

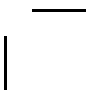


Real-Time Control Systems with Delays

Johan Nilsson

Department of Automatic Control, Lund Institute of Technology

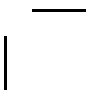






Real-Time Control Systems with Delays









Real-Time Control Systems with Delays

Johan Nilsson

Lund 1998



To my parents

Published by
Department of Automatic Control
Lund Institute of Technology
Box 118
S-221 00 LUND
Sweden

ISSN 0280-5316
ISRN LUTFD2/TFRT--1049--SE

©1998 by Johan Nilsson
All rights reserved

Printed in Sweden by Lunds Offset AB
Lund 1998

Contents

| | |
|------------------------------------------------------------------|-----------|
| Acknowledgments | 7 |
| 1. Introduction | 8 |
| Outline of the Thesis and Publications | 9 |
| 2. Problem Formulation | 13 |
| 2.1 Distributed Control | 13 |
| 2.2 Networks | 18 |
| 2.3 Clock Synchronization | 20 |
| 2.4 Related Work | 23 |
| 3. Modeling of Network Delays | 29 |
| 3.1 Network Modeled as Constant Delay | 29 |
| 3.2 Network Modeled as Delays Being Independent | 30 |
| 3.3 Network Modeled Using Markov Chain | 31 |
| 3.4 Sampling of Systems with Network Delays | 31 |
| 4. Experimental Delay Measurements | 35 |
| 4.1 CAN | 35 |
| 4.2 Ethernet Network | 50 |
| 4.3 Summary | 54 |
| 5. Analysis and Control with Independent Delays | 56 |
| 5.1 Closed Loop System | 56 |
| 5.2 Analysis | 58 |
| 5.3 Simulation of Systems with Network Delays | 62 |
| 5.4 Optimal Stochastic Control | 63 |
| 5.5 Example | 72 |
| 5.6 Summary | 73 |

Contents

| | |
|--------------------------------------------------------------|-----|
| 6. Analysis and Control with Markov Delays | 76 |
| 6.1 Setup | 76 |
| 6.2 Analysis | 79 |
| 6.3 Proof of Theorem 6.1 | 89 |
| 6.4 Optimal Stochastic Control | 92 |
| 6.5 Proof of Theorem 6.4 | 98 |
| 6.6 Summary | 101 |
| 7. Special Topics | 102 |
| 7.1 Markov Chain with Two Transitions Every Sample | 103 |
| 7.2 Sampling Interval Jitter | 106 |
| 7.3 Estimation of Markov State | 110 |
| 7.4 The MIMO Problem | 111 |
| 7.5 Timeout | 116 |
| 7.6 Timeout and Vacant Sampling | 120 |
| 7.7 Asynchronous Loops | 122 |
| 8. Conclusions | 125 |
| 9. References | 129 |
| A. Kronecker Products | 133 |
| A.1 Definitions | 133 |
| A.2 Basic Rules of Calculation | 134 |
| B. Some Results from Probability Theory | 135 |
| B.1 Markov Chains | 135 |
| B.2 Conditional Independence | 136 |

Acknowledgments

I would like to take the opportunity to thank all friends and colleagues that have given me support and encouragement in my thesis work. Thank you!

My work has been supervised by Bo Bernhardsson and Björn Wittenmark. During the work they have continuously provided me with enthusiasm, vision, and wisdom. They are outstanding researchers to have as role models for a PhD-student. During my time at the department and during the work with this thesis I have also been inspired by Karl Johan Åström, who is the most skilled engineer I have met.

I would like to express my sincere gratitude to the group at the department working in the field of real-time systems. My own research has benefited from them. I would especially like to mention Karl-Erik Årzén, Anders Blomdell, Klas Nilsson, Leif Andersson, and Johan Eker.

I have had the good fortune to work in a very inspiring environment. It has been a privilege to work together with the intelligent and friendly colleagues at the department. Especially I would like to thank Lennart Andersson, Johan Eker, Karl Henrik Johansson, Ulf Jönsson, Jörgen Malmberg, Erik Möllerstedt, and Henrik Olsson, with whom I have spent a lot of time both at work and in leisure hours.

I would like to thank the support staff at the department. Throughout the years they have been very helpful. Excellent computer facilities have been provided by Leif Andersson and Anders Blomdell. They have also served as excellent teachers for me in computer science. The secretariat of the department has always been run with perfectionism, especially I would like to mention Eva Schildt for doing a great job. Rolf Braun has always been very helpful with the laboratory work.

The CAN experimental setup was borrowed from DAMEK Mechatronics Division at the Royal Institute of Technology. I would like to thank Martin Törngren and Ola Redell at DAMEK for helping me getting the CAN-platform running.

The work has been partly supported by the Swedish National Board for Industrial and Technical Development (NUTEK), Project Dicosmos, Contract 93-3485.

Finally, I am indebted to my parents and family for always supporting and encouraging me.

J.N.

1

Introduction

Control loops that are closed over a communication network get more and more common as the hardware devices for network and network nodes become cheaper. A control system communicating with sensors and actuators over a communication network will be called a *distributed real-time control system*. In distributed real-time control systems, see Figure 1.1, data are sent and received by network nodes of different kind and manufacturers. Network nodes that are of specific interest for distributed control are sensor nodes, actuator nodes, and controller nodes. Sensor nodes measure process values and transmit these over the communication network. Actuator nodes receive new values for the process inputs over the communication network and apply these on the process input. Controller nodes read process values from sensor nodes. Using a control algorithm control signals are calculated and sent to the actuator nodes. The system setup with a common communication network reduces cost of cabling and offers modularity and flexibility in system design.

The distributed control setup is powerful, but some caution must be taken. Communication networks inevitably introduce delays, both due to limited bandwidth, but also due to overhead in the communicating nodes and in the network. The delays will in many systems be varying in a random fashion. From a control perspective the control system with varying delays will no longer be time-invariant. As an effect of this the standard computer control theory can not be used in analysis and design of distributed real-time control systems. The thesis addresses the problem of analysis and design of control systems when the communication delays are varying in a random fashion. Models for communication network delays are developed. The most advanced model has an underlying Markov chain that generates the probability distributions of the time delays. Measurements of transfer delays are presented for two commercially used networks, a CAN-network (Controller Area Network) and an Ethernet network. For the derived network models closed loop stability and evaluation

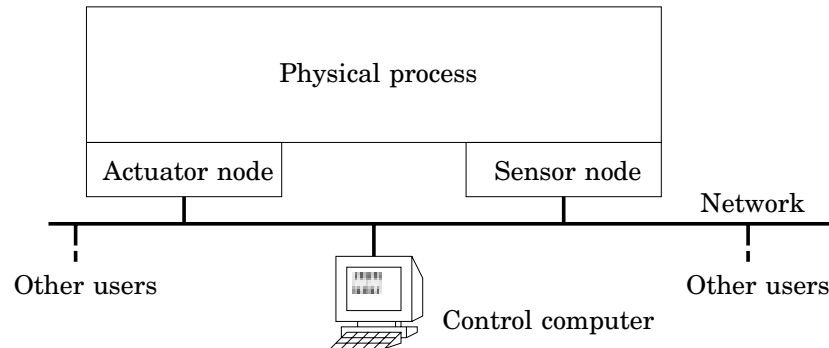


Figure 1.1 Distributed real-time control system with sensor node, controller node, and actuator node. The communication network is also used for other applications in the system.

of a quadratic cost function are analyzed. The LQG-optimal controller is derived both for a network where the time delays are independent from sample to sample, and for a network where the probability distribution functions for the delays are governed by an underlying Markov chain. The derived controllers use knowledge of old time delays. This can be achieved by so called “timestamping”, all transferred signals are marked with the time they were generated. By comparing the “timestamp” with the internal clock of the controller the time delay can be calculated. It is shown that the optimal controller is the combination of a state feedback controller and a Kalman filter, i.e., the separation principle applies.

Outline of the Thesis and Publications

The contents of the thesis are as follows:

Chapter 2: Problem Formulation

This chapter gives an introduction to the problem formulation. A short review of clock synchronization and networks for distributed control is also presented. The chapter is concluded with a summary of work related to this thesis.

Our first study of the problem with varying network delays are published in

WITTENMARK, B., J. NILSSON, and M. TÖRNGREN (1995): “Timing problems in real-time control systems.” In *Proceedings of the 1995 American Control Conference, Seattle, Washington*.

Chapter 1. Introduction

The problem with distributed real-time control systems was also studied in my licentiate thesis

NILSSON, J (1996): "Analysis and design of real-time systems with random delays." Report ISRN LUTFD2/TFRT--3215--SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Chapter 3: Modeling of Network Delays

Two models for the network delays are developed. The first model is memoryless and assumes that the delays have a constant probability distribution function. The second model assumes that the probability distribution functions for the delays are given by an underlying Markov chain. The models are well suited for analysis and design of distributed real-time control systems.

The network models were first developed in the licentiate thesis

NILSSON, J (1996): "Analysis and design of real-time systems with random delays." Report ISRN LUTFD2/TFRT--3215--SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Chapter 4: Experimental Delay Measurements

Network delay measurements are presented for two commercial networks, CAN (Controller Area Network) and Ethernet. The networks are shown to have more or less, depending on load situation, random delays for transmissions. This chapter serves as a motivation for studies of the control problem outlined in Chapter 2, and for use of the models derived in Chapter 3.

Chapter 5: Analysis and Control with Independent Delays

In this chapter the delays are assumed to have a constant probability distribution function. The network delay model is derived in Chapter 3. Results are developed to determine system stability and values of quadratic cost functions given a proposed controller. The LQG-optimal stochastic control problem is solved. Some examples are given.

The results of this chapter are published in

NILSSON, J., B. BERNHARDSSON, and B. WITTENMARK (1996): "Stochastic analysis and control of real-time systems with random time delays." In *Proceedings of the 13th International Federation of Automatic Control World Congress, San Francisco, California*.

NILSSON, J (1996): "Analysis and design of real-time systems with random delays." Report ISRN LUTFD2/TFRT--3215--SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Outline of the Thesis and Publications

NILSSON, J., B. BERNHARDSSON, and B. WITTENMARK (1998): “Stochastic analysis and control of real-time systems with random time delays.” *Automatica*, 34:1.

Chapter 6: Analysis and Control with Markov Delays

In this chapter we study a network model having a Markov chain that gives the probability distribution functions for the delays. Results for evaluation of covariances and mean square stability are derived. The LQG-optimal controller is derived for the Markov communication network. Examples are given along with the theoretical presentation.

The analysis results are published in

NILSSON, J (1996): “Analysis and design of real-time systems with random delays.” Report ISRN LUTFD2/TFRT--3215--SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

NILSSON, J., and B. BERNHARDSSON (1996): “Analysis of real-time control systems with time delays.” In *Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan*.

and the LQG-control results are published in

NILSSON, J., and B. BERNHARDSSON (1997): “LQG control over a Markov communication network.” In *Proceedings of the 36th IEEE Conference on Decision and Control, San Diego, California*.

NILSSON, J., and B. BERNHARDSSON (1997): “LQG control over a Markov communication network.” Submitted for journal publication.

Chapter 7: Special Topics

This chapter contains several subproblems that are connected to the thesis. Some are extensions to the theory developed in Chapter 5 and Chapter 6, including sampling interval jitter and setups with multiple sensors and actuators. A problem that is related to the delay in measurements is the use of “timeouts”. We present and analyze a controller that uses a timeout for waiting on a new measurement. Time variations resulting from use of asynchronous loops are also studied.

The results in this chapter are mainly from

NILSSON, J., B. BERNHARDSSON, and B. WITTENMARK (1997): “Some topics in real-time control.” Submitted for conference publication.

Chapter 8: Conclusions

In the last chapter conclusions are presented. Extensions and open problems are discussed.

Chapter 1. Introduction

Appendices

The thesis is concluded with two appendices, one on Kronecker products, and one containing some results from probability theory.

2

Problem Formulation

2.1 Distributed Control

We will study the closed loop system depicted in Figure 2.1. The actuators and sensors are connected to a communication network. These units receive respectively send control information to the centralized controller. The centralized controller is connected to the network, and communicates with sensors and actuators by sending messages over the network. Sending a message over a network typically takes some time. Depending on the network and scheduling policy in the system this transfer time can have different characteristics. The transfer time can in some setups be nearly constant, but in many cases it is varying in a random fashion. The length of the transfer delay can, for instance, depend on the network load, priorities of the other ongoing communications, and electrical disturbances, Ray (1987). Depending on how the sensor, actuator, and controller nodes are synchronized several setups can be found. Several previous authors have suggested control schemes with slightly different timing setups. The different setups come from whether a node is event-driven or clock-driven. By event-driven we mean that the node starts its activity when an event occurs, for instance, when it receives information from another node over the data network. Clock-driven means that the node starts its activity at a prespecified time, for instance, the node can run periodically. There are essentially three kinds of computer delays in the system, see Figure 2.1:

- Communication delay between the sensor and the controller, τ_k^{sc} .
- Computational delay in the controller, τ_k^c .
- Communication delay between the controller and the actuator, τ_k^{ca} .

The subscript k is used to indicate a possible time dependence of the

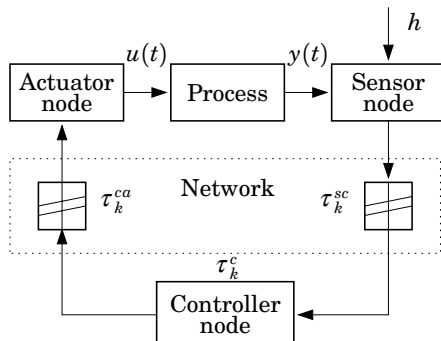


Figure 2.1 Distributed digital control system with induced delays, τ_k^{sc} and τ_k^{ca} . The computational delay in the controller node, τ_k^c , is also indicated.

delays.

The *control delay*, τ_k , for the control system, the time from when a measurement signal is sampled to when it is used in the actuator, equals the sum of these delays, i.e., $\tau_k = \tau_k^{sc} + \tau_k^c + \tau_k^{ca}$.

One important problem in this control system setup is the delays, which are varying in a random fashion. This makes the system time-varying and theoretical results for analysis and design for time-invariant systems can not be used directly. One way to get rid of the time variations is to introduce clocked buffers on the input in the controller node and the actuator node. If these buffers are chosen large enough, larger than the worst case delay, the delay for a transfer between two nodes is deterministic. This scheme was proposed in e.g. Luck and Ray (1990). Introduction of buffers in the loop means that we sometimes are using older information than we need to. It is shown in Chapter 5 that this can lead to a degradation of performance in comparison with an event-driven setup.

From a sampled-data control perspective it is natural to sample the process output equidistantly with a sample period of h . It is also natural to keep the control delay as short as possible. The reason is that time delays give rise to phase lag, which often degenerate system stability and performance. This motivation suggests a system setup with event-driven controller node and event-driven actuator node, which means that calculation of the new control signal respectively D/A-conversion of the new control signal takes place as soon as the new information arrives from the sensor node and the controller node respectively. The timing in such a system is illustrated in Figure 2.2. A drawback with this setup

is that the system becomes time-varying. This is seen from Figure 2.2 in that the process input is changed at irregular times.

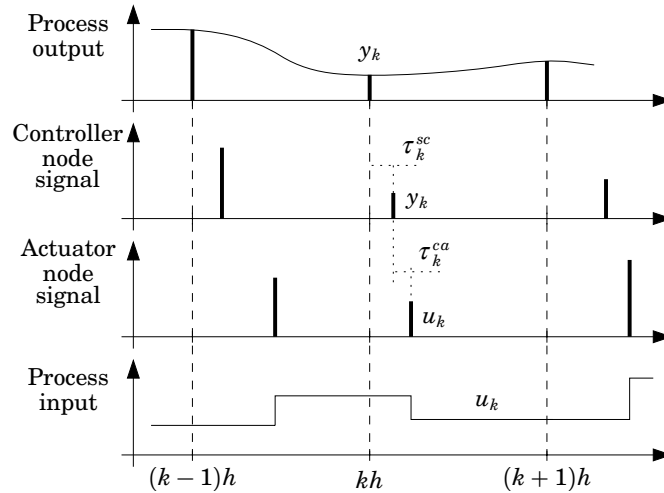


Figure 2.2 Timing of signals in the control system. The first diagram illustrates the process output and the sampling instants, the second diagram illustrates the signal into the controller node, the third diagram illustrates the signal into the actuator node, and the fourth diagram illustrates the process input, compare with Figure 2.1.

In the subsequent chapters we will analyze and design controllers with an equidistantly sampling sensor node and event-driven controller and actuator nodes. We will also make the assumption that the control delay is less than the sampling period, i.e., $\tau_k \leq h$. This can be motivated in several ways. From a control point of view, a normal design usually has $0.2 \leq \omega h \leq 0.6$, where h is the sampling period and ω is the natural frequency of the closed loop system, see Åström and Wittenmark (1997). With a delay equal to one sample this design has a phase lag induced by the controller, ϕ_{lc} , of $11^\circ \leq \phi_{lc} \leq 34^\circ$. An even larger phase lag would make many processes hazardous to control. If we have a larger control delay than the sampling period, h , samples may arrive in a non-chronological order at the actuator-node. This would make both implementation of algorithms and system analysis much harder. The condition that the control delay is less than h can be replaced with the assumption that the control delay may not vary more than h , which also guarantees that samples arrive in chronological order. In Sections 7.5–7.6 we will look at a controller that does not need the assumption that the delay variation is less than h . This controller uses a timeout for the arrival of a new measurement. If a

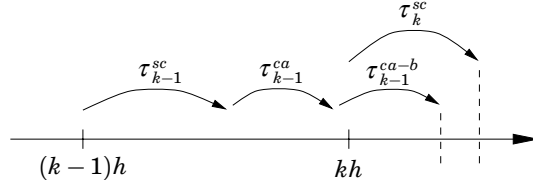


Figure 2.3 Timing plot showing delays during a clock cycle.

timeout occurs, control signal calculation is done based on prediction. A control scheme with this feature is motivated if the probability function of the delays have long “tails”. The controller can also be used to detect, and control, when we lose samples. This phenomena is called *vacant sampling*.

We will in the following only look at the influences from τ_k^{sc} and τ_k^{ca} . The effect of τ_k^c can be embedded in τ_k^{ca} . We will also assume that we have the knowledge of how large the previous transfer delays in the loop were. Ways to implement this feature is discussed in the sequel.

Knowledge of Old Time Delays – Timestamps

Using synchronized clocks in the nodes, delay information can be extracted by including the time of generation (timestamp) to every message. Clock synchronization is discussed in Section 2.3. In most networks the extra network load introduced by the timestamp is negligible in comparison with message and network overhead. The delay information can be used in the controller node. The timestamp will tell the controller how old the received measurement is. This information can then be used by the controller in the control signal calculation.

The controller node can easily calculate τ_k^{sc} by comparing the timestamp of the measurement with the internal clock of the controller node. The controller can also obtain information about τ_{k-1}^{ca} . In many network implementations it is possible to get information from the network interface when the last control signal was sent on the network. If this information is available, the transfer delay τ_{k-1}^{ca} is known when u_k is calculated. Another way to achieve propagation of τ_{k-1}^{ca} to the controller is to immediately send a message back from the actuator-node to the controller containing the transfer time τ_{k-1}^{ca} . It is, however, not certain that the controller will have received this message when the next control signal is to be calculated. The timing for the control system between two control signal calculations is shown in Figure 2.3. We have here introduced the transfer delay τ_k^{ca-b} for the transfer-time to send back information about the length of τ_{k-1}^{ca} to the controller. The condition that the τ_{k-1}^{ca} is known when we calculate the control signal can in the general case be written

2.1 Distributed Control

as

$$\tau_{k-1}^{sc} + \tau_{k-1}^{ca} + \tau_{k-1}^{ca-b} < \tau_k^{sc} + h. \quad (2.1)$$

We will assume that the sum of the time delays in one control cycle, the control delay, is less than the sampling interval h . As the control delay is the sum of τ_k^{sc} and τ_k^{ca} , it is reasonable to assume that each of the delays is distributed on $[0, \alpha h]$, where $\alpha < 0.5$. The problem of guaranteed knowledge of old time delays will be analyzed for some special cases of communication network behavior.

Total randomness If the communication network gives time delays that are random and independent it is seen from (2.1) that the condition on the distribution of the time delays is

$$\alpha < \frac{1}{3}. \quad (2.2)$$

Total order If we add the requirement that network messages are transmitted in the order they were generated, the message containing the length of τ_k^{ca} will always be sent before the next measurement if

$$\alpha < \frac{1}{2}. \quad (2.3)$$

An implementation of such a system would require something like a global queue for messages to be sent in the network.

Priority Some communication networks have the possibility to give messages priorities and guarantee that waiting messages are sent in priority order. In such a system we can give the message containing τ_{k-1}^{ca} a higher priority than the message containing the new measurement y_k . This guarantees knowledge of τ_{k-1}^{ca} if

$$\alpha < \frac{1}{2}. \quad (2.4)$$

Summary of Assumptions

We will make the following assumptions about the control system:

- The sensor node is time-driven. The output of the process is sampled periodically without any scheduling disturbances. The sampling period is h .
- The controller node is event-driven. The control signal is calculated as soon as the sensor data arrives at the controller node.

Chapter 2. Problem Formulation

- The actuator node is event-driven. The control signal is applied to the process as soon as the data arrives at the actuator node.
- The communication delays τ_k^{sc} and τ_k^{ca} are randomly varying with known stochastic properties. The variation of the total time delay, $\tau_k^{sc} + \tau_k^{ca}$, is less than one sampling interval.
- The lengths of the past time delays are known to the controller.

Several models for the time delays will be studied. These are further discussed in Chapter 3. In Chapter 7 some of the assumptions will be relaxed. A controller not needing a constant sampling interval will be investigated in Section 7.2. A controller that can handle longer delays than a sampling interval is developed in Section 7.5. The controller is based on timeout for new measurements. The controller node will in this case sometimes be event-driven, and sometimes clock-driven. This controller can also be used when we have vacant samples.

2.2 Networks

Communication networks were introduced in digital control systems in the 1970's. At that time the driving force was the car industry. The motives for introducing communication networks were reduced cost for cabling, modularization of systems, and flexibility in system setup. Since then, several types of communication networks have been developed. Communication protocols can be grouped into *fieldbuses* (e.g. FIP and PROFIBUS), *automotive buses* (e.g. CAN), "other" *machine buses* (e.g. 1553B and the IEC train communication network), *general purpose networks* (e.g. IEEE LAN's and ATM-LAN) and a number of *research protocols* (e.g. TTP), see Törngren (1995). Fieldbuses are intended for real-time control applications, but in some applications other networks may have to be used for control. For instance, if another network already is used for other functions it could be cost effective to use this network for control too. The fieldbuses are usually only made for connection of low-level devices. If high-level function, for instance, a work station, is to be connected, other networks may be more suitable. There is vast number of communication protocols and fieldbuses. A short summary is now given of some of the most used fieldbuses, see also Olsson and Piani (1992) and Tindell and Hansson (1995).

Foundation Fieldbus

The Foundation Fieldbus was developed by the organization Fieldbus Foundation, a not for profit organization with over 100 member compa-

nies, including several major international automation companies. Foundation Fieldbus is released for two speeds, 31.25 kbit/s, and 1 Mbit/s. A faster bus with bus speed 2.5 Mbit/s, is announced. The low speed bus, 31.25 kbit/s, is intended for replacement of traditional 4 – 20 mA analog signals, without changing the wiring. Each bus can have 32 devices. By use of bridges a hierarchical network topology can be built. Using a hierarchical network structure more devices can be connected. Access to the bus is controlled by a centralized bus scheduler called the *Link Active Scheduler*, *LAS*. During configuration of the fieldbus all devices on the bus will inform the LAS which data it needs, and at which times the data is needed. During runtime the LAS will tell the devices to broadcast data to the bus using a schedule. All subscribers to this data will receive it simultaneously. Spare time is reserved in the schedule for unscheduled messages. A system global clock is also distributed on the fieldbus. The distributed clock will allow connected devices to know the time within 1 ms.

FIP (Factory Instrumentation Protocol)

FIP was developed by a group of French, German, and Italian companies. FIP uses a twisted pair conductor and the transmission speeds are from 31.25 kbit/s up to 2.5 Mbit/s, depending on the spatial dimension of the bus. For a transmission speed of 1 Mbit/s the maximum length of the bus is 500 m. The maximum number of nodes in a FIP network is 256.

In a FIP-network one node acts as *bus arbitrator*. The *bus arbitrator* cyclically polls all nodes in the network to broadcast its data on the network. The inactive nodes listen to the communication and recognize when data of interest to the node is sent. The FIP-network can be seen as a distributed database, where the database is updated periodically.

PROFIBUS (Process Fieldbus)

PROFIBUS was developed by a group of German companies and is now a German standard. A screened twisted pair is used as conductor. The transfer speed can be from 9.6 kbit/s to 500 kbit/s. The maximum length of the bus is 1200 m. Up to 127 stations can be connected to the network. PROFIBUS messages can be up to 256 bytes long. PROFIBUS is a token-passing network. The nodes are divided into *active* and *passive* nodes. The node which holds the token has the permission to send data on the network. The token is passed around in the network between the *active* nodes. *Active* nodes can transmit when they hold the token. *Passive* nodes need to be addressed by an *active* node to be allowed to send data on the network.

CAN (Controller Area Network)

CAN was developed by the German company Bosch for the automation industry. CAN was one of the first fieldbuses and is now in use in cars from several manufacturers. CAN is defined in the ISO standards 11898 and 11519-1. The transfer speed on the bus can be programmed. The transfer speed can be 1 Mbit/s if the bus is no longer than 50 m, and 500 kbit/s if the bus is longer than 50 m. If the cable quality is low, as it can be in mass produced cars, the maximum transfer speed may be lower. There is no limit on the number of nodes. A node can start transmitting at any time if the bus is silent. If several nodes are trying to transmit an arbitration starts. The node trying to send the message with highest priority gets the right to use the bus. There are 2^{29} different priority levels for messages. CAN-controllers can usually be programmed to cause an interrupt when a message is sent. This feature makes back-propagation of the size of the delay from controller to actuator, τ_k^{ca} , simple to implement.

Ethernet

Ethernet is one of the most used *local area network* (LAN) technologies. It transmits data with the speeds 10 Mbit/s or 100 Mbit/s. Ethernet is not intended for real-time communications. However, the large number of installed Ethernets will make it attractive for use in real-time control systems. There is no central bus controller, instead Ethernet uses a bus access method called CSMA/CD, Carrier Sense Multiple Access with Collision Detection. This means that before sending to the network the station listens to the channel, and when the channel appears to be idle transmission starts. If several stations start sending to the bus the collision is detected, and the colliding stations back off, and try a retransmission after a random wait. An almost unlimited number of stations can be connected to an Ethernet. The number of stations is limited by the six bytes address. The first three bytes are used as a vendor ID, and the last three bytes are defined by the vendor, so every Ethernet interface has a unique address. An Ethernet frame, or packet, is between 64 and roughly 1500 bytes in length. For details about Ethernet, see IEEE (1985). In Section 4.2 delay measurement experiments using an Ethernet network are presented.

2.3 Clock Synchronization

Clock synchronization is a research area in itself. The purpose of clock synchronization is to give the internal clocks of two or more nodes corresponding values. We will only consider software synchronization, i.e., the synchronization signals are sent over the communication network.

2.3 Clock Synchronization

Hardware synchronization is also a possibility, for instance, using special wiring just to distribute a global clock signal in the system. As all communication between nodes is over the data network, all messages between nodes will be subject to random delays. Several schemes for synchronization are available in the literature, see Christian and Fetzer (1994), van Oorschot (1993), Schedl (1996). Some of the algorithms are designed for special application areas. One example is NTP, which is used for clock synchronization on Internet, see Mills (1991).

Most schemes build on the concept of estimating the difference between the clocks of two nodes by sending clock-requests back and forth as described in the following. Let S be the node which wants to estimate its clock difference to node R . Let the absolute time be t_i , let the local time in node S be t_i^S , and let the local time in node R be t_i^R . The local clocks in S and R have a skew to the absolute time such that

$$t_i^S = t_i + \delta^S \quad (2.5)$$

$$t_i^R = t_i + \delta^R, \quad (2.6)$$

where δ^S and δ^R are the clock mismatches. We define the clock offset, δ , as

$$\delta = \delta^R - \delta^S. \quad (2.7)$$

From (2.5) and (2.6) it follows that

$$t_i^S = t_i^R - \delta. \quad (2.8)$$

The clock offset will have a drift in time due to inaccuracies in the local clocks. For the moment we assume that δ is constant. The synchronization sequence starts with a clock-read request from node S to node R , this message is sent at time t_a^S , see Figure 2.4. As node R receives the message from node S it immediately sends a message back containing the local clock value t_b^R . This message arrives at node S at time t_c^S . Introduce T_{SR}

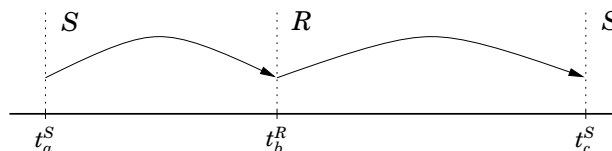


Figure 2.4 Clock synchronization sequence. First a request is sent from node S to node R , then R responds by sending the value of its local clock to S .

Chapter 2. Problem Formulation

and T_{RS} as the transfer times for the from S to R and from R to S , respectively. The transfer times can be written as

$$T_{SR} = t_b^S - t_a^S = (t_b^R - \delta) - t_a^S \quad (2.9)$$

$$T_{RS} = t_c^S - t_b^S = t_c^S - (t_b^R - \delta). \quad (2.10)$$

Assuming that $E(T_{SR} - T_{RS}) = 0$ we find that

$$\delta = E \left\{ \frac{2t_b^R - t_a^S - t_c^S}{2} \right\}, \quad (2.11)$$

where E denotes the expectation operator. By repeating the clock synchronization experiment we can find an accurate estimate of δ by (2.11). There are other clock synchronization algorithms not depending on the assumption $E(T_{SR} - T_{RS}) = 0$, see van Oorschot (1993). There are also algorithms addressing fault-tolerant synchronization. These faults can be failures in the network, in a local clock etc., see Christian and Fetzer (1994). If the clock offset, δ , is drifting due to inaccuracies in the local clocks, resynchronization must be done after a while. The drift in a clock is often defined as clock drift ρ . If $H(t)$ is the value of the clock at time t , the following holds for a time interval $[s, t]$:

$$(1 - \rho)(t - s) \leq H(t) - H(s) \leq (1 + \rho)(t - s). \quad (2.12)$$

It is claimed in Christian and Fetzer (1994) that clocks in modern computers have ρ of the order 10^{-5} or 10^{-6} , and high precision quartz clocks have ρ of the order 10^{-7} or 10^{-8} . With $\rho = 10^{-6}$ we would over one hour have

$$1\text{h} - 3.6\text{ms} \leq H(t + 1\text{h}) - H(t) \leq 1\text{h} + 3.6\text{ms}. \quad (2.13)$$

These drift values have to be compared with the time scales in the actual system to determine how often resynchronization must be done to keep an acceptable clock synchronization in the system. A simple way to estimate the drift is by calculating how much the clock offset has changed between two resynchronizations. In Section 4.1 a new clock synchronization algorithm is developed. The new algorithm can be used for both on-line and off-line clock synchronization. In off-line clock synchronization it suffices to be able to estimate a clock reading time afterwards. This could, for instance, be done after an experiment has finished. The new algorithm is based on least squares estimation, and directly estimates both clock offset and clock drift.

2.4 Related Work

Some work has been done on setups related to the one described in Section 2.1. No work by other authors is known on exactly the setup in Section 2.1. In this section some of the related work is described.

Make the System Time-Invariant

In Luck and Ray (1990) the closed loop system is made time-invariant by introduction of buffers at the controller and actuator nodes as illustrated in Figure 2.5. All nodes are clocked and act synchronized. By making the

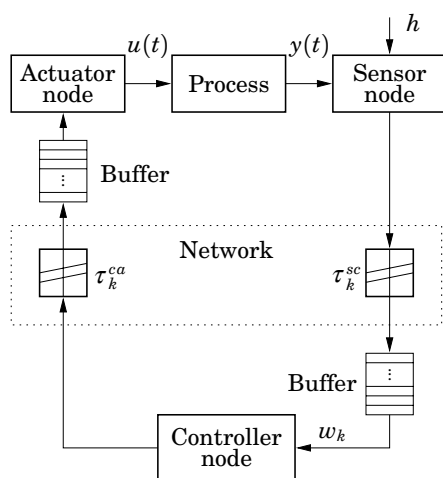


Figure 2.5 In Luck and Ray (1990) buffers are introduced after the varying communication delays to make the system time-invariant. The buffers must be longer than the worst case communication delay.

buffers longer than the worst case delay the process state can be written as

$$x_{k+1} = Ax_k + Bu_{k-\Delta_1} \quad (2.14)$$

$$y_k = Cx_k, \quad (2.15)$$

where Δ_1 is the length in samples of the buffer at the actuator node. If the buffer at the controller node is assumed to have the length Δ_2 samples, the process output available for the controller at time k is $w_k = y_{k-\Delta_2}$. The design problem is now reformulated as a standard sampled data control problem. The information set available for calculation of u_k is

$$W_k = \{w_k, w_{k-1}, \dots\}. \quad (2.16)$$

Chapter 2. Problem Formulation

In Luck and Ray (1990) the LQG-optimal controller,

$$u_k = \xi(W_k), \quad (2.17)$$

is derived. An advantage with the method is that it handles control delays that are longer than the sampling period. A serious disadvantage with the method is that it makes the control delay longer than necessary. In Chapter 5 it will be shown that performance can be increased by having event-driven controller and actuator nodes. By using event-driven nodes we make the control delay smaller, but we get a time-varying system, which is harder to analyze.

Stochastic Approaches

In Liou and Ray (1991) a scheme with time-driven sensor, time-driven controller, and event-driven actuator, is studied. The sensor and the controller is started with a time skew of Δ_s . The probability that the new sensor value has reached the controller when the control signal is calculated, $P(\tau_k^{sc} < \Delta_s)$, is assumed to be known. If $\tau_k^{sc} > \Delta_s$ the new control signal is calculated without knowledge of the new measurement signal. The actuator node D/A-converts the new control signal as soon as it is transmitted to the actuator node. A discrete time *augmented plant model* is derived by introducing the delayed signals as states in the augmented plant model. The *augmented plant model* can be written as

$$x_{k+1} = A_k x_k + B_k u_k, \quad (2.18)$$

where A_k and B_k are stochastic matrices due to the random communication delays. The LQ-optimal controller is solved for the problem setup by (2.18). It is also discussed how to construct a state estimator in the case when all states are not measured. It is claimed that timestamping of signals is important for estimation of process state. For the problem setup in Liou and Ray (1991) it is not known if the combination of optimal controller and the proposed state estimator is the optimal output feedback controller, i.e., if the *separation principle* applies.

The LQ-controller of Liou and Ray (1991) is used in Ray (1994) together with a stochastic state estimator. The timing setup is the same as the one used in Liou and Ray (1991). The estimator is designed to minimize the variance of the state prediction errors. The combination of the LQ-controller and the minimum variance estimator is introduced as the *DCLQG*-controller, delay compensated LQG. It is stressed that the separation principle does not hold for the *DCLQG*-controller, i.e., *DCLQG*-controller is a suboptimal control scheme.

In Krtolica *et al.* (1994) control systems with random communication delays are studied. The delays, from sensor to controller and from controller to actuator, are modeled as being generated from a Markov chain, see Figure 2.6. Only one of the β_i coefficients in Figure 2.6 is one, all the

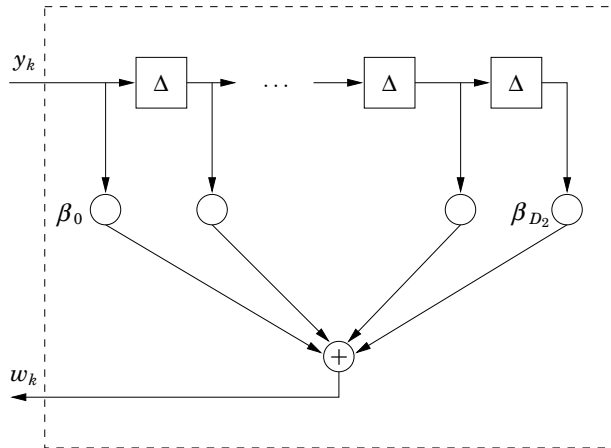


Figure 2.6 The network model used in Krtolica *et al.* (1994). The sampled signal, y_k , is delayed a number of samples due to communication delay. The controller reads the signal w_k . Only one of the β_i coefficients is one, the others are zero.

others are zero. Notice that the delay must be a multiple of the sampling period, i.e., all nodes are clock driven. It is shown that the closed loop system can be written as

$$z_{k+1} = H_k z_k, \quad (2.19)$$

where H_k depends on the state of the delay elements depicted in Figure 2.6. The sequence of β_i is generated by a Markov chain. Necessary and sufficient conditions are found for zero-state mean-square exponential stability. The resulting stability criterion is that the solution of two coupled (Lyapunov-like) equations need to be positive definite for stability of the closed loop. The problem formulation and solution used in this approach has clear connections to the theory of *jump linear systems*, which will be reviewed in the next section.

In Chan and Özgüner (1995) a setup using the *Ford SCP Multiplex Network hardware* is studied. The communication delay is modeled as in Figure 2.7. There is a queue of unsent sensor readings at the sensor node. A simple form of timestamping is done by appending the size of the queue to every message that is sent from the sensor node to the controller

Chapter 2. Problem Formulation

node. It is shown that by this method the controller node can reduce its uncertainty about which sensor reading it is using down to two possible cases. By knowing the probability for the two possible cases of delay, a state estimator is constructed. The sensor node and the controller node are both time-driven with a skew of Δ_{sp} . It is also shown how pole placement can be done for the described setup.

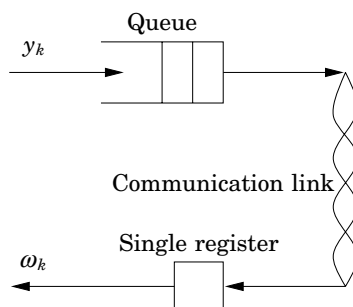


Figure 2.7 Block diagram of the transmission from the sensor node to the controller node in Chan and Özgüner (1995). The sampled signal y_k is delayed during the transmission to the controller node. The controller reads the sensor value ω_k from a register in the controller node. A simple form of timestamping is done by appending every message with the size of the queue when the message was sent.

Jump Linear Systems

Jump systems in continuous time was introduced and studied in the 1960's by Krasovskii and Lidskii (1961). *Jump linear systems* can in discrete time be written as

$$x_{k+1} = A(r_k)x_k + B(r_k)u_k, \quad (2.20)$$

where $A(r_k)$ and $B(r_k)$ are real-valued matrix functions of the random process $\{r_k\}$, see Ji *et al.* (1991). An interesting special case of $\{r_k\}$ -process is to let $\{r_k\}$ be a time homogeneous Markov chain taking values in a finite set $\{1, \dots, s\}$. The Markov chain has the transition probabilities

$$P(r_{k+1} = j \mid r_k = i) = q_{ij} \geq 0, \quad (2.21)$$

where

$$\sum_{j=1}^s q_{ij} = 1. \quad (2.22)$$

If $\{r_k\}$ is generated by a time homogeneous Markov chain the system is called a *discrete-time Markovian jump linear system*. As will be discussed in Chapter 3 this is an attractive model for control systems with randomly varying communication delays.

Different notations of stability have been considered for jump linear systems. The following definition of three stability notions is taken from Ji *et al.* (1991).

DEFINITION 2.1—STOCHASTIC STABILITY

For systems (2.20) and (2.21) with $u_k \equiv 0$, the equilibrium point 0 is

Stochastically Stable if for every initial state (x_0, r_0)

$$\mathbf{E} \left\{ \sum_{k=0}^{\infty} |x_k(x_0, r_0)|^2 \mid x_0, r_0 \right\} < \infty, \quad (2.23)$$

Mean Square Stable if for every initial state (x_0, r_0)

$$\lim_{k \rightarrow \infty} \mathbf{E} \left\{ |x_k(x_0, r_0)|^2 \mid x_0, r_0 \right\} = 0, \quad (2.24)$$

Exponentially Mean Square Stable if for every initial state (x_0, r_0) , there exist constants $0 < \alpha < 1$ and $\beta > 0$ such that for all $k \geq 0$

$$\mathbf{E} \left\{ |x_k(x_0, r_0)|^2 \mid x_0, r_0 \right\} \leq \beta \alpha^k |x_0|^2, \quad (2.25)$$

where α and β are independent of x_0 and r_0 . □

In Ji *et al.* (1991) the three stability concepts of Definition 2.1 are shown to be equivalent. Another, less conservative, stability notion is *almost sure stability*. The definition of almost sure stability, from Ji *et al.* (1991), is

DEFINITION 2.2—ALMOST SURE STABILITY

Systems (2.20) and (2.21) with $u_k \equiv 0$ are said to be *almost surely stable*, if for every initial state (x_0, r_0) , we have

$$\mathbf{P} \left\{ \lim_{k \rightarrow \infty} |x_k(x_0, r_0)| = 0 \right\} = 1. \quad (2.26)$$

□

The stability concepts of Definition 2.1 imply almost sure stability, but almost sure stability does not imply the stability concepts of Definition 2.1. This relation is illustrated with an example.

Chapter 2. Problem Formulation

EXAMPLE 2.1

Consider a jump linear system with two Markov states and the transition matrix

$$Q = \begin{bmatrix} 1 - \alpha & \alpha \\ 0 & 1 \end{bmatrix}, \quad (2.27)$$

where $\alpha < 1/2$. This system has the property that when it jumps to state 2 it will remain in state 2 forever. Let the system matrices be

$$\begin{aligned} A(1) &= 2, & B(1) &= 0, \\ A(2) &= 0, & B(2) &= 0. \end{aligned} \quad (2.28)$$

The state x_k will grow exponentially as long as we stay in state 1, but x_k will be reset to 0 as soon as we jump to state 2, where we will stay forever. For this system we have

$$\lim_{k \rightarrow \infty} \mathbb{E} \left\{ |x_k(x_0, r_0)|^2 \mid x_0, r_0 = 1 \right\} = \lim_{k \rightarrow \infty} \left((1 - \alpha)^k 2^k x_0 \right)^2 = \infty, \quad (2.29)$$

which means that the system is not mean square stable. We also have that

$$\mathbb{P} \left\{ \lim_{k \rightarrow \infty} |x_k(x_0, r_0)| = 0 \right\} = 1, \quad (2.30)$$

which implies that the system is almost surely stable. \square

Many of the ideas from control theory have been studied for jump linear systems. The LQ-problem was solved with finite and infinite time horizon by Sworder (1969) and Wonham (1971). The discrete-time jump LQ-problem was solved for a finite-horizon by Blair and Sworder (1975). Extensions of the LQ-problem, such as control of jump linear systems with Gaussian input and measurement noise, has been done in Ji and Chizeck (1990). The H_∞ -control state-feedback problem has been studied for both continuous- and discrete-time jump linear systems, see de Souza and Fragoso (1993) and Fragoso *et al.* (1995), respectively.

All the above mentioned work rely on the availability of the value r_k at time k . If this is not the case, r_k has to be estimated. Less work has been done in this area, see Ji *et al.* (1991) for a discussion. This problem is also known as the *Hidden Markov Model* problem, see Elliot *et al.* (1995).

An interesting extension of jump linear systems is to let the Markov chain postulate the distribution of the system matrices A , B etc. instead of values for these. As an example the states of the Markov chain could be “Low network load”, “Medium network load” and “High network load”. This is further discussed in Chapter 3.

3

Modeling of Network Delays

Network delays, or network transfer times, have different characteristics depending on the network hardware and software. To analyze control systems with network delays in the loop we have to model these. The network delay is typically varying due to varying network load, scheduling policies in the network and the nodes, and due to network failures. We will use three models of the network delay:

- Constant delay,
- Random delay, which is independent from transfer to transfer,
- Random delay, with probability distributions governed by an underlying Markov chain.

The validity for these models when modeling network delays will be commented in the next chapter, where experimental delay measurements are presented. The control loop usually also contains computational delays. The effect of these will not be investigated separately, as they can be embedded in the network delays.

In the last section of this chapter we will study how the controlled process can be modeled when we have communication delays in the loop.

In Chapter 4 experimental network delay measurements will be presented, in connection with this, applicability of the models developed in this chapter will be discussed.

3.1 Network Modeled as Constant Delay

The simplest model of the network delay is to model it as being constant for all transfers in the communication network. This can be a good model even if the network has varying delays, for instance, if the time scale in the process is much larger than the delay introduced by the communication.

Chapter 3. Modeling of Network Delays

In this case the mean value or maybe the worst case delay can be used in the analysis. If this is not the case, wrong conclusions can be drawn regarding system stability and performance.

One way to achieve constant delays is by introduction of timed buffers after each transfer. By making these buffers longer than the worst case delay time the transfer time can be viewed as being constant. This method was proposed in Luck and Ray (1990). A drawback with this method is that the control delay becomes longer than necessary. This can lead to decreased performance as will be shown in Chapter 5.

3.2 Network Modeled as Delays Being Independent

Network delays are usually random. The network delay can have several sources, for instance,

- waiting for the network to become idle,
- if several messages are pending, the wait can include transmission of the waiting messages,
- if transmission errors occur, a retransmission can be needed,
- in some networks collisions can occur if two nodes try to send at the same time, the resolution of this can include a random wait to avoid a collision at the next try.

As the activities in the system usually are not synchronized with each other, the number of the above listed delay causes that will occur is random. To take the randomness of the network delays into account in the model, the time delays can be modeled as being taken from a probabilistic distribution. To keep the model simple to analyze one can assume the transfer delay to be independent of previous delay times. In a real communication system the transfer time will, however, usually be correlated with the last transfer delay. For example, the network load, which is one of the factors affecting the delay, is typically varying with a slower time constant than the sampling period in a control system, i.e., the time between two transfers. We will allow the model to have different probability distributions for the delay from sensor to controller, τ_k^{sc} , and for the delay from controller to actuator, τ_k^{ca} . In Chapter 4 we develop a CAN network model which uses different distributions for τ_k^{sc} and τ_k^{ca} .

3.3 Network Modeled Using Markov Chain

To model phenomena as network queues, and varying network loads, our network model needs to have a memory, or a state. One way to model dependence between samples is by letting the distribution of the network delays be governed by the state of an underlying Markov chain. Effects such as varying network load can be modeled by making the Markov chain do a transition every time a transfer is done in the communication network. The model is closely related to the models used in *jump systems*, see Section 2.4. A difference is that in our network model each state of the Markov chain postulates probability distributions for τ_k^{sc} and τ_k^{ca} . In *jump linear system* each Markov state defines a set of system matrices, $\{A(r_k), B(r_k), C(r_k), D(r_k)\}$. It can also be noticed that the previously discussed model, where delays are independent from transfer to transfer, constitutes a Markov model with only one state. A short discussion of the theory of Markov chains is given in Appendix B.

EXAMPLE 3.1—SIMPLE NETWORK MODEL

A simple network model capturing different network loads can have three states, one for low network load, one for medium network load, and one for high network load. In Figure 3.1 the transitions between different states in the communication network are modeled with a Markov chain. The transition probabilities are indicated on the arcs. The transition probabilities are defined as

$$q_{ij} = P\{r_{k+1} = j \mid r_k = i\}, \quad i, j \in [L, M, H], \quad (3.1)$$

and will model how frequent changes of the network state will be. Together with every state in the Markov chain we have a corresponding delay distribution modeling the delay for that network state. These distributions could typically look like the probabilistic distributions in Figure 3.2. The distributions are assumed to have a lower mean if the network has low load, and a higher mean if the network has a high load. In a realistic model, the delay variation would probably be smaller for low loads, and larger for high loads. When the load is low, networks delay are quite deterministic, the network is often idle when we want to transmit. When the network load is high, we can have a short delay, but we could also have to wait for multiple messages to be sent. \square

3.4 Sampling of Systems with Network Delays

Continuous-time systems with time-delays are infinite dimensional systems. A finite dimensional description of the control loop can be formu-

Chapter 3. Modeling of Network Delays

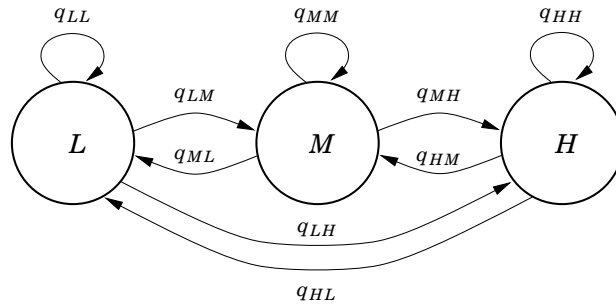


Figure 3.1 An example of a Markov chain modeling the state in a communication network. L is the state for low network load, M is the state for medium network load, and H is the state for high network load. The arrows show possible transitions in the system.

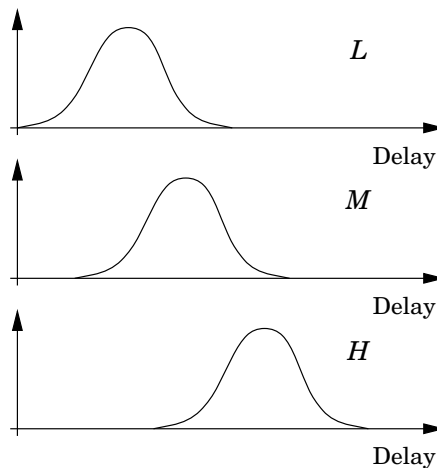


Figure 3.2 The delay distributions corresponding to the states of the Markov chain in Figure 3.1. L is the state for low network load, M is the state for medium network load, and H is the state for high network load.

lated by sampling of the continuous-time process. Let the controlled process be

$$\frac{dx}{dt} = Ax(t) + Bu(t) + v(t), \quad (3.2)$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$ and $v(t) \in \mathbb{R}^n$. A and B are matrices of appropriate sizes, $u(t)$ is the controlled input and $v(t)$ is white noise with zero

3.4 Sampling of Systems with Network Delays

mean and incremental covariance R_v . We will treat the MIMO-problem, multiple input multiple output, in the sense that we allow process outputs, and process inputs to be vectors. We will, however, not allow the different signals within y_k and u_k to have different delays. This means that if we have several sensors, the measurements have to be delivered from the same node. For the actuators it means that all actuators have to be connected to the same node. The MIMO-problem with individual delays for the signal is treated in Section 7.4. The timing of the signals in the system is shown in Figure 3.3. Notice that the control signal segments are active a varying time. Assume that the delay from sensor to actuator

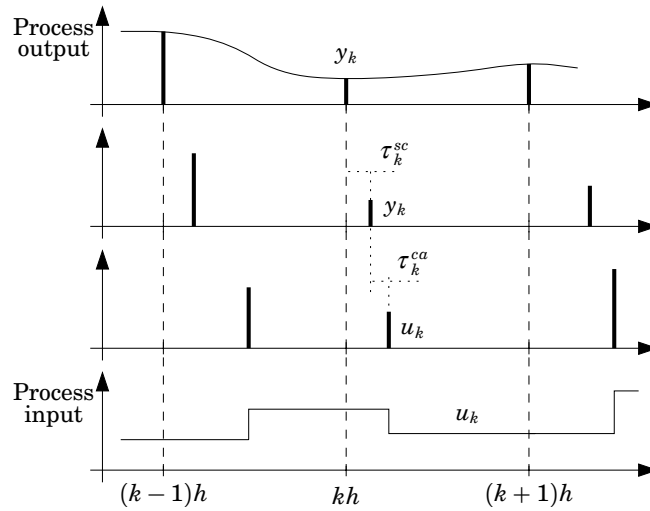


Figure 3.3 Timing of signals in the control system. The first diagram illustrates the process output and the sampling instants, the second diagram illustrates the signal into the controller node, the third diagram illustrates the signal into the actuator node, and the fourth diagram illustrates the process input.

is less than the sampling period h , i.e., $\tau_k^{sc} + \tau_k^{ca} < h$. Integration of (3.2) over a sampling interval gives

$$x_{k+1} = \Phi x_k + \Gamma_0(\tau_k^{sc}, \tau_k^{ca})u_k + \Gamma_1(\tau_k^{sc}, \tau_k^{ca})u_{k-1} + v_k, \quad (3.3)$$

Chapter 3. Modeling of Network Delays

where

$$\Phi = e^{Ah} \quad (3.4)$$

$$\Gamma_0(\tau_k^{sc}, \tau_k^{ca}) = \int_0^{h-\tau_k^{sc}-\tau_k^{ca}} e^{As} ds B \quad (3.5)$$

$$\Gamma_1(\tau_k^{sc}, \tau_k^{ca}) = \int_{h-\tau_k^{sc}-\tau_k^{ca}}^h e^{As} ds B. \quad (3.6)$$

The index k is used to indicate sample number, i.e., $x_k = x(kh)$. The state noise v_k has zero mean and the variance

$$R_1 = \mathbb{E}\{v_k v_k^T\} = \int_0^h e^{A(h-s)} R_v e^{A^T(h-s)} ds. \quad (3.7)$$

This is a standard result on sampling of systems with time-delays, see, for instance, Åström and Wittenmark (1997). The infinite dimensional continuous-time system has now been reformulated to the time-varying, finite-dimensional, discrete-time system (3.3). The drawback is that we do not have direct control over intersample behavior using the discrete-time model. This is, however, easy to study if needed. Some early work on modeling of imperfections in sampled data systems, such as random sampling and imperfect hold etc., was published already 1959, see Kalman and Bertram (1959).

4

Experimental Delay Measurements

This chapter shows the results of delay measurements from two communication networks, CAN (Controller Area Network) and Ethernet. The experiments are done to verify that the statistical assumptions about the delay distributions are relevant. The experiments also show that the distribution of communication delays can vary significantly depending on the setup and the load on the network. The chapter also describes a new clock synchronization algorithm that can be used for off-line as well as on-line clock synchronization.

4.1 CAN

The first platform for which delay measurements will be presented is CAN, Controller Area Network. The experimental platform was four one-card computers with CAN-controllers connected to a CAN-network. The network nodes were originally built by the Department of Computer Engineering at Chalmers University of Technology. The nodes used in our experiments were borrowed from the DAMEK Mechatronics Division at the Royal Institute of Technology.

Setup

The hardware platform used for delay measurements on the CAN-bus was one-card computers equipped with CAN-controllers. The one-card computers are based on the Motorola MC68340 processor, Motorola (1992). This is an 8 MHz processor with special properties making it suitable for embedded systems. The problems with this CPU are that it lacks floating point arithmetics and that it is quite slow. The experimental setup

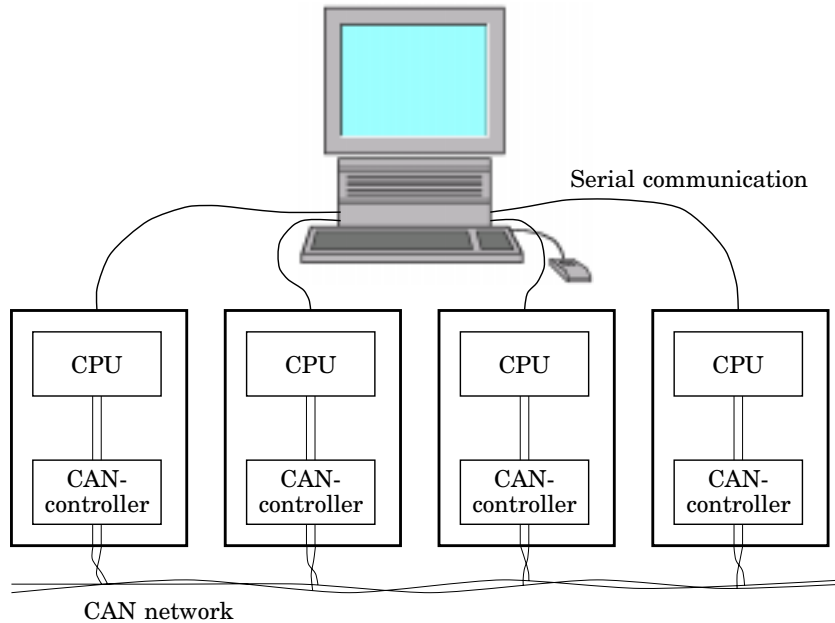


Figure 4.1 The CAN experimental system with the four one-card computers. The nodes are connected with a twisted pair CAN. Serial communications to the nodes are used for data collection.

is shown in Figure 4.1. The communication between the four test computers was done on a CAN-bus. Operator interface was done in text terminals on a workstation connected to the nodes with serial connections. The CAN-communication is handled by a special chip in the nodes, the CAN-controller. The CAN-controller is interfaced to the CPU and can, for instance, generate an interrupt when a new message arrives. The CPU can then read the received message with a memory read operation. The CAN-controller used was the Intel 82527, Intel (1995).

The programming of the nodes was done using the real-time kernel from the Department of Automatic Control, Lund Institute of Technology. This is a C-kernel with a Modula-2 interface, so all programming is done in Modula-2. The software interface to the CAN-controller is designed to allow a user specified interrupt handler to be executed when a message arrives or when a message is sent. This feature makes it possible to get high resolution in the measurement of send and receive times.

The delay measurement experiment is done by letting one node be the master node, and one of the other nodes be a slave node. The master

node periodically sends a message to the slave node. The times when the messages were sent are stored in the master node. When the slave node receives a message from the master node its local clock is read. The clock value is stored in the slave node. Then a message immediately is returned to the master node. When this message arrives the master reads its local clock and stores the value. This experiment will result in a number of triples of clock readings, one for each periodic message sent by the master node. When the experiment is finished the clock readings are echoed to the terminals on the workstation, where further data analysis is done. This experiment is close to the communications done in a system with distributed I/O. The master node would then be the distributed I/O, and the slave node would be the controller node. If the time delays were to be evaluated in a distributed I/O system the local clock readings would be sent together with each message, see also Section 2.1 on timestamping. For these timestamps to provide any information to the controller the system needs to have synchronized clocks. Here it suffices to be able to correct the local clock readings afterwards. For this purpose a special algorithm for off-line clock synchronization has been developed, see below.

The CAN-bus can be programmed for several transmission speeds, or bit rates. The bit rate is limited by, for instance, noise environment, bus length, and cable quality. The standard bit rates range from 10 kbit/s to 1 Mbit/s, see CiA (1994). In the experiments we used the bite rate 10 kbit/s. The motive for this relatively slow bit rate is that the node CPUs are quite slow, and a higher bit rate would give bad resolution in the time measurements. With a higher bit rate a significant part of the transmission delay could be the time for the node program to start the interrupt handler. As the transmission delays mainly is proportional to the bit rate the delays would scale down with the bit rate if a higher bit rate is used. This will hold as long as we do not get a large amount of retransmissions due to a too high bit rate. The measurements obtained hence reflect the behavior of a high-rate bus quite well.

CAN is a priority based network. The sending of a message on the bus starts with a wait for the bus to become idle. When the bus becomes idle an arbitration starts to find which node has the message with highest priority. The arbitration results in that all nodes except the one with the message of highest priority stay off the bus. In a CAN network the message transmission delay, τ , can be split into three parts

$$\tau = \tau_w + \tau_{hp} + \tau_s, \quad (4.1)$$

where

- τ_w is the wait for an ongoing transmission to end,

Chapter 4. Experimental Delay Measurements

- τ_{hp} is the wait for messages with higher priorities to be sent, and
- τ_s is the time for the message to be transmitted.

The delay parts originating from other messages, τ_w and τ_{hp} , will depend on the load on the bus and the message priorities. The transmission delay, τ_s , will depend on the message length and random retransmission. In CAN a message can range from zero to eight bytes. Together with address bits and control bits a CAN message transmission ranges from 64 to 128 bits depending on the message length. In practice, the message lengths differ because “stuff” bits are added during transmission. A “stuff” bit, with opposite sign, is added if five consecutive bits have the same sign. This is done to assist the bus synchronization. In the experiments we used messages with eight bytes. This would allow transmission of both a measurement and a timestamp. The transmission delay will clearly also depend on the bus load and the priority of this load. The effect on the delays of these two parameters are presented later in this section. The following section presents the off-line clock synchronization method used in the experiments.

Off-line clock synchronization

The purpose of clock synchronization is to estimate the difference between the local clocks of two nodes. The synchronization is done by sending clock requests back and forth between the two nodes, see Section 2.3. For the experiments in this chapter we do not need the clock correction in real-time, it suffices to do it off-line, after the experiment has finished. Let S be the node sending the requests, and R be the node responding to the requests from S . This is exactly the situation described earlier; there S would be the distributed I/O, and R would be the controller node. Let the absolute time be t_i , let the local time in node S be t_i^S , and let the local time in node R be t_i^R . The local clocks in S and R are assumed to have a linear growing skew to the absolute time such that

$$t_i^S = t_i + \delta^S + \rho^S t_i \quad (4.2)$$

$$t_i^R = t_i + \delta^R + \rho^R t_i, \quad (4.3)$$

where δ^S and δ^R are skews due to different start times of the nodes, and ρ^S and ρ^R are clock drifts due to different crystal speeds in the nodes. The clock drift is varying due to inaccuracies in the clocks, aging, and crystal temperature variations, see Schedl (1996). Over short time interval, such as the time to do the delay experiment, the drift parameters ρ^S and ρ^R can be considered as being constant. By eliminating t_i from (4.2) and (4.3) it follows that

$$t_i^R = \beta t_i^S + \alpha, \quad (4.4)$$

where

$$\alpha = \delta^R - \frac{1 + \rho^R}{1 + \rho^S} \delta^S, \quad (4.5)$$

$$\beta = \frac{1 + \rho^R}{1 + \rho^S}. \quad (4.6)$$

Remark: It is seen from (4.6) that clocks having the same speed, $\rho^S = \rho^R$, results in $\beta = 1$. \square

The delay measurement experiment described in Section 4.1 will result in three time series. The series are:

- The times when a clock request was sent from node S to node R . Define this series as $\{t_a^S(k)\}_{k=1}^n$, where the superscript S indicates that the time is measured by the local clock in node S , and the superscript n that the experiment contains n data points.
- The times when a clock request was received by node R and immediately echoed back to node S . Define this series as $\{t_b^R(k)\}_{k=1}^n$, where the superscript R indicates that the time is measured by the local clock in node R .
- The times when an answer to a clock request is received by node S . Define this series as $\{t_c^S(k)\}_{k=1}^n$.

Let the transfer time from node S to node R , and from node R to node S , be T_k^{SR} and T_k^{RS} , respectively. The transfer times T_k^{SR} and T_k^{RS} are measured with the time base of the clock in node S . The off-line clock synchronization algorithm described in the sequel will need one assumption.

ASSUMPTION 4.1

The transfer times from node S to node R , and from node R to node S , have the same mean value $E(T_k^{SR}) = E(T_l^{RS}) = \sigma$ for $l, k \in [1 \dots n]$. \square

Assumption 4.1 is common in network clock synchronization. In Figure 4.2 typical realizations of the three time series are plotted against time measured by the local clock in node S . The series $\{t_a^S(k)\}_{k=1}^n$ will, of course, form a line with unit slope. The series $\{t_b^R(k)\}_{k=1}^n$ will also be on a line with unit slope, as this is also read with the clock in node S . Due to the random transfer time for messages the clock readings in $\{t_c^S(k)\}_{k=1}^n$ will be a random time later than the corresponding request time. The times, measured by the clock in node S , when the requests were handled by node R are unknown, but we know that in mean they were handled σ after the corresponding request time. In Figure 4.2 the series $\{t_c^S(k)\}_{k=1}^n$ is drawn

Chapter 4. Experimental Delay Measurements

against the mean clock request handling time, i.e., $\{t_a^S(k) + \sigma\}_{k=1}^n$. The data points in the time series $\{t_b^R(k)\}_{k=1}^n$ can be written as

$$t_b^R(k) = \beta (t_a^S(k) + T_k^{SR}) + \alpha, \quad (4.7)$$

and the data points in the series $\{t_c^S(k)\}_{k=1}^n$ can be written as

$$t_c^S(k) = t_a^S(k) + T_k^{SR} + T_k^{RS}. \quad (4.8)$$

The idea is to evaluate the mean transfer delay, σ , by measuring the round trip delay, and then to fit a line to the series $\{t_b^R(k)\}_{k=1}^n$ to estimate the unknown parameters $\{\delta, \rho, \sigma\}$. Let the line fitted to the data series $\{t_b^R(k)\}_{k=1}^n$ have the equation $a + bt$, see Figure 4.2. Introduce the mean transfer time estimate as

$$\hat{\sigma} = \frac{1}{2n} \sum_{k=1}^n (t_c^S(k) - t_a^S(k)). \quad (4.9)$$

From (4.7) and (4.8) we get the estimates

$$\hat{\alpha} = a \quad (4.10)$$

$$\hat{\beta} = b \quad (4.11)$$

With these estimates of α and β , a clock reading in node R can be transformed to a clock reading in node S using (4.4), and vice versa.

The line fitting can, for instance, be done using the least squares algorithm, see Åström and Wittenmark (1997). In the experiments, a and b were estimated with the least squares method, and σ was estimated by taking the mean of the round trip delays.

The clock synchronization algorithm, which is believed to be new, can easily be generalized to an on-line procedure by using recursive least squares estimation (RLS) with a constant or varying forgetting factor, see Ljung and Söderström (1983) and Åström and Wittenmark (1997).

System load

The experiments done on the CAN system were performed under several load conditions. The load was introduced as “dummy” traffic between the two nodes not involved in the delay measurements. From (4.1) it can be seen that the delays will depend on, not only how much traffic there is on the bus, but also on the priority profile of the traffic. The experiment was performed with the following four load types:

No load The bus was empty except for the experiment.

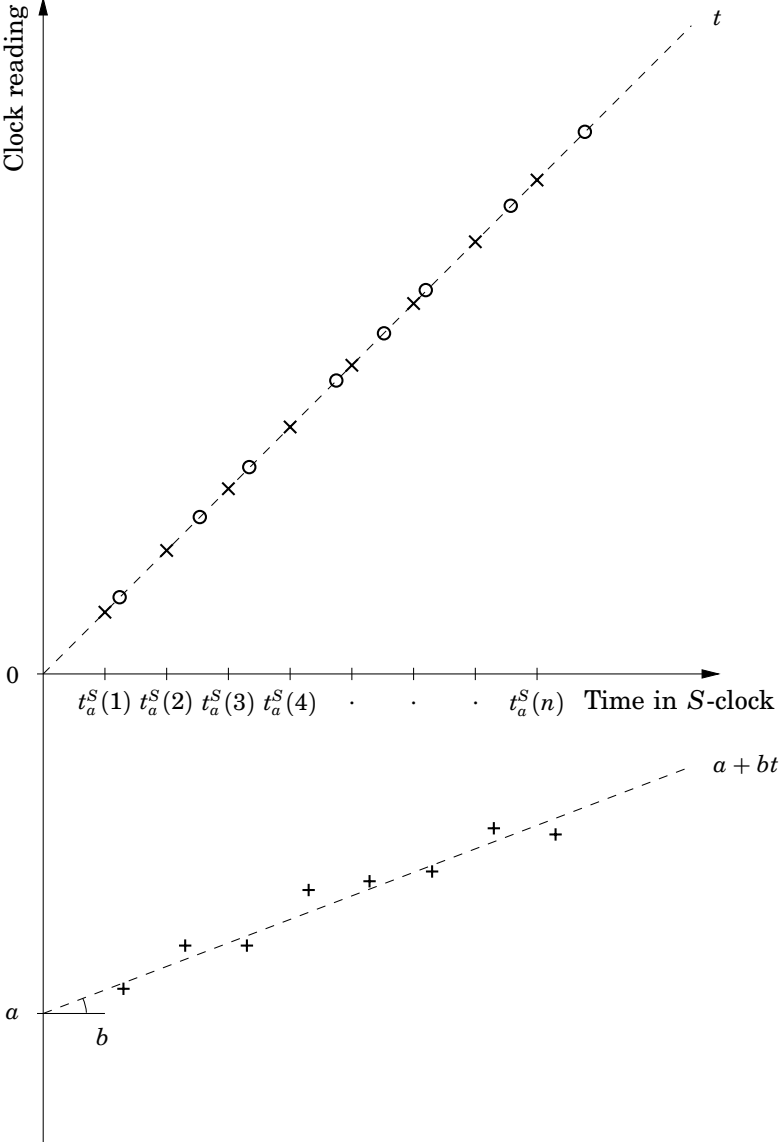


Figure 4.2 The three series of clock readings. The markings are request times $\{t_a^S(k)\}_{k=1}^n$ (\times), request bounce times $\{t_b^R(k)\}_{k=1}^n$ ($+$), and request return times $\{t_c^S(k)\}_{k=1}^n$ (\circ). The request bouncing times are drawn as if they occurred σ later than the request time. A line is fitted for estimation of a and b . Here the request times are drawn as if they were equally spaced, which is not needed for the method to work.

Chapter 4. Experimental Delay Measurements

Periodic message One periodic process was loading the bus,

Periodic messages Several periodic processes were loading the bus.

Random message interval The load process had a random interval between messages.

In the following the load cases are described in detail, and the results from the delay measurements are presented.

No load The only bus load was the load from the experiment itself. The time interval between two requests was designed to be larger than the round trip delay. In this case the interval was $h = 50$ ms. This means that the bus will always be empty when a transmission is initiated. A constant delay was expected from this experiment, as both $\tau_w = 0$ and $\tau_{hp} = 0$. The delay will only be the part τ_s , which originates from the message transmission. The time to transmit a message will depend on the message length. In Figure 4.3 and Figure 4.4 measured delays are displayed for a message length of 4 and 8 bytes. The delays shown are the sensor to controller delay, τ_k^{sc} , and the controller to actuator delay, τ_k^{ca} . As seen from Figure 4.3 and Figure 4.4 the delays are almost constant if the bus is not loaded. In Figure 4.3 a small variation can be seen in the delays. It appears as a variation in τ_k^{sc} and τ_k^{ca} , but the sum $\tau_k^{sc} + \tau_k^{ca}$ is almost constant. The effect probably comes from a varying clock drift in the local clocks, maybe a varying crystal temperature. This effect is not corrected for by the clock synchronization described in Section 4.1, which will only correct for a constant clock drift. A synchronization error of this size, some tenths of a millisecond, will be seen throughout the experiments.

The utilization, see Tindell and Hansson (1995), of the bus is defined as

$$U = \sum_i \frac{C_i}{T_i}, \quad (4.12)$$

where i is the number of periodic transmissions on the bus, C_i is the transfer time for this message, i.e., τ_s , and T_i is the period for sending of message i . The utilization is a measure of how much load there is on the bus. The shortest transfer time for a message can be calculated from the knowledge of the bit rate on the bus, and from how many bits a message will be sent as. An 8 bytes long message will be sent as at least 128 bits. It could possibly be longer if “stuff” bits are needed. The bit rate is 10 kbits/s, which gives a shortest delay of 12.8 ms. Comparing with the measured delay mean of 16.2 ms we see that we have a software overhead in the delays of 3.4 ms. This overhead comes from delay to start

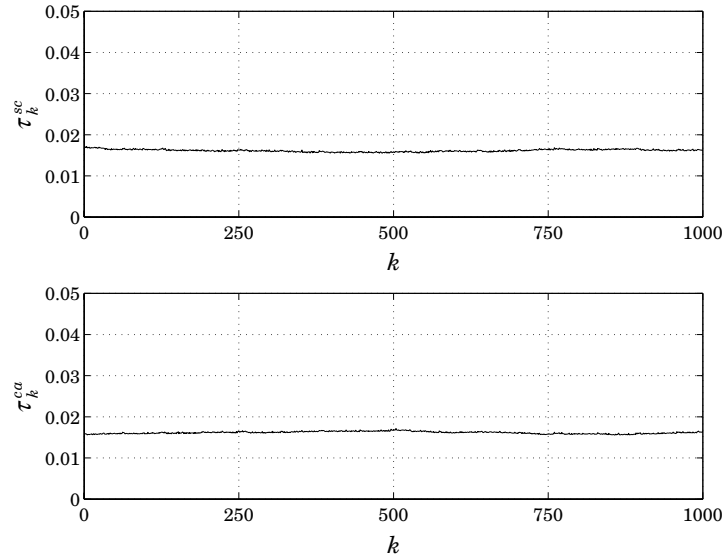


Figure 4.3 Measured delays for sensor to controller, τ_k^{sc} , and from controller to actuator, τ_k^{ca} . The experiment was the only load on the bus. The messages were 8 bytes long. The delay is almost constant.

the interrupt handler, and to execute the code in the interrupt handler, such as reading clocks etc. The utilization with message length 8 bytes is

$$U = \frac{0.0128}{0.05} + \frac{0.0128}{0.05} = 0.51. \quad (4.13)$$

As the utilization is a measure of how much load there is on the bus, the utilization 0.512 indicates the bus load is low. If the utilization is greater than 1 the bus has overload.

Periodic message In the next experiment the load was a periodic sending of a “dummy” message with length 8 bytes. The period for the “dummy” message was around 100 ms. The experiment was performed both with a higher priority load, and a lower priority load. The measured delays of the control loop are shown in Figure 4.5 and Figure 4.6. Now the delays are varying, this comes from the fact that the bus could be occupied when we want to send a message, giving a nonzero and varying τ_w . A surprising observation is that the delays are not longer in the case with a higher priority load. The explanation is that there is never a queue of messages, and hence $\tau_{hp} = 0$. This is because the period of the control loop (50

Chapter 4. Experimental Delay Measurements

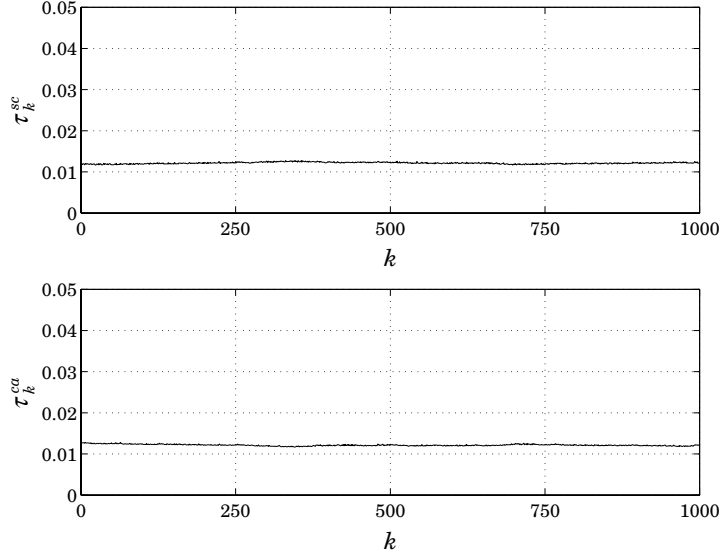


Figure 4.4 Measured delays for sensor to controller, τ_k^{sc} , and from controller to actuator, τ_k^{ca} . The experiment was the only load on the bus. The messages were 4 bytes long. The delay is almost constant.

ms) is much longer than the time to send. The only occasion when there could be a queue is when the clock request answer is to be sent. But as the response to a request takes some time, the load message, if there is one, will have time to take the bus no matter its priority. From the delay measurements it can be seen that the delay will have a periodic behavior. This comes from that all processes on the bus are periodic. The load will disturb the experiment messages for a time, and then the messages will not collide for a while, after which the collisions will appear again, and so on. The utilization for this experiment is

$$U = \frac{0.0128}{0.05} + \frac{0.0128}{0.05} + \frac{0.0128}{0.1} = 0.64, \quad (4.14)$$

which also is clearly less than 1. Despite the low utilization we get large variations in the delays.

Periodic messages To get a situation where we could have queued messages, and hence a nonzero τ_{hp} , we let the load be generated by two processes. In the experiment we used the load periods 80 ms and 100 ms,

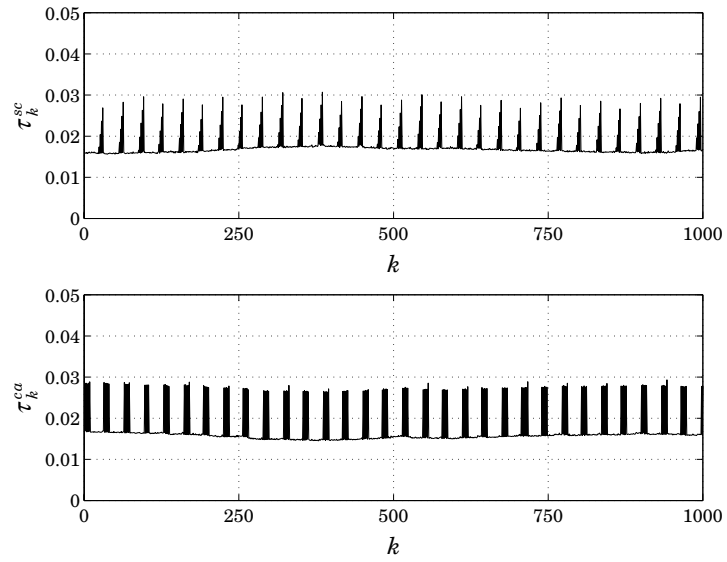


Figure 4.5 Load: One sender with period 100 ms and lower priority. Message length: 8 bytes.

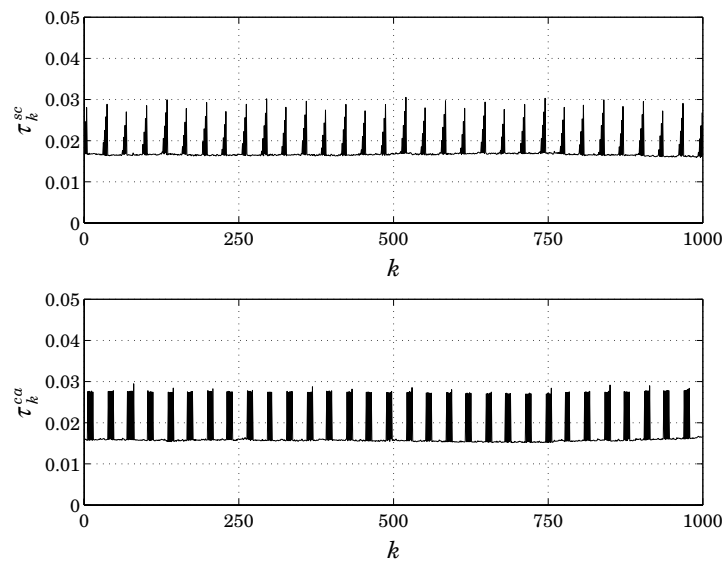


Figure 4.6 Load: One sender with period 100 ms and higher priority. Message length: 8 bytes. No difference to Figure 4.5, because there is never a queue of messages.

Chapter 4. Experimental Delay Measurements

giving the utilization

$$U = \frac{0.0128}{0.05} + \frac{0.0128}{0.05} + \frac{0.0128}{0.08} + \frac{0.0128}{0.1} = 0.80. \quad (4.15)$$

The utilization is still less than 1. The measured delays are shown in Figure 4.7 and Figure 4.8. It is seen that the delays can be substantially longer in the case where one of the load messages have a higher priority than the experiment messages. In the experiment where both load processes have lower priority it is noticed that the delay for the clock request answer, τ_k^{ca} , can take two values, either the bus is empty, or the message will have to wait for one message to be transmitted. In the case with one load process with higher priority and one with lower priority, the delay τ_k^{ca} can take three values. The longest wait will occur if the lower priority message is started when the request is served, and then the high priority message arrives before the clock request answer is sent. We will get a shorter delay if just one message is transmitted in between the request and the answer. If the bus is empty after a request the delay will have $\tau_{hp} = 0$ and $\tau_w = 0$. The other delay, τ_k^{sc} can take any value in an interval, depending on if there are queued messages when a request is to be sent. As the load processes are periodic the load pattern will be periodic, which can be seen in the periodic behavior of the delays.

Random message interval If a random message interval is used for the load process, the collision pattern is expected to disappear. In Figure 4.9 the experiment is shown with one load message which has a uniformly distributed period on the interval [40, 80]ms. The load message has higher priority than the experiment messages. As described earlier, the priority of the load is not expected to influence the delays, as we only have two processes on the bus. The mean utilization is

$$E\{U\} = \frac{0.0128}{0.05} + \frac{0.0128}{0.05} + \frac{0.0128}{0.04} \int_{0.04}^{0.08} \frac{1}{x} dx = 0.73, \quad (4.16)$$

which is clearly less than 1. The delay plot in Figure 4.9 shows that the periodic delay behavior disappears when we have a random period.

Modeling of Delays

The easiest way to get a model of the delays is by estimating the probability distribution functions of the delays from the delay measurements. As an example we will use the delay experiment presented in Figure 4.9. The load process has a period that is uniformly distributed on [40, 80]ms. The histograms of the delays are shown in Figure 4.10. From the time series

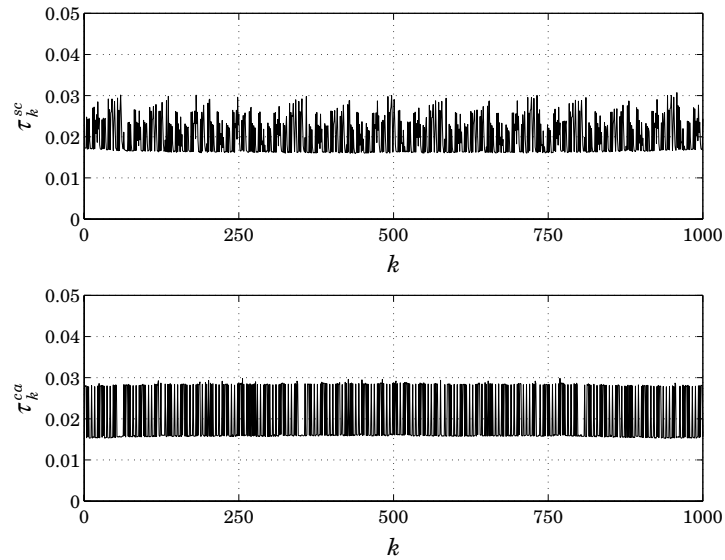


Figure 4.7 Load: Two senders with periods 80 ms and 100 ms and lower priority. Message length: 8 bytes.

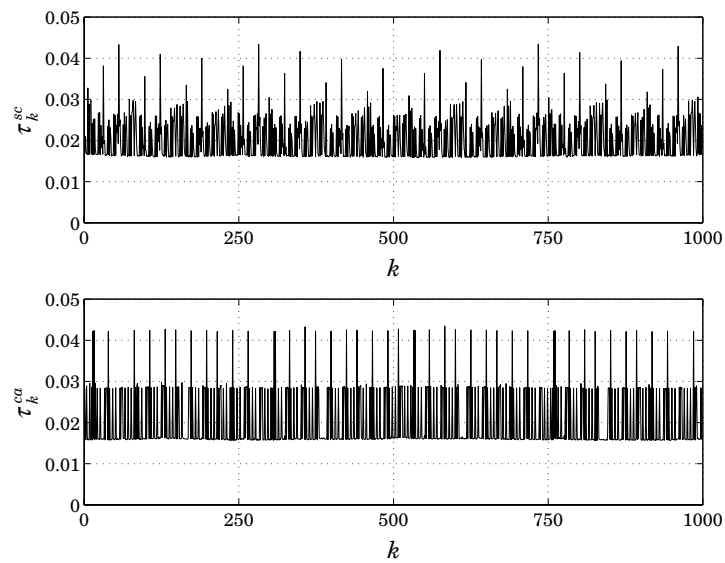


Figure 4.8 Load: Two load processes, one with period 80 ms and higher priority, and one with period 100 ms and lower priority. Message length: 8 bytes.

Chapter 4. Experimental Delay Measurements

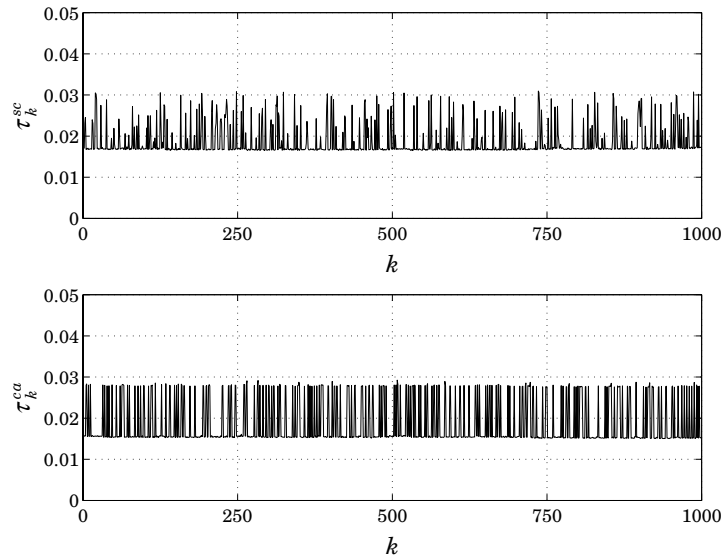


Figure 4.9 Load: One sender with period $\text{rect}(40, 80)$ ms and higher priority. Message length: 8 bytes. The delay behavior is non-periodic.

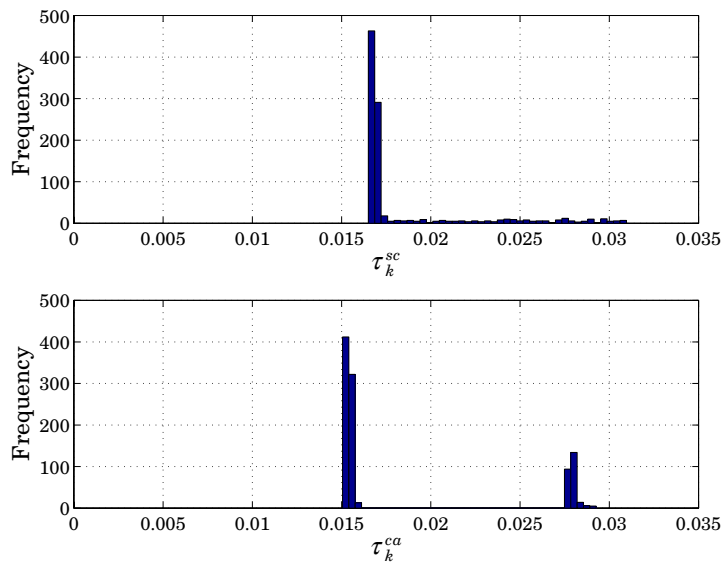


Figure 4.10 Distribution of the 1000 samples in the experiment. Load: One sender with period $\text{rect}(40, 80)$ ms and higher priority. Message length: 8 bytes.

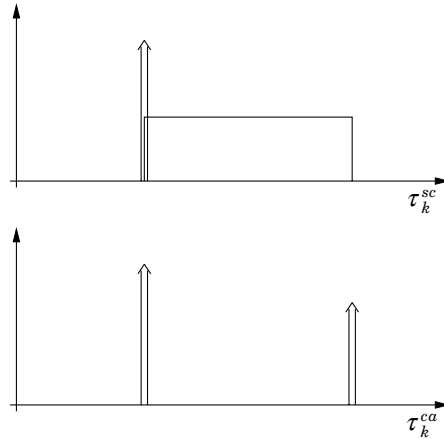


Figure 4.11 Model of the probability distributions with one load process.

in Figure 4.9 it can be seen that τ_k^{sc} takes several values in the interval $[0.017, 0.031]$, opposed to τ_k^{ca} which only takes the values 0.016 and 0.028. This is also seen from the histograms in Figure 4.10. The distributions can be explained from the experimental setup.

Sensor to Controller Delay When the message is to be sent the bus can be idle or a message can be under transmission. The probability for bus idle depends on the period of the load process. It will show up as a Dirac in the probability distribution function. If the bus is busy we will get a nonzero τ_w but there will never be a queue of messages. The delay τ_w will be uniformly distributed from 0 to the time it takes to send a message. The parts of the modeled distribution of τ_k^{sc} is shown in Figure 4.11.

Controller to Actuator Delay The delay from controller to actuator can only take two values when we have one load process. The reason for this is that if there was a message waiting when the message was sent from the sensor, the transmission of the waiting message starts before the message to the actuator is ready for transmission. In this case, the delay until the transmission starts will be the time to transmit the load message. If there is no waiting message the message to the actuator will be sent immediately after some computation time in the controller node. The modeled distribution is shown in Figure 4.11.

By comparing the measured distributions, Figure 4.10, with the model we see that there is a very good agreement.

Chapter 4. Experimental Delay Measurements

The developed model, Figure 4.11, captures the probability distribution of the time delay in the data, Figure 4.10, very well. What it does not capture is the time behavior, when there is correlation between consecutive delays. The covariance has been estimated for the experimental data with deterministic traffic, and with random traffic. In the case with random traffic no correlation can be seen between the delays. In the cases with one or several periodic load processes correlation can be seen between delays. This type of behavior can be captured with the Markov models discussed in Chapter 3.

Results

The CAN-bus gives varying delays. Message priorities can be used to lower the mean delay, but the delay will anyway be varying. The delay variation depends on the bus utilization, the priority of other messages on the bus, and if the sending processes are periodic. A simple delay model can be built by estimating the probability density functions for the delays. If this model is not sufficient, Markov models can be used. The advantage with the Markov models is that they can capture behaviors changing with time.

4.2 Ethernet Network

Delay measurement experiments have also been done using the Ethernet network at the Department of Automatic Control, Lund Institute of Technology. This network also serves as network connection for the work stations at the department, which means that we will not have control of all network load as in the CAN experiments. The network uses TCP/IP as communication protocol. This protocol is not intended for real-time services, since it lacks, for instance, message priorities. For more information on Ethernet, see IEEE (1985).

Setup

The experimental setup is shown in Figure 4.12. Experimental data for the network delays, τ_k^{sc} and τ_k^{ca} , were collected using two networked computers. The sampling interval was set to $h = 0.5$ s, this means that the sensor node sends a new measurement to the controller node every 0.5 seconds. After a short delay, due to execution of code in the controller node, a message containing the new control signal is returned to the distributed I/O node. All messages sent on the network were transmitted together with a timestamp. By comparing the timestamps the delays can be evaluated. Both the controller node and the distributed I/O node were

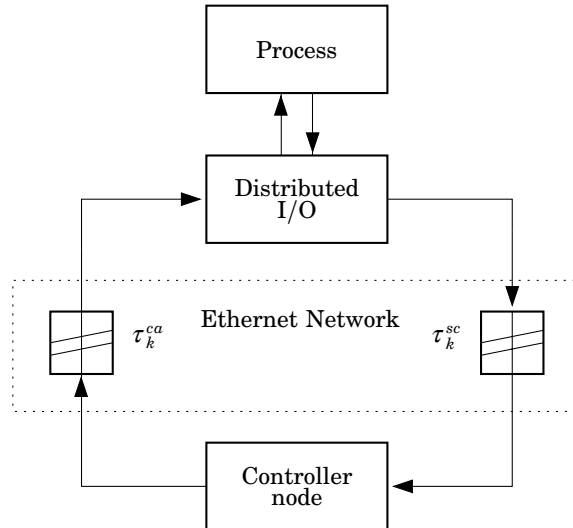


Figure 4.12 Experimental setup for the delay measurements using an Ethernet network. The network delays are τ_k^{sc} and τ_k^{ca} .

implemented on Pentium PCs. The PCs were running Windows NT as operating system. Windows NT is not intended for real-time control, this is expected to give some unpredictable software overhead in the communication delays. The experiment programs were written using the Modula-2 real-time kernel from the Department of Automatic Control. The programs were executed in Windows NT using the real-time priority option. This is the highest priority a user program can have in Windows NT. As in the experiments on CAN, clock synchronization was done with the off-line clock synchronization algorithm, see Section 4.1.

System load

The network used for experiments connects around 40 computers at the department. It is a 10 Mbit/s network, which means that the transfer time for a message is quite short. If, for instance, if a message is sent as 100 bytes, the transmission time is 0.08 ms. The software overhead in the computer nodes will be much larger than the transmission time. A network sending collision will occur if two network nodes try to send at the same time. If a collision is detected the two nodes stay off the bus for a random time. If the network is not loaded, only the normal load by computer users at the department is on the network, collisions on the network are unlikely, and the delays are expected to be constant. If

Chapter 4. Experimental Delay Measurements

the load is increased the collision probability will increase. The load can increase or decrease during an experiment due to actions made by other users of the network. To increase the load further we started one or two network intensive applications. The load cases for the experiments were:

Basic load The network load is the load on the network by other users, and the load introduced by the experiment.

One extra load Extra load was introduced by a network intensive application on a computer on the network.

Two extra loads Two extra loads was introduced by two network intensive applications on a computer on the network.

The delay measurements with low load on the network is shown in Figure 4.13. The delays are quite constant, as expected. In the delay from sensor to controller a periodic delay variation can be seen, this is expected to come from software overhead in Windows NT. If the load is increased by one extra load process on the network, the delay variation increases, see Figure 4.14, but the delays are still quite constant. The delay measurements from the experiment with two load processes are shown in Figure 4.15. Here the variations are even larger, a trend in the network can also be seen in the variations. It looks as if the delays have large variations for a while, after which the delays are more constant again.

It can be concluded that the delay variations vary with the network load. If the load during, for instance, one day will vary between the investigated load cases, the network delays will show several behaviors. As seen in the plots, both mean delay and the delay variation are influenced by the network load.

Time variations introduced by Windows NT

During the experiments it was noticed that the periodic process sending sensor data to the controller node was not executed with a regular interval. The sampling interval was programmed to be 500 ms. The resulting sampling interval, h_k , is shown together with its histogram in Figure 4.16. The programming style used for the period process is:

```
VAR t : Time;

GetTime(t);
LOOP
  IncTime(t,h);
  ...
  WaitUntil(t);
END;
```

4.2 Ethernet Network

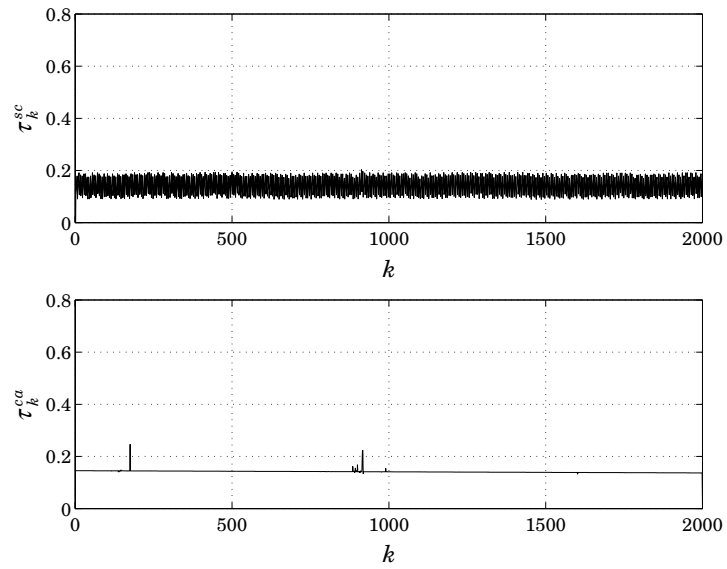


Figure 4.13 Delay measurements with low load on the Ethernet network.

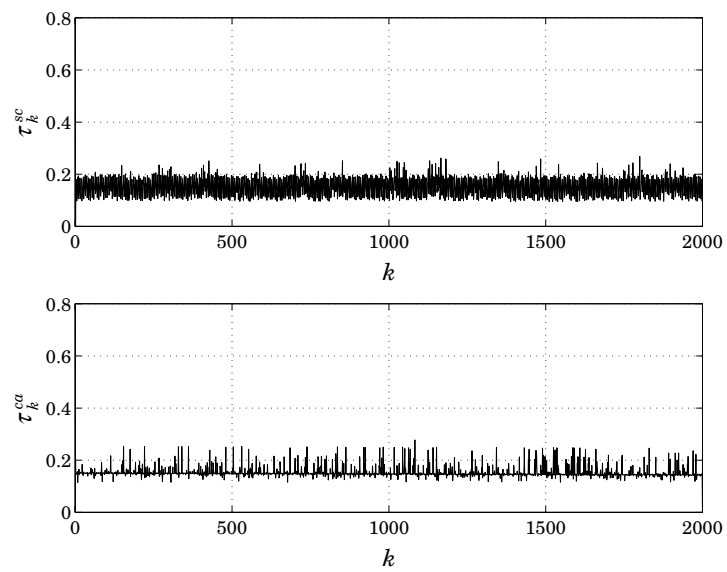


Figure 4.14 Delay measurements with one extra load on the Ethernet network.

Chapter 4. Experimental Delay Measurements

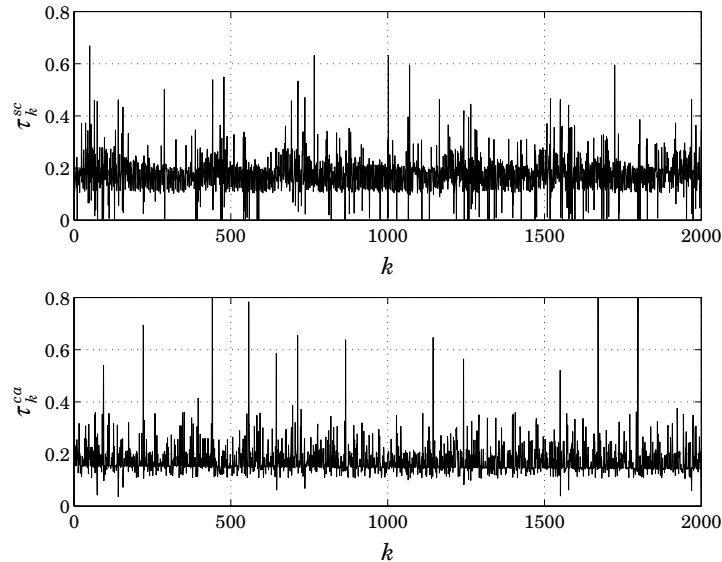


Figure 4.15 Delay measurements with two extra loads on the Ethernet network.

The use of a time reference ensures that the mean sampling interval will be correct. For instance, if we get a sampling interval longer than h , it is likely that the next sampling interval will be scheduled to be shorter than h . For the used sampling interval, 500 ms, the variation is relatively small, but it could be more critical if a shorter sampling interval was used.

Results

The Ethernet network gives varying delays if it is loaded. The delay variations occur when two transmissions collide. A collision results in that the colliding nodes stays off the bus for a random time, resulting in a random transmission delay. The delay characteristic depends highly on the network load. If the network load is varying a Markov chain model could capture the network delay behavior.

4.3 Summary

In this chapter we have presented delay experiments on two computer networks. The first was CAN, which is a network intended for real-time use. The delay behavior could be explained from knowledge about the CAN

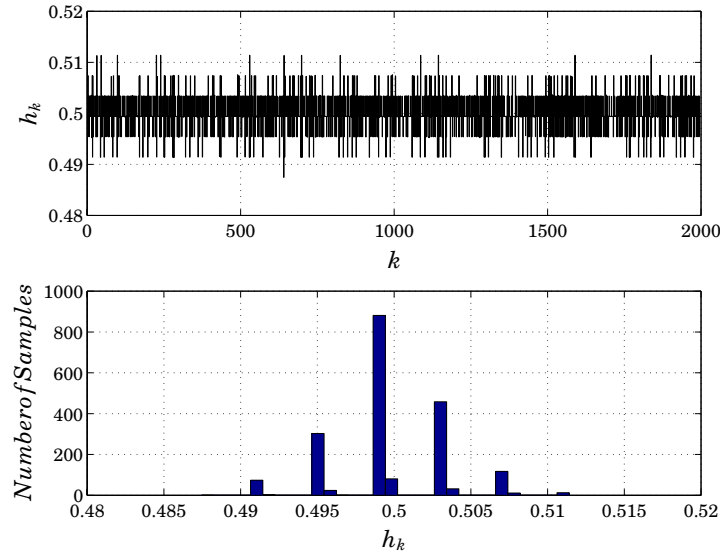


Figure 4.16 The sampling interval variation introduced by Windows NT. The sampling interval was programmed to be 500 ms. The lower plot shows the histogram for the sampling interval.

protocol. Even if CAN is intended for real-time use, we get delay variations. For a case with random load intervals a simple model was built. The model captured the probability distribution functions of the delays. More advanced models can be built if dependencies between delays are taken into account, for instance, using Markov models. The second network used for experiments was Ethernet with the TCP/IP protocol. This network is not intended for real-time use. When the network is working with low load the delays are deterministic. If network load is increased the probability for network collisions increases. When a collision occurs we get a substantially longer delay. This gives an unpredictable behavior when the load is high.

The delay measurement experiments use synchronized clocks. A new off-line clock synchronization algorithm was developed. If the clock synchronization algorithm is implemented with recursive algorithms it can also be used on-line.

5

Analysis and Control with Independent Delays

To design a controller for a distributed real-time control system it is important to know how to analyze such systems. We will study the control system setup described in Section 2.1. We will assume that the sensor node is sampled regularly at a constant sampling period h . The actuator node is assumed to be event driven, i.e., the control signal will be used as soon as it arrives. In this chapter we will develop both analysis methods and an LQG-design method when the delays have a constant known probability density function, see Chapter 3.

5.1 Closed Loop System

In Figure 5.1 the control system is illustrated in a block diagram. In this section we will determine the closed loop system when a linear control law is used. In Section 5.2 we will analyze closed loop systems. Optimal control laws are calculated in Section 5.4. The controlled process is assumed to be

$$\frac{dx}{dt} = Ax(t) + Bu(t) + v(t), \quad (5.1)$$

where $x(t) \in R^n$, $u(t) \in R^m$ and $v(t) \in R^n$. A and B are matrices of appropriate sizes. $u(t)$ is the controlled input and $v(t)$ is white noise with zero mean and covariance R_v . We will assume that the delay from sensor to actuator is less than the sampling period h , i.e., $\tau_k^{sc} + \tau_k^{ca} < h$. If this condition is not satisfied control signals may arrive at the actuator in corrupted order, which makes the analysis much harder. As described earlier, this condition can be relaxed to a condition on the delay variation

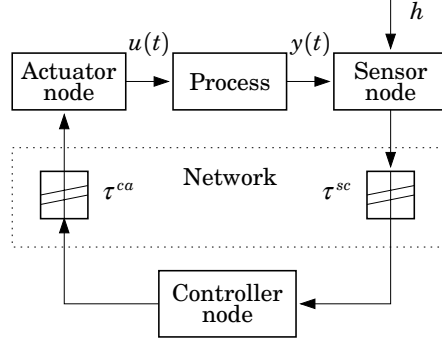


Figure 5.1 Distributed digital control system with induced delays, τ^{sc} and τ^{ca} .

being less than h . The influence from the network is collected in the variable τ_k . For instance τ_k can be a vector with the delays in the loop, i.e., $\tau_k = [\tau_k^{sc}, \tau_k^{ca}]^T$. Periodic sampling of (5.1) gives a description of the system at the sampling instants, see Chapter 3,

$$x_{k+1} = \Phi x_k + \Gamma_0(\tau_k)u_k + \Gamma_1(\tau_k)u_{k-1} + v_k. \quad (5.2)$$

The output equation is

$$y_k = C x_k + w_k, \quad (5.3)$$

where $y_k \in \mathbb{R}^p$. The stochastic processes v_k and w_k are uncorrelated Gaussian white noise with zero mean and covariance matrices R_1 and R_2 respectively.

A linear controller for this system can be written as

$$x_{k+1}^c = \Phi^c(\tau_k)x_k^c + \Gamma^c(\tau_k)y_k \quad (5.4)$$

$$u_k = C^c(\tau_k)x_k^c + D^c(\tau_k)y_k, \quad (5.5)$$

where appearance of τ_k in Φ^c , Γ^c , C^c or D^c , means that the controller knows the network delays completely or partly. Examples of such controllers are published in Krtolica *et al.* (1994), Ray (1994), and Nilsson *et al.* (1998).

From (5.2) – (5.5) we see that the closed loop system can be written as

$$z_{k+1} = \Phi(\tau_k)z_k + \Gamma(\tau_k)e_k, \quad (5.6)$$

where

$$z_k = \begin{bmatrix} x_k \\ x_k^c \\ u_{k-1} \end{bmatrix}, \quad (5.7)$$

$$\Phi(\tau_k) = \begin{bmatrix} \Phi + \Gamma_0(\tau_k)D^c(\tau_k)C & \Gamma_0(\tau_k)C^c(\tau_k) & \Gamma_1(\tau_k) \\ \Gamma^c(\tau_k)C & \Phi^c(\tau_k) & 0 \\ D^c(\tau_k)C & C^c(\tau_k) & 0 \end{bmatrix}, \quad (5.8)$$

$$e_k = \begin{bmatrix} v_k \\ w_k \end{bmatrix}, \quad (5.9)$$

and

$$\Gamma(\tau_k) = \begin{bmatrix} I & \Gamma_0(\tau_k)D^c(\tau_k) \\ 0 & \Gamma^c(\tau_k) \\ 0 & D^c(\tau_k) \end{bmatrix}. \quad (5.10)$$

The variance R of e_k is

$$R = \mathbf{E}(e_k e_k^T) = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix}.$$

The next section investigates properties of the closed loop system (5.6). Section 5.3 describes how to analyze systems with random delays using computer simulations. In Chapter 6 the analysis is made for the Markov network model described in Chapter 3.

5.2 Analysis

As described in Chapter 3, communication delays in a data network usually vary from transfer to transfer. In this situation the standard methods from linear time-invariant discrete-time systems cannot be applied. There are examples where the closed loop system is stable for all constant delays, but give instability when the delay is varying. This section develops some analysis tools for systems where consecutive delays are random and independent. First it is investigated how to calculate covariances of signals generated by (5.6), this naturally leads to a stability criterion for systems with random and independent delays.

Evaluation of Covariance

Let the closed loop system be given by (5.6), where $\{\tau_k\}$ is a random process uncorrelated with $\{e_k\}$. The form of $\Phi(\tau_k)$ and $\Gamma(\tau_k)$ is determined by the process, the communication network, and the controller structure. τ_k can be a vector consisting of the delay from sensor to controller, τ_k^{sc} , and the delay from controller to actuator, τ_k^{ca} . We assume that τ_k has known distribution, and that τ_k is independent for different k . This can be an unrealistic assumption and will be relaxed in Chapter 6, where a Markov network model is used. To keep track of the noise processes we collect the random components up to time k in

$$Y_k = \{\tau_0, \dots, \tau_k, e_0, \dots, e_k\}.$$

Introduce the state covariance P_k as

$$P_k = \mathbb{E}_{Y_{k-1}}(z_k z_k^T), \quad (5.11)$$

where the expectation is calculated with respect to noise in the process and randomness in the communication delays. By iterating (5.11) we get

$$\begin{aligned} P_{k+1} &= \mathbb{E}_{Y_k}(z_{k+1} z_{k+1}^T) \\ &= \mathbb{E}_{Y_k}((\Phi(\tau_k)z_k + \Gamma(\tau_k)e_k)(\Phi(\tau_k)z_k + \Gamma(\tau_k)e_k)^T) \\ &= \mathbb{E}_{Y_k}((\Phi(\tau_k)z_k z_k^T \Phi(\tau_k)^T + \Gamma(\tau_k)e_k e_k^T \Gamma(\tau_k)^T)) \\ &= \mathbb{E}_{\tau_k}((\Phi(\tau_k)P_k \Phi(\tau_k)^T + \Gamma(\tau_k)R\Gamma(\tau_k)^T)). \end{aligned}$$

Here we have used that τ_k , z_k and e_k are independent, and that e_k has mean zero. This is crucial for the applied technique to work and indirectly requires that τ_k and τ_{k-1} are independent. Using Kronecker products, see Appendix A, this can be written as

$$\begin{aligned} \text{vec}(P_{k+1}) &= \mathbb{E}_{\tau_k}(\Phi(\tau_k) \otimes \Phi(\tau_k)) \text{vec}(P_k) + \text{vec} \mathbb{E}_{\tau_k}(\Gamma(\tau_k)R\Gamma(\tau_k)^T) \\ &= A \text{vec}(P_k) + G, \end{aligned} \quad (5.12)$$

where

$$A = \mathbb{E}_{\tau_k}(\Phi(\tau_k) \otimes \Phi(\tau_k)), \quad (5.13)$$

$$G = \mathbb{E}_{\tau_k}(\Gamma(\tau_k) \otimes \Gamma(\tau_k)) \text{vec}(R). \quad (5.14)$$

From (5.12) we see that stability in the sense of bounded $\mathbb{E}(z_k^T z_k)$, i.e., second moment stability, is guaranteed if $\rho(\mathbb{E}(\Phi(\tau_k) \otimes \Phi(\tau_k))) < 1$, where $\rho(A)$ denotes the spectral radius of the matrix A . This stability condition appeared in Kalman (1962) in a slightly different setting. For a discussion of the connection between second moment stability and other stability concepts such as mean square stability, stochastic stability, and exponential mean square stability see Section 2.4.

Calculation of Stationary Covariance

If the recursion (5.12) is stable, $\rho(\mathbb{E}(\Phi(\tau_k) \otimes \Phi(\tau_k))) < 1$, the stationary covariance

$$P^\infty = \lim_{k \rightarrow \infty} P_k \quad (5.15)$$

can be found from the unique solution of the linear equation

$$\text{vec}(P^\infty) = \mathbb{E}(\Phi(\tau_k) \otimes \Phi(\tau_k)) \text{vec}(P^\infty) + \text{vec} \mathbb{E}(\Gamma(\tau_k) R \Gamma(\tau_k)^T). \quad (5.16)$$

Calculation of Quadratic Cost Function

In LQG-control it is of importance to evaluate quadratic cost functions like $\mathbb{E} z_k^T S(\tau_k) z_k$. This can be done as

$$\mathbb{E}_{Y_k} z_k^T S(\tau_k) z_k = \text{tr} \mathbb{E}_{Y_k} z_k^T S(\tau_k) z_k = \text{tr}(\mathbb{E}_{\tau_k} S(\tau_k) \mathbb{E}_{Y_{k-1}} z_k z_k^T), \quad (5.17)$$

which as $k \rightarrow \infty$ gives

$$\lim_{k \rightarrow \infty} \mathbb{E} z_k^T S(\tau_k) z_k = \text{tr}(\mathbb{E}_{\tau_k} S(\tau_k) P^\infty). \quad (5.18)$$

This quantity can now be calculated using the previous result.

Normally we want to calculate a cost function of the form $\mathbb{E}(x_k^T S_{11} x_k + u_k^T S_{22} u_k)$. As u_k is not an element of the vector z_k , see (5.7), this cost function can not always directly be cast into the formalism of (5.17). A solution to this problem is to rewrite u_k of (5.5) using the output equation (5.3) as

$$\begin{aligned} u_k &= C^c(\tau_k) x_k^c + D^c(\tau_k)(C x_k + w_k) \\ &= [D^c(\tau_k) C \quad C^c(\tau_k) \quad 0] z_k + D^c(\tau_k) w_k. \end{aligned}$$

Noting that τ_k and w_k are independent, and that w_k has zero mean, the cost function can be written as

$$\mathbb{E}_{Y_k}(x_k^T S_{11} x_k + u_k^T S_{22} u_k) = \mathbb{E}_{Y_k}(z_k^T S(\tau_k) z_k) + J_1,$$

where

$$S(\tau_k) = \begin{bmatrix} S_{11} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} (D^c(\tau_k)C)^T \\ C^c(\tau_k)^T \\ 0 \end{bmatrix} S_{22} [D^c(\tau_k)C \quad C^c(\tau_k) \quad 0]$$

$$J_1 = \text{tr} \left(\mathbf{E}_{\tau_k} \{ D^c(\tau_k)^T S_{22} D^c(\tau_k) \} R_2 \right),$$

where the first part again is on the form of (5.17).

EXAMPLE 5.1—CONSTANT VS RANDOM DELAY

Consider the control system in Figure 5.2 with one delay τ_k in the loop. The process output is sampled with the sampling period h . We will consider two cases for the delay.

- The delay is constant with the value $\tau_k = h/2$.
- The delay is uniformly distributed on the interval $[0, h]$.

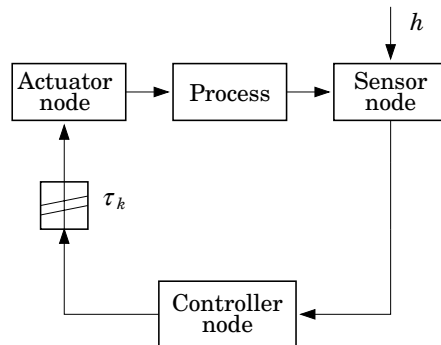


Figure 5.2 Digital control system with induced delay.

Let the process be, this can for instance be an inverted pendulum,

$$Y(s) = \frac{1}{s^2 - 1} U(s).$$

Discretizing this at the sampling instants and assuming that white noise v_k affects the state gives

$$x_{k+1} = \Phi x_k + \Gamma_0(\tau_k)u_k + \Gamma_1(\tau_k)u_{k-1} + \sqrt{h}v_k, \quad (5.19)$$

Chapter 5. Analysis and Control with Independent Delays

where v_k is white noise with variance I and zero mean. The \sqrt{h} -factor comes from sampling of continuous time white noise with the sampling period h . We will control the process with an LQ-controller which minimizes

$$J = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{k=1}^N (y_k^2 + u_k^2).$$

We let the control design take the nominal delay into account by designing the controller for (5.19) with $\tau_k = h/2$. This gives a control law

$$u_k = -L \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix},$$

where we partition L as $L = [l_x, l_u]$. Assuming that we can measure the process state the closed loop system can be written as (5.7), where

$$\Phi = \begin{bmatrix} \Phi - \Gamma_0(\tau_k)l_x & \Gamma_1(\tau_k) - \Gamma_0(\tau_k)l_u \\ -l_x & -l_u \end{bmatrix}, \quad \Gamma = \begin{bmatrix} \sqrt{h}I \\ 0 \end{bmatrix},$$

and $z_k = [x_k \quad u_{k-1}]^T$.

In Figure 5.3 the value of the cost function J is plotted for the two models of the delay for $h \in [0, 1]$. It is seen that the controller is most successful in minimizing J when the delay is constant instead of being spread over an interval. This is not surprising since the controller was designed assuming a constant delay. From calculations using (5.12) it is found that the controller fails to stabilize the process for $h > 0.785$ if the time delays are randomly varying. \square

5.3 Simulation of Systems with Network Delays

An alternative to the algebraic analysis of control systems with network induced delays is to use simulation. This can also be useful in cases when there are no analytic results available. Typical cases are time domain responses such as step responses, noise amplification, response to load disturbances, etc. Simulation can also be used when we want to study effects of nonlinear elements in the control loop, for example actuator saturation. An alternative way to evaluate a cost function, which was done exactly for the linear case in Section 5.2, is to do Monte Carlo simulations and calculate the mean value of the cost function. In Section 5.5 this technique is used for an example. A problem with this approach is the

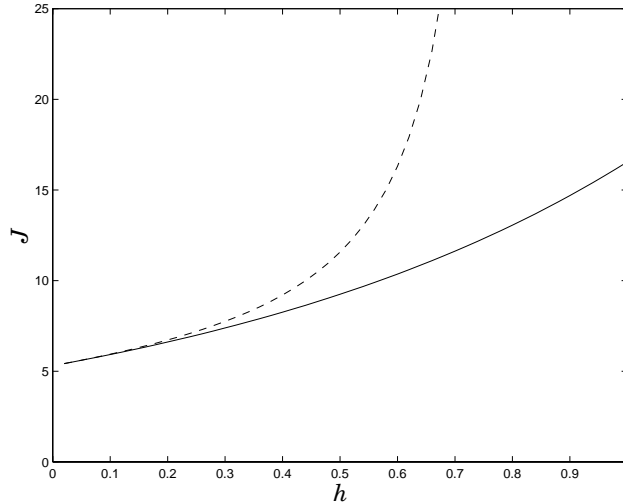


Figure 5.3 Values of the cost function J for constant delay (full line) and for uniformly distributed delay (dashed line) vs sampling period for the system in Example 5.1.

large number of samples needed to get a small confidence interval for the cost function. Although this can be lowered using variance reduction techniques, see Morgan (1984).

5.4 Optimal Stochastic Control

This section deals with controller design for the distributed digital control system in Figure 5.1. The controlled process is assumed to be

$$\frac{dx}{dt} = Ax(t) + Bu(t) + v(t), \quad (5.20)$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$ and $v(t) \in \mathbb{R}^n$. A and B are matrices of appropriate sizes, $u(t)$ is the controlled input and $v(t)$ is white noise with zero mean and covariance R_v . The influence from the network is collected in the variables τ_k^{sc} and τ_k^{ca} , which is the delay from sensor to controller and from controller to actuator. We will assume that the delay from sensor to actuator is less than the sampling period h , i.e., $\tau_k^{sc} + \tau_k^{ca} < h$. We also assume that old time delays are known when the control signal is calculated, i.e., when we calculate u_k the set $\{\tau_0^{sc}, \dots, \tau_k^{sc}, \tau_0^{ca}, \dots, \tau_{k-1}^{ca}\}$ of time

Chapter 5. Analysis and Control with Independent Delays

delays is known. The knowledge of old time delays can for instance be solved by clock synchronization and timestamping as described in Chapter 2. Sampling (5.20) at the sampling instants determined by the sensor node, see Chapter 2, gives

$$x_{k+1} = \Phi x_k + \Gamma_0(\tau_k^{sc}, \tau_k^{ca})u_k + \Gamma_1(\tau_k^{sc}, \tau_k^{ca})u_{k-1} + v_k. \quad (5.21)$$

The output equation is

$$y_k = C x_k + w_k, \quad (5.22)$$

where $y_k \in \mathbb{R}^p$. The stochastic processes v_k and w_k are uncorrelated Gaussian white noise with zero mean and covariance matrices R_1 and R_2 respectively.

If the network delays can be assumed to be constant, (5.21) degenerates to the standard process used in sampled data control theory, and thus the standard design tools can be used for synthesis, see Åström and Wittenmark (1997).

In this section we assume that the network delays are stochastically independent, see Chapter 3. First we will solve the LQ-problem for (5.21) with full state information. Then the optimal state estimator is derived, and finally the LQG-problem with output feedback is solved. Through a separation theorem it is seen that the optimal controller is the combination of the LQ-controller and the optimal state estimator. The parameters of the optimal controller can be precalculated and interpolated from a tabular. The proof of the separation property follows the lines for the delay-free case in Åström (1970). In Section 5.5 the optimal controller is compared to different control schemes in an example of distributed digital control.

Optimal State Feedback

In this section we derive the controller that minimizes the cost function

$$J_N = \mathbb{E} x_N^T Q_N x_N + \mathbb{E} \sum_{k=0}^{N-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T Q \begin{bmatrix} x_k \\ u_k \end{bmatrix}, \quad (5.23)$$

where Q is symmetric with the structure

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{bmatrix}. \quad (5.24)$$

Here Q is positive semi-definite and Q_{22} is positive definite. The solution is obtained by the same technique as for the standard LQG problem. We have the following result:

5.4 Optimal Stochastic Control

THEOREM 5.1—OPTIMAL STATE FEEDBACK

Given the plant (5.21), with noise free measurement of the state vector x_k , i.e., $y_k = x_k$. The control law that minimizes the cost function (5.23) is given by

$$u_k = -L_k(\tau_k^{sc}) \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} \quad (5.25)$$

where

$$\begin{aligned} L_k(\tau_k^{sc}) &= (\mathbf{Q}_{22} + \tilde{\mathbf{S}}_{k+1}^{22})^{-1} \begin{bmatrix} \mathbf{Q}_{12}^T + \tilde{\mathbf{S}}_{k+1}^{21} & \tilde{\mathbf{S}}_{k+1}^{23} \end{bmatrix} \\ \tilde{\mathbf{S}}_{k+1}(\tau_k^{sc}) &= \mathbf{E}_{\tau_k^{ca}} \{ \mathbf{G}^T(\tau_k^{sc}, \tau_k^{ca}) \mathbf{S}_{k+1} \mathbf{G}(\tau_k^{sc}, \tau_k^{ca}) | \tau_k^{sc} \} \\ \mathbf{G}(\tau_k^{sc}, \tau_k^{ca}) &= \begin{bmatrix} \Phi & \Gamma_0(\tau_k^{sc}, \tau_k^{ca}) & \Gamma_1(\tau_k^{sc}, \tau_k^{ca}) \\ 0 & \mathbf{I} & 0 \end{bmatrix} \\ \mathbf{S}_k &= \mathbf{E}_{\tau_k^{sc}} \{ \mathbf{F}_1^T(\tau_k^{sc}) \mathbf{Q} \mathbf{F}_1(\tau_k^{sc}) + \mathbf{F}_2^T(\tau_k^{sc}) \tilde{\mathbf{S}}_{k+1}(\tau_k^{sc}) \mathbf{F}_2(\tau_k^{sc}) \} \\ \mathbf{F}_1(\tau_k^{sc}) &= (\mathbf{Q}_{22} + \tilde{\mathbf{S}}_{k+1}^{22})^{-1} \begin{bmatrix} (\mathbf{Q}_{22} + \tilde{\mathbf{S}}_{k+1}^{22}) \mathbf{I} & 0 \\ -(\mathbf{Q}_{12}^T + \tilde{\mathbf{S}}_{k+1}^{21}) & -\tilde{\mathbf{S}}_{k+1}^{23} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} & 0 \\ -L_k(\tau_k^{sc}) & \end{bmatrix} \\ \mathbf{F}_2(\tau_k^{sc}) &= (\mathbf{Q}_{22} + \tilde{\mathbf{S}}_{k+1}^{22})^{-1} \begin{bmatrix} (\mathbf{Q}_{22} + \tilde{\mathbf{S}}_{k+1}^{22}) \mathbf{I} & 0 \\ -(\mathbf{Q}_{12}^T + \tilde{\mathbf{S}}_{k+1}^{21}) & -\tilde{\mathbf{S}}_{k+1}^{23} \\ 0 & (\mathbf{Q}_{22} + \tilde{\mathbf{S}}_{k+1}^{22}) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} & 0 \\ -L_k(\tau_k^{sc}) & \\ 0 & \mathbf{I} \end{bmatrix} \\ \mathbf{S}_N &= \begin{bmatrix} \mathbf{Q}_N & 0 \\ 0 & 0 \end{bmatrix}. \end{aligned}$$

Here $\tilde{\mathbf{S}}_{k+1}^{ij}$ is block (i, j) of the symmetric matrix $\tilde{\mathbf{S}}_{k+1}(\tau_k^{sc})$, and \mathbf{Q}_{ij} is block (i, j) of \mathbf{Q} . \square

Proof Introduce a new state variable $z_k = \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix}$. Using dynamic programming with S_k the cost to go at time k , and with α_k the part of

the cost function that cannot be affected by control, gives

$$\begin{aligned}
 z_k^T S_k z_k + \alpha_k &= \min_{u_k} \mathbf{E}_{\tau_k^{sc}, \tau_k^{ca}, v_k} \left\{ \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \mathbf{Q} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + z_{k+1}^T S_{k+1} z_{k+1} \right\} + \alpha_{k+1} \\
 &= \mathbf{E}_{\tau_k^{sc}} \min_{u_k} \mathbf{E}_{\tau_k^{ca}, v_k} \left\{ \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \mathbf{Q} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + z_{k+1}^T S_{k+1} z_{k+1} \mid \tau_k^{sc} \right\} + \alpha_{k+1} \\
 &= \mathbf{E}_{\tau_k^{sc}} \min_{u_k} \left\{ \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \mathbf{Q} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \begin{bmatrix} x_k \\ u_k \\ u_{k-1} \end{bmatrix}^T \tilde{S}_{k+1}(\tau_k^{sc}) \begin{bmatrix} x_k \\ u_k \\ u_{k-1} \end{bmatrix} \right\} \\
 &\quad + \alpha_{k+1} + \text{tr } S_{k+1}^{11} R_1.
 \end{aligned}$$

The second equality follows from the fact that τ_k^{sc} is known when u_k is determined. The third equality follows from independence of $\begin{bmatrix} x_k \\ u_k \end{bmatrix}$ and τ_k^{ca} , and from the definition of $\tilde{S}_{k+1}(\tau_k^{sc})$. The resulting expression is a quadratic form in u_k . Minimizing this with respect to u_k gives the optimal control law (5.25). From the assumption that \mathbf{Q} is symmetric it follows that S_k and \tilde{S}_k are symmetric. \square

Theorem 5.1 states that the optimal controller with full state information is a linear τ_k^{sc} -depending feedback from the state and the previous control signal

$$u_k = -L_k(\tau_k^{sc}) \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix}.$$

The equation involved in going from S_{k+1} to S_k is a stochastic Riccati equation evolving backwards in time. Each step in this iteration will contain expectation calculations with respect to the stochastic variables τ_k^{sc} and τ_k^{ca} . Under reasonable assumptions, that we will not discuss here, a stationary value S_∞ of S_k can be found by iterating the stochastic Riccati equation. In practice a tabular for $L_\infty(\tau_k^{sc})$ can then be calculated to get a control law on the form

$$u_k = -L(\tau_k^{sc}) \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix},$$

where $L(\tau_k^{sc})$ is interpolated from the tabular values of $L_\infty(\tau_k^{sc})$ in real-time. Notice that the tabular will be one-dimensional.

Optimal State Estimate

It is often impossible to get full state information. A common solution to this is to construct a state estimate from the available data. In our setup there is the problem of the random time delays which enter (5.21) in a nonlinear fashion. The fact that the old time delays up to time $k - 1$ are known at time k , however, allows the standard Kalman filter of the process state to be optimal. This is because x_k only depend on delays in the set $\{\tau_0^{sc}, \dots, \tau_{k-1}^{sc}, \tau_0^{ca}, \dots, \tau_{k-1}^{ca}\}$, as seen from (5.21).

When we are to calculate an estimate of x_k we assume that we know old values of the process output and process input. These can simply be stored in the controller for later use. We also assume that old values of the transfer delays for process output measurements and control signals are known. One way to achieve this is by timestamping of signals transferred in the control system, see Chapter 2. Denote the information available when the control signal u_k is calculated by Y_k . This has the structure

$$Y_k = \{y_0, \dots, y_k, u_0, \dots, u_{k-1}, \tau_0^{sc}, \dots, \tau_k^{sc}, \tau_0^{ca}, \dots, \tau_{k-1}^{ca}\}.$$

Notice that the sensor to controller delay τ_k^{sc} at time k and older are available, but the controller to actuator delays τ_k^{ca} are only known up to time $k - 1$.

The state estimator that minimizes the error covariance is given in the following theorem.

THEOREM 5.2—OPTIMAL STATE ESTIMATE

Given the plant (5.21)–(5.22). The estimator

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \bar{K}_k(y_k - C\hat{x}_{k|k-1}) \quad (5.26)$$

with

$$\begin{aligned} \hat{x}_{k+1|k} &= \Phi\hat{x}_{k|k-1} + \Gamma_0(\tau_k^{sc}, \tau_k^{ca})u_k + \Gamma_1(\tau_k^{sc}, \tau_k^{ca})u_{k-1} + K_k(y_k - C\hat{x}_{k|k-1}) \\ \hat{x}_{0|-1} &= \mathbf{E}(x_0) \\ P_{k+1} &= \Phi P_k \Phi^T + R_1 - \Phi P_k C^T [C P_k C^T + R_2]^{-1} C P_k \Phi \\ P_0 &= R_0 = \mathbf{E}(x_0 x_0^T) \\ K_k &= \Phi P_k C^T [C P_k C^T + R_2]^{-1} \\ \bar{K}_k &= P_k C^T [C P_k C^T + R_2]^{-1} \end{aligned}$$

minimizes the error variance $\mathbf{E}\{[x_k - \hat{x}_{k|k}]^T [x_k - \hat{x}_{k|k}] \mid Y_k\}$. The estimation error is Gaussian with zero mean and covariance $P_{k|k} = P_k - P_k C^T [C P_k C^T + R_2]^{-1} C P_k$. \square

Proof Note that the random matrices in the process (5.21), $\Gamma_0(\tau_k^{sc}, \tau_k^{ca})$ and $\Gamma_1(\tau_k^{sc}, \tau_k^{ca})$, are known when the estimate $\hat{x}_{k+1|k}$ is calculated. This simply follows from the assumption that old time delays are known when we make the estimate. By this we know how the control signal enters x_{k+1} , and the optimality of the estimator can be proved in the same way as the standard Kalman filter for time-varying, linear systems, see Anderson and Moore (1979). See also Chen *et al.* (1989). \square

Remark: Note that the filter gains K_k and \bar{K}_k do not depend on τ_k^{sc} and τ_k^{ca} . This follows from that τ_k^{sc} and τ_k^{ca} do not enter the matrix Φ . \square

Optimal Output Feedback

The following theorem justifies use of the estimated state in the optimal controller.

THEOREM 5.3—SEPARATION PROPERTY

Given the plant (5.21)–(5.22), with Y_k known when the control signal is calculated. The controller that minimizes the cost function (5.23) is given by

$$u_k = -L_k(\tau_k^{sc}) \begin{bmatrix} \hat{x}_{k|k} \\ u_{k-1} \end{bmatrix} \quad (5.27)$$

with

$$L_k(\tau_k^{sc}) = (\mathbf{Q}_{22} + \tilde{\mathbf{S}}_{k+1}^{22})^{-1} [\mathbf{Q}_{12}^T + \tilde{\mathbf{S}}_{k+1}^{21} \quad \tilde{\mathbf{S}}_{k+1}^{23}], \quad (5.28)$$

where $\tilde{\mathbf{S}}_{k+1}$ is calculated as in Theorem 5.1, and $\hat{x}_{k|k}$ is the minimum variance estimate from Theorem 5.2. \square

To prove Theorem 5.3 we will need some lemmas. The first lemma is from Åström (1970).

LEMMA 5.1

Let $\mathbb{E}[\cdot | y]$ denote the conditional mean given y . Assume that the function $f(y, u) = \mathbb{E}[l(x, y, u) | y]$ has a unique minimum with respect to $u \in U$ for all $y \in Y$. Let $u^0(y)$ denote the value of u for which the minimum is achieved. Then

$$\min_{u(y)} \mathbb{E} l(x, y, u) = \mathbb{E} l(x, y, u^0(y)) = \mathbb{E}_y \{ \min_u \mathbb{E}[l(x, y, u) | y] \}, \quad (5.29)$$

where \mathbb{E}_y denotes the mean value with respect to the distribution of y . \square

Proof This is Lemma 3.2 in Chapter 8 of Åström (1970). \square

LEMMA 5.2

With the notation in (5.26) and under the conditions for Theorem 5.2 the following holds.

$$\begin{aligned} & \mathbb{E}_{\tau_k^{ca}, v_k, w_{k+1}} \left\{ \begin{bmatrix} \hat{x}_{k+1|k+1} \\ u_k \end{bmatrix}^T S_{k+1} \begin{bmatrix} \hat{x}_{k+1|k+1} \\ u_k \end{bmatrix} \middle| Y_k \right\} \\ &= \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix}^T \tilde{S}_{k+1}(\tau_k^{sc}) \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix} + \text{tr}(R_1 C^T \bar{K}_{k+1}^T S_{k+1}^{11} \bar{K}_{k+1} C) \\ & \quad + \text{tr}(R_2 \bar{K}_{k+1}^T S_{k+1}^{11} \bar{K}_{k+1}) + \text{tr}(P_{k|k} \Phi^T C^T \bar{K}_{k+1}^T S_{k+1}^{11} \bar{K}_{k+1} C \Phi), \end{aligned}$$

where S_k^{11} is block (1, 1) of the matrix S_k . \square

Proof The calculations are similar to those in Theorem 5.1. In Theorem 5.2 the state estimate recursion is written as a recursion in $\hat{x}_{k|k-1}$. This can by use of the equations in Theorem 5.2 be rewritten as a recursion in $\hat{x}_{k|k}$.

$$\begin{aligned} \hat{x}_{k+1|k+1} &= (I - \bar{K}_{k+1} C) \hat{x}_{k+1|k} + \bar{K}_{k+1} y_{k+1} \\ &= (I - \bar{K}_{k+1} C) \{ \Phi \hat{x}_{k|k} + \Gamma_0(\tau_k^{sc}, \tau_k^{ca}) u_k + \Gamma_1(\tau_k^{sc}, \tau_k^{ca}) u_{k-1} \} \\ & \quad + \bar{K}_{k+1} \{ C(\Phi x_k + \Gamma_0(\tau_k^{sc}, \tau_k^{ca}) u_k + \Gamma_1(\tau_k^{sc}, \tau_k^{ca}) u_{k-1} + v_k) \\ & \quad + w_{k+1} \}. \end{aligned} \quad (5.30)$$

By introducing the estimation error $\tilde{x}_k = x_k - \hat{x}_{k|k}$, which we know is orthogonal to $\hat{x}_{k|k}$ from Theorem 5.2, equation (5.30) can be written as

$$\begin{aligned} \hat{x}_{k+1|k+1} &= \Phi \hat{x}_{k|k} + \Gamma_0(\tau_k^{sc}, \tau_k^{ca}) u_k + \Gamma_1(\tau_k^{sc}, \tau_k^{ca}) u_{k-1} \\ & \quad + \bar{K}_{k+1} C \Phi \tilde{x}_k + \bar{K}_{k+1} C v_k + \bar{K}_{k+1} w_{k+1}. \end{aligned} \quad (5.31)$$

From this it follows that

$$\begin{bmatrix} \hat{x}_{k+1|k+1} \\ u_k \end{bmatrix} = G(\tau_k^{sc}, \tau_k^{ca}) \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix} + H \begin{bmatrix} \tilde{x}_k \\ v_k \\ w_{k+1} \end{bmatrix}, \quad (5.32)$$

where

$$G(\tau_k^{sc}, \tau_k^{ca}) = \begin{bmatrix} \Phi & \Gamma_0(\tau_k^{sc}, \tau_k^{ca}) & \Gamma_1(\tau_k^{sc}, \tau_k^{ca}) \\ 0 & I & 0 \end{bmatrix} \quad (5.33)$$

$$H = \begin{bmatrix} \bar{K}_{k+1} C \Phi & \bar{K}_{k+1} C & \bar{K}_{k+1} \\ 0 & 0 & 0 \end{bmatrix}. \quad (5.34)$$

Chapter 5. Analysis and Control with Independent Delays

The equality in the formulation of the lemma can now be written as

$$\begin{aligned}
& \mathbf{E}_{\tau_k^{ca}, v_k, w_{k+1}} \left\{ \begin{bmatrix} \hat{x}_{k+1|k+1} \\ u_k \end{bmatrix}^T S_{k+1} \begin{bmatrix} \hat{x}_{k+1|k+1} \\ u_k \end{bmatrix} \middle| Y_k \right\} \\
&= \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix}^T \mathbf{E}_{\tau_k^{ca}} \left\{ G^T(\tau_k^{sc}, \tau_k^{ca}) S_{k+1} G(\tau_k^{sc}, \tau_k^{ca}) \middle| \tau_k^{sc} \right\} \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix} \\
&\quad + \mathbf{E}_{v_k, w_{k+1}} \left\{ \begin{bmatrix} \tilde{x}_k \\ v_k \\ w_{k+1} \end{bmatrix}^T H^T S_{k+1} H \begin{bmatrix} \tilde{x}_k \\ v_k \\ w_{k+1} \end{bmatrix} \middle| Y_k \right\} \\
&= \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix}^T \tilde{S}_{k+1}(\tau_k^{sc}) \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix} + \text{tr}(R_1 C^T \bar{K}_{k+1}^T S_{k+1}^{11} \bar{K}_{k+1} C) \\
&\quad + \text{tr}(R_2 \bar{K}_{k+1}^T S_{k+1}^{11} \bar{K}_{k+1}) + \text{tr}(P_{k|k} \Phi^T C^T \bar{K}_{k+1}^T S_{k+1}^{11} \bar{K}_{k+1} C \Phi), \quad (5.35)
\end{aligned}$$

where

$$\tilde{S}_{k+1}(\tau_k^{sc}) = \mathbf{E}_{\tau_k^{ca}} \left\{ G^T(\tau_k^{sc}, \tau_k^{ca}) S_{k+1} G(\tau_k^{sc}, \tau_k^{ca}) \middle| \tau_k^{sc} \right\}. \quad (5.36)$$

The first part of the first equality follows from that $\hat{x}_{k|k}$, u_k , and u_{k-1} are independent of \tilde{x}_k , τ_k^{ca} , v_k , and w_{k+1} . The second part of the first equality follows from that H is independent of τ_k^{ca} . The second equality follows from that \tilde{x}_k is independent of v_k and w_{k+1} . \square

Proof of Theorem 5.3 By repeated use of Lemma 5.1 we obtain a dynamic programming recursion for the future loss W . Since knowledge of $\hat{x}_{k|k}$ and $P_{k|k}$ is a sufficient statistic for the conditional distribution of x_k given Y_k and since τ_k^{sc} is known at time k we obtain the functional equation

$$\begin{aligned}
W(\hat{x}_{k|k}, P_{k|k}, k) &= \mathbf{E}_{\tau_k^{sc}} \min_{u_k} \mathbf{E} \left\{ \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T Q \begin{bmatrix} x_k \\ u_k \end{bmatrix} \right. \\
&\quad \left. + W(\hat{x}_{k+1|k+1}, P_{k+1|k+1}, k+1) \middle| Y_k \right\} \\
&= \mathbf{E}_{\tau_k^{sc}} \min_{u_k} \mathbf{E} \left\{ \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T Q \begin{bmatrix} x_k \\ u_k \end{bmatrix} \right. \\
&\quad \left. + W(\hat{x}_{k+1|k+1}, P_{k+1|k+1}, k+1) \middle| \hat{x}_{k|k}, P_{k|k}, \tau_k^{sc} \right\}. \quad (5.37)
\end{aligned}$$

5.4 Optimal Stochastic Control

The initial condition for the functional (5.37) is

$$W(\hat{x}_{N|N}, P_{N|N}, N) = \mathbf{E} \{ x_N^T Q_N x_N \mid \hat{x}_{N|N}, P_{N|N} \}. \quad (5.38)$$

In (5.37) \mathbf{E} is brought outside the minimization using Lemma 5.1, i.e., τ_k^{sc} is known when we calculate the control signal. We will now show that the functional (5.37) has a solution which is a quadratic form

$$W(\hat{x}_{k|k}, P_{k|k}, k) = \begin{bmatrix} \hat{x}_{k|k} \\ u_{k-1} \end{bmatrix}^T S_k \begin{bmatrix} \hat{x}_{k|k} \\ u_{k-1} \end{bmatrix} + s_k, \quad (5.39)$$

and that the functional is minimized by the controller of Theorem 5.1 with x_k replaced by $\hat{x}_{k|k}$. Using Theorem 5.2 we can rewrite the initial condition (5.38) as

$$W(\hat{x}_{N|N}, P_{N|N}, N) = \hat{x}_{N|N}^T Q_N \hat{x}_{N|N} + \text{tr}(Q_N P_{N|N}), \quad (5.40)$$

which clearly is on the quadratic form (5.39). Proceeding by induction we assume that (5.39) holds for $k+1$ and we will then show that it also holds for k . We have that

$$\begin{aligned} W(\hat{x}_{k|k}, P_{k|k}, k) &= \mathbf{E} \min_{\tau_k^{sc} u_k} \left\{ \begin{bmatrix} \hat{x}_{k|k} \\ u_k \end{bmatrix}^T Q \begin{bmatrix} \hat{x}_{k|k} \\ u_k \end{bmatrix} + \text{tr}(P_{k|k} Q_{11}) \right. \\ &\quad + \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix}^T \tilde{S}_{k+1}(\tau_k^{sc}) \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix} + \text{tr}(R_1 C^T \bar{K}_{k+1}^T S_{k+1}^{11} \bar{K}_{k+1} C) \\ &\quad \left. + \text{tr}(R_2 \bar{K}_{k+1}^T S_{k+1}^{11} \bar{K}_{k+1}) + \text{tr}(P_{k|k} \Phi^T C^T \bar{K}_{k+1}^T S_{k+1}^{11} \bar{K}_{k+1} C \Phi) + s_{k+1} \right\}, \end{aligned} \quad (5.41)$$

where we have used Lemma 5.2 to rewrite $\mathbf{E} W(\hat{x}_{k+1|k+1}, P_{k+1|k+1}, k+1)$. Comparing (5.41) with the quadratic form in the proof of Theorem 5.1 we see that it is minimized by the control law

$$u_k = -(Q_{22} + \tilde{S}_{k+1}^{22})^{-1} [Q_{12}^T + \tilde{S}_{k+1}^{21} \quad \tilde{S}_{k+1}^{23}] \begin{bmatrix} \hat{x}_{k|k} \\ u_{k-1} \end{bmatrix}, \quad (5.42)$$

where \tilde{S}_{k+1} is as stated in Theorem 5.1. Using the optimal control in (5.41) and applying \mathbf{E} , which can be moved inside $[\hat{x}_{k|k}^T \quad u_{k-1}^T]$, we find that $W(\hat{x}_{k|k}, P_{k|k}, k)$ is on the quadratic form (5.39). The induction is thus

completed and the criterion is minimized by the controller stated in the theorem. \square

Remark: The assumption $\tau_k^{sc} + \tau_k^{ca} < h$ implies that all measurements are available and that the control signal can be applied within each sampling period, i.e., there are no missing measurements or lost control actions. \square

5.5 Example

Consider the following plant, both plant and design specifications are taken from Doyle and Stein (1979),

$$\begin{aligned} \frac{dx}{dt} &= \begin{bmatrix} 0 & 1 \\ -3 & -4 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 35 \\ -61 \end{bmatrix} \xi \\ y &= [2 \quad 1]x + \eta, \end{aligned} \quad (5.43)$$

where $\xi(t)$ and $\eta(t)$ have mean zero and unit incremental variance. The control objective is to minimize the cost function

$$J = E \lim_{T \rightarrow \infty} \frac{1}{T} \int_0^T (x^T H^T H x + u^2) dt,$$

where $H = 4\sqrt{5} [\sqrt{35} \quad 1]$. The sampling period for the controller is chosen as $h = 0.05$. This is in accordance with the rule of thumb that is given in Åström and Wittenmark (1997). The time delays, τ_k^{sc} and τ_k^{ca} , are assumed to be uniformly distributed on the interval $[0, \alpha h/2]$ where $0 \leq \alpha \leq 1$.

The stationary cost function will be evaluated and compared for four different schemes:

- an LQG-controller neglecting the time delays,
- an LQG-controller designed for the mean delay,
- the scheme with buffers proposed in Luck and Ray (1990),
- the optimal controller derived in Section 5.4.

The first design is done without taking any time delays into account. The process and the cost function are sampled to get discrete time equivalents, and the standard LQG-controller is calculated. This gives the design

$$L = \begin{bmatrix} 38.911 \\ 8.094 \end{bmatrix}^T, K = \begin{bmatrix} 2.690 \\ -4.484 \end{bmatrix}, \bar{K} = \begin{bmatrix} 2.927 \\ -5.012 \end{bmatrix}.$$

This choice of L , K and \bar{K} gives the following closed loop poles

$$\begin{aligned}\text{sp}(\Phi - \Gamma L) &= \{0.700 \pm 0.0702i\} \\ \text{sp}(\Phi - KC) &= \{0.743, 0.173\}.\end{aligned}$$

Even if these looks reasonable, a Nyquist plot of the loop transfer function reveals a small phase margin, $\phi_m = 10.9^\circ$. The small phase margin indicates that there could be problems to handle unmodeled time delays. Numerical evaluation of (5.13) gives the stability limit $\alpha_{crit} = 0.425$ for the controller neglecting the time delays.

The second design takes account of the mean delay, $\alpha h/2$, by being designed for a process containing the mean delay.

The scheme in Luck and Ray (1990) eliminates the randomness of the time delays by introduction of timed buffers. This will, however, introduce extra time delay in the loop. The design for this scheme is done in the same way as in the standard LQG-problem.

The fourth scheme we will compare with is the optimal controller described in Section 5.4. Notice that the optimal state estimator gains \bar{K} and K will be the same for the optimal controller as if the time delays were neglected. The feedback from the estimated state will have the form

$$u_k = -L(\tau_k^{sc}) \begin{bmatrix} \hat{x}_k \\ u_{k-1} \end{bmatrix}.$$

The stationary cost function has been evaluated for the four schemes by solving the linear equation (5.16). For comparison the stationary cost has also been evaluated by Monte Carlo simulation, calculating the mean cost during $2 \cdot 10^4$ simulated samples. The results agree very well, see Figure 5.4. From Figure 5.4 it is seen that the controller neglecting the time delays fails to stabilize the process for $\alpha > \alpha_{crit}$. The optimal controller outperforms the scheme proposed in Luck and Ray (1990).

5.6 Summary

In this chapter we have developed methods for analysis and design for real-time control systems with induced network delays. The network delay model implied that delays have a known probability distribution and that the delays are mutually independent. The analysis results were done by formulating the closed loop system as

$$z_{k+1} = \Phi(\tau_k)z_k + \Gamma(\tau_k)e_k. \quad (5.44)$$

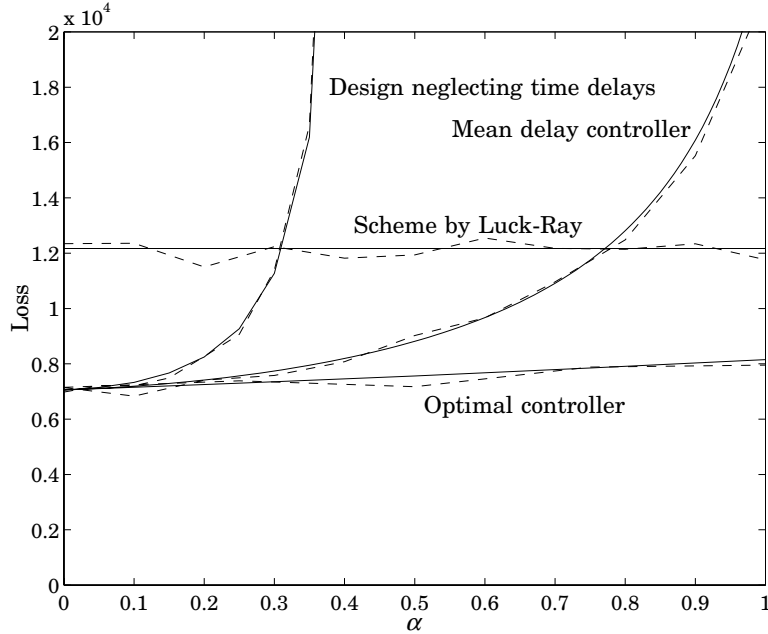


Figure 5.4 Exact calculated performance (solid lines) of the four schemes, and simulated performance (dashed lines) of system (5.43) as a function of the amount of stochastics in the time-delays. The time-delays are uniformly distributed on $[0, \alpha h/2]$. For $\alpha > 0.425$ the controller neglecting the time delays fails to stabilize the process.

The analysis results give methods for evaluation of signal covariances, and mean square stability.

The LQG-optimal control problem was solved in three steps. First it was shown that the optimal state feedback has the form

$$u_k = -L_k(\tau_k^{sc}) \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix}, \quad (5.45)$$

where τ_k^{sc} is the delay from sensor to controller. Secondly, the minimum error variance state estimator was derived. By use of timestamps we assumed that old network delays are known when we make the estimate. This lead to the conclusion that the standard Kalman filter is optimal. As the third piece it was show that the separation principle can be used, i.e., the combination of optimal state feedback and optimal state estimate is optimal in the output feedback case. The resulting controller is easy to

5.6 Summary

implement on-line. The feedback gain $L(\tau_k^{sc})$ can, for instance, be interpolated from a one dimensional tabular.

The chapter was concluded with a design example where the optimal controller was compared with some controllers from the literature. The optimal controller was shown to outperform the other controllers in the example.

6

Analysis and Control with Markov Delays

As described in Chapter 3 a more realistic model for communication delays in data networks is to model the delays as being random with the distribution selected from an underlying Markov chain. In this chapter some analysis and design tools for these systems are developed. First variances of signals and stability of the closed loop are studied for a system with a Markov chain which makes one transition every sample. Then the LQG-optimal controller is derived. In Section 7.1 the analysis results are generalized to the case when the Markov chain makes two transitions every sample, this to allow for the state of the Markov chain to change both when sending measurement and control signals.

6.1 Setup

The Markov Communication Network

The network delays are collected in the variable τ_k , where τ_k is a random variable with probability distribution given by the state of a Markov chain. For instance, τ_k can be a vector with the delays in the loop, i.e., $\tau_k = [\tau_k^{sc}, \tau_k^{ca}]^T$. The Markov chain has the state $r_k \in \{1, \dots, s\}$ when τ_k is generated. The Markov chain then makes a transition between k and $k+1$. The transition matrix for the Markov chain is $\mathbf{Q} = \{q_{ij}\}$, $i, j \in \{1, \dots, s\}$, where

$$q_{ij} = \mathbf{P}(r_{k+1} = j \mid r_k = i).$$

Introduce the Markov state probability

$$\pi_i(k) = \mathbf{P}(r_k = i), \tag{6.1}$$

and the Markov state distribution vector

$$\pi(k) = [\pi_1(k) \quad \pi_2(k) \quad \dots \quad \pi_s(k)].$$

The probability distribution for r_k is given by the recursion

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

where π_0 is the probability distribution for r_0 .

Let Y_k denote the σ -algebra generated by the random components up to time k , i.e.,

$$Y_k = \{e_0, \dots, e_k, \tau_0, \dots, \tau_k, r_0, \dots, r_k\},$$

where e_i is the noise in the process. That the probability distribution of τ_k is assumed given by the state r_k of the Markov chain means that

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

for all measurable sets F . Markov chains with continuous observation densities will be used. For Example 3.1 with varying network load we will have the probability distribution functions

$$f_i(\tau_k) = \mathbb{P}(\tau_k \mid r_k = i)$$

for $i \in [L, M, H]$ corresponding to low, medium, and high load on the network. Note that discrete observation densities are covered as a special case obtained by letting f_i be a Dirac function.

Closed Loop Sampled System

The controlled process is assumed to be linear of the form

$$\frac{dx}{dt} = Ax(t) + Bu(t) + v(t), \quad (6.2)$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, $v(t) \in \mathbb{R}^n$ and where A and B are matrices of appropriate sizes. The controlled input is denoted by $u(t)$ and $v(t)$ is white noise with zero mean and covariance R_v . As a simplification we will assume that the delay from sensor to actuator always is less than the sampling period h , i.e., $\tau_k^{sc} + \tau_k^{ca} < h$. If this condition is not satisfied control signals may arrive at the actuator in corrupted order, which makes the analysis much harder.

Remark: The assumption that the delays from sensor to actuator are less than h can be changed to that the variation in the delays from sensor to controller are less than h . This generalization requires that the process state is extended with some more old control signals. \square

The influence from the network is collected in the variable τ_k , which has a probability distribution governed by an underlying Markov chain. Discretizing (6.2) at the sampling instants, see Chapter 3, gives

$$x_{k+1} = \Phi x_k + \Gamma_0(\tau_k)u_k + \Gamma_1(\tau_k)u_{k-1} + v_k, \quad (6.3)$$

where

$$\Phi = e^{Ah} \quad (6.4)$$

$$\Gamma_0(\tau_k^{sc}, \tau_k^{ca}) = \int_0^{h-\tau_k^{sc}-\tau_k^{ca}} e^{As} ds B \quad (6.5)$$

$$\Gamma_1(\tau_k^{sc}, \tau_k^{ca}) = \int_{h-\tau_k^{sc}-\tau_k^{ca}}^h e^{As} ds B. \quad (6.6)$$

The output equation is

$$y_k = C x_k + w_k, \quad (6.7)$$

where $y_k \in \mathbb{R}^p$. The stochastic processes v_k and w_k are uncorrelated Gaussian white noise with zero mean and covariance matrices R_1 and R_2 , respectively.

A general linear controller for system (6.3)–(6.7) can be written as

$$x_{k+1}^c = \Phi^c(\tau_k, r_k)x_k^c + \Gamma^c(\tau_k, r_k)y_k \quad (6.8)$$

$$u_k = C^c(\tau_k, r_k)x_k^c + D^c(\tau_k, r_k)y_k, \quad (6.9)$$

where appearance of τ_k and r_k in Φ^c , Γ^c , C^c or D^c , means that the controller knows the network delays and network state completely or partly. Examples of such controllers are given in Krtolica *et al.* (1994), Ray (1994), and Nilsson *et al.* (1998).

From (6.3) – (6.9) we see that the closed loop system can be written as

$$z_{k+1} = \Phi(\tau_k, r_k)z_k + \Gamma(\tau_k, r_k)e_k, \quad (6.10)$$

where

$$z_k = \begin{bmatrix} x_k \\ x_k^c \\ u_{k-1} \end{bmatrix}, \quad (6.11)$$

$$\Phi(\tau_k, r_k) = \begin{bmatrix} \Phi + \Gamma_0(\tau_k)D^c(\tau_k, r_k)C & \Gamma_0(\tau_k)C^c(\tau_k, r_k) & \Gamma_1(\tau_k) \\ \Gamma^c(\tau_k, r_k)C & \Phi^c(\tau_k, r_k) & 0 \\ D^c(\tau_k, r_k)C & C^c(\tau_k, r_k) & 0 \end{bmatrix}, \quad (6.12)$$

$$e_k = \begin{bmatrix} v_k \\ w_k \end{bmatrix}, \quad (6.13)$$

and

$$\Gamma(\tau_k, r_k) = \begin{bmatrix} I & \Gamma_0(\tau_k)D^c(\tau_k, r_k) \\ 0 & \Gamma^c(\tau_k, r_k) \\ 0 & D^c(\tau_k, r_k) \end{bmatrix}. \quad (6.14)$$

The variance R of e_k is

$$R = \mathbf{E}(e_k e_k^T) = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix}.$$

6.2 Analysis

Stability and Covariances

In this section we will study how to evaluate stability and covariances for the closed loop system (6.10). The trick to avoid combinatorial explosion is to use the Markov property to collect past information in a small number of variables. There are several ways to do this. For completeness we present four different ways that are pairwise dual. One possibility is to introduce the conditional state covariance as

$$P_i(k) = \mathbf{E}_{Y_{k-1}}(z_k z_k^T \mid r_k = i), \quad (6.15)$$

and

$$\tilde{P}_i(k) = P_i(k)\pi_i(k) = \mathbf{E}_{Y_{k-1}}(z_k z_k^T \mathbf{1}_{r_k=i}), \quad (6.16)$$

Chapter 6. Analysis and Control with Markov Delays

where 1_A is a function which is 1 on (the measurable event) A , see Appendix B for a short review of the 1-function. The state covariance $P(k) = \mathbf{E}(z_k z_k^T)$ is given by

$$P(k) = \sum_{i=1}^s P_i(k) \pi_i(k) = \sum_{i=1}^s \tilde{P}_i(k). \quad (6.17)$$

The following theorem gives a forward recursion to evaluate $\tilde{P}_i(k)$.

THEOREM 6.1

Let z_k be given by a Markov driven linear system with continuous probability densities, then the covariance matrices $\tilde{P}_i(k)$ in (6.16) are given by the forward recursion

$$\tilde{P}_j(k+1) = \sum_{i=1}^s q_{ij} \mathbf{E} \left(\Phi \tilde{P}_i(k) \Phi^T + \pi_i(k) \Gamma R \Gamma^T \middle| r_k = i \right). \quad (6.18)$$

□

Proof The proof of Theorem 6.1 is presented in Section 6.3. □

It is also possible to obtain a different recursion by conditioning on r_{k-1} instead of on r_k . With the definition

$$\bar{P}_i(k) = \mathbf{E}_{Y_{k-1}} (z_k z_k^T \mathbf{1}_{r_{k-1}=i}). \quad (6.19)$$

one easily obtains

$$\bar{P}_j(k+1) = \mathbf{E} \left(\Phi \left(\sum_{i=1}^s q_{ij} \bar{P}_i(k) \right) \Phi^T + \pi_j(k) \Gamma R \Gamma^T \middle| r_k = j \right). \quad (6.20)$$

It is easily shown that the relation

$$\tilde{P}_i(k) = \sum_{j=1}^s q_{ji} \bar{P}_j(k) \quad (6.21)$$

holds between $\tilde{P}_i(k)$ and $\bar{P}_j(k)$. There are also two dual backwards recursions that evaluate future loss. If we define

$$\tilde{V}(k, i, z) = \mathbf{E} \left(\sum_{t=k}^N z_t^T \mathbf{Q} z_t \middle| r_k = i, z_k = z \right). \quad (6.22)$$

we get that $\tilde{V}(k, i, z) = z^T \tilde{S}_i(k) z + \tilde{\alpha}_i(k)$ where

$$\tilde{S}_i(k) = \mathbf{Q} + \mathbf{E} \left(\Phi^T \left(\sum_{j=1}^s q_{ij} \tilde{S}_j(k+1) \right) \Phi \middle| r_k = i \right) \quad (6.23)$$

$$\begin{aligned} \tilde{\alpha}_i(k) = \sum_{j=1}^s q_{ij} \operatorname{tr} \mathbf{E} \left(\Gamma^T \tilde{S}_j(k+1) \Gamma \mathbf{R} \middle| r_k = i \right) \\ + \sum_{j=1}^s q_{ij} \tilde{\alpha}_j(k+1). \end{aligned} \quad (6.24)$$

An alternative is to condition on r_{k-1} , i.e.,

$$\bar{V}(k, i, z) = \mathbf{E} \left(\sum_{t=k}^N z_t^T \mathbf{Q} z_t \middle| r_{k-1} = i, z_k = z \right)$$

One then gets $\bar{V}(k, i, z) = z^T \bar{S}_i(k) z + \bar{\alpha}_i(k)$ where

$$\bar{S}_i(k) = \mathbf{Q} + \sum_{j=1}^s q_{ij} \mathbf{E} \left(\Phi^T \bar{S}_j(k+1) \Phi \middle| r_k = j \right) \quad (6.25)$$

$$\begin{aligned} \bar{\alpha}_i(k) = \sum_{j=1}^s q_{ij} \left(\operatorname{tr} \mathbf{E} \left(\Gamma^T \bar{S}_j(k+1) \Gamma \mathbf{R} \middle| r_k = j \right) \right) \\ + \sum_{j=1}^s q_{ij} \bar{\alpha}_j(k+1). \end{aligned} \quad (6.26)$$

Using the Kronecker and vec notation the recursions above can be written as standard linear recursions. This is exemplified in the following corollary.

COROLLARY 6.1

The vectorized state covariance matrix $\tilde{\mathbf{P}}(k)$ satisfies the recursion

$$\tilde{\mathbf{P}}(k+1) = (\mathbf{Q}^T \otimes \mathbf{I}) \operatorname{diag}(A_i) \tilde{\mathbf{P}}(k) + (\mathbf{Q}^T \otimes \mathbf{I}) (\operatorname{diag}(\pi_i(k)) \otimes \mathbf{I}) \mathbf{G}. \quad (6.27)$$

where

$$A_i = \mathbf{E}_{\tau_k} (\Phi(\tau_k) \otimes \Phi(\tau_k) \mid r_k = i) \quad G_i = \mathbf{E}_{\tau_k} (\Gamma(\tau_k) \mathbf{R} \Gamma^T(\tau_k) \mid r_k = i)$$

$$\tilde{\mathbf{P}}(k) = \begin{bmatrix} \text{vec } \tilde{P}_1(k) \\ \text{vec } \tilde{P}_2(k) \\ \vdots \\ \text{vec } \tilde{P}_s(k) \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} \text{vec } G_1 \\ \text{vec } G_2 \\ \vdots \\ \text{vec } G_s \end{bmatrix},$$

where $\tilde{P}_i(k) = E(z_k z_k^T \mathbf{1}_{r_k=i})$. □

Proof The proof of Corollary 6.1 is presented in Section 6.3. □

The iterations described above are closely related to the Viterbi algorithm used to determine the most probably state history of Hidden Markov Models. The recursion gives a lattice (or so called Trellis) structure that can be used to efficiently implement the computations, see Rabiner (1989).

From (6.27) it is seen that the closed loop will be stable, in the sense that the covariance is bounded, if the matrix $(Q^T \otimes I) \text{diag}(A_i)$ has all its eigenvalues in the unit circle. The connections between stability in covariance and other stochastic stability criteria for jump linear systems are discussed in Feng *et al.* (1992).

This result generalizes the results in Ji *et al.* (1991) and Gajic and Qureshi (1995) in the sense that we let the Markov chain postulate the distribution of $\Phi(\tau_k)$ and $\Gamma(\tau_k)$, while Ji *et al.* (1991) and Gajic and Qureshi (1995) let the Markov chain postulate a deterministic $\Phi(\tau_k)$ and $\Gamma(\tau_k)$ for every Markov state. The results in Gajic and Qureshi (1995) are for the continuous time case.

Calculation of Stationary Covariance

We will assume that the Markov chain is stationary and regular, see Appendix B. This assures that

$$\lim_{k \rightarrow \infty} \pi(k) = \pi^\infty$$

exists and is independent of $\pi(0)$. The stationary distribution π^∞ of the Markov chain is then given uniquely by

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

In the stable case the recursion (6.27) will converge as $k \rightarrow \infty$,

$$\tilde{\mathbf{P}}^\infty = \lim_{k \rightarrow \infty} \tilde{\mathbf{P}}(k).$$

When (6.27) is a stable linear difference equation it follows that $\tilde{\mathbf{P}}^\infty$ will be the unique solution of the linear equation

$$\tilde{\mathbf{P}}^\infty = (\mathbf{Q}^T \otimes I) \text{diag}(A_i) \tilde{\mathbf{P}}^\infty + (\mathbf{Q}^T \otimes I) (\text{diag}(\pi_i^\infty) \otimes I) \mathbf{G}.$$

The stationary value of $\mathbf{E}(z_k z_k^T)$ is given by

$$P^\infty = \lim_{k \rightarrow \infty} \mathbf{E}(z_k z_k^T) = \lim_{k \rightarrow \infty} \sum_{i=1}^s \mathbf{E}(z_k z_k^T | r_k = i) P(r_k = i) = \sum_{i=1}^s \tilde{P}_i^\infty,$$

where \tilde{P}_i^∞ is the corresponding part of $\tilde{\mathbf{P}}^\infty$, see Corollary 6.1.

Calculation of Quadratic Cost Function

In LQG-control it is of importance to evaluate quadratic cost functions like $\mathbf{E} z_k^T \mathbf{S}(\tau_k, r_k) z_k$. This can be done as

$$\begin{aligned} \mathbf{E} z_k^T \mathbf{S}(\tau_k, r_k) z_k &= \text{tr} \left(\mathbf{E}_{Y_k} z_k^T \mathbf{S}(\tau_k, r_k) z_k \right) \\ &= \text{tr} \left(\sum_{i=1}^s \pi_i(k) \mathbf{E}_{Y_k} \left(z_k^T \mathbf{S}(\tau_k, r_k) z_k | r_k = i \right) \right) \\ &= \text{tr} \left(\sum_{i=1}^s \tilde{P}_i(k) \mathbf{E}_{\tau_k} \left(\mathbf{S}(\tau_k, i) | r_k = i \right) \right) \end{aligned} \quad (6.28)$$

which as $k \rightarrow \infty$ gives

$$\lim_{k \rightarrow \infty} \mathbf{E} z_k^T \mathbf{S}(\tau_k, r_k) z_k = \text{tr} \left(\sum_{i=1}^s \tilde{P}_i^\infty \mathbf{E}_{\tau_k} \left(\mathbf{S}(\tau_k, i) | r_k = i \right) \right). \quad (6.29)$$

This quantity can now be calculated using the previous result.

Normally we want to calculate a cost function of the form $\mathbf{E}(x_k^T \mathbf{S}_{11} x_k + u_k^T \mathbf{S}_{22} u_k)$. As u_k is not an element of the vector z_k , see (6.11), this cost function can not always directly be cast into the formalism of (6.28). A solution to this problem is to rewrite u_k of (6.9) using the output equation (6.7) as

$$\begin{aligned} u_k &= \mathbf{C}^c(\tau_k, r_k) x_k^c + \mathbf{D}^c(\tau_k, r_k) (\mathbf{C} x_k + w_k) \\ &= [\mathbf{D}^c(\tau_k, r_k) \mathbf{C} \quad \mathbf{C}^c(\tau_k, r_k) \quad 0] z_k + \mathbf{D}^c(\tau_k, r_k) w_k. \end{aligned}$$

Chapter 6. Analysis and Control with Markov Delays

Noting that τ_k and w_k are independent, and that w_k has zero mean, the cost function can be written as

$$\mathbf{E}(x_k^T S_{11} x_k + u_k^T S_{22} u_k) = \mathbf{E}(z_k^T S(\tau_k, r_k) z_k) + J_1,$$

where

$$S(\tau_k, r_k) = \begin{bmatrix} S_{11} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} (D^c(\tau_k, r_k)C)^T \\ C^c(\tau_k, r_k)^T \\ 0 \end{bmatrix} S_{22} \begin{bmatrix} D^c(\tau_k, r_k)C & C^c(\tau_k, r_k) & 0 \end{bmatrix}$$

$$J_1 = \text{tr} \left(\sum_{i=1}^s \pi_i(k) \mathbf{E}(D^c(\tau_k, i)^T S_{22} D^c(\tau_k, i)) R_2 \right),$$

where the first part again is on the form of (6.28).

EXAMPLE 6.1—VARIABLE DELAY

Consider the closed loop system in Figure 6.1. Assume that the distribu-

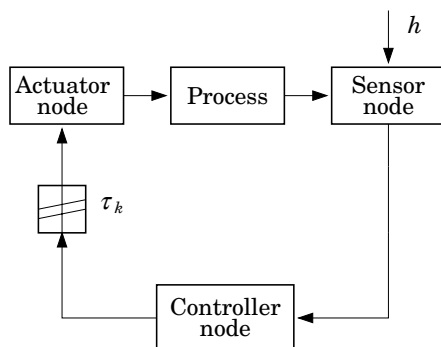


Figure 6.1 Digital control system with induced delay. The time-delay τ_k is determined by the state r_k of the Markov chain in Figure 6.2

tion of the communication delay τ_k from controller to actuator is given by the state r_k of a Markov chain. The Markov chain has two states, see Figure 6.2. The delay is

$$\tau_k = \begin{cases} 0 & \text{if } r_k = 1, \\ \text{rect}(d - a, d + a) & \text{if } r_k = 2, \end{cases} \quad (6.30)$$

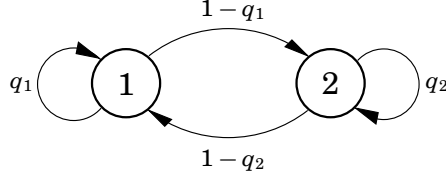


Figure 6.2 Markov chain with two states. State 1 corresponds to no delay, and state 2 corresponds to a time-delay in the interval $[d-a, d+a]$, see Equation (6.30).

where $\text{rect}(d-a, d+a)$ denotes a uniform distribution on the interval $[d-a, d+a]$. It is also assumed that $d-a > 0$ and $d+a < h$. The controlled process is

$$\begin{cases} \frac{dx}{dt} = x + u + e \\ y = x. \end{cases}$$

Let the control strategy be given by $u_k = -Lx_k$. Discretizing the process at the sampling instants determined by the sensor we get

$$x_{k+1} = \Phi x_k + \Gamma_0(\tau_k)u_k + \Gamma_1(\tau_k)u_{k-1} + \Gamma_e e_k,$$

where

$$\begin{aligned} \Phi &= e^{Ah} = e^h, \\ \Gamma_0(\tau_k) &= \begin{cases} \int_0^h e^{As} ds B = e^h - 1, & \text{if } r_k = 1, \\ \int_0^{h-\tau_k} e^{As} ds B = e^{h-\tau_k} - 1, & \text{if } r_k = 2, \end{cases} \\ \Gamma_1(\tau_k) &= \begin{cases} 0, & \text{if } r_k = 1, \\ \int_{h-\tau_k}^h e^{As} ds B = e^h - e^{h-\tau_k}, & \text{if } r_k = 2. \end{cases} \end{aligned}$$

Introduce the closed loop state z_k as

$$z_k = \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix}.$$

The closed loop system can then be written as

$$z_{k+1} = A(\tau_k)z_k + \Gamma(\tau_k)e_k,$$

Chapter 6. Analysis and Control with Markov Delays

where

$$A(\tau_k) = \begin{bmatrix} \Phi - \Gamma_0(\tau_k)L & \Gamma_1(\tau_k) \\ -L & 0 \end{bmatrix} \quad \Gamma(\tau_k) = \begin{bmatrix} \Gamma_e \\ 0 \end{bmatrix}.$$

Stability of the closed loop system is determined by the spectral radius of $(Q^T \otimes I) \text{diag}(A_i)$, where

$$\begin{aligned} A_1 &= A(0) \otimes A(0), \\ A_2 &= \mathbb{E}_{\tau_k}(A(\tau_k) \otimes A(\tau_k) | r_k = 2) \end{aligned}$$

and the transition matrix for the Markov chain is

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

Figure 6.3 shows the stability region in the $q_1 - q_2$ space for $h = 0.3$, $d = 0.8h$, $a = 0.1h$ and $L = 4$. This corresponds to a control close to deadbeat for the nominal case. In Figure 6.3 the upper left corner ($q_1 = 1$ and $q_2 = 0$) corresponds to the nominal system, i.e., a system without delay. The lower right corner ($q_1 = 0$ and $q_2 = 1$) corresponds to the system with a delay uniformly distributed on $[d - a, d + a]$. As seen from Figure 6.3 the controller does not stabilize the process in this case. When $q_1 = q_2$ the stationary distribution of the state in the Markov chain is $\pi_1 = \pi_2 = 0.5$. In Figure 6.3 this is a line from the lower left corner to the upper right corner. Note that if the Markov chain stays a too long or a too short time in the states (i.e., if $q_1 = q_2 \approx 1$ or $q_1 = q_2 \approx 0$) the closed loop is not stable, but for a region in between the closed loop is stable (i.e., if $q_1 = q_2 \approx 0.5$). \square

EXAMPLE 6.2—VACANT SAMPLING - TWO HEURISTIC CONTROL STRATEGIES

Consider the digital control system in Figure 6.4. Due to sensor failure or communication problem samples can be lost. This is called vacant sampling. Vacant sampling is modeled with a Markov chain, see Figure 6.2. If the Markov chain is in State 2 the sample is lost, and State 1 corresponds to normal operation. Let the process be

$$\begin{cases} \frac{dx}{dt} = x + u + e \\ y = x. \end{cases}$$

Discretizing the process at the sampling instants we get

$$x_{k+1} = \Phi x_k + \Gamma u_k + \Gamma_e e_k.$$

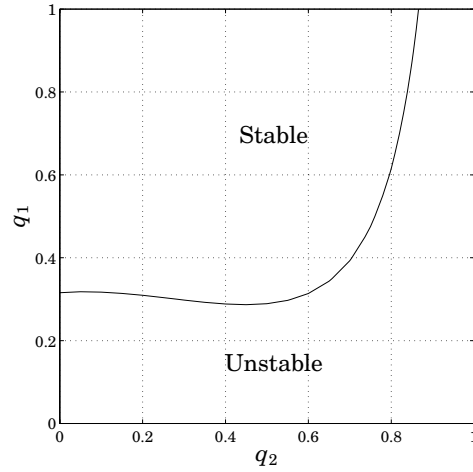


Figure 6.3 Stability region in $q_1 - q_2$ space for the system in Example 6.1. Here r_1 corresponds to a Markov state with no time delay and r_2 to a state with uniform delay on $[d - a, d + a]$. A non-optimal controller $u_k = -4x_k$ is used.

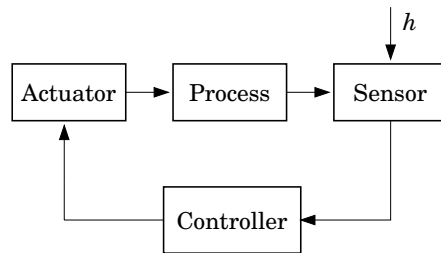


Figure 6.4 Digital control system.

We will now compare two heuristic control strategies. The first is given by $u_k = -Lx_k$ if we get a new sample, and $u_k = u_{k-1}$ if the sample is lost. The closed loop system will be

$$z_{k+1} = A(r_k)z_k + B(r_k)e_k,$$

Chapter 6. Analysis and Control with Markov Delays

where

$$\begin{aligned} z_k &= \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} \\ A(r_k) &= \begin{cases} \begin{bmatrix} \Phi - \Gamma L & 0 \\ -L & 0 \end{bmatrix} & \text{if } r_k = 1, \\ \begin{bmatrix} \Phi & \Gamma \\ 0 & I \end{bmatrix} & \text{if } r_k = 2 \end{cases} \\ B(r_k) &= \begin{bmatrix} \Gamma_e \\ 0 \end{bmatrix}. \end{aligned}$$

The sampling period is chosen to $h = 0.3$, and the noise variance such that $\Gamma_e = 1$. The feedback is designed to minimize the nominal LQ cost function, i.e., assuming $r_k \equiv 1$,

$$J = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{k=1}^N (x_k x_k^T + u_k u_k^T).$$

This gives $u_k = -Lx_k$, where $L = 1.776$. Using (6.27) we can determine stability and calculate stationary performance. The stability region and level curves for J in the $q_1 - q_2$ space are shown in Figure 6.5. Note that we fail to stabilize the system if we have long periods with lost samples (large q_2).

The second choice of controller is to use state feedback when the sensor is working, and in case of sensor failure use feedback from an estimated state \hat{x}_k . An estimate of x_k can be formed as

$$\hat{x}_{k+1} = \begin{cases} x_{k+1} & \text{if } r_k = 1, \\ \Phi \hat{x}_k + \Gamma u_k & \text{if } r_k = 2. \end{cases}$$

The closed loop system will be

$$z_{k+1} = A(r_k)z_k + B(r_k)e_k,$$

where

$$\begin{aligned} z_k &= \begin{bmatrix} x_k \\ \hat{x}_k \end{bmatrix} \\ A(r_k) &= \begin{cases} \begin{bmatrix} \Phi - \Gamma L & 0 \\ \Phi - \Gamma L & 0 \end{bmatrix} & \text{if } r_k = 1, \\ \begin{bmatrix} \Phi & -\Gamma L \\ 0 & \Phi - \Gamma L \end{bmatrix} & \text{if } r_k = 2, \end{cases} \\ B(r_k) &= \begin{bmatrix} \Gamma_e \\ 0 \end{bmatrix}. \end{aligned}$$

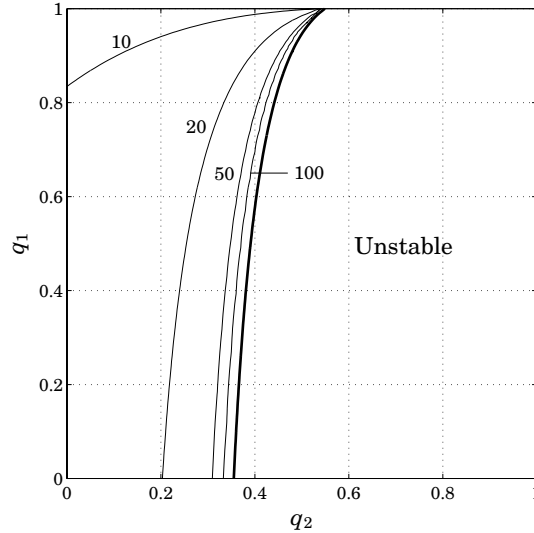


Figure 6.5 Stability region and level curves for J in the $q_1 - q_2$ space for Example 6.2. The level curves are plotted for $J = \{10, 20, 50, 100, \infty(\text{thick})\}$.

The stability region and level curves for J in the $q_1 - q_2$ space are shown in Figure 6.6. In Figure 6.5 and Figure 6.6 the upper left corner corresponds to the nominal case without lost samples. The lower right corner corresponds to a system where all samples are lost, of course the controllers fail to stabilize the system in this region. It can be seen from Figure 6.5 and Figure 6.6 that the stability region of the closed loop system is increased using the second control with an estimator. There are however regions in the $q_1 - q_2$ space where the first controller is better. In Figure 6.7 the area where the controller without prediction outperforms the controller using prediction is shown. This is not surprising, although the controller using prediction has a larger stability area it is not an optimal controller. \square

6.3 Proof of Theorem 6.1

We will need the following property of τ_k and z_k .

LEMMA 6.1—CONDITIONAL INDEPENDENCE LEMMA

The random variables τ_k and z_k are conditionally independent relative r_k ,

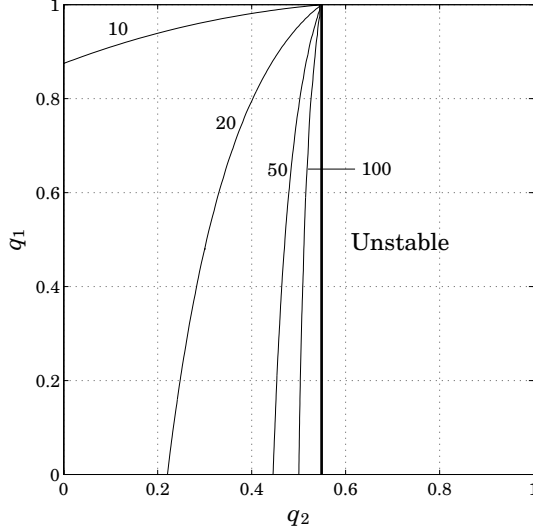


Figure 6.6 Stability region and level curves for J in the $q_1 - q_2$ space with controller using prediction. The level curves are plotted for $J = \{10, 20, 50, 100, \infty(\text{thick})\}$. Compare with Figure 6.5. Note that the stability region is increased using the second control with an estimator.

i.e.,

$$\mathbb{E}_{Y_k}(f(\tau_k)g(z_k) \mid r_k = i) = \mathbb{E}_{Y_k}(f(\tau_k) \mid r_k = i)\mathbb{E}_{Y_k}(g(z_k) \mid r_k = i),$$

for all measurable functions $f(\cdot)$ and $g(\cdot)$. \square

Proof The proof is based on Chapter 9.1 of Chung (1974), especially Theorem 9.2.1. The conditional independence property follows by Theorem 9.2.1 of Chung (1974) and from the fact that

$$\mathbb{P}(\tau_k \in E \mid z_k, r_k = i) = \mathbb{P}(\tau_k \in E \mid r_k = i).$$

This is a consequence of (6.10) and the fact that z_k is measurable with respect to Y_k . \square

Remark: z_k and τ_k are generally not independent. \square

We will also need the following Markov property.

LEMMA 6.2

Under the stated Markov assumptions it holds that

$$\mathbb{E}_{Y_k}(f(z_k, \tau_k) \mid r_{k+1} = j, r_k = i) = \mathbb{E}_{Y_k}(f(z_k, \tau_k) \mid r_k = i),$$

6.3 Proof of Theorem 6.1

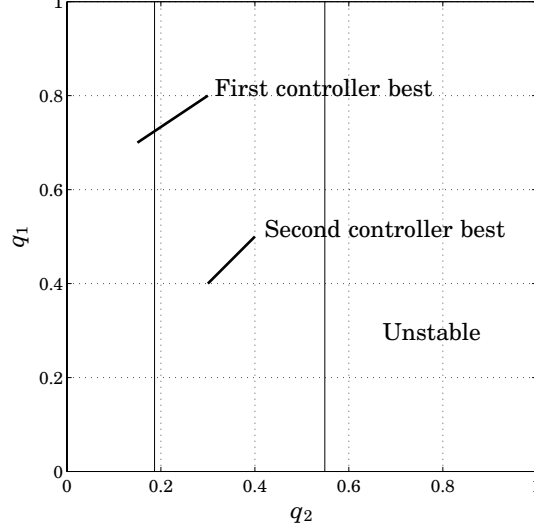


Figure 6.7 Regions in the $q_1 - q_2$ space where the two controllers are performing best. Remember that none of the controllers are optimal.

where $f(\cdot)$ is a measurable function. \square

Proof From Theorem 9.2.1 of Chung (1974) it follows that an equivalent condition is

$$P(r_{k+1} = j \mid f(z_k, \tau_k), r_k = i) = P(r_{k+1} = j \mid r_k = i).$$

The validity of this condition follows directly from the Markov property. \square

We will use the fact that if f and A are independent given B then

$$E(f \mathbf{1}_A) = \sum_B P(A \mid B) E(f \mathbf{1}_B) \quad (6.31)$$

with A being the event that $r_{k+1} = j$ and B the event $r_k = i$ and $f = z_{k+1} z_{k+1}^T$. For the proof of (6.31), see Theorem B.3. We have

$$\begin{aligned} \tilde{P}_j(k+1) &= \mathbb{E}_{Y_k} \{ z_{k+1} z_{k+1}^T \mathbf{1}_{r_{k+1}=j} \} \\ &= \mathbb{E}_{Y_k} \{ (\Phi(\tau_k) z_k z_k^T \Phi^T(\tau_k) + \Gamma(\tau_k) e_k e_k^T \Gamma^T(\tau_k)) \mathbf{1}_{r_{k+1}=j} \} \\ &= \sum_{i=1}^s q_{ij} \mathbb{E}_{Y_k} \{ (\Phi(\tau_k) z_k z_k^T \Phi^T(\tau_k) + \Gamma(\tau_k) e_k e_k^T \Gamma^T(\tau_k)) \mathbf{1}_{r_k=i} \}. \end{aligned} \quad (6.32)$$

Chapter 6. Analysis and Control with Markov Delays

By vectorizing $\tilde{P}_j(k+1)$ and using the Conditional independence lemma, Lemma 6.1, we get

$$\text{vec } \tilde{P}_j(k+1) = \sum_{i=1}^s q_{ij} A_i \text{vec } \tilde{P}_i(k) + \sum_{i=1}^s q_{ij} \pi_i(k) \text{vec } G_i,$$

where A_i and G_i are as stated in Corollary 6.1. Rewriting the sums as matrix multiplications and using Kronecker products, the recursion can be written as the linear recursion

$$\tilde{\mathbf{P}}(k+1) = (\mathbf{Q}^T \otimes I) \text{diag}(A_i) \tilde{\mathbf{P}}(k) + (\mathbf{Q}^T \otimes I) (\text{diag}(\pi_i(k)) \otimes I) \mathbf{G}, \quad (6.33)$$

which completes the proof. \square

6.4 Optimal Stochastic Control

Optimal State Feedback

In this section we solve the control problem set up by the cost function

$$J_N = \mathbf{E} x_N^T Q_N x_N + \mathbf{E} \sum_{k=0}^{N-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T Q \begin{bmatrix} x_k \\ u_k \end{bmatrix}, \quad (6.34)$$

where

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{bmatrix}. \quad (6.35)$$

is symmetric, positive semi-definite, and Q_{22} is positive definite. The solution of this problem follows by the same technique as for the standard LQG problem. We have the following result:

THEOREM 6.2—OPTIMAL STATE FEEDBACK

Given the plant (6.3), with noise free measurement of the state vector x_k , i.e., $y_k = x_k$, and knowledge of the Markov state r_k . The control law that minimizes the cost function (6.34) is given by

$$u_k = -L_k(\tau_k^{sc}, r_k) \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} \quad (6.36)$$

where, for $r_k = i$, $i = 1, \dots, s$, we have

$$\begin{aligned}
 L_k(\tau_k^{sc}, i) &= (\mathbf{Q}_{22} + \tilde{S}_i^{22}(k+1))^{-1} [\mathbf{Q}_{12}^T + \tilde{S}_i^{21}(k+1) \quad \tilde{S}_i^{23}(k+1)] \\
 \tilde{S}_i(k+1) &= \mathbf{E}_{\tau_k^{ca}} \left(G^T \sum_{j=1}^s q_{ij} S_j(k+1) G \mid \tau_k^{sc}, r_k = i \right) \\
 G &= \begin{bmatrix} \Phi & \Gamma_0(\tau_k^{sc}, \tau_k^{ca}) & \Gamma_1(\tau_k^{sc}, \tau_k^{ca}) \\ 0 & I & 0 \end{bmatrix} \\
 S_i(k) &= \mathbf{E}_{\tau_k^{sc}} \left(F_1^T \mathbf{Q} F_1 + F_2^T \tilde{S}_i(k+1) F_2 \mid r_k = i \right) \\
 F_1 &= \begin{bmatrix} I & 0 \\ -L_k(\tau_k^{sc}, r_k) \end{bmatrix} \\
 F_2 &= \begin{bmatrix} I & 0 \\ -L_k(\tau_k^{sc}, r_k) \\ 0 & I \end{bmatrix} \\
 S_i(N) &= \begin{bmatrix} \mathbf{Q}_N & 0 \\ 0 & 0 \end{bmatrix}.
 \end{aligned}$$

$\tilde{S}_i^{ab}(k)$ is block (a, b) of the symmetric matrix $\tilde{S}_i(k)$, and \mathbf{Q}_{ab} is block (a, b) of \mathbf{Q} . \square

Proof Introduce a new state variable $z_k = \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix}$. Using dynamic programming with $S_i(k)$ the cost to go at time k if $r_k = i$, and with $\alpha_i(k)$ the part of the cost function that cannot be affected by control, gives

$$\begin{aligned}
 & z_k^T S_i(k) z_k + \alpha_i(k) \\
 &= \min_{u_k} \mathbf{E}_{\tau_k^{sc}, \tau_k^{ca}, v_k} \left\{ \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \mathbf{Q} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + z_{k+1}^T \sum_{j=1}^s q_{ij} S_j(k+1) z_{k+1} \mid r_k = i \right\} \\
 & \qquad \qquad \qquad + \sum_{j=1}^s q_{ij} \alpha_j(k+1) \\
 &= \mathbf{E}_{\tau_k^{sc}} \min_{u_k} \mathbf{E}_{\tau_k^{ca}, v_k} \left\{ \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \mathbf{Q} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + z_{k+1}^T \sum_{j=1}^s q_{ij} S_j(k+1) z_{k+1} \mid \tau_k^{sc}, r_k = i \right\} \\
 & \qquad \qquad \qquad + \sum_{j=1}^s q_{ij} \alpha_j(k+1)
 \end{aligned}$$

$$\begin{aligned}
 = \mathbf{E} \min_{\tau_k^{sc}, u_k} & \left\{ \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T Q \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \begin{bmatrix} x_k \\ u_k \\ u_{k-1} \end{bmatrix}^T \tilde{S}_i(k+1) \begin{bmatrix} x_k \\ u_k \\ u_{k-1} \end{bmatrix} \middle| r_k = i \right\} \\
 & + \sum_{j=1}^s q_{ij} \alpha_j(k+1) + \text{tr} \sum_{j=1}^s q_{ij} S_j^{11}(k+1) R_1.
 \end{aligned}$$

The second equality follows from the fact that τ_k^{sc} is known when u_k is determined. The third equality follows from independence of $\begin{bmatrix} x_k \\ u_k \end{bmatrix}$ and τ_k^{ca} , and from the definition of $\tilde{S}_i(k+1)$. The resulting expression is a quadratic form in u_k . Minimizing this with respect to u_k gives the optimal control law (6.36). From the assumption that Q is symmetric it follows that $S(k)$ and $\tilde{S}_i(k)$ are symmetric. \square

The result is closely related to both standard LQ-results and the LQ-result of Chapter 5. The difference is that the optimal control law uses knowledge about the delays, τ_k^{sc} , and the delay generating system, r_k . Theorem 6.2 states that the optimal controller with full state information is a linear feedback depending on τ_k^{sc} and r_k

$$u_k = -L_k(\tau_k^{sc}, r_k) \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix}.$$

The equation involved in going from $S_j(k+1)$ to $S_i(k)$ is a coupled Riccati equation evolving backwards in time. Each step in this iteration will contain expectation calculations with respect to the stochastic variables τ_k^{sc} and τ_k^{ca} . Under reasonable assumptions, that we will not discuss here, a stationary value S_i^∞ of $S_i(k)$ can be found by iterating the stochastic Riccati equation. In practice a tabular for $L_\infty(\tau_k^{sc}, r_k)$ can then be calculated to get a control law on the form

$$u_k = -L(\tau_k^{sc}, r_k) \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix},$$

where $L(\tau_k^{sc}, r_k)$ is interpolated from the tabular values of $L_\infty(\tau_k^{sc}, r_k)$ in real-time.

EXAMPLE 6.3—VARIABLE DELAY LQ-DESIGN

Consider the process setup in Example 6.1. This is a system where we can use the derived synthesis method. The design is made by setting up

6.4 Optimal Stochastic Control

a cost function to be minimized. Here we will use (6.34) with

$$Q_{11} = C^T C = 1 \quad Q_{12} = 0 \quad Q_{22} = 1.$$

In the example $\tau_k^{sc} = 0$ and we only have statistical information about the control delay when we calculate the control signal. The optimal control will be

$$u_k = -L(r_k) \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix},$$

where r_k is the state of the Markov chain. The Markov chain state is assumed known when the control signal is calculated. Solving the coupled Riccati equations in Theorem 6.2 gives $L(r_k)$. In Figure 6.8 the LQ-cost is evaluated in the $q_1 - q_2$ space for $h = 0.3$, $d = 0.8h$ and $a = 0.1h$. Compared to the controller used in Example 6.1, Figure 6.3, we see that this controller stabilizes the system for all $q_1 - q_2$ values. From Figure 6.8

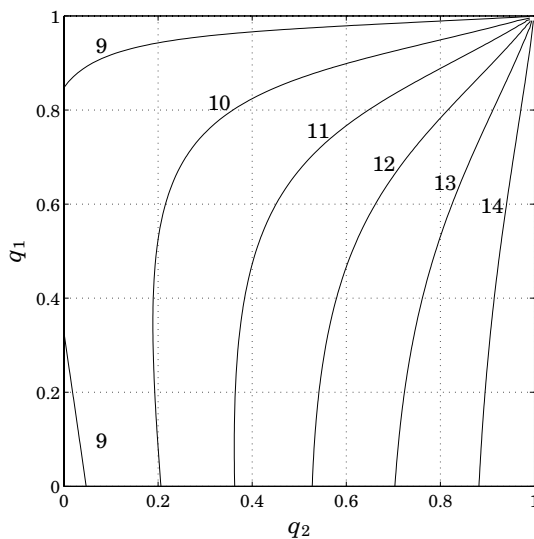


Figure 6.8 Level curves for the cost function in $q_1 - q_2$ space for the system in Example 6.3. The minimal value is 8.66, which is attained for $(q_1 = 1, q_2 = 0)$. The following level curves are $\{9, 10, 11, 12, 13, 14\}$.

it can be observed that the most advantageous cases for control is when q_1 is close to 1, and when both q_1 and q_2 are close to 0. In both these cases we have what can be called a “close to deterministic system”. If q_1

is close to 1 we will be in the no delay state most of the time. In the other case where both q_1 and q_2 are small we will switch between state 1 and state 2 almost every sample, which could be characterized as a periodic deterministic system with a period of 2 samples. \square

Optimal State Estimate

It is often impossible to get full state information. A common solution to this is to construct a state estimate from the available data. In our setup there is the problem of the random time delays which enter (6.3) in a nonlinear fashion. The fact that the old time delays up to time $k - 1$ are known at time k , however, allows the standard Kalman filter of the process state to be optimal. This is because x_k only depends on delays in the set $\{\tau_0^{sc}, \dots, \tau_{k-1}^{sc}, \tau_0^{ca}, \dots, \tau_{k-1}^{ca}\}$, as seen from (6.3).

Denote the information available when the control signal u_k is calculated by Y_k . This has the structure

$$Y_k = \{y_0, \dots, y_k, u_0, \dots, u_{k-1}, \tau_0^{sc}, \dots, \tau_k^{sc}, \tau_0^{ca}, \dots, \tau_{k-1}^{ca}, r_0, \dots, r_k\}.$$

Notice that the sensor to controller delay τ_k^{sc} at time k and older are available, but the controller to actuator delays τ_k^{ca} are only known up to time $k - 1$.

The state estimation problem considered is exactly the same as the state estimation problem considered in Theorem 5.2. The key is that old time delays are known when the state estimate is calculated. The state estimator that minimizes the error covariance is given in the following theorem.

THEOREM 6.3—OPTIMAL STATE ESTIMATE

Given the plant (6.3)–(6.7). The estimator

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \bar{K}_k(y_k - C\hat{x}_{k|k-1}) \quad (6.37)$$

with

$$\begin{aligned} \hat{x}_{k+1|k} &= \Phi\hat{x}_{k|k-1} + \Gamma_0(\tau_k^{sc}, \tau_k^{ca})u_k + \Gamma_1(\tau_k^{sc}, \tau_k^{ca})u_{k-1} + K_k(y_k - C\hat{x}_{k|k-1}) \\ \hat{x}_{0|-1} &= \mathbf{E}(x_0) \\ P_{k+1} &= \Phi P_k \Phi^T + R_1 - \Phi P_k C^T [C P_k C^T + R_2]^{-1} C P_k \Phi \\ P_0 &= R_0 = \mathbf{E}(x_0 x_0^T) \\ K_k &= \Phi P_k C^T [C P_k C^T + R_2]^{-1} \\ \bar{K}_k &= P_k C^T [C P_k C^T + R_2]^{-1} \end{aligned}$$

6.4 Optimal Stochastic Control

minimizes the error variance $E\{[x_k - \hat{x}_{k|k}]^T [x_k - \hat{x}_{k|k}] \mid Y_k\}$. Note that the filter gains K_k and \bar{K}_k do not depend on τ_k^{sc} and τ_k^{ca} . Moreover, the estimation error is Gaussian with zero mean and covariance $P_{k|k} = P_k - P_k C^T [C P_k C^T + R_2]^{-1} C P_k$. \square

Proof See proof for Theorem 5.2. \square

Optimal Output Feedback

The optimal controller satisfies the separation property and is given by the following gain-scheduled controller.

THEOREM 6.4—SEPARATION PROPERTY

Given the plant (6.3)–(6.7), with Y_k known when the control signal is calculated. The controller that minimizes the cost function (6.34) is given by

$$u_k = -L_k(\tau_k^{sc}, r_k) \begin{bmatrix} \hat{x}_{k|k} \\ u_{k-1} \end{bmatrix} \quad (6.38)$$

with

$$L_k(\tau_k^{sc}, r_k) = (Q_{22} + \tilde{S}_i^{22}(k+1))^{-1} [Q_{12}^T + \tilde{S}_i^{21}(k+1) \quad \tilde{S}_i^{23}(k+1)], \quad (6.39)$$

where $\tilde{S}_i(k)$ is calculated as in Theorem 6.2, and $\hat{x}_{k|k}$ is the minimum variance estimate from Theorem 6.3. \square

Proof The proof is presented in Section 6.5. \square

It is easy to see that this controller reduces to the optimal controller previously known, see Ji *et al.* (1991), when the probability distributions reduce to Dirac functions, i.e., when each Markov state has a deterministic Φ, Γ, C vector associated with it.

The separation property is important for use of Theorem 6.2 and Theorem 6.3. The properties that make the separation principle valid are the following.

- The knowledge of old time delays together with the assumption on noises to be Gaussian white noises makes $\{\hat{x}_{k|k}, P_{k|k}\}$ sufficient statistic, i.e., all historical information is collected in $\{\hat{x}_{k|k}, P_{k|k}\}$.
- The process is linear, which makes the state feedback controller linear.

Together the above properties make the separation principle, as known from LQG-theory, valid.

6.5 Proof of Theorem 6.4

To prove Theorem 6.4 we will need some lemmas. We will use Lemma 5.1, which also was used in the proof of the separation principle in Chapter 5. The following lemma corresponds to Lemma 5.2 in Chapter 5.

LEMMA 6.3

With the notation in (6.37) and under the conditions for Theorem 6.3 the following holds.

$$\begin{aligned} & \sum_{j=1}^s q_{ij} \mathbf{E}_{\tau_k^{ca}, v_k, w_{k+1}} \left\{ \begin{bmatrix} \hat{x}_{k+1|k+1} \\ u_k \end{bmatrix}^T S_j(k+1) \begin{bmatrix} \hat{x}_{k+1|k+1} \\ u_k \end{bmatrix} \mid Y_k, r_k = i \right\} \\ &= \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix}^T \tilde{S}_i(k+1) \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix} + \sum_{j=1}^s q_{ij} \left(\text{tr}(R_1 C^T \bar{K}_{k+1}^T S_j^{11}(k+1) \bar{K}_{k+1} C) \right. \\ & \left. + \text{tr}(R_2 \bar{K}_{k+1}^T S_j^{11}(k+1) \bar{K}_{k+1}) + \text{tr}(P_{k|k} \Phi^T C^T \bar{K}_{k+1}^T S_j^{11}(k+1) \bar{K}_{k+1} C \Phi) \right), \end{aligned}$$

where $S_j^{11}(k)$ is block (1, 1) of the matrix $S_i(k)$. \square

Proof The calculations are similar to those in Theorem 6.2. In Theorem 6.3 the state estimate recursion is written as a recursion in $\hat{x}_{k|k-1}$. This can by use of the equations in Theorem 6.3 be rewritten as a recursion in $\hat{x}_{k|k}$.

$$\begin{aligned} \hat{x}_{k+1|k+1} &= (I - \bar{K}_{k+1} C) \hat{x}_{k+1|k} + \bar{K}_{k+1} y_{k+1} \\ &= (I - \bar{K}_{k+1} C) \{ \Phi \hat{x}_{k|k} + \Gamma_0(\tau_k^{sc}, \tau_k^{ca}) u_k + \Gamma_1(\tau_k^{sc}, \tau_k^{ca}) u_{k-1} \} \\ & \quad + \bar{K}_{k+1} \{ C(\Phi x_k + \Gamma_0(\tau_k^{sc}, \tau_k^{ca}) u_k + \Gamma_1(\tau_k^{sc}, \tau_k^{ca}) u_{k-1} + v_k) \\ & \quad + w_{k+1} \}. \end{aligned} \quad (6.40)$$

By introducing the estimation error $\tilde{x}_k = x_k - \hat{x}_{k|k}$, which we know is orthogonal to $\hat{x}_{k|k}$ from Theorem 6.3, (6.40) can be written as

$$\begin{aligned} \hat{x}_{k+1|k+1} &= \Phi \hat{x}_{k|k} + \Gamma_0(\tau_k^{sc}, \tau_k^{ca}) u_k + \Gamma_1(\tau_k^{sc}, \tau_k^{ca}) u_{k-1} \\ & \quad + \bar{K}_{k+1} C \Phi \tilde{x}_k + \bar{K}_{k+1} C v_k + \bar{K}_{k+1} w_{k+1}. \end{aligned} \quad (6.41)$$

From this it follows that

$$\begin{bmatrix} \hat{x}_{k+1|k+1} \\ u_k \end{bmatrix} = G(\tau_k^{sc}, \tau_k^{ca}) \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix} + H \begin{bmatrix} \tilde{x}_k \\ v_k \\ w_{k+1} \end{bmatrix}, \quad (6.42)$$

where

$$G(\tau_k^{sc}, \tau_k^{ca}) = \begin{bmatrix} \Phi & \Gamma_0(\tau_k^{sc}, \tau_k^{ca}) & \Gamma_1(\tau_k^{sc}, \tau_k^{ca}) \\ 0 & I & 0 \end{bmatrix} \quad (6.43)$$

$$H = \begin{bmatrix} \bar{K}_{k+1}C\Phi & \bar{K}_{k+1}C & \bar{K}_{k+1} \\ 0 & 0 & 0 \end{bmatrix}. \quad (6.44)$$

The sought equality can now be written as

$$\begin{aligned} & \sum_{j=1}^s q_{ij} \mathbf{E}_{\tau_k^{ca}, v_k, w_{k+1}} \left\{ \begin{bmatrix} \hat{x}_{k+1|k+1} \\ u_k \end{bmatrix}^T S_j(k+1) \begin{bmatrix} \hat{x}_{k+1|k+1} \\ u_k \end{bmatrix} \mid Y_k, r_k = i \right\} \\ &= \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix}^T \mathbf{E}_{\tau_k^{ca}} \left\{ G^T(\tau_k^{sc}, \tau_k^{ca}) \sum_{j=1}^s q_{ij} S_j(k+1) G(\tau_k^{sc}, \tau_k^{ca}) \mid r_k = i \right\} \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix} \\ & \quad + \mathbf{E}_{v_k, w_{k+1}} \left\{ \begin{bmatrix} \tilde{x}_k \\ v_k \\ w_{k+1} \end{bmatrix}^T H^T \sum_{j=1}^s q_{ij} S_j(k+1) H \begin{bmatrix} \tilde{x}_k \\ v_k \\ w_{k+1} \end{bmatrix} \mid Y_k \right\} \\ &= \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix}^T \tilde{S}_i(k+1) \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix} + \sum_{j=1}^s q_{ij} \text{tr} S_j(k+1) H \begin{bmatrix} P_{k|k} & & \\ & R_1 & \\ & & R_2 \end{bmatrix} H^T, \end{aligned} \quad (6.45)$$

where

$$\tilde{S}_i(k+1) = \mathbf{E}_{\tau_k^{ca}} \left\{ G^T(\tau_k^{sc}, \tau_k^{ca}) \sum_{j=1}^s q_{ij} S_j(k+1) G(\tau_k^{sc}, \tau_k^{ca}) \mid r_k = i \right\}. \quad (6.46)$$

The first part of the first equality follows from that $\hat{x}_{k|k}$, u_k , and u_{k-1} are uncorrelated with \tilde{x}_k , τ_k^{ca} , v_k , and w_{k+1} . The second part of the first equality follows from that H is independent of τ_k^{ca} . The second equality follows from that \tilde{x}_k is independent of v_k and w_{k+1} . \square

By repeated use of Lemma 5.1, and the knowledge that $\hat{x}_{k|k}$ is a sufficient statistic for the conditional distribution of x_k given Y_k , we find the

functional equation

$$\begin{aligned}
 W_i(\hat{x}_{k|k}, k) &= \mathbf{E} \min_{\tau_k^{sc} u_k} \mathbf{E} \left\{ \begin{aligned} &\begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \mathbf{Q} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \\ &+ \sum_{j=1}^s q_{ij} W_j(\hat{x}_{k+1|k+1}, k+1) \mid Y_k, r_k = i, r_{k+1} = j \end{aligned} \right\} \\
 &= \mathbf{E} \min_{\tau_k^{sc} u_k} \mathbf{E} \left\{ \begin{aligned} &\begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \mathbf{Q} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \\ &+ \sum_{j=1}^s q_{ij} W_j(\hat{x}_{k+1|k+1}, k+1) \mid \hat{x}_{k|k}, \tau_k^{sc}, r_k = i, r_{k+1} = j \end{aligned} \right\}. \quad (6.47)
 \end{aligned}$$

The initial condition for the functional (6.47) is

$$W_i(\hat{x}_{N|N}, N) = \mathbf{E} \{ x_N^T \mathbf{Q}_N x_N \mid \hat{x}_{N|N}, r_N = i \}. \quad (6.48)$$

In (6.47) \mathbf{E} is brought outside the minimization using Lemma 5.1, i.e., τ_k^{sc} is known when we calculate the control signal. We will now show that the functional (6.47) has a solution which is a quadratic form

$$W_i(\hat{x}_{k|k}, k) = \begin{bmatrix} \hat{x}_{k|k} \\ u_{k-1} \end{bmatrix}^T \mathbf{S}_i(k) \begin{bmatrix} \hat{x}_{k|k} \\ u_{k-1} \end{bmatrix} + \alpha_i(k), \quad (6.49)$$

and that the functional is minimized by the controller of Theorem 6.2 with x_k replaced by $\hat{x}_{k|k}$. Using Theorem 6.3 we can rewrite the initial condition (6.48) as

$$W_i(\hat{x}_{N|N}, N) = \hat{x}_{N|N}^T \mathbf{Q}_N \hat{x}_{N|N} + \text{tr}(\mathbf{Q}_N \mathbf{P}_{N|N}), \quad (6.50)$$

which clearly is on the quadratic form (6.49). Proceeding by induction we assume that (6.49) holds for $k+1$ and we will then show that it also holds for k . We have that

$$\begin{aligned}
 W_i(\hat{x}_{k|k}, k) &= \mathbf{E} \min_{\tau_k^{sc} u_k} \left\{ \begin{aligned} &\begin{bmatrix} \hat{x}_{k|k} \\ u_k \end{bmatrix}^T \mathbf{Q} \begin{bmatrix} \hat{x}_{k|k} \\ u_k \end{bmatrix} + \text{tr}(\mathbf{P}_{k|k} \mathbf{Q}_{11}) \\ &+ \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix}^T \tilde{\mathbf{S}}_i(k+1) \begin{bmatrix} \hat{x}_{k|k} \\ u_k \\ u_{k-1} \end{bmatrix} + \sum_{j=1}^s q_{ij} \left(\text{tr}(\mathbf{R}_1 \mathbf{C}^T \bar{\mathbf{K}}_{k+1}^T \mathbf{S}_j^{11}(k+1) \bar{\mathbf{K}}_{k+1} \mathbf{C}) \right. \\ &+ \text{tr}(\mathbf{R}_2 \bar{\mathbf{K}}_{k+1}^T \mathbf{S}_j^{11}(k+1) \bar{\mathbf{K}}_{k+1}) + \text{tr}(\mathbf{P}_{k|k} \Phi^T \mathbf{C}^T \bar{\mathbf{K}}_{k+1}^T \mathbf{S}_j^{11}(k+1) \bar{\mathbf{K}}_{k+1} \mathbf{C} \Phi) \\ &\left. + \alpha_j(k+1) \right) \end{aligned} \right\}, \quad (6.51)
 \end{aligned}$$

6.6 Summary

where we have used Lemma 6.3 to rewrite $E W_j(\hat{x}_{k+1|k+1}, k+1)$. Comparing (6.51) with the quadratic form in the proof of Theorem 6.2 we see that it is minimized by the control law

$$u_k = -(Q_{22} + \tilde{S}_i^{22}(k+1))^{-1} [Q_{12}^T + \tilde{S}_i^{21}(k+1) \quad \tilde{S}_i^{23}(k+1)] \begin{bmatrix} \hat{x}_{k|k} \\ u_{k-1} \end{bmatrix}, \quad (6.52)$$

where $\tilde{S}_i(k+1)$ is as stated in Theorem 6.2. Using the optimal control in (6.51) and applying $E_{\tau_k^{sc}}$, which can be moved inside $[\hat{x}_{k|k}^T \quad u_{k-1}^T]$, we find that $W_i(\hat{x}_{k|k}, k)$ is on the quadratic form (6.49). The induction is thus completed and the criterion is minimized by the controller stated in the theorem. \square

6.6 Summary

In this chapter we have developed methods for analysis and design of distributed real-time control systems with induced network delays. The network delay model was that the delays have a known probability distribution given by an underlying known Markov chain. The analysis results were derived by formulating the closed loop system as

$$z_{k+1} = \Phi(\tau_k, r_k)z_k + \Gamma(\tau_k, r_k)e_k, \quad (6.53)$$

where τ_k is a vector with the network delays, and r_k is the state of the Markov chain. The analysis results give methods for evaluation of signal covariances, and mean square stability.

The LQG-optimal control problem was solved in three steps. First it was shown that the optimal state feedback has the form

$$u_k = -L_k(\tau_k^{sc}, r_k) \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix}, \quad (6.54)$$

where τ_k^{sc} is the delay from sensor to controller, and r_k is the state of the Markov chain. Secondly, the minimum error variance state estimator was derived. By use of timestamps we assumed that old network delays are known when we make the estimate. As a result of this, the standard Kalman filter is optimal. As the third step it was shown that the separation principle can be used, i.e., the combination of optimal state feedback and optimal state estimate is the optimal output feedback controller. A restriction with the optimal controller is that the Markov state is used in the control law. A solution to this problem could be to use an estimated Markov state in the controller. This is further discussed in Section 7.3.

7

Special Topics

The theory of Chapter 5 and Chapter 6 has many possible extensions and generalizations. It can also be used on some problems where the time variations do not come from a varying network delay. Most sections of this chapter extend some of the results in Chapter 5 and Chapter 6. Other sections treat related problems. In Section 7.1 the analysis results of Section 6.2 are generalized to a network model that makes two transitions between two sampling instants. This models the network closer. In Section 7.2 the LQG-results of Chapter 5 are generalized to the case with a varying sampling interval. The LQG-optimal controller of Chapter 6 uses the state of the Markov chain in the control law. In Section 7.3 a solution based on estimated Markov state is presented. The LQG-optimal controllers derived so far require that all measurements come from the same sensor node, and that all actuators are connected to the same actuator node. In Section 7.4 the LQG-optimal controller of Chapter 5 is generalized to the case where measurements come from several different sensor nodes, and several actuator nodes are used. The concept of “timeout” is sometimes used in computer science. Can control performance be increased using a timeout for measurements? In Section 7.5 a controller using timeout is derived. It is also shown how to analyze systems with timeouts. A closely related problem is missing samples, or vacant samples. A controller for the case with vacant samples is derived in Section 7.6. In industrial distributed control systems it is common to find asynchronous loops. For instance, different sampling intervals are used for sampling of inputs, and mirroring of signals over fieldbuses. This, together with the lack of clock synchronization, leads to variations in the control delay for these systems. In Section 7.7 a typical example is studied by simulations.

7.1 Markov Chain with Two Transitions Every Sample

In the Markov network model used in Chapter 6 the Markov chain makes one transition every sample. If we want the Markov model to make a transition between sending of the measurement signal and sending of the actuator signal we can model this by introducing extra states. In Example 3.1 we could have the states, LL , LM , LH , ML , etc. Corresponding to low load when sending the measurement and low load sending the actuator signal, etc. The problem with this model is that we need s^2 states in the Markov chain, where s is the number of network states. To model the situation in the communication network more closely we can let the Markov chain make two transitions between successive samples, one transition for each message being sent in the communication network. With this approach the Markov chain only need to have $2s$ number of states. Let the Markov chain be divided into two groups of states, one group R^{sc} that generates the probability distributions for delays from sensor to controller, and one group R^{ca} of states that generates the probability distributions for delays from controller to actuator. The transitions between states are then such that the state of the Markov chain will alter between the two groups of states, see Figure 7.1. In this section we will generalize the analysis results in Section 6.2 to the network model doing two transitions every sample. The optimal control results in Section 6.4 should be possible to generalize using the same method. When we transmit the

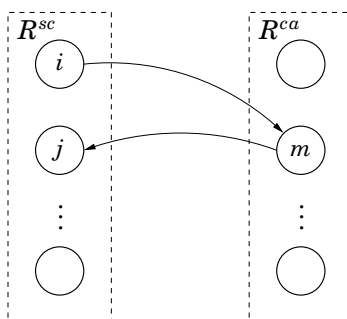


Figure 7.1 Transitions between the two sets of states. First a transition is made from state $i \in R^{sc}$ to state $m \in R^{ca}$, then a transition is made from $m \in R^{ca}$ to $j \in R^{sc}$.

data package containing y_k the state of the Markov chain will be r_k^{sc} , and when u_k is transmitted the state will be r_k^{ca} . The probability distribution of τ_k^{sc} is given by r_k^{sc} , and the distribution of τ_k^{ca} by r_k^{ca} . Let the closed loop

Chapter 7. Special Topics

system be (6.10) with

$$\begin{aligned}\tau_k &= [\tau_k^{sc} \quad \tau_k^{ca}], \\ r_k &= [r_k^{sc} \quad r_k^{ca}].\end{aligned}$$

To simplify writing we introduce the notation

$$\begin{aligned}\Phi_k &= \Phi(\tau_k, r_k) \\ \Gamma_k &= \Gamma(\tau_k, r_k),\end{aligned}$$

for the matrices in the closed loop system. Introduce the Markov state probabilities

$$\begin{aligned}\pi_j^{sc}(k) &= P(r_k^{sc} = j) \\ \pi_j^{ca}(k) &= P(r_k^{ca} = j).\end{aligned}$$

Collect the random components up to time k in

$$Y_k = \{e_0, \dots, e_k, r_0^{sc}, \dots, r_k^{sc}, \tau_0^{sc}, \dots, \tau_k^{sc}, r_0^{ca}, \dots, r_k^{ca}, \tau_0^{ca}, \dots, \tau_k^{ca}\}.$$

Introduce the conditional state covariance

$$P_i(k) = \mathbf{E}_{Y_{k-1}}(z_k z_k^T \mid r_k^{sc} = i)$$

and

$$\tilde{P}_i(k) = \pi_i^{sc}(k) P_i(k) = \mathbf{E}(z_k z_k^T \mathbf{1}_{r_k^{sc}=i}).$$

The following theorem gives a procedure to evaluate $\tilde{P}_i(k)$.

THEOREM 7.1

The vectorized state covariance matrix $\tilde{\mathbf{P}}(k)$ satisfies the recursion

$$\tilde{\mathbf{P}}(k+1) = (Q^T \otimes I) \text{block}_{mi}(q_{im} A_{im}) \tilde{\mathbf{P}}(k) + \tilde{\mathbf{G}}, \quad (7.1)$$

where

$$\begin{aligned}A_{im} &= \mathbf{E}_{\tau_k^{ca}, \tau_k^{sc}}(\Phi_k \otimes \Phi_k \mid r_k^{ca} = m, r_k^{sc} = i), \\ G_{im} &= \mathbf{E}_{\tau_k^{ca}, \tau_k^{sc}}(\Gamma_k R \Gamma_k^T \mid r_k^{ca} = m, r_k^{sc} = i), \\ \text{vec } G_j &= \sum_{m=1}^s q_{mj} \sum_{i=1}^s q_{im} \pi_i^{sc}(k) \text{vec } G_{im},\end{aligned}$$

7.1 Markov Chain with Two Transitions Every Sample

$$\tilde{\mathbf{P}}(k) = \begin{bmatrix} \text{vec } \tilde{P}_1(k) \\ \text{vec } \tilde{P}_2(k) \\ \vdots \\ \text{vec } \tilde{P}_s(k) \end{bmatrix}, \quad \tilde{\mathbf{G}} = \begin{bmatrix} \text{vec } G_1 \\ \text{vec } G_2 \\ \vdots \\ \text{vec } G_s \end{bmatrix},$$

and $\text{block}_{mi}(A(m, i))$ means the block matrix with elements $A(m, i)$ in block position (m, i) for $m, i \in \{1, \dots, s\}$. \square

Proof Using the Markov property lemma, Lemma 6.2, we get

$$\begin{aligned} \tilde{P}_j(k+1) &= \mathbf{E}_{Y_k} \{ z_{k+1} z_{k+1}^T \mathbf{1}_{r_{k+1}^{sc}=j} \} \\ &= \mathbf{E}_{Y_k} \{ (\Phi_k z_k z_k^T \Phi_k^T + \Gamma_k e_k e_k^T \Gamma_k^T) \mathbf{1}_{r_{k+1}^{sc}=j} \} \\ &= \sum_{m=1}^s \sum_{i=1}^s \mathbf{E}_{Y_k} \{ (\Phi_k z_k z_k^T \Phi_k^T + \Gamma_k e_k e_k^T \Gamma_k^T) \mathbf{1}_{r_{k+1}^{sc}=j} \mathbf{1}_{r_k^{ca}=m} \mathbf{1}_{r_k^{sc}=i} \} \\ &= \sum_{m=1}^s q_{mj} \sum_{i=1}^s q_{im} \pi_i^{sc}(k) \mathbf{E}_{Y_k} \{ \Phi_k z_k z_k^T \Phi_k^T \\ &\quad + \Gamma_k e_k e_k^T \Gamma_k^T \mid r_{k+1}^{sc} = j, r_k^{ca} = m, r_k^{sc} = i \} \\ &= \sum_{m=1}^s q_{mj} \sum_{i=1}^s q_{im} \pi_i^{sc}(k) \mathbf{E}_{Y_k} \{ \Phi_k z_k z_k^T \Phi_k^T \\ &\quad + \Gamma_k e_k e_k^T \Gamma_k^T \mid r_k^{ca} = m, r_k^{sc} = i \}. \end{aligned}$$

In the fourth equality we have used that

$$\mathbf{E}(f \mathbf{1}_A \mathbf{1}_B \mathbf{1}_C) = \mathbf{E}(f \mid A B C) \mathbf{P}(A \mid B C) \mathbf{P}(B \mid C) \mathbf{P}(C).$$

Vectorizing $\tilde{P}_j(k+1)$ and using the conditional independence lemma, Lemma 6.1, we get

$$\begin{aligned} \text{vec } \tilde{P}_j(k+1) &= \sum_{m=1}^s q_{mj} \sum_{i=1}^s q_{im} A_{im} \text{vec } \tilde{P}_i(k) \\ &\quad + \sum_{m=1}^s q_{mj} \sum_{i=1}^s q_{im} \pi_i^{sc}(k) \text{vec } G_{im}. \end{aligned}$$

Using matrix notation and Kronecker products, see Appendix A, this linear recursion can be written as the Lyapunov recursion (7.1), which completes the proof. \square

From (7.1) it is seen that the closed loop will be stable, in the sense that the covariance is finite, if the matrix $(\mathbf{Q}^T \otimes I) \text{block}_{im}(q_{im} A_{im})$ has all its eigenvalues inside the unit circle.

7.2 Sampling Interval Jitter

In some systems sampling of the process outputs is done with an irregular interval. One such example was described in Section 4.2, where it was noticed that the sampling interval for a process implemented in Windows NT was randomly varying around the nominal value. A similar effect can be seen in industrial control systems where the user can not directly setup sampling intervals. In these systems process outputs are stored in the computer memory with a fixed period. If the controller runs with another period the variation will appear as a random sampling interval. See also Section 7.7 for a discussion on time variations due to asynchronous loops.

In this section we will study the control system in Figure 7.2. We will use the network model where delays are independent from sample to sample, see Section 3.2. It is also assumed that the process output is sampled with a non-constant interval, h_k . This problem without the variations in sampling period, i.e., $h_k \equiv h$, was studied in Chapter 5. The sensor to controller delay and controller to actuator delay are denoted τ_k^{sc} and τ_k^{ca} , respectively. We assume the random sequences $\{\tau_k^{sc}\}$, $\{\tau_k^{ca}\}$, and $\{h_k\}$, are stochastically independent with known probability distributions. As in earlier chapters we make the assumption $\tau_k^{sc} + \tau_k^{ca} < h_k$. This means that control actions will arrive at the actuator node in correct order. We

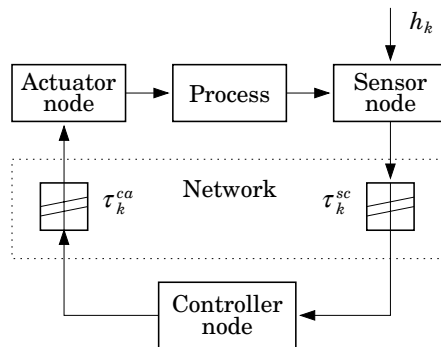


Figure 7.2 Control system with distributed I/O. The sampling is done with the irregular interval h_k . The network delays are τ_k^{sc} and τ_k^{ca} .

will also assume that old delays and old sampling intervals are known to the controller. This can be achieved by timestamping of the messages sent in the system, see Chapter 2. This means that when u_k is calculated the following delays and sample intervals are known to the controller node:

$$\{\tau_0^{sc}, \dots, \tau_k^{sc}, \tau_0^{ca}, \dots, \tau_{k-1}^{ca}, h_0, \dots, h_{k-1}\}.$$

7.2 Sampling Interval Jitter

Assumptions on the continuous time process to be linear with process noise, gives the discrete time process

$$x_{k+1} = \Phi(h_k)x_k + \Gamma_0(h_k, \tau_k^{sc}, \tau_k^{ca})u_k + \Gamma_1(h_k, \tau_k^{sc}, \tau_k^{ca})u_{k-1} + \Gamma_v(h_k)v_k, \quad (7.2)$$

where $x \in R^n$ is the process state and $u \in R^m$ the input. Here the time discretization is done at the sampling instants. The output equation is

$$y_k = Cx_k + w_k, \quad (7.3)$$

where $y_k \in R^p$. The stochastic processes v_k and w_k are uncorrelated Gaussian white noise with zero mean and covariance matrices R_1 and R_2 , respectively.

We solve the LQG-problem set up by the cost function

$$J_N = \mathbf{E} x_N^T Q_N x_N + \mathbf{E} \sum_{k=0}^{N-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T Q \begin{bmatrix} x_k \\ u_k \end{bmatrix}, \quad (7.4)$$

where

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{bmatrix} \quad (7.5)$$

is symmetric, positive semi-definite, and Q_{22} is positive definite. The solution of the stochastic optimal control problem follows the method that was used in Chapter 5.

THEOREM 7.2—OPTIMAL STATE FEEDBACK

Given the plant (7.2), with noise free measurement of the state vector x_k , i.e., $y_k = x_k$. The control law that minimizes the cost function (7.4) is

$$u_k = -L_k(\tau_k^{sc}) \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} \quad (7.6)$$

Chapter 7. Special Topics

where

$$\begin{aligned}
 L_k(\tau_k^{sc}) &= (\mathbf{Q}_{22} + \tilde{\mathbf{S}}_{k+1}^{22})^{-1} [\mathbf{Q}_{12}^T + \tilde{\mathbf{S}}_{k+1}^{21} \quad \tilde{\mathbf{S}}_{k+1}^{23}] \\
 \tilde{\mathbf{S}}_{k+1}(\tau_k^{sc}) &= \mathbf{E}_{h_k, \tau_k^{ca}} (G^T \mathbf{S}_{k+1} G | \tau_k^{sc}) \\
 G &= \begin{bmatrix} \Phi(h_k) & \Gamma_0(h_k, \tau_k^{sc}, \tau_k^{ca}) & \Gamma_1(h_k, \tau_k^{sc}, \tau_k^{ca}) \\ 0 & I & 0 \end{bmatrix} \\
 \mathbf{S}_k &= \mathbf{E}_{\tau_k^{sc}} (F_1^T(\tau_k^{sc}) \mathbf{Q} F_1(\tau_k^{sc}) + F_2^T(\tau_k^{sc}) \tilde{\mathbf{S}}_{k+1}(\tau_k^{sc}) F_2(\tau_k^{sc})) \\
 F_1(\tau_k^{sc}) &= \begin{bmatrix} I & 0 \\ -L_k(\tau_k^{sc}) & \end{bmatrix} \\
 F_2(\tau_k^{sc}) &= \begin{bmatrix} I & 0 \\ -L_k(\tau_k^{sc}) & \\ 0 & I \end{bmatrix} \\
 \mathbf{S}_N &= \begin{bmatrix} \mathbf{Q}_N & 0 \\ 0 & 0 \end{bmatrix}.
 \end{aligned}$$

$\tilde{\mathbf{S}}_k^{ij}$ is block (i, j) of $\tilde{\mathbf{S}}_k(\tau_k^{sc})$, and \mathbf{Q}_{ij} is block (i, j) of \mathbf{Q} . □

Proof The theorem is proved by dynamic programming. The proof is a simple generalization of Theorem 5.1. □

If the complete process state is not measured, a Kalman filter is used for estimation. The varying system matrix gives a time varying Kalman filter. Denote the information available to the controller when control signal u_k is calculated by Y_k . This has the structure

$$Y_k = \{y_k, y_{k-1}, \dots, \tau_k^{sc}, \tau_{k-1}^{sc}, \dots, \tau_{k-1}^{ca}, \tau_{k-2}^{ca}, \dots, h_{k-1}, h_{k-2}, \dots, u_{k-1}, u_{k-2}, \dots\}.$$

The estimator minimizing the estimation error variance is given by the following theorem.

THEOREM 7.3—OPTIMAL STATE ESTIMATE
Given the plant (7.2)–(7.3). The estimator

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \bar{\mathbf{K}}_k (y_k - C \hat{x}_{k|k-1}) \quad (7.7)$$

with

$$\begin{aligned}
\hat{x}_{k+1|k} &= \Phi(h_k)\hat{x}_{k|k-1} + \Gamma_0(h_k, \tau_k^{sc}, \tau_k^{ca})u_k \\
&\quad + \Gamma_1(h_k, \tau_k^{sc}, \tau_k^{ca})u_{k-1} + K_k(y_k - C\hat{x}_{k|k-1}) \\
\hat{x}_{0|-1} &= \mathbf{E}(x_0) \\
P_{k+1} &= \Phi(h_k)P_k\Phi^T(h_k) + \Gamma_v(h_k)R_1\Gamma_v^T(h_k) \\
&\quad - \Phi(h_k)P_kC^T[CP_kC^T + R_2]^{-1}CP_k\Phi(h_k) \\
P_0 &= R_0 = \mathbf{E}(x_0x_0^T) \\
K_k &= \Phi(h_k)P_kC^T[CP_kC^T + R_2]^{-1} \\
\bar{K}_k &= P_kC^T[CP_kC^T + R_2]^{-1}
\end{aligned}$$

minimizes the error variance $\mathbf{E}\{[x_k - \hat{x}_k]^T[x_k - \hat{x}_k] \mid Y_k\}$. The estimation error is Gaussian with zero mean and covariance $P_{k|k} = P_k - P_kC^T[CP_kC^T + R_2]^{-1}CP_k$. \square

Proof The proof follows from the same ideas as was used in Theorem 5.2. The complete knowledge of the matrices $\Phi(h_k)$, $\Gamma_0(h_k, \tau_k^{sc}, \tau_k^{ca})$, and $\Gamma_1(h_k, \tau_k^{sc}, \tau_k^{ca})$, when $\hat{x}_{k+1|k}$ is calculated makes the standard Kalman filter for time-varying systems optimal. \square

A consequence of the varying sampling interval formulation is that the Kalman filter will be time-varying. This can easily be seen in the state recursion for P_k . In the case of a constant sampling interval, $h_k \equiv h$, the P_k -equation will be a time invariant state equation, and the filter gains will converge to stationary values. For an implementation of the Kalman filter this means that the state estimation error covariance recursion, P_k , needs to be updated every sample. This recursion involves several table lookups and could be costly to implement. This complication introduced by non-periodical sampling encourages use of periodical sampling of process outputs.

In case of output feedback the LQG-optimal controller is given by the following theorem.

THEOREM 7.4—SEPARATION PROPERTY

Given the plant (7.2)–(7.3). The controller that minimizes the cost function (7.4) is given by

$$u_k = -L_k(\tau_k^{sc}) \begin{bmatrix} \hat{x}_{k|k} \\ u_{k-1} \end{bmatrix} \quad (7.8)$$

with $L_k(\tau_k^{sc})$ as in Theorem 7.2, and where $\hat{x}_{k|k}$ is the minimum variance estimate from Theorem 7.3. \square

Proof The proof is a simple generalization of Theorem 5.3. \square

7.3 Estimation of Markov State

The LQG-optimal controller for Markov communication network was derived in Chapter 6. The optimal controller was written as

$$u_k = -L_k(\tau_k^{sc}, r_k) \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix}, \quad (7.9)$$

where τ_k^{sc} is the delay from sensor to controller and r_k is the state of the Markov chain postulating probability distributions for the delays. If the state of the Markov chain can not be measured, a suboptimal controller including estimation of Markov state can be used. Several estimators of the Markov state have been proposed in the literature, see Rabiner (1989) and Elliot *et al.* (1995). What makes the proposed estimators different is the objective the estimator optimizes. One common estimator is the maximum likelihood estimator, which has the objective

$$\hat{r}_k = \max_{j=1, \dots, s} P(r_k = j | T_k), \quad (7.10)$$

where T_k is the known delays when the control signal is calculated, i.e.,

$$T_k = \{\tau_k^{sc}, \tau_{k-1}^{ca}, \tau_{k-1}^{sc}, \dots, \tau_0^{ca}, \tau_0^{sc}\}. \quad (7.11)$$

Notice that when the control calculation is made τ_k^{sc} is known, but τ_k^{ca} is unknown. The estimation objective (7.10) gives the estimator which maximizes the expected number of correct Markov states. A drawback with (7.10) is that the estimated state sequence may not even be a valid state sequence. Other approaches, see Rabiner (1989), optimizes number of correct pairs (r_k, r_{k+1}) , or triples (r_k, r_{k+1}, r_{k+2}) , etc.

Our setup has a small difference to the standard case found in the literature. When we make an estimate of the Markov state we have two new delay measurements, τ_{k-1}^{ca} and τ_k^{sc} , where the first has a distribution given by r_{k-1} and the latter has a distribution given by r_k . The evaluation of the probability for the Markov chain to be in state r_k can be split in two parts. The first part updates the probability for the state being r_k by using the measurement τ_{k-1}^{ca} . The second part updates the probability for the state being r_k by using τ_k^{sc} . The probability for the Markov chain to be in state r_k given the observed delays can be updated as

$$P(r_k = j | T_k) = \frac{P(\tau_k^{sc} | r_k = j) \sum_{i=1}^s q_{ij} P(r_{k-1} = i | \tau_{k-1}^{ca}, T_{k-1})}{\sum_{j=1}^s \{P(\tau_k^{sc} | r_k = j) \sum_{i=1}^s q_{ij} P(r_{k-1} = i | \tau_{k-1}^{ca}, T_{k-1})\}}, \quad (7.12)$$

where

$$P(r_{k-1} = i | \tau_{k-1}^{ca}, T_{k-1}) = \frac{P(\tau_{k-1}^{ca} | r_{k-1} = i) P(r_{k-1} = i | T_{k-1})}{\sum_{i=1}^s P(\tau_{k-1}^{ca} | r_{k-1} = i) P(r_{k-1} = i | T_{k-1})}. \quad (7.13)$$

The estimator uses a lattice structure, where the estimator state is $P(r_k = j | T_k)$, $j = \{1, \dots, s\}$. If the probability distribution functions for the delays have a continuous part, $f_i(\tau_k)$, the estimator (7.12)–(7.13) has to be modified. The modification can be done by discretizing the distribution function $f_i(\tau_k)$ as

$$P(\tau_k \pm d\tau) = 2 f_i(\tau_k) d\tau, \quad (7.14)$$

where $d\tau$ is chosen to reflect the resolution in the delay measurements and delay models. Using the described Markov state estimator together with the optimal controller in Chapter 6 results in a suboptimal controller. Properties of this controller is not investigated in the thesis. The resulting system of process, controller, network, and Markov state estimation, will be hard to analyze. One possible way of analysis could be to handle the system as being an adaptive system adapting to the network state, then methods from the area of adaptive control could be tried.

Another important problem is how to identify a Markov chain model from delay measurements, i.e., finding the number of Markov states, s , transition probabilities, q_{ij} , and the probability distribution functions for the delays in each Markov state. The area of identification of hidden Markov models is a large active research area, see Rabiner (1989) for a review.

7.4 The MIMO Problem

In this section we relax the assumption that we only have one sensor node and one actuator node. The generalization will be made for the LQG-optimal controller in Section 5.4, but a similar generalization can be done in the case with a Markov chain controlling the delays. We will study the MIMO, multiple input multiple output, control problem in Figure 7.3. Contrary to the control problem in Section 5.4 every signal transferred in the system may have an individual delay. We assume that there are p sensor nodes, and m actuator nodes. Let the controlled process be

$$\frac{dx}{dt} = Ax(t) + Bu(t) + v(t), \quad (7.15)$$

Chapter 7. Special Topics

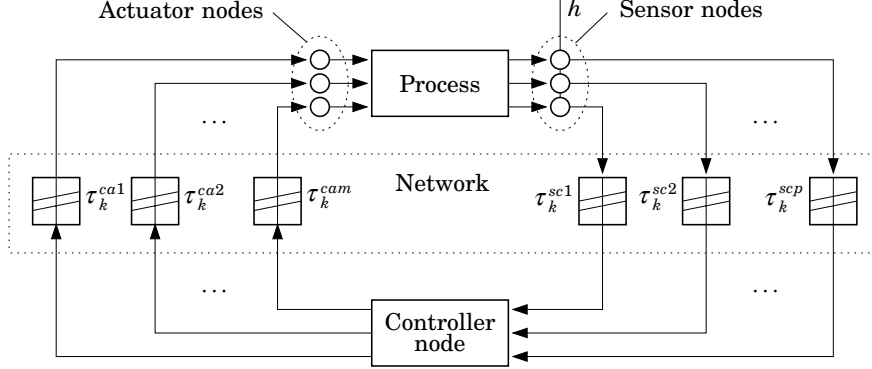


Figure 7.3 MIMO control system with several sensor nodes and actuator nodes. The transmission delays are random with individual values. Sampling of the process outputs are done synchronized for the sensor nodes.

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$ and $v(t) \in \mathbb{R}^n$. Here A and B are matrices of appropriate sizes, $u(t)$ is the controlled input and $v(t)$ is white noise with zero mean and incremental covariance R_v . We will assume that sampling in the sensor nodes are done synchronized with the period h . The measurement signals are immediately after sampling sent to the controller node. The p measurement signals will have individual random delays, τ_k^{sci} , $i \in \{1, \dots, p\}$, to the controller node. When all measurements have arrived at the controller, new control signals are calculated and sent to the actuator nodes. The control signals will have random delays, τ_k^{cai} , $i \in \{1, \dots, m\}$, to the actuator nodes. We assume that all delays in the system have constant known probability distributions. This corresponds to the network model in Section 3.2. Introduce the longest sensor to controller delay, $\bar{\tau}_k^{sc}$, as

$$\bar{\tau}_k^{sc} = \max(\tau_k^{sc1}, \tau_k^{sc2}, \dots, \tau_k^{scp}). \quad (7.16)$$

We will assume that all nodes have synchronized clocks. This will be needed both for synchronized sampling, and for timestamping of signals. By use of timestamping we assume that all old delays are known to the controller node. Discretizing the process in the sampling instants gives

$$\begin{aligned} x_{k+1} = & \Phi x_k + \Gamma_0(\bar{\tau}_k^{sc}, \tau_k^{ca1}, \dots, \tau_k^{cam})u_k \\ & + \Gamma_1(\bar{\tau}_k^{sc}, \tau_k^{ca1}, \dots, \tau_k^{cam})u_{k-1} + v_k, \end{aligned} \quad (7.17)$$

where

$$\Phi = e^{Ah} \quad (7.18)$$

$$\Gamma_0(\bar{\tau}_k^{sc}, \tau_k^{ca1}, \dots, \tau_k^{cam}) = [\Gamma_0^1(\bar{\tau}_k^{sc}, \tau_k^{ca1}) \quad \dots \quad \Gamma_0^m(\bar{\tau}_k^{sc}, \tau_k^{cam})] \quad (7.19)$$

$$\Gamma_1(\bar{\tau}_k^{sc}, \tau_k^{ca1}, \dots, \tau_k^{cam}) = [\Gamma_1^1(\bar{\tau}_k^{sc}, \tau_k^{ca1}) \quad \dots \quad \Gamma_1^m(\bar{\tau}_k^{sc}, \tau_k^{cam})] \quad (7.20)$$

$$\Gamma_0^i(\bar{\tau}_k^{sc}, \tau_k^{cai}) = \int_0^{h-\bar{\tau}_k^{sc}-\tau_k^{cai}} e^{As} ds B \quad (7.21)$$

$$\Gamma_1^i(\bar{\tau}_k^{sc}, \tau_k^{cai}) = \int_{h-\bar{\tau}_k^{sc}-\tau_k^{cai}}^h e^{As} ds B. \quad (7.22)$$

The state noise v_k has zero mean and the variance

$$R_1 = \mathbf{E}(v_k v_k^T) = \int_0^h e^{A(h-s)} R_v e^{A^T(h-s)} ds. \quad (7.23)$$

The measurement signals are

$$y_k = C x_k + w_k, \quad (7.24)$$

where $y_k \in \mathbf{R}^p$. The stochastic process w_k is uncorrelated with v_k and has zero mean and covariance matrices R_2 . As in the previous chapters we assume that the control delay, or the control delay variation, is less than a sampling interval, i.e.,

$$\max(\tau_k^{sc1}, \tau_k^{sc2}, \dots, \tau_k^{scp}) + \max(\tau_k^{ca1}, \tau_k^{ca2}, \dots, \tau_k^{cam}) < h. \quad (7.25)$$

We will solve the LQG-control problem set up by the cost function

$$J_N = \mathbf{E} x_N^T Q_N x_N + \mathbf{E} \sum_{k=0}^{N-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T Q \begin{bmatrix} x_k \\ u_k \end{bmatrix}, \quad (7.26)$$

where Q is symmetric with the structure

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{bmatrix}. \quad (7.27)$$

Here Q is positive semi-definite and Q_{22} is positive definite.

The solution of this problem follows by the same technique as was used in Chapter 5.

Chapter 7. Special Topics

THEOREM 7.5—OPTIMAL STATE FEEDBACK

Given the plant (7.17), with noise free measurement of the state vector x_k , i.e., $y_k = x_k$. The control law that minimizes the cost function (7.26) is given by

$$u_k = -L_k(\bar{\tau}_k^{sc}) \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} \quad (7.28)$$

where

$$\begin{aligned} L_k(\bar{\tau}_k^{sc}) &= (\mathbf{Q}_{22} + \tilde{\mathbf{S}}_{k+1}^{22})^{-1} [\mathbf{Q}_{12}^T + \tilde{\mathbf{S}}_{k+1}^{21} \quad \tilde{\mathbf{S}}_{k+1}^{23}] \\ \tilde{\mathbf{S}}_{k+1}(\bar{\tau}_k^{sc}) &= \mathbf{E}_{\tau_k^{ca1}, \dots, \tau_k^{cam}} (G^T \mathbf{S}_{k+1} G | \bar{\tau}_k^{sc}) \\ G &= \begin{bmatrix} \Phi & \Gamma_0(\bar{\tau}_k^{sc}, \tau_k^{ca1}, \dots, \tau_k^{cam}) & \Gamma_1(\bar{\tau}_k^{sc}, \tau_k^{ca1}, \dots, \tau_k^{cam}) \\ 0 & I & 0 \end{bmatrix} \\ \mathbf{S}_k &= \mathbf{E}_{\bar{\tau}_k^{sc}} (F_1^T(\bar{\tau}_k^{sc}) \mathbf{Q} F_1(\bar{\tau}_k^{sc}) + F_2^T(\bar{\tau}_k^{sc}) \tilde{\mathbf{S}}_{k+1}(\bar{\tau}_k^{sc}) F_2(\bar{\tau}_k^{sc})) \\ F_1(\bar{\tau}_k^{sc}) &= (\mathbf{Q}_{22} + \tilde{\mathbf{S}}_{k+1}^{22})^{-1} \begin{bmatrix} (\mathbf{Q}_{22} + \tilde{\mathbf{S}}_{k+1}^{22}) I & 0 \\ -(\mathbf{Q}_{12}^T + \tilde{\mathbf{S}}_{k+1}^{21}) & -\tilde{\mathbf{S}}_{k+1}^{23} \end{bmatrix} \\ &= \begin{bmatrix} I & 0 \\ -L_k(\bar{\tau}_k^{sc}) & \end{bmatrix} \\ F_2(\bar{\tau}_k^{sc}) &= (\mathbf{Q}_{22} + \tilde{\mathbf{S}}_{k+1}^{22})^{-1} \begin{bmatrix} (\mathbf{Q}_{22} + \tilde{\mathbf{S}}_{k+1}^{22}) I & 0 \\ -(\mathbf{Q}_{12}^T + \tilde{\mathbf{S}}_{k+1}^{21}) & -\tilde{\mathbf{S}}_{k+1}^{23} \\ 0 & (\mathbf{Q}_{22} + \tilde{\mathbf{S}}_{k+1}^{22}) \end{bmatrix} \\ &= \begin{bmatrix} I & 0 \\ -L_k(\bar{\tau}_k^{sc}) & \\ 0 & I \end{bmatrix} \\ \mathbf{S}_N &= \begin{bmatrix} \mathbf{Q}_N & 0 \\ 0 & 0 \end{bmatrix}. \end{aligned}$$

Here $\tilde{\mathbf{S}}_{k+1}^{ij}$ is block (i, j) of the symmetric matrix $\tilde{\mathbf{S}}_{k+1}(\bar{\tau}_k^{sc})$, and \mathbf{Q}_{ij} is block (i, j) of \mathbf{Q} . \square

Proof The proof is a simple generalization of the proof for Theorem 5.1. The idea is to replace τ_k^{sc} with $\bar{\tau}_k^{sc}$, and τ_k^{ca} with $\tau_k^{ca1}, \dots, \tau_k^{cam}$. \square

A similar generalization can be used for the Kalman filter. Again the Kalman filter relies on the knowledge of old time delays.

THEOREM 7.6—OPTIMAL STATE ESTIMATE

Given the plant (7.17)–(7.24). The estimator

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \bar{K}_k(y_k - C\hat{x}_{k|k-1}) \quad (7.29)$$

with

$$\begin{aligned} \hat{x}_{k+1|k} &= \Phi\hat{x}_{k|k-1} + \Gamma_0(\bar{\tau}_k^{sc}, \tau_k^{ca1}, \dots, \tau_k^{cam})u_k \\ &\quad + \Gamma_1(\bar{\tau}_k^{sc}, \tau_k^{ca1}, \dots, \tau_k^{cam})u_{k-1} + K_k(y_k - C\hat{x}_{k|k-1}) \\ \hat{x}_{0|-1} &= \mathbf{E}(x_0) \\ P_{k+1} &= \Phi P_k \Phi^T + R_1 - \Phi P_k C^T [C P_k C^T + R_2]^{-1} C P_k \Phi \\ P_0 &= R_0 = \mathbf{E}(x_0 x_0^T) \\ K_k &= \Phi P_k C^T [C P_k C^T + R_2]^{-1} \\ \bar{K}_k &= P_k C^T [C P_k C^T + R_2]^{-1} \end{aligned}$$

minimizes the error variance $\mathbf{E}\{[x_k - \hat{x}_k]^T [x_k - \hat{x}_k] \mid Y_k\}$. The estimation error is Gaussian with zero mean and covariance $P_{k|k} = P_k - P_k C^T [C P_k C^T + R_2]^{-1} C P_k$. \square

Proof The proof is a simple generalization of the proof for Theorem 5.2. The idea is to replace τ_k^{sc} with $\bar{\tau}_k^{sc}$, and τ_k^{ca} with $\tau_k^{ca1}, \dots, \tau_k^{cam}$. \square

THEOREM 7.7—SEPARATION PROPERTY

Given the plant (7.17)–(7.24). The controller that minimizes the cost function (7.26) is given by

$$u_k = -L_k(\bar{\tau}_k^{sc}) \begin{bmatrix} \hat{x}_{k|k} \\ u_{k-1} \end{bmatrix} \quad (7.30)$$

with $L_k(\tau_k^{sc})$ as in Theorem 7.5, and where $\hat{x}_{k|k}$ is the minimum variance estimate from Theorem 7.6. \square

Proof The proof is a simple generalization of the proof for Theorem 5.3. The idea is to replace τ_k^{sc} with $\bar{\tau}_k^{sc}$, and τ_k^{ca} with $\tau_k^{ca1}, \dots, \tau_k^{cam}$. \square

Sending of signals in the system will be synchronized. For instance, when the process outputs are sampled all sensor nodes want to use the network at the same time. This leads to a dependence between $\tau_k^{sci}, i \in \{1, \dots, p\}$. This dependence can, however, be modeled by making a model for $\bar{\tau}_k^{sc}$ from data measured under operation of the control system. The argument also

holds for τ_k^{cai} , $i \in \{1, \dots, m\}$, which also will be correlated. If τ_k^{cai} is modeled under control system operation these dependencies can be captured. For instance, in priority based networks the messages will always be sent in the same order, the priority order.

Other control principles can also be interesting for the MIMO problem. If the controlled process has several modes with large differences in time constants it could be advantageous to sample process outputs with different sampling intervals. It could also be possible to calculate new control signals with different intervals for the actuators. In this case we could use a multirate controller. For a discussion on multirate control, see Feuer and Goodwin (1996). Here are several open questions worthy of further investigation.

7.5 Timeout

In this section we will investigate if there is a reason to introduce the concept of “timeout” from a control perspective. Is it always beneficial to wait for a new measurement before doing control? If not, how do we design a controller that uses a timeout, and how long should this timeout be?

Problem Formulation and Design

Consider controlling the process in Figure 7.4, where τ_k is a random delay with probability distribution function $f_\tau(t)$. Assume that control signal

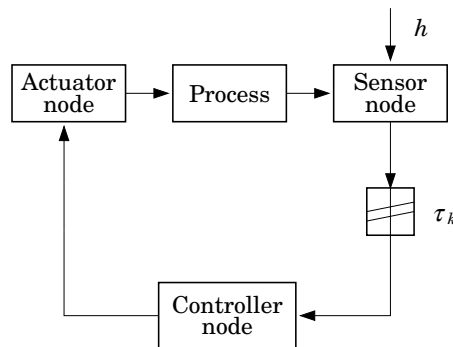


Figure 7.4 Control system with random delay from sensor to controller.

calculation is made when a new measurement arrives at the controller node, but never later than $kh + \tau^o$. The time τ^o can be considered as a timeout for the wait on a new measurement. If the controlled process is

a continuous-time time-invariant linear process, the sampled process can be written as

$$x_{k+1} = \Phi x_k + \Gamma_0(\tau_k^o)u_k + \Gamma_1(\tau_k^o)u_{k-1} + \Gamma_v v_k, \quad (7.31)$$

where

$$\tau_k^o = \min(\tau_k, \tau^o) \quad (7.32)$$

With this problem formulation τ^o is one of the design parameters in the controller design. The idea for controller design is to do control based on prediction if we get a timeout. The control instants have the probability distribution function

$$f_{\tau^o}(t) = \Psi(0, \tau^o, t)f_{\tau}(t) + \int_{\tau^o}^h f_{\tau}(s) ds \delta(t - \tau^o), \quad (7.33)$$

where

$$\Psi(a, b, t) = \begin{cases} 0 & \text{if } t < a \text{ or } t \geq b, \\ 1 & \text{if } a \leq t < b. \end{cases} \quad (7.34)$$

If we at these instants would have the latest measurement y_k available, the LQG-optimal controller would be

$$u_k = -L(\tau_k) \begin{bmatrix} \hat{x}_{k|k} \\ u_{k-1} \end{bmatrix}. \quad (7.35)$$

The measurement y_k is, however, not available when there is a timeout. The newest measurement we will have is y_{k-1} . This leads to the idea of using the control structure

$$u_k = \begin{cases} -L(\tau_k) \begin{bmatrix} \hat{x}_{k|k} \\ u_{k-1} \end{bmatrix} & \text{if } \tau_k < \tau^o, \\ -L(\tau^o) \begin{bmatrix} \hat{x}_{k|k-1} \\ u_{k-1} \end{bmatrix} & \text{if } \tau_k \geq \tau^o. \end{cases} \quad (7.36)$$

If we get a timeout we assume that the measurement will arrive later than the sampling interval. When it arrives $\hat{x}_{k+1|k}$ is computed for use in the next control calculation. Either $\hat{x}_{k+1|k}$ will be used directly as a state estimate, or if a new measurement arrives before the timeout it will be used together with the estimator (5.26). This means that no measurements are thrown away. A generalization to the case with lost samples, *vacant sampling*, is discussed in Section 7.6.

Analysis

In the previous section we formulated a suboptimal controller design. To help to choose the timeout in the design procedure we will analyze the resulting closed-loop system for different choices of the timeout, τ^o . The system using timeout can be written as a Markov chain model with two states, see Figure 7.5. The first state is modeling $\tau_k \leq \tau^o$, and the second

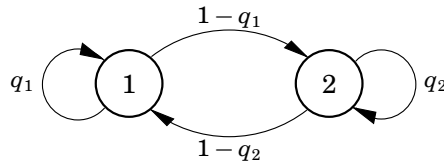


Figure 7.5 Markov chain with two states. State 1 is modeling normal operation, and state 2 is modeling timeout.

state is modeling timeout. The Markov chain has the transition matrix

$$Q = \begin{bmatrix} 1 - P_{to} & P_{to} \\ 1 - P_{to} & P_{to} \end{bmatrix}, \quad (7.37)$$

where

$$P_{to} = P(\tau_k > \tau^o) = \int_{\tau^o}^h f_{\tau}(s) ds \quad (7.38)$$

is the timeout probability. The probability distribution functions for τ_k^o associated with the states in the Markov chain are

$$f_1(t) = \frac{1}{1 - P_{to}} \Psi(0, \tau^o, t) f_{\tau}(t) \quad (7.39)$$

$$f_2(t) = \delta(t - \tau^o), \quad (7.40)$$

for the Markov states, respectively. The closed-loop system using the described controller and timing setup can be written on the form

$$z_{k+1} = \Phi(\tau_k^o, i) z_k + \Gamma(\tau_k^o, i) e_k, \quad (7.41)$$

where

$$z_k = \begin{bmatrix} x_k \\ u_{k-1} \\ \hat{x}_{k|k-1} \end{bmatrix} \quad (7.42)$$

$$e_k = \begin{bmatrix} v_k \\ w_k \end{bmatrix} \quad (7.43)$$

$$\Phi(\tau_k^o, 1) = \Phi_{t_o} + \begin{bmatrix} \Gamma_0 \\ 0 \\ \Gamma_0 \end{bmatrix} L_1 \bar{K} C [-I \ 0 \ I] \quad (7.44)$$

$$\Gamma(\tau_k^o, 1) = \Gamma_{t_o} + \begin{bmatrix} \Gamma_0 \\ 0 \\ \Gamma_0 \end{bmatrix} L_1 \bar{K} [0 \ -I] \quad (7.45)$$

$$\Phi(\tau_k^o, 2) = \Phi_{t_o} \quad (7.46)$$

$$\Gamma(\tau_k^o, 2) = \Gamma_{t_o} \quad (7.47)$$

$$\Phi_{t_o} = \begin{bmatrix} \Phi & \Gamma_1 - \Gamma_0 L_2 & -\Gamma_0 L_1 \\ 0 & -L_2 & -L_1 \\ KC & \Gamma_1 - \Gamma_0 L_2 & \Phi - \Gamma_0 L_1 - KC \end{bmatrix} \quad (7.48)$$

$$\Gamma_{t_o} = \begin{bmatrix} \Gamma_v & 0 \\ 0 & 0 \\ 0 & K \end{bmatrix}, \quad (7.49)$$

where the shortened notation

$$\begin{aligned} \Gamma_0 &= \Gamma_0(\tau_k^o) \\ \Gamma_1 &= \Gamma_1(\tau_k^o) \\ L_1 &= L_{:,1:n}(\tau_k^o) \\ L_2 &= L_{:,n+1:n+m}(\tau_k^o), \end{aligned}$$

has been used. The notation $L_{:,a:b}$ is used to indicate the sub-matrix of L having the columns a to b . This system can be analyzed with the method of Theorem 6.1. The analysis method gives tools for stability analysis, covariance evaluation, and quadratic cost function evaluation. In the controller design this analysis has to be done for a number of timeouts to find the optimal timeout. The design technique and properties of the controller will be illustrated with a simple example.

Chapter 7. Special Topics

EXAMPLE 7.1—DOYLE-STEIN WITH TIMEOUT

Consider the following plant, both plant and design specifications are taken from Doyle and Stein (1979),

$$\begin{aligned}\frac{dx}{dt} &= \begin{bmatrix} 0 & 1 \\ -3 & -4 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 35 \\ -61 \end{bmatrix} \xi \\ y &= [2 \quad 1] x + D_w \eta,\end{aligned}\tag{7.50}$$

where $\xi(t)$ and $\eta(t)$ have mean zero and unit incremental variance. The control objective is to minimize the cost function

$$J = \mathbf{E} \lim_{T \rightarrow \infty} \frac{1}{T} \int_0^T (x^T H^T H x + u^2) dt,$$

where $H = 4\sqrt{5} [\sqrt{35} \quad 1]$. The sampling period for the controller is chosen as $h = 0.05$. This is in accordance with the rule of thumb given in Åström and Wittenmark (1997). The time delay from sensor to controller, τ_k , is assumed to be uniformly distributed on the interval $[0, h]$.

The achieved cost will be evaluated for different values of the timeout, τ^o , in the interval $[0, h]$. The controller design and analysis is done with the previous described method. In Figure 7.6 the cost is plotted against the timeout. With $\tau^o = h$ the design corresponds to a design without a timeout. The case with $\tau^o = h$ and $D_w = 1$ is the case that was compared to other design methods in Section 5.5. For the cases of low measurement noise, small D_w , the cost function has its minimum at $\tau^o = h$, so no timeout is needed. For cases with more measurement noise, $D_w = \{3, 4\}$, controller performance is increased using a timeout. The intuition is that we do not gain anything by waiting for a noisy measurement, it is better to do control based on prediction. \square

The example demonstrates that performance in some situations can be increased by use of timeout. It is, however, not known if the controller (7.36) with timeout is optimal in any sense. The described control methodology can be extended to include a random delay from controller to actuator.

7.6 Timeout and Vacant Sampling

In the previous section we assumed that the measurement arrives at the controller node even if we get a timeout. If we get a timeout the measurement is used for state estimation at the next control instant. This section discusses what can be done if a measurement does not always arrive. This problem is known as *vacant sampling*. This can also be used to handle delay distributions with very long tails, in which case long delays will be considered as a vacant sample.

7.6 Timeout and Vacant Sampling

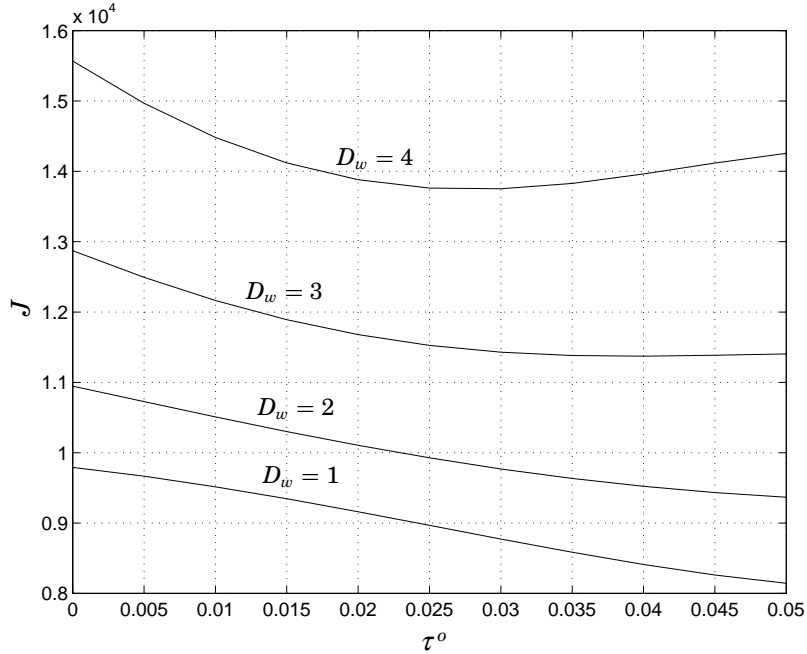


Figure 7.6 Cost vs timeout for $D_w = \{1, 2, 3, 4\}$. Notice that for $D_w = \{3, 4\}$ the cost function has a minimum. In these cases the optimal timeout is less than h .

Design

The state feedback part of the control law can be done as in Section 7.5. The state estimation part has to be revised. The vacant sample can be seen as measurement with infinite variance, $R_2 = \infty$, which makes the implementation of the Kalman filter feasible. This will also require a third state in the Markov chain to model a vacant sample. The states in the Markov chain will model:

- normal operation,
- timeout, but measurement can be used next sample, and
- vacant sample, measurement lost.

A difference to the case without vacant sampling is that the Kalman filter will be time varying. We need to keep track of the state estimation error covariance, P_k , which will be a state in the controller.

Analysis

The varying gains in the Kalman filter, K_k and \bar{K}_k , make the close loop system nonlinear in the state P_k . This makes it impossible to use an analysis method analog to the one used in Section 7.5. In this case methods for analysis, except simulation, are unknown to the author. Possible analysis methods can perhaps be found in the field of adaptive control. Nevertheless, the described controller design method seems reasonable. The described controller is a small extension of the controller in Section 7.5, which was shown to perform well in Example 7.1.

7.7 Asynchronous Loops

In this section we study another source of random time variations in real-time control systems. A problem occurring in commercial control systems is time variations due to asynchronous loops. As an example we will study the control system in Figure 7.7. The control system has two units, an I/O-

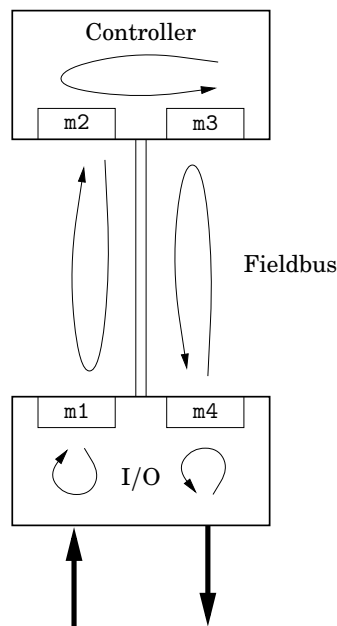


Figure 7.7 Control system with asynchronous loops.

module and a controller, which communicate using a fieldbus. In the I/O-module the process output is AD-converted and written in the memory

7.7 Asynchronous Loops

cell m1 every 250 ms. The I/O-module also reads the memory cell m4 and DA-converts the value to the process input. This is also done every 250 ms. The fieldbus mirrors the memory cell m1 in the I/O-module to the memory cell m2 in the controller module. The controller memory cell m3 is mirrored to the I/O-module memory cell m4. The fieldbus mirroring is done every 256 ms, and is asynchronous to the activities in the controller and I/O-module. The controller is executed with a period of h , it reads the process output from memory cell m2 and writes the new control signal to memory cell m3.

This type of setup with asynchronous periodic activities is common in commercial control systems with distributed I/O and control. As no global clock is used in the system the periodic activities will drift relative to each other. In some cases loops can also be closed on an even higher level, where we would have even more asynchronous loops in series. In this case the higher level could, for instance, be a plant-wide Ethernet with limited real-time capabilities.

The setup above gives a system with varying delay from sensor to actuator. We will look at the variations in the *control delay*, the time from when a measurement signal is sampled to when it is used in the actuator. In Figure 7.8 a typical sequence of control delays is plotted. The delay sequence was captured by simulating the system in Figure 7.7. The signal mirroring over the fieldbus was assumed to be instant, i.e., the fieldbus delays were neglected. If activities on the bus are not synchronized, which can be done using a schedule, we will also get a delay variation from transfers on the fieldbus. In the simulated sequence the sampling interval was $h = 1.1\text{s}$. The pattern of delays will vary over time due to the varying synchronization of the loops. The loop skew will be varying over time as no global clock is used in the system. The local clocks will also have differences in speed, which will give the delay sequence a very long period. Even if each loop have an updating interval of 250ms, 256ms, or 1.1s, the total control delay can be slightly greater than 2.1s, the sum of the periods, if the loops have the most unfavorable synchronization.

The described time variations can lead to decreased control performance if the size and variation of the delay variations is not negligible compared to the time constants in the loop. More work can certainly be done on modeling of the delays originating from asynchronous loops. An accurate model of this timing behavior will also make it possible to do better controllers for these systems. The delays can be modeled using the network models of Chapter 3. The methods of Chapter 5–6 can then be used for analysis and controller design.

Chapter 7. Special Topics

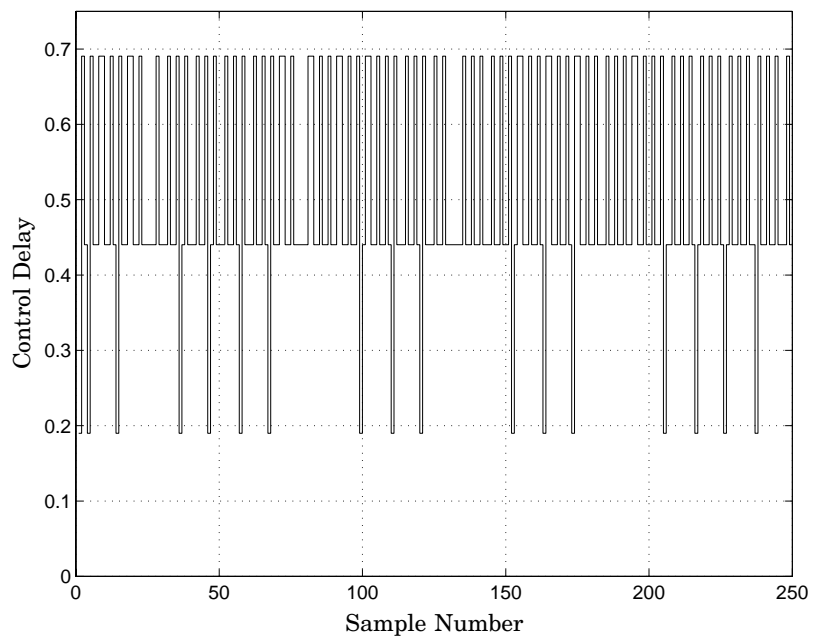


Figure 7.8 A typical sequence of control delays in the system with asynchronous loops.

8

Conclusions

This thesis has presented a control problem that arises when control loops are closed over a communication network, as is being more and more common. The communication network introduces time delays in the control loop. The network induced time delays can have effect on system stability and performance.

Modeling Two delay models have been studied:

- Random delays that are independent from transfer to transfer,
- Random delays with probability distribution functions governed by an underlying Markov chain.

The model including a Markov chain has the attractive property of making modeling of trends in the network delays possible. These trends can, for instance, arise due to varying network load. The delay models were verified by delay measurement experiments on two commercially used networks, CAN (Controller Area Network) and Ethernet. An off-line clock synchronization algorithm for estimation of clock-offset and clock-drift was derived. The clock synchronization algorithm can be reformulated as an on-line synchronization algorithm. The experiments show that network delays are varying if the network is loaded. The experimental measurements show that both models can be useful in modeling of network delays.

Independent Delays Using a linear controller it was shown that the closed loop can be written as

$$z_{k+1} = \Phi(\tau_k)z_k + \Gamma(\tau_k)e_k, \quad (8.1)$$

where the stochastic properties of τ_k depend on the network model. A Lyapunov recursion was found for evaluation of signal covariances in the closed loop system. A condition for mean square stability followed directly

Chapter 8. Conclusions

from the Lyapunov recursion. The LQG-problem was studied. A separation theorem was shown for the LQG-optimal controller. This means that the optimal controller is the combination of the optimal state feedback, and the optimal state estimator. The optimal controller uses knowledge of old time delays. These can be calculated using timestamps on messages sent in the network.

Markov Delays When the probability distributions of the delays are generated from a Markov chain the closed loop system can be written as

$$z_{k+1} = \Phi(\tau_k, r_k)z_k + \Gamma(\tau_k, r_k)e_k, \quad (8.2)$$

where τ_k is the network delay and r_k is the state of the Markov chain. Using a lattice structure for the covariance, a Lyapunov recursion for evaluation of covariances of signal in the system was found. A stability criterion for mean square stability followed directly from the Lyapunov recursion. The LQG-problem was solved by splitting it into a state feedback problem and an optimal state estimation problem. The optimality of the combination of state feedback and optimal state estimator was proven in a separation theorem. The optimal controller uses knowledge of old time delays together with the knowledge of the state of the Markov chain.

If the state of the Markov chain is unknown it has to be estimated. The combination of LQG-controller and Markov state estimator will not be the LQG-optimal controller.

Special Topics Several possible extensions and generalizations to the basic problem have been presented. The network model was extended to a model that makes one transition when sending measurement, and one transition when sending control signal. This closer models the real network, which can change state at any time. The LQG-controller was also generalized to the case with random sampling interval. This behavior was noted for one of the experimental setups. The controller for varying sample interval will be more complicated than the controller for equidistant sampling. The reason for this is that in this case the estimation error recursion needs to be updated every sample. The LQG-controller was generalized to the case with multiple sensor and actuator nodes. This controller calculates a new control signal when measurements have arrived from all sensor nodes. The controller uses timestamps for calculation of old time delays. If the delays have long “tails” a controller with timeout can be beneficial. A suboptimal controller using timeout was proposed. The proposed timeout controller is shown to outperform other controllers if we have much measurement noise. A closely related problem is vacant sampling, which is detected with a timeout. The timeout controller

is useful also in cases with vacant samples. Finally, time delay variations originating from use of asynchronous loops was studied. Asynchronous loops inevitable introduce varying time delays. This was exemplified with a simulation of the timing in a typical industrial controller.

Main Contributions

The main contributions of this thesis are

- The problem of control using distributed real-time control systems has been structured and studied. Earlier approaches exist, for instance, Luck and Ray (1990), Liou and Ray (1991), Ray (1994), and Krtolica *et al.* (1994), but this thesis gives methods that allow comparison with earlier results within a common structure.
- Using the methods developed in the thesis, distributed real-time control systems can be analyzed and controllers can be designed taking the timing behavior into account. The results are nice generalizations of well known theory for sampled-data control. Engineers with background in sampled-data control will recognize the similarities with the case without delays.
- The importance of timestamps are pointed out. This allows an approach where the history of the system is considered known.
- The problem with vacant samples has been studied. It was also shown that a distributed real-time control system can gain in performance by using a timeout.

Open Problems

There are several interesting problems in the area of distributed real-time control systems still to be solved. Some of the problems not treated in this thesis are:

- Throughout the thesis, except for the controller with timeout, we have assumed that the delay variation is less than a sampling interval. How can controllers be designed if the delay variations can be longer than a sampling interval? One motivation for keeping the delay less than a sampling interval is that if we have a fieldbus, only one message can be on the network at each time. If we have a longer delay than a sampling interval a measurement signal y_k would be waiting for the sending of y_{k-1} . In this case maybe we should send y_k instead of y_{k-1} , and consider y_{k-1} as a vacant sample. However, in other network types we can think of messages arriving in another order than the order of sending.

Chapter 8. Conclusions

- The properties of the controller with timeout is not completely studied. For instance, when do we gain by using a timeout?
- Special models of delay variations in asynchronous loops need to be developed. One possibility could be to use a Markov model. Using better models better controllers can be designed for these systems.
- The mathematical treatment of when the Riccati-iteration in the LQG-design will converge is not studied in this thesis. As in standard LQG-theory this would probably lead to conditions on both the cost function, and on the controlled process.
- LMI-formulations have been found for the standard LQG output control problem. For the case with jump linear system an LQG-formulation has been found for the state feedback problem, see Rami and Ghaoui (1996). Can LMI-formulations be found for the Riccati equations in this thesis?
- The hidden Markov case, where the state of the Markov chain is unknown, is not treated in the thesis. A method for prediction of controller performance when using an estimated Markov state would be very useful. This problem is very hard. An interesting study would be to try theory from adaptive control.

9

References

- ANDERSON, B. and J. MOORE (1979): *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, N.J.
- ÅSTRÖM, K. J. (1970): *Introduction to Stochastic Control Theory*. Academic Press, New York. Translated into Russian, Japanese and Chinese.
- ÅSTRÖM, K. J. and B. WITTENMARK (1997): *Computer-Controlled Systems*, third edition. Prentice Hall.
- BERMAN, A. and R. J. PLEMMONS (1969): *Theory of Matrices*. Academic Press, New York.
- BLAIR, W. P. and D. D. SWORDER (1975): “Feedback control of a class of linear discrete systems with jump parameters and quadratic cost criteria.” *International Journal of Control*, **21:5**, pp. 833–841.
- CHAN, H. and Ü. ÖZGÜNER (1995): “Closed-loop control of systems over a communications network with queues.” *International Journal of Control*, **62:3**, pp. 493–510.
- CHEN, H.-F., P. R. KUMAR, and J. H. VAN SCHUPPEN (1989): “On Kalman filtering for conditionally Gaussian systems with random matrices.” *Systems & Control Letters*, pp. 397–404.
- CHRISTIAN, F. and C. FETZER (1994): “Probabilistic internal clock synchronization.” In *Proceedings of the Thirteenth Symposium on Reliable Distributed Systems*.
- CHUNG, K. L. (1974): *A Course in Probability Theory*. Academic Press, New York.
- CiA (1994): “CiA draft standard 102 version 2.0.” <http://can-cia.de/>.
- DE SOUZA, C. E. and M. D. FRAGOSO (1993): “ H_∞ control for linear systems with Markovian jumping parameters.” *Control-Theory and Advanced Technology (C-TAT)*, **9:2**, pp. 457–466.

Chapter 9. References

- DOYLE, J. C. and G. STEIN (1979): "Robustness with observers." *IEEE Transaction on Automatic Control*, **AC-24:4**, pp. 607–611.
- ELLIOT, J. E., L. AGGOUN, and J. B. MOORE (1995): *Hidden Markov Models, Estimation and Control*. Springer-Verlag.
- FENG, X., K. A. LOPARO, Y. JI, and H. J. CHIZECK (1992): "Stochastic stability properties of jump linear systems." *IEEE Transaction on Automatic Control*, **37:1**, pp. 38–52.
- FEUER, A. and G. GOODWIN (1996): *Sampling in Digital Signal Processing and Control*. Birkhäuser.
- FRAGOSO, M. D., J. B. R. DO VAL, and D. L. PINTO, JR. (1995): "Jump linear H_∞ control: the discrete-time case." *Control-Theory and Advanced Technology (C-TAT)*, **10:4**, pp. 1459–1474.
- GAJIC, Z. and M. T. J. QURESHI (1995): *Lyapunov Matrix Equation in System Stability and Control*. Academic Press.
- HALMOS, P. R. (1958): *Finite-Dimensional Vector Spaces*. D. Van Nostrand Company, Inc.
- IEEE (1985): "IEEE 802.3 Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method and physical layer specifications."
- INTEL (1995): *82527 Serial Communications Controller, Architectural Overview*.
- Ji, Y. and H. J. CHIZECK (1990): "Controllability, stabilizability, and continuous-time Markovian jump linear quadratic control." *IEEE Transactions on Automatic Control*, **35:7**, pp. 777–788.
- Ji, Y., H. J. CHIZECK, X. FENG, and K. A. LOPARO (1991): "Stability and control of discrete-time jump linear systems." *Control-Theory and Advanced Applications*, **7:2**, pp. 447–270.
- KALMAN, R. E. (1962): "Control of randomly varying linear dynamical systems." *Proceedings of Symposia in Applied Mathematics*, **13**, pp. 287–298.
- KALMAN, R. E. and J. E. BERTRAM (1959): "A unified approach to the theory of sampling systems." *Journal of the Franklin Institute*, **267:5**, pp. 405–436.
- KRASOVSKII, N. N. and E. A. LIDSKII (1961): "Analytic design of controllers in systems with random attributes, I, II, III." *Automation and Remote Control*, **22:9–11**, pp. 1021–1025, 1141–1146, 1289–1294.

- KRTOLICA, R., Ü. ÖZGÜNER, H. CHAN, H. GÖKTAS, J. WINKELMAN, and M. LIUBAKKA (1994): "Stability of linear feedback systems with random communication delays." *International Journal of Control*, **59:4**, pp. 925–953.
- LANCASTER, P. (1969): *Theory of Matrices*. Academic Press, New York.
- LIU, L.-W. and A. RAY (1991): "A stochastic regulator for integrated communication and control systems: Part I - Formulation of control law." *Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control*, **113**, pp. 604–611.
- LJUNG, L. and T. SÖDERSTRÖM (1983): *Theory and Practice of Recursive Identification*. MIT Press, Cambridge, Massachusetts.
- LUCK, R. and A. RAY (1990): "An observer-based compensator for distributed delays." *Automatica*, **26:5**, pp. 903–908.
- MILLS, D. (1991): "Internet time synchronization: the network time protocol." *IEEE Transactions on Communications*, **39:10**, pp. 1482–1493.
- MORGAN, B. J. T. (1984): *Elements of Simulation*. Chapman and Hall.
- MOTOROLA (1992): *MC68340 User's Manual*.
- NILSSON, J., B. BERNHARDSSON, and B. WITTENMARK (1998): "Stochastic analysis and control of real-time systems with random time delays." *Automatica*, **34:1**.
- OLSSON, G. and G. PIANI (1992): *Computer Systems for Automation and Control*. Prentice-Hall, Englewood Cliffs, New Jersey.
- RABINER, L. R. (1989): "A tutorial on hidden Markov models and selected applications in speech recognition." *Proceedings of the IEEE*, **77:2**, pp. 257–286.
- RAMI, A. R. and L. E. GHAOUI (1996): "LMI optimization for nonstandard Riccati equations arising in stochastic control." *IEEE Transactions on Automatic Control*, **41:11**, pp. 1666–1671.
- RAY, A. (1987): "Performance evaluation of medium access control protocols for distributed digital avionics." *Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control*, **109**, December, pp. 370–377.
- RAY, A. (1994): "Output feedback control under randomly varying distributed delays." *Journal of Guidance, Control, and Dynamics*, **17:4**, pp. 701–711.

Chapter 9. References

- SCHEDL, A. V. (1996): *Design and Simulation of Clock Synchronization in Distributed Systems*. PhD thesis, Technische Universität Wien, Institut für Technische Informatik.
- SWORDER, D. D. (1969): "Feedback control of a class of linear systems with jump parameters." *IEEE Transactions on Automatic Control*, **14:1**, pp. 9–14.
- TINDELL, K. and H. HANSSON (1995): "Real time systems and fixed priority scheduling." Technical Report. Department of Computer Systems, Uppsala University.
- TÖRNGREN, M. (1995): *Modelling and design of distributed real-time control applications*. PhD thesis, Royal Institute of Technology, KTH, Sweden.
- VAN OORSCHOT, J. (1993): *Measuring and Modeling Computer Networks*. PhD thesis, Delft University of Technology.
- WONHAM, W. M. (1971): "Random differential equations in control theory." *Probabilistic Methods in Applied Mathematics*, pp. 131–212.

A

Kronecker Products

This appendix contains the definition of *Kronecker product* and some results for calculation with Kronecker products. For a more thorough discussion, see Halmos (1958) and Lancaster (1969).

A.1 Definitions

Let $A \in \mathcal{R}^{m \times n}$ and $B \in \mathcal{R}^{p \times q}$. The *Kronecker product* $A \otimes B \in \mathcal{R}^{mp \times nq}$ is defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{bmatrix}, \quad (\text{A.1})$$

where a_{ij} are the elements of A . Let $X \in \mathcal{R}^{m \times n}$, with the structure

$$X = [X_1 \quad X_2 \quad \dots \quad X_n]. \quad (\text{A.2})$$

The vectorized form of X , $\text{vec}\{X\} \in \mathcal{R}^{mn \times 1}$, is defined by stacking the columns into a vector as

$$\text{vec}\{X\} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}. \quad (\text{A.3})$$

A.2 Basic Rules of Calculation

The Kronecker product fulfills the following rules of calculation:

$$\alpha A \otimes \beta B = \alpha\beta(A \otimes B), \quad \alpha, \beta \in \mathcal{R} \quad (\text{A.4})$$

$$(A + B) \otimes C = A \otimes C + B \otimes C \quad (\text{A.5})$$

$$A \otimes (B + C) = A \otimes B + A \otimes C \quad (\text{A.6})$$

$$A \otimes (B \otimes C) = (A \otimes B) \otimes C \quad (\text{A.7})$$

$$(A \otimes B)^T = A^T \otimes B^T \quad (\text{A.8})$$

$$(A \otimes B)(C \otimes D) = AC \otimes BD \quad (\text{A.9})$$

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}. \quad (\text{A.10})$$

The proofs follow directly from the definition of Kronecker products.

LEMMA A.1

$$\text{vec}\{AXB\} = (B^T \otimes A) \text{vec}\{X\}. \quad (\text{A.11})$$

□

For a proof see Lancaster (1969).

B

Some Results from Probability Theory

B.1 Markov Chains

This section presents some results on Markov chains. See Berman and Plemmons (1969), and Elliot *et al.* (1995) for a more thorough discussion on Markov chains.

The Markov process is characterized by having a state. This implies that if its state is given at a time t , its future evolution is independent of the history that gave the state at time t .

DEFINITION B.1—MARKOV PROCESS

A sequence of random variables $x(t)$ is said to be a *Markov process* or to possess the *Markov property* if the associated probability measure has the property

$$P(x(t_n + h) \mid x(t_1), x(t_2), \dots, x(t_n)) = P(x(t_n + h) \mid x(t_n)), \quad (\text{B.1})$$

where $t_i < t_n$ for $i = 1, \dots, n - 1$. □

DEFINITION B.2—MARKOV CHAIN

A finite *Markov chain* is a Markov process that takes values $\{r_k\}$ in a finite set $S = \{1, 2, \dots, s\}$, with transition probabilities

$$P(r_{k+1} = j \mid r_k = i) = q_{ij}. \quad (\text{B.2})$$

Chapter B. Some Results from Probability Theory

The transition probabilities, q_{ij} , fulfill $q_{ij} \geq 0$ for all $i, j \in \mathcal{S}$, and

$$\sum_{j=1}^s q_{ij} = 1. \quad (\text{B.3})$$

□

Introduce the Markov state probability distribution

$$\pi(k) = [\pi_1(k) \quad \pi_2(k) \quad \dots \quad \pi_s(k)], \quad (\text{B.4})$$

where $\pi_i(k)$ is the probability that the Markov chain state at time k is i . The probability distribution for r_k is given by

$$\pi(k+1) = \pi(k)Q \quad (\text{B.5})$$

$$\pi(0) = \pi^0, \quad (\text{B.6})$$

where π^0 is the distribution for r_0 .

A Markov chain is said to be regular if the transition matrix Q is a primitive matrix. A primitive matrix fulfills $Q^k \gg 0$ for a positive integer k , $A \gg B$ denotes that the matrix elements satisfies $a_{ij} > b_{ij}$. That a Markov chain is regular means that all states will be possible to reach in the future, there are no “dead ends” in the Markov chain.

If a Markov chain is primitive the stationary probability distribution $\pi^\infty = \lim_{k \rightarrow \infty} \pi(k)$ is given uniquely by

$$\pi^\infty = \pi^\infty Q, \quad (\text{B.7})$$

where π^∞ is a probability distribution.

B.2 Conditional Independence

Stochastic independence of two events X and Y is usually defined by

$$P(X \& Y) = P(X)P(Y), \quad (\text{B.8})$$

where $P(X)$ is the probability that event X occurs. A weaker condition on two events is *conditional independence*.

DEFINITION B.3—CONDITIONAL INDEPENDENCE

X and Y are said to be *conditional independent relative to Z* if

$$P(X \& Y | Z) = P(X | Z)P(Y | Z). \quad (\text{B.9})$$

□

The following theorem is often useful when conditional independence of two events are to be shown.

B.2 Conditional Independence

THEOREM B.1

The following three conditions for X and Y being conditional independent relative to Z are equivalent

1. $P(X \& Y \mid Z) = P(X \mid Z) P(Y \mid Z)$
2. $P(X \mid Y \& Z) = P(X \mid Z)$
3. $P(Y \mid X \& Z) = P(Y \mid Z)$

□

Proof We will use the following three equalities in the proof.

$$\begin{aligned} P(X \& Y \& Z) &= P(X \mid Y \& Z) P(Y \& Z) \\ &= P(X \mid Y \& Z) P(Y \mid Z) P(Z) \end{aligned} \quad (\text{B.10})$$

$$\begin{aligned} P(X \& Y \& Z) &= P(Y \mid X \& Z) P(X \& Z) \\ &= P(Y \mid X \& Z) P(X \mid Z) P(Z) \end{aligned} \quad (\text{B.11})$$

$$P(X \& Y \& Z) = P(X \& Y \mid Z) P(Z) \quad (\text{B.12})$$

1 \Rightarrow 2: Use Condition 1 in (B.12), and compare with (B.10).

2 \Rightarrow 3: Use Condition 2 in (B.11), and compare with (B.10).

3 \Rightarrow 1: Use Condition 3 in (B.11), and compare with (B.12).

□

Theorem B.1 can also be formulated using random variables and expected values.

THEOREM B.2

Let x and y be random variables which are conditionally independent relative to z . The following relations hold for all measurable functions $f(\cdot)$ and $g(\cdot)$.

1. $E(f(x)g(y) \mid z) = E(f(x) \mid z) E(g(y) \mid z)$
2. $E(f(x) \mid y, z) = E(f(x) \mid z)$
3. $E(f(y) \mid x, z) = E(f(y) \mid z)$

□

Proof This is proven by use of Theorem B.1 and the technique in Chapter 9.1 of Chung (1974). □

Using the 1-function we will formulate another formula for conditional independent variables. The 1-function is defined as:

Chapter B. Some Results from Probability Theory

DEFINITION B.4—1-FUNCTION

1_A is a function which is 1 on (the measurable event) A . \square

Easy calculations give the following rules for calculations using the 1-function.

$$E(1_A) = P(A) \quad (\text{B.13})$$

$$E(1_A | B) = P(A|B) \quad (\text{B.14})$$

$$E(f 1_A) = P(A) E(f|A) \quad (\text{B.15})$$

THEOREM B.3

If f and A are independent given B then

$$E(f 1_A) = \sum_B P(A | B) E(f 1_B). \quad (\text{B.16})$$

Proof

$$\begin{aligned} E(f 1_A) &= \sum_B P(B) E(f 1_A | B) \\ &= \sum_B E(1_A | B) P(B) E(f | B) = \sum_B P(A | B) E(f 1_B) \quad (\text{B.17}) \end{aligned}$$

\square



ISSN 0280-5316
ISRN LUTFD2/TFRT--1049--SE