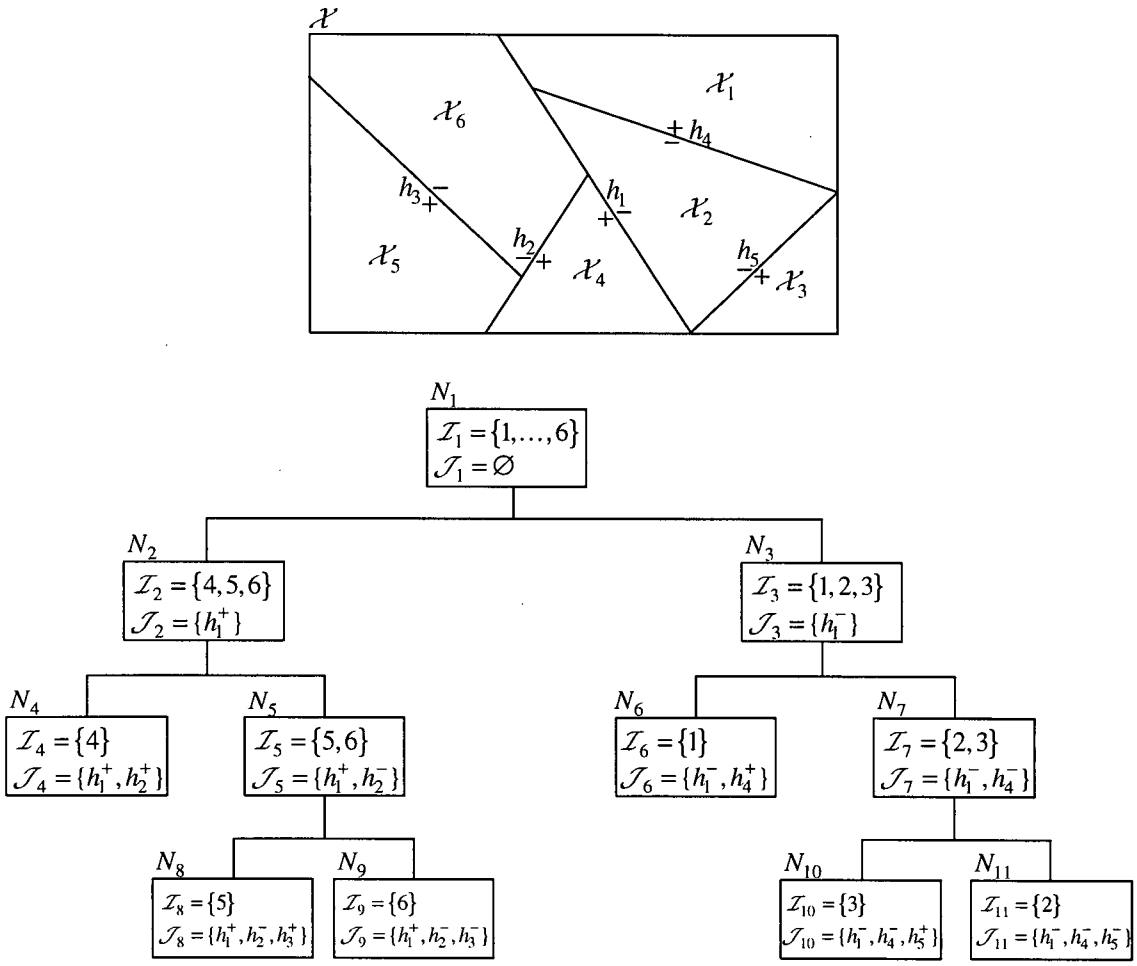


Figure B.1: An illustrative example of a partitioned polyhedron with its corresponding binary search tree.

set $\mathcal{U}_{ex}$ is used to maintain a list of indices of unexplored nodes. An unexplored non-leaf node N_k will contain $(\mathcal{I}_k, \mathcal{J}_k)$ in which $\mathcal{J}_k$ is the signed index set of hyperplanes gathered while traversing the search tree from the root node to N_k and $\mathcal{I}_k = \mathcal{I}(\mathcal{J}_k)$. It should be noted that $\mathcal{I}(\mathcal{J} \cup h_l^{\pm}) \subseteq \left(\mathcal{I}(\mathcal{J}) \cap \mathcal{I}(h_j^{\pm})\right)$. The following lemma distinguishes the difference between the two sets.

Lemma B.1 (cf. Tøndel, 2003, pg. 71) : *If* $i \in \mathcal{I}(\mathcal{J}) \cap \mathcal{I}(h_l^+)$ *but* $i \notin \mathcal{I}(\mathcal{J} \cup h_l^+)$, *then* $\mathcal{X}_i$ *is divided into two full-dimensional polyhedra by the hyperplane* h_l ; *i.e.,* $i \in \mathcal{I}(\mathcal{J}) \cap \mathcal{I}(h_l^+) \cap \mathcal{I}(h_l^-)$. *The same applies with* h_l^-.

The exact $\mathcal{I}_k^{\pm}$ can be determined by solving the two Linear Programming (LP) problems of $\min_{x\in\mathcal{X}_i} d_l(x)$ and $\max_{x\in\mathcal{X}_i} d_l(x)$ for each $i\in\mathcal{I}(\mathcal{J})\bigcap\mathcal{I}(h_l^+)\bigcap\mathcal{I}(h_l^-)$. By examining the signs of the optimal values of these two LP problems, the side of each hyperplane h_l a particular region $\mathcal{X}_i$ belongs to or intersects can be determined. An optimal value of zero indicates that the hyperplane is one of the faces of the polyhedral region. It should be noted that each hyperplane has its own "sense" of '+' and '−' directions independent of the states x. In the present thesis, the LP problems are solved using C-Library cddlib (Fukuda, 2003) in C++ programming language, which implements a dual simplex method.

The algorithm for building the binary search tree is given in Table B.1 in pseudocode form. $\mathcal{N}(\cdot)$ denotes the number of elements in a set. The memory requirement for this algorithm is large since each node and each hyperplane needs to store its associated regions. The number of regions and hyperplanes of an MPC controller typically grows in an exponential manner with the dimension of the system, the horizon length, and the imposed constraints. These numbers can be easily in the order of thousands for a low order system (see Section B.3.2) requiring hundreds of Megabytes of memory. In order not exceed the capacity of the random access memory (RAM) of the computer, without adversely affecting the computation speed, the less frequently accessed variables are stored in a binary file while the frequently accessed variables; e.g., $\mathcal{I}_k^{\pm}$, are allocated memory dynamically on an as needed basis. It can be observed from the binary search tree building algorithm that it contains two most computationally intensive double loops (lines 9-15 and 36-42) in which each cycle requires the optimization of the LP problems of $\min_{x\in\mathcal{X}_i} d_l(x)$ and $\max_{x\in\mathcal{X}_i} d_l(x)$. In order to reduce the computational time as well as the tree depth and hence the total number of nodes for a particular MPC controller, a minimum number of remaining regions $\mathcal{N}_{\min}(\mathcal{I}_k)$ for defining a leaf node is preset (instead of branching the search tree until each leaf node corresponds to one region, as shown in Figure B.1). During run-time of the real-time controller, a sequential search (see Section B.4) is carried out to find the target region (and hence the control input function) after traversing the search tree and reaching a leaf node. In the present work, the worst case sequential search time is less than the sampling period of the MPC controller (using a Pentium III 1 GHz processor) if $\mathcal{N}_{\min}(\mathcal{I}_k)$ is set below 250 regions.

Table B.1: Pseudocode for building the binary search tree of an MPC controller.

```
1   Load all partitioned regions into structured memory
2   Create files for temporary storage of index sets I(h_l^+) and I(h_l^-)
3   for (each region) {
4     for (each associating hyperplane) {
5       Sort, index and store all unique hyperplanes
6       Skip hyperplanes that define the state-space borders
7     }
8   }
9   for (each unique hyperplane) {
10    for (each region) {
11      Compute the index sets I(h_l^+) and I(h_l^-) by solving and comparing
12       the solutions to min_{x∈X_i} d_l(x) and max_{x∈X_i} d_l(x)
13      Included split regions in both index sets
14    }
15  }
16  for (each unique hyperplane) {
17    for (each region) {
18      Associate each unique hyperplane with its defining regions
19    }
20  }
21  Initialize the index set of the root node N_1 with I_1 = { all regions }
22  Initialize the set of unexplored nodes as U_ex = {N_1}
23  Set h_1 = 0 and the minimum N_min(I_k)
24  while (N(U_ex) > 0) {
25    Arbitrarily select and remove a node N_k from U_ex
26    Determine the set of hyperplanes corresponding to the regions in I_k
27    Sort the hyperplanes to be considered in ascending numerical order
28    for (each remaining hyperplane to be consider) {
29      Find the intersecting sets of I(J_k) ∩ I(h_l^+) and I(J_k) ∩ I(h_l^-)
30      Determine max (N(I(J_k) ∩ I(h_l^+)), N(I(J_k) ∩ I(h_l^-)))
31    }
32    Sort the remaining hyperplanes in ascending order of
33     max (N(I(J_k) ∩ I(h_l^+)), N(I(J_k) ∩ I(h_l^-)))
34    Determine the hyperplane candidates h_k's with
35     min max (N(I(J_k) ∩ I(h_l^+)), N(I(J_k) ∩ I(h_l^-))) from the sorted list
36    for (each hyperplane candidate) {
37      for (each region in the set I(J_k) ∩ I(h_l^+) ∩ I(h_l^-)) {
38        Find the sets I_k^+ = I(J_k ∪ h_l^+) and I_k^- = I(J_k ∪ h_l^-) by solving and
39         comparing the solutions to min_{x∈X_i} d_l(x) and max_{x∈X_i} d_l(x)
40      }
41      Rank the hyperplane candidates according to max (N(I_k^+), N(I_k^-))
42    }
43    Select the nodal hyperplane h_k with min max (N(I_k^+), N(I_k^-)) and associate
44     it with the current node N_k
45    Create 2 child nodes N_k^+ and N_k^- and set the current node N_k
46     as their parent
47    Store the set of regions I_k^+ and I_k^- in the respective child node
48    if (N(I_k^+) > N_min(I_k)) Add the child node N_k^+ into U_ex
49    if (N(I_k^-) > N_min(I_k)) Add the child node N_k^- into U_ex
50  } // end of while
51  Save the binary search tree in a pre-specified text format
```

Table B.2 shows an excerpt from the text format for saving the binary search tree of an MPC controller used in this thesis. Each node is stored in a line where the first 4 numerical entries are the corresponding nodal hyperplane, the 2 child nodes, and the number of regions $\mathcal{N}(\mathcal{I}_k)$, respectively. The subsequent entries are the indices of the regions. Each hyperplane is stored in a line in the form $\begin{bmatrix} \boldsymbol{r}_l^T & q_l \end{bmatrix}$ while each region is stored in the form $\begin{bmatrix} \boldsymbol{R}_i & \boldsymbol{q}_i \end{bmatrix}$ where the number after the text "::R1:Rq" indicates the number of rows of $\begin{bmatrix} \boldsymbol{R}_i & \boldsymbol{q}_i \end{bmatrix}$. The corresponding control input function is located after each region, in the form $\begin{bmatrix} \boldsymbol{H}_g^i & \boldsymbol{k}_g^i \end{bmatrix}$.

Table B.2: An excerpt from the text format for saving a binary search tree of an MPC controller.

```
@// Controller parameters
::Horizon length = 15
::Number of states = 4
::Number of inputs = 1
::Number of nodes = 909
::Number of hyperplanes = 18896
::Number of regions = 6154
// Nodes (Depth = 9)
::N1: 10440 2 3 6154 1 2 3 4 5 6 7 8 ··· 6150 6151 6152 6153 6154
  ⋮
::N909: 0 0 0 75 4630 4832 5217 5218 ··· 6062 6063 6066 6067 6068 6077
// Hyperplanes
::H1: -0.488419 -0.115701 -0.651252 0.569149 0.031239
  ⋮
::H18896: -0.352372 -0.175715 0.700649 -0.595020 -0.058510
// Regions
::R1:Rq24
0.000000 1.000000 0.000000 0.000000 1.000000
0.000000 -1.000000 0.000000 0.000000 1.000000
  ⋮
-0.136572 -0.024657 0.717317 -0.682785 0.065461
-0.334505 0.020349 0.558848 -0.758538 0.055683
::R1:Hk
-3.127023 -0.740758 -4.169526 3.643878 0.000000
-1.912031 -0.570619 -2.999832 3.044070 0.000000
  ⋮
2.683680 0.291840 1.710521 -0.410551 0.000000
2.987808 0.331897 1.966618 -0.538040 0.000000
::R2:Rq13
  ⋮
::R6154:Rq6
0.000000 0.000000 0.000000 -1.000000 1.000000
-0.276432 0.028682 0.582242 -0.764039 0.056276
  ⋮
0.296278 -0.028400 -0.566065 0.768753 -0.056192
::R6154:Hk
0.000000 -0.000000 -0.000000 0.000000 -0.199999
  ⋮
-0.00000 0.000000 0.000000 -0.000000 0.199999
```

B.3.2 Computational Complexity

The complexity of the binary search tree mainly depends on the number of states $\boldsymbol{x} \in \mathbb{R}^n$, the number of inputs $\boldsymbol{u} \in \mathbb{R}^m$, the length of the prediction horizon H, and the number of imposed constraints. Table B.3 shows a simple comparison of the time required to build the binary search trees subjected to different lengths of prediction horizon and state-input constraints, for controlling one axis of the electro-hydraulic manipulator system of the fish processing machine, which is experimental platform used in the present thesis (see chapters 2 and 4). The minimum number of remaining regions in leaf nodes is set as $\mathcal{N}_{\min}(\mathcal{I}_k) = 100$. Input and state constraints are of the forms $-c_1 \le u_i \le c_2$ and $-c_3 \le x_i \le c_4$, respectively. The binary search trees building algorithm is coded in C/C++ programming language for optimal speed (approximately 30 times faster than the identical algorithm coded in Matlab®) and they are executed on a computer with a Pentium IV 2.2 GHz processor and 1 GB of memory. It can be observed that the number of regions and hyperplanes increase rapidly with the length of the prediction horizon and more rapidly with the number of imposed constraints. Computation time increases exponentially with the number of regions and hyperplanes. Hence, although able to achieve very fast on-line optimization of the MPC control policy, the concept of partitioning the state-space using the method of mpQP and then constructing a binary search tree may not be attractive for higher order servo systems.

Table B.3: Comparison of the computation time requirements for building binary search trees subjected to different complexities of an MPC controller for a 4-states-1-input system.

Horizon length	No. of regions	No. of hyperplanes	No. of inputs constrained	No. of states constrained	Time taken
13	1411	4052	1	0	9 mins.
15	1589	4274	1	0	12 mins.
10	2271	13009	1	1	2 hrs.
10	3115	14194	1	2	4 hrs.
13	10358	51106	1	2	13 hrs.
18	52948	65512	1	2	15 days

B.4 On-line Traversing of the Binary Search Tree

Table B.4 gives the pseudo-coded on-line algorithm for arriving at the right optimal control input function for a given state vector $\boldsymbol{x}$. The binary search tree is first traversed from the root node to a leaf node containing a set of regions in which one of them encloses the particular state vector $\boldsymbol{x}$. The target region is then determined by sequentially testing each of the remaining regions. Note that the binary search tree is read in advance during controller initialization.

Table B.4: Pseudocode for traversing and sequential searching of the binary search tree during controller operation.

```
1   Read the current state x
2   Initialize the starting node to the root node N_1
3   // Traverse the binary search tree
4   while (N_k is not a leaf node) {
5     Evaluate the distance d_l(x)=r_l^T x-q_l between the nodal hyperplane and x
6     if (d_l(x)>0)
7       Set N_{k+1} to the corresponding child node N_k^+
8     else
9       Set N_{k+1} to the corresponding child node N_k^-
10  }
11  // Sequential search through the remaining regions
12  for (each remaining region N(I_k)) {
13    if (R_i x-q_i ≤ 0) {
14      Evaluated the optimal control input function Π_H^* = g(x)=H_g^i x+k_g^i
15      break
16    }
17  }
```