Timing Problems in Real-Time Control Systems: Problem Formulation

Wittenmark, Björn; Törngren, Martin

1994

Document Version:
Publisher's PDF, also known as Version of record

ISSN 0280–5316
ISRN LUTFD2/TFRT--7518--SE

Timing Problems in Real-time Control Systems: Problem Formulation

Björn Wittenmark
Martin Törngren

Department of Automatic Control
Lund Institute of Technology
May 1994

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
INTERNAL REPORT

Date of issue
May 1994

Document Number
ISRN LUTFD2/TFRT--7518--SE

Author(s)
B. Wittenmark and M. Törngren

Supervisor

Sponsoring organisation
NUTEK

Title and subtitle
Timing Problems in Real-time Control Systems: Problem Formulation

Abstract

This report gives descriptions of different timing problems in connection with scheduling in and architectural properties of real-time systems. Many real-time systems are today built up in a decentralized way using several processors. The communication between the processors causes delays, which must be considered in the control algorithms and when scheduling the different tasks in the system. The execution of several tasks on one processor can cause unacceptable time-variations for control purposes. Transient errors can also affect control system performance. A literature survey is given and typical problems are formulated and discussed.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
20

Recipient's notes

Security classification

Timing Problems in Real-time Control Systems: Problem Formulation*

Björn Wittenmark† **Martin Törngren**‡

Abstract

This report gives descriptions of different timing problems in connection with scheduling in and architectural properties of real-time systems. Many real-time systems are today built up in a decentralized way using several processors. The communication between the processors causes delays, which must be considered in the control algorithms and when scheduling the different tasks in the system. The execution of several tasks on one processor can cause unacceptable time-variations for control purposes. Transient errors can also affect control system performance. A literature survey is given and typical problems are formulated and discussed.

1. Introduction

Many real-time systems are today implemented as multiprocessor or distributed computer systems and the different tasks are performed in different processors. Processors may be connected to different sensors and actuators and configured to cooperate in performing one or more feedback control functions. As a consequence timing problems can arise when implementing real-time control systems. For example, the network can cause time-varying delays in the communication between different parts of the system, and the multiplexing of several tasks by operating systems can cause unacceptable time-variations for control purposes.

The focus in this research is on timing problems for applications based on ordinary discrete-time control theory. In this theory time-invariance is usually assumed and it is common to use equidistant sampling. Time-invariance in

* This work is supported by NUTEK, Swedish National Board for Industrial and Technical Development, Project Dicosmos, 93-3485.

† Department of Automatic Control, Lund Institute of Technology, Box 118, S-221 00 Lund, Sweden

‡ DAMEK, Department of Machine Design, KTH, S-100 44 Stockholm, Sweden

turn requires that control delays are constant and known, that control and sampling actions take place a well defined instants in time, and that no transient errors take place. Alternatively, if time-variations exist, they must be negligible compared to the dynamics of the controlled process or included in the model of the process.

Traditionally, it has often been assumed that a computer system implementation of an automatic control system does not violate any assumptions made in design, simulation and verification. That is, while inaccuracies, disturbances etc. have been extensively treated at the process side, very little work has treated deficiencies in the computer system implementation of the automatic control system. Sometimes conservative scheduling is used by increasing sampling and communication periods in order to make time variations negligible.

A *control delay* corresponds to the time interval between related sampling and actuation actions. To relate this to real-time system terminology, the validity interval of sensor data - i.e. the allowed data age - corresponds to the maximum allowed control delay. In the simplest case, the control delay corresponds to the computational delay in the control algorithm. In a distributed control system, feedback control loops in many cases have to be closed over a communication network because sensing and actuating devices are physically separated. This type of systems are sometimes called Integrated Communication and Control Systems (ICCS), Ray (1989). In such systems a control delay belonging to a feedback control loop closed over a network will depend on a number of parameters including related computation delays, communication delays, synchronism between distributed control functions and the sampling (communication/computation) periods of the control functions, Törngren (1992). Thus, careful design is needed either by the control engineer to compensate for time-varying delays, or by the computer engineer to provide a computer system in which control delays are guaranteed to be constant. These two options illustrate the need for multidisciplinary interaction between control and computer designers.

Analysis of control systems with time-varying delays has been treated to some extent, see Hirai and Satoh (1980), Ray (1989), and Shin and Kim (1992). The applicability of the theory for performance and stability robustness to such problems have so far not been investigated. New results for robust control can be used for this kind of investigation.

Jitter can be defined as time-variations in actual start times of actions, as opposed to stipulated start times. Jitter depends on clock accuracy, scheduling algorithms, and computer hardware architecture. Without proper design large jitter effects will result, e.g. due to variations in execution times. In improved designs the worst case variation (jitter) in the sampling period can be predicted. However, in practice no system will have zero jitter. It is therefore important to derive robust controllers that can guarantee performance despite jitter. Similarly to the above problem there are two solutions here from the control or computer side, and interaction between the two disciplines is needed.

Transient errors, for instance, caused by loss or corruption of control related state variables during network communication, may temporarily violate the timing behavior of the computer. One way to recover from such errors is to detect loss of measurement and base the next control action on prediction of the output of the process. A more serious error, a *temporary blackout*, refers to a transient fault which causes the control system to behave in a non predicted way (e.g. no action at all or erroneous action) for a certain period of time. It

is more difficult to detect and handle such situations when the control actions do not reach the process. Especially in safety related systems it is important that these problems are considered and that adequate robustness is provided. The analysis of such phenomena has not until recently been treated, Ray and Halevi (1988), Shin and Kim (1992).

A related issue connected to scheduling is intentional changes in the sampling period. This might, for instance, be done due to computer failure in a multi-processor system. A graceful degradation may then be obtained by controlling less often. In such cases it could be necessary to redesign the controllers. In time critical systems there may not be time to do this. Simpler ways to do the recalculations can be used. See Wittenmark and Åström (1980), Åström and Wittenmark (1990), and Albertos and Salt (1990).

As the use of automatic control systems move towards more and more demanding applications with higher sampling rates, distributed computer implementations and higher demands on dependability, "second order effects" due to computer system deficiencies with respect to the timing behaviour become more and more apparent, and as a consequence can no longer be neglected.

There is a lack of design and analysis tools/methodology, and so far real-time control systems have been designed either by neglecting the effects of improper timing, by using conservative scheduling or by ad hoc methods, which are not guaranteed to work.

Goal of the project

This report is part of the NUTEK project 93-3485 *Distributed Control of Safety Critical Motion Systems (Dicosmos)*. This is a joint research project between the Department of Computer Technology at CTH, DAMEK at KTH and the Department of Automatic Control at LTH. The goal of the part of the project that is to be done at LTH is to investigate, theoretically and experimentally, the influence on performance of time scheduling and jitter. Further the goal is to use robust design methods to derive controllers that are robust against jitter in sampling period. Redesign methods when making intentional changes in the sampling period will be investigated. In this work we will cooperate with CTH and KTH in the formulation and investigation. The expertise at CTH will help in the modeling phase. The proposed research follows previous work at DAMEK on distributed control. The goal of the part of the project that is to be done at KTH is to investigate from the real-time point of view the influence on performance of time-varying delays and various compensation methods. There is a common interest between KTH and LTH in the described research work.

The project is addressing problems that are highly relevant for the theory and practice of sampled-data systems as well as for real-time systems. It is expected that the described work will provide a base of knowledge for establishing models, useful for multidisciplinary design of ICCS.

Outline of the report

Section 2 contains a description of a real-time system for robotics applications designed at the Department of Automatic Control at LTH. The intention is to show a specific system and to discuss possible causes of problems and their solutions. A more detailed description of important timing problems in real-time control systems is given in Section 3. Different problems that will be considered in the project are discussed and references are given. Section 4 contains

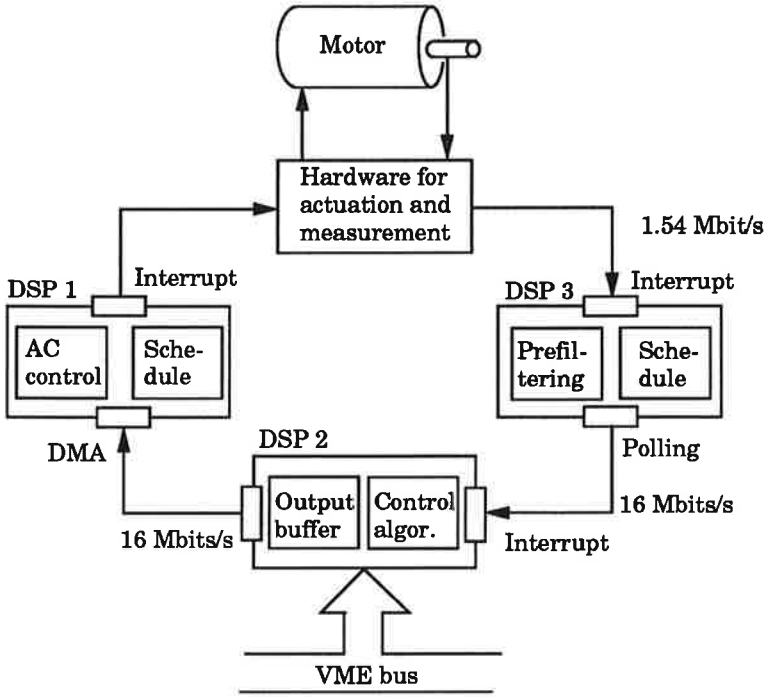

Figure 1. Schematic block diagram of a real-time system for robotics applications. Data between the processing DSP nodes and the actuation and measurement nodes is transferred via serial communication.

different models for systems with network induced delays. The different problems are briefly discussed and exemplified in Section 5. Conclusions are given in Section 6 and references in Section 7. The appendix gives a glossary of the concepts used in the project.

Acknowledgements

The work is made possible thanks to support by NUTEK, Swedish National Board for Industrial and Technical Development under contract 93-3485. We also want to thank Klas Nilsson for enlightning discussions about scheduling problems in real-time systems.

2. A real-time system for robotics applications

To illustrate different timing problems we will describe a specific system where detailed knowledge is available. Figure 1 shows a schematic block diagram of a real-time system for robotics applications. The system has been designed and built at the Department of Automatic Control, LTH. For further details we refer to Nilsson (1992). The intention has been to build an experimental system that is reliable and flexible. It is used for experiments with new controllers and design concepts for robot programming.

The controlled robot is an ABB Irb 2000 equipped with three phase AC-motors. The drive electronics of the original system is still used, but the control computers and the measurement system has been replaced by new hardware. This means that even the commutation control of the AC-motors is performed in the new controller. An outline of the DSP part of the system is shown in Figure 1. One digital signal processor, DSP1, schedules the control actions to

the actuator. It follows a schedule for the different control loops and uses data like set points and measurement values from the input buffer. Each output package from DSP1 contains current references for two phases (12 bits each) of the motor. In connection with the drive electronics for each motor there is hardware which unpacks the data and determines the desired currents for all three phases of the motor. The serial data also continues to the measurement nodes, where the address and control bits initiates a sampling of the position of the motor. The data in all the serial communication lines are transferred synchronously in 32 bit packages. Each package contains one sample including address and control bits. The sampling rate is 48 ksamples/s. The data is then transferred to DSP3, which collects the data using interrupts. The data here contains 24 bits for the measurement. DSP3 is using polling to check if the control processor DSP2 is ready to receive new data. If so data is transferred with a rate of 16 Mbits/s. There is also a time-marking of the data. This is used to indicate if the controller processor was late to receive the data or if data is missing. The control algorithm computes the control signal and delivers it to an output buffer. The content of the output buffer is transferred to the input buffer in DSP1 using Direct Memory Access (DMA) with a speed of 16 Mbits/s. Typical sampling rates are 2–4 kHz per control loop. The control processor is also connected to a VME bus for communication with a work-station for operator communication, display, controller design, and cross-compilation of new control algorithms.

The real-time system for the robot application is not a truly ICCS, since the communication is specially designed for the special application and it is not straight forward to incorporate new nodes in the system. Nevertheless the system can be used to illustrate many of the timing problems in ICCS.

In the robot system much effort has been put into minimizing delays in the communication. The DMA transfer of data from DSP2 to DSP1 is done to not slow down the controller processor with communication. The computing time in DSP2 is mainly devoted to the control algorithms and the operator communication. Immediately after sending data to the outbuffer, the controller processor can continue with the next control loop. The activation of the next control action at the motor is done according to the schedule in DSP1. The specially designed hardware for actuation and measurement at each motor makes it possible to immediately measure the new output and apply the control signal. Considering each individual unit the skew between sampling and actuation within the node is thus negligible in this system.

Despite the carefully designed system it may happen that data is delayed or lost. This can be detected in DSP3. The data arriving at DSP3 has an address part, which indicates the output signal associated with the data. This is compared with the schedule in DSP3 and loss of data can be detected. The two processors DSP1 and DSP3 must thus be synchronized. This is done using a common clock. This implies that the time-marking or synchronization is very important. In the transfer of data from DSP3 to DSP2 it is possible to indicate if data is delayed or missing. This information may then be used in the control algorithm. Modifications of the controller can thus be done since it is possible to detect delays or blackouts. For the control of the unit, however, the skew can be up to several samples due to the communication time, computation time, and the static schedule of the control. A typical minimum skew for robot control is half a sampling interval, but the effect of longer delays can easily be emulated.

From this brief description of a real-time system we can point at sev-

eral crucial timing problems in real-time systems. The problems will be more difficult in a truly integrated communication and control system. The main problem areas are:

- Synchronization of sampling and actuation
- Detection of delayed or missing data
- Surveillance and minimization of communication delay

Different ways to model and describe these problems are discussed in the next section.

3. Timing problems in ICCS

In the analysis and design of sampled-data control systems it is in general assumed that the controlled process and the computer are both time-invariant. For the computer control system it is usually assumed that:

- The control delay and skew is negligible or constant, i.e., related sampling and actuation actions are separated by a constant time-delay.
- Jitter, i.e. time-variations in scheduled sampling instants and control actions, is negligible.
- There are no transient errors causing information to be lost or control actions to not reach the process.

As discussed in the introductory section, time-varying control delays and jitter will exist in the system unless special care is devoted to the design of computer architectures and software, specialized for real-time control, Halang (1992).

To clarify the issues the three mentioned problems, their effects and their origin, are further discussed in the following.

Single processor implementation

Consider the case where a number of feedback control loops are implemented by a single processor. The operating system then needs to schedule: sampling actions, the control algorithms, and control actions in a periodic fashion. However, in the traditional approach based on preemptive prioritized scheduling and the above actions performed by software, the system will typically exhibit time-varying control delays as well as jitter caused by the scheduling policy and system load. Solutions can be provided, either from the computer or control side. The latter exemplified by means of dedicated hardware performing sampling and actuation or by use of more appropriate scheduling policies. The former, by use of control design which compensates for the timing problems. This approach, however, requires that the characteristics of time-variations are known, either prior to run-time or measurable so that compensation can be provided during run-time.

Distributed computer systems implementation

Timing problems become more apparent when control loops are closed over several processors and a communication network. This is due to distributed system characteristics including among other things true parallelism, internode synchronization, communication events, transient communication failures, and end-to-end delays depending on many parameters.

It is more difficult to guarantee constant control delays. This is because global scheduling considerations and synchronization is then required. If these

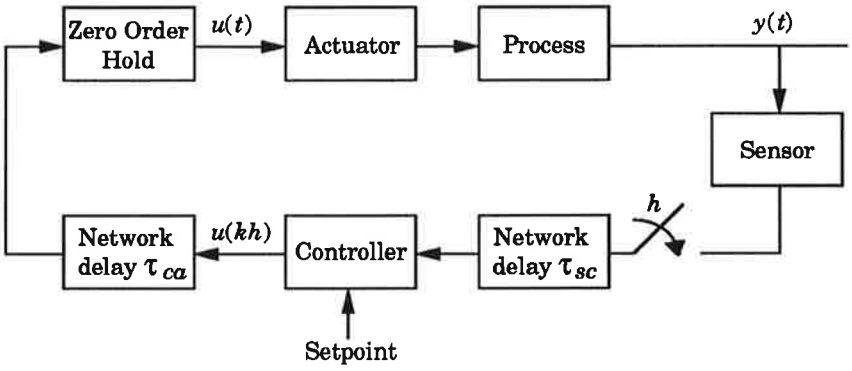

Figure 2. Digital control system with induced delays.

considerations are made, ordinary discrete-time control theory can in principle be directly applied. This is further elaborated in Törngren (1992). Such considerations constrain the implementation and as a result some flexibility is lost. On the other hand, without these considerations control delays will be time-varying. Control performance due to network induced time-varying delays in ICCS is discussed in Halevi and Ray (1988) and Ray (1989). Different models for varying delays in the control loop were used. A block diagram of a digital control system closed over a communication network is shown in Figure 2. The ICCS has three nodes, each dedicated to the sensor, controller, and actuator, respectively.

There are essentially three kinds of delays in the system:

- Communication delay between the sensor and the controller τ_{sc}
- Computational delay in the control algorithm τ_c
- Communication delay between the controller and the actuator τ_{ca}

The control delay for the control system equals the sum of these delays. Figure 3 shows a possible scenario for the timing in an ICSS. It is assumed that the sensor is sampling periodically and that there is a time-varying delay $\tau_{sc}(t)$ between the sensor and the controller. Further the control algorithm is scheduled periodically with a constant skew τ_s from the sampling. After the computation of the control signal it is sent to the actuator, which is receiving it after $\tau_{ca}(t)$ time units. The actuator is sending out the control signal to the process immediately.

If $\tau_{sc}(t) > \tau_s$ the data will not be received before it is needed, as shown with the third sample in Figure 3. This situation is referred to as vacant sampling, Halevi and Ray (1988). Vacant sampling is thus caused by a time-varying end-to-end communication delay, resulting in a corresponding time-varying control delay. It is very important that such delays are considered, as they otherwise may cause a deterioration of the closed loop performance.

Depending on the chosen solution in the real-time system different scenarios are possible. The simplest one used in the figure is that the previous data is used once again, since it is still in the memory location where the new sample should occur. Another possibility is used in the real-time system described in Section 2, where the hardware guarantees that actuation and sampling are synchronized and where the communication processor DSP3 is waiting until new data is arriving. After the data is received, or loss of data is detected, information is sent to the controller processor DSP2 together with a time-stamp showing how much the data is delayed. This information can be used to modify the control algorithm. For instance, the control calculation can

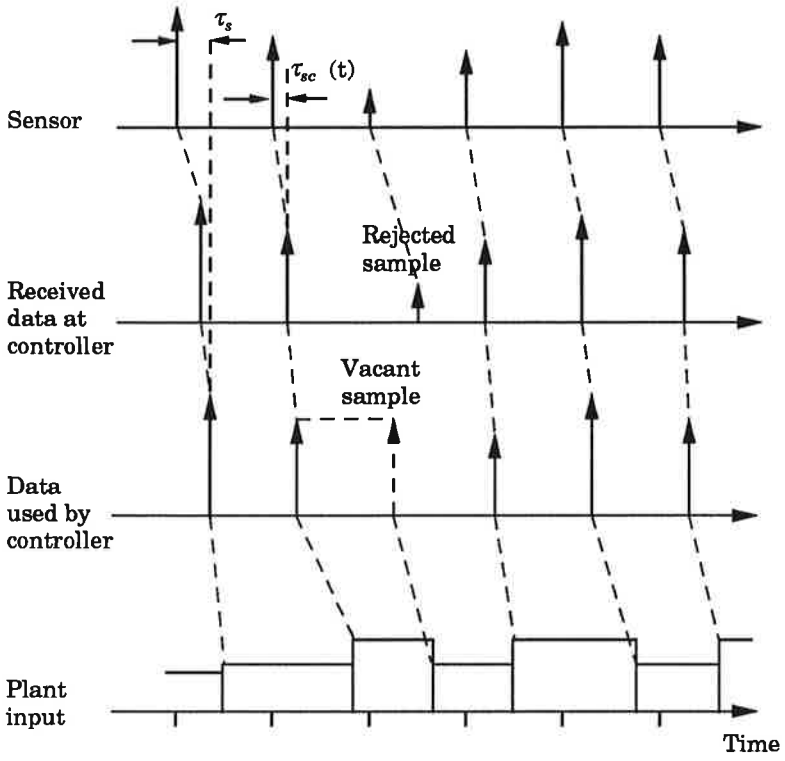

Figure 3. Timing between sampling, control signal calculation, and actuation.

be based on a predicted value of the output.

Constant vs time-varying delays

In the simplest case it can be assumed that the communication delay and computational delay are constant. To analyze the performance of the sampled-data system in this case the controller, process, and communication delays can be lumped provided they are constant. Using different design methods it is possible to take the delay into consideration when designing the controller, see for instance Åström and Wittenmark (1990). Most of the literature on time-delay systems is dealing with constant delays. Some results for systems with time- varying delays are found in Hirai and Satoh (1980), Ikeda and Ashida (1979), and Drouin *et al.* (1985). A review of control of delayed systems is found in Liou and Ray (1991a).

It is important to note that a system with a time-varying delay, not in a straightforward fashion can be designed using ordinary control theory. For instance, consider a system with a delay, $\tau(t)$. Suppose that the delay is time-varying in an unknown fashion but is bounded as follows: $\tau(t) < \tau_{max}$. If the system is designed for $\tau(t) = \tau_{max}$, i.e. assuming time-invariance, it may happen that the actual closed loop system performance becomes worse than expected, and even that instability occurs, Hirai and Satoh (1980).

A continuous-time system with time delay is an infinite dimensional system. The system will, however, become finite dimensional when sampled. The case with one internal delay is considered in Wittenmark (1985). A generalization to the case with several delays is found in Bernhardsson (1992).

In real-time systems it is likely that the time-delay can change with time. The variation may be described as a stochastic process. The sampled-data system can then be described by a finite-dimensional system with time-varying

or stochastic coefficients.

Jitter

The sampling interval h may vary both at the sensor and at the actuator. The reason might come from inappropriate scheduling, e.g. allowing higher priority interrupts; preemption and dispatching delays; or execution times being longer than expected. Typically, the jitter can vary at each sampling interval depending on the current system load. For a particular load, and if the system is reasonably predictable, it should be possible to determine the values that the sampling interval actually can take. E.g. for a smaller system with a few control tasks, a few values will be obtained depending on the execution scenario, scheduling policy and computer characteristics.

Solutions can, as described previously, be provided either from the computer or control side. The latter exemplified by means of dedicated hardware performing sampling and actuation. Alternatively known jitter can be compensated for, which is further discussed in the next section.

The change in the sampling interval can also be intentional. It would be valuable if control design determined and specified a range of allowable sampling intervals, where larger periods implicates deteriorated but still acceptable control performance. This would be very useful and open up a new degree of freedom for computer system design, resource management, and exception handling.

Transient errors

Transient errors, basically, can have the following three effects:

- Information loss
- Increased control delay
- Incorrect control actions

These types of problems are exceptions that must be handled by the logic of the control algorithm.

The former problem can, for instance, be due to a disturbance on the communication medium, causing information to be lost. The immediate effect is that of vacant sampling, i.e. an increase of control delay. However, the effect may be more serious since the corresponding data-item may be completely lost unless retransmission takes place. As discussed previously one solution is to provide error detection mechanisms and to base the next control action on prediction of the output of the process. This problem can be well formulated and solutions can be derived. For a particular system it would be useful to know the acceptable limit on the "amount of" lost information, such as sensor values, that can be handled by use of prediction. This can give a basis for the decision when to use more elaborate exception handling.

The result of other types of transient errors or a temporary blackout, may inhibit control actions to be performed for a certain period of time. An important question then is how long time the computer system can spend on system recovery until control actions must be continued if control system performance is to be guaranteed. I.e. it is of interest to derive "hard deadlines" similar to the work described by Shin and Kim (1992).

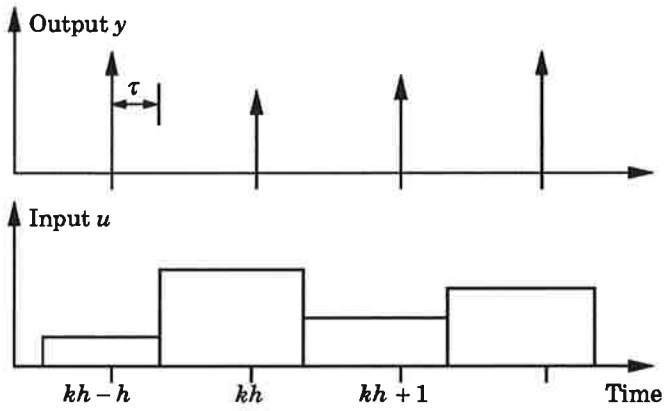

Figure 4. Illustration of the delay (skew) τ in a sampled-data system.

4. Models for systems with network induced delays

In this section we will describe different ways to model systems with time-delays, constant or time-varying.

Constant delay

The first case to be considered is when there is one constant time-delay in the system. I.e. that there is a delay τ_{ca} or τ_{sc} in Figure 2. Assume that the system can be described by the state-space model

$$\begin{aligned} \frac{dx(t)}{dt} &= Ax(t) + Bu(t-\tau) \\ y(t) &= Cx(t) \end{aligned} \tag{1}$$

The timing of the input and output is shown in Figure 4. Equation (1) represents an infinite dimensional continuous-time system. The zero-order-hold sampling of the system will now be calculated. It is initially assumed that τ is less than the sampling period h. Integrating over one sampling interval gives, see Åström and Wittenmark (1990),

$$x(kh+h) = e^{Ah}x(kh) + \int_{kh}^{kh+h} e^{A(kh+h-s')}Bu(s'-\tau)\,ds'$$

Since $u(t-\tau)$ is piecewise constant the integral can be divided into

$$\begin{aligned} \int_{kh}^{kh+h} \cdots &= \int_{kh}^{kh+\tau} e^{A(kh+h-s')}B\,ds'\,u(kh-h) + \int_{kh+\tau}^{kh+h} e^{A(kh+h-s')}B\,ds'\,u(kh) \\ &= \Gamma_1 u(kh-h) + \Gamma_0 u(kh) \end{aligned}$$

This gives the sampled-data system

$$x(kh+h) = \Phi(h)x(kh) + \Gamma_0(h,\tau)u(kh) + \Gamma_1(h,\tau)u(kh-h) \tag{2}$$

where

$$\begin{aligned} \Phi(h) &= e^{Ah} \\ \Gamma_0(h,\tau) &= \int_0^{h-\tau} e^{As}\,ds B \\ \Gamma_1(h,\tau) &= e^{A(h-\tau)} \int_0^{\tau} e^{As}\,ds B \end{aligned} \tag{3}$$

and the state-space model of (1) is

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

The sampled-data system is finite dimensional, since the input is piecewise constant and we only are looking at the sampling instances kh. Assuming that the input signal has dimension r then r new states $u(kh-h)$ have to be introduced.

If the time-delay is larger than h then we introduce

$$\tau = (d-1)h + \tau' \qquad 0 < \tau' \leq h$$

where d is an integer. The following state equation is now obtained

$$x(kh+h) = \Phi x(kh) + \Gamma_0 u(kh-dh+h) + \Gamma_1 u(kh-dh)$$
$$y(kh) = Cx(kh)$$

where Γ_0 and Γ_1 are given by (3) with τ replaced by τ'. Equation (3) shows how the system is changed when h or τ are changed.

From the analysis above it is found that there is no problem to model and compensate for if the system has one constant time-delay.

Now consider the case where there are two delays as in Figure 2. Assume that the delays are such that there will not be any vacant samples. If the control algorithm is independent of τ_{sc} and τ_{ca} the two delays can be lumped into one delay with $\tau = \tau_{sc} + \tau_{ca}$ and the process can be modeled as (2).

Time-varying delays

The case with time-varying delays will now be considered. It is assumed that the output of the process is sampled at equally spaced times and that we are interested in the relation between the inputs and outputs of the system at these sample points. The model (2) is still valid if the time-delays are varying with time. The system matrices will, however, be time-varying. Further the control signal will be constant over time intervals that will vary. This implies that we must assume that the total delay is varying less than one sample for (2) to be valid. Otherwise it will not be possible to integrate (1) over the different parts of the sampling interval.

In real-time systems the delays may vary stochastically and the system matrices will then have stochastic elements. The statistical properties of the elements may be difficult to describe due to the nonlinear function between the system matrices and the time-delay.

5. Discussion and examples

The models for systems with network induced delays are in general quite complex. In general, there will be delays that may vary with time in a complicated way. There is comparable little research done in the area of control of systems with varying delays. One approach is discussed in Liou and Ray (1991a) and Liou and Ray (1991b). There are, however, several other possible ways

to approach the problem. One way is to use robust design methods to investigate possible ways to design the controllers such that they are insensitive to variations in the control delays. Another very interesting area is to detect and compensate for vacant samples. This is closely connected to diagnosis of process control systems. The missing sample must be detected and the control algorithm has to be modified to improve the control performance.

We will end this section by exemplifying some of the problems that are of interest for scheduling in real-time systems.

Effect of jitter in sampling interval

The effect of jitter in a control system will be illustrated on a simple example. Assume that the process can be described by

$$G(s) = \frac{1}{s^2} \tag{4}$$

The process is a double integrator. A controller is designed using pole placement, see Åström and Wittenmark (1990). The controller has the structure

$$u(kh) + r_1 u(kh - h) = t_0 u_c(kh) - s_0 y(kh) - s_1 y(kh - h)$$

where u, y, and u_c are the control signal, process output, and reference signal, respectively.

The desired natural frequency is $\omega_0 = 1$ and the desired damping is $\zeta = 0.7$. Using standard rules of thumb for chosing the sampling interval gives a nominal sampling time $h_0 = 0.5$. The closed loop system is simulated with variations in the sampling interval. At each sampling instance the next sampling time is determined as

$$h(k) = h_0 + \Delta_h \cdot v(k) \tag{5}$$

where $v(k)$ is a random signal that is rectangular distributed over the interval $[-1, 1]$. Figure 5 shows the output for $\Delta_h = 0.05$, 0.1, 0.15, and 0.2. For each value of Δ_h 10 different simulations are done, where the sampling interval is changed at each sampling instance according to (5). Small values of the jitter will not influence the performance of the system very much, while the degradation is increasing with increasing Δ_h. The example shows that it important to reduce the jitter or to compensate for it.

Network induced delay

In the next example we again consider the double integrator and assume that there is a constant delay τ introduced by the network. The measurements are delayed τ time units before they are used in the controller. The controller is designed for the case $\tau = 0$. The result of simulations for different values of τ is shown in Figure 6. The delay will cause the system to become less stable and the overshoot increases.

Redesign when changing the sampling interval

In some situations it is necessary to increase the sampling period. One occasion can be when there is a degradation in the communication capacity of the network or if the computing load increases in a processor. To maintain the best possible control it may be necessary to redesign the controller to the new

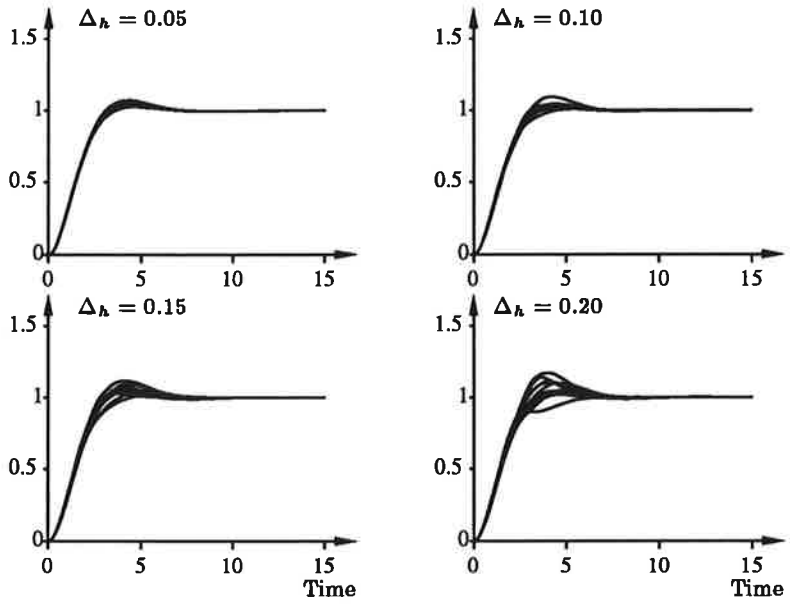

Figure 5. The output signal of the double integrator when the sampling period at each instant is a rectangular distributed random variable, when $\Delta_h = 0.05$, 0.1, 0.15, 0.2. The reference signal is a step.

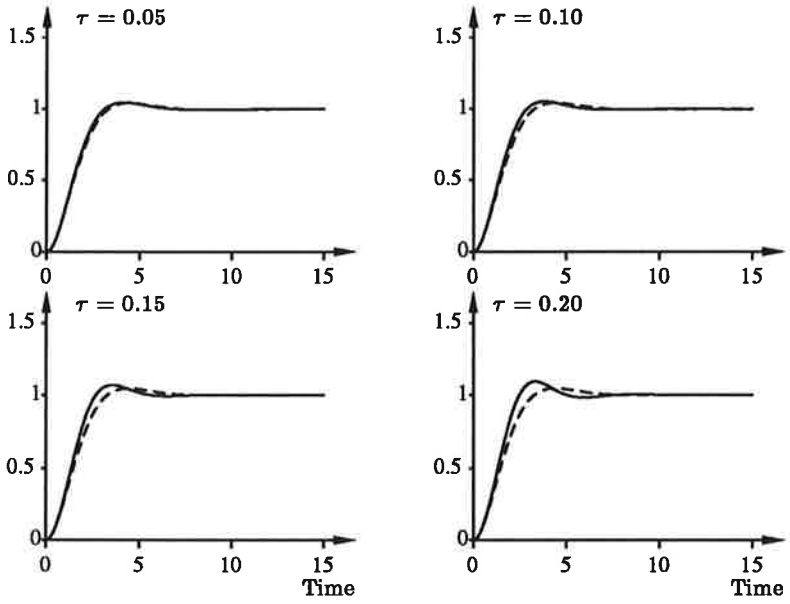

Figure 6. The output signal of the double integrator when there is a delay introduced by the network when $\tau = 0.05$, 0.1, 0.15, and 0.2. The reference signal is a step. The output when $\tau = 0$ is indicated by dashed lines.

sampling period. This can be done on-line or by using precomputed controller parameters. The redesign approach is also applicable to compensate for time-varying but known changes in the time-delay. We will illustrate this by a simple example. The process is a double integrator, i.e. it has the transfer function given by (4). The controller is designed to give a closed loop system with natural frequency $\omega_0 = 1$ and a damping of $\zeta = 0.7$. The nominal sampling period is chosen to $h = 0.5$. It is required that the closed loop system has an integrator. The performance of the closed loop system is shown in Figure 7

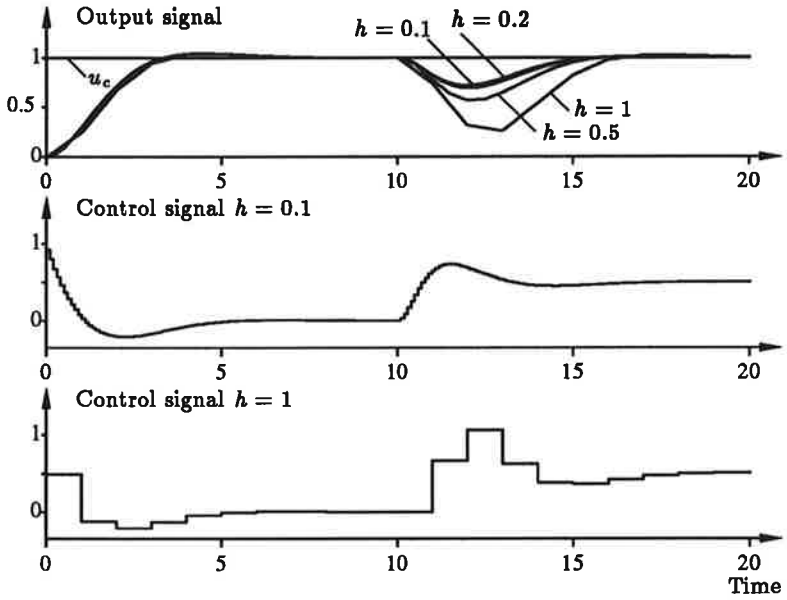

Figure 7. Control of the double integrator when the controller is designed for different sampling intervals. (a) Output signal when the reference signal is a step and when there is an input load disturbance. The control signal is shown for $h = 0.1$ and 1.

when $h = 0.1$, 0.2, 0.5, and 1. The control signal is also shown for $h = 0.1$ and 1. The reference signal is a step at $t = 0$ and there is an input load disturbance of -0.5 at $t = 10$. It is seen that the step response is not sensitive to the chosen sampling interval. The response to the input disturbance is more sensitive. When $h = 1$ the disturbance is not seen in the output until $t = 11$ and the output is deviating more when the sampling interval is increased. There is virtually no difference in the responses when h is 0.1 or 0.2.

The example shows that it can be of great advantage to redesign the controller when the sampling period is changed.

Simplified modification of controller parameters

Above it was shown how changes in the sampling interval could be compensated for by redesigning the controller. In this example we will illustrate a simplified way of changing the controller signal. This has been discussed, for instance, in Wittenmark and Åström (1980) and Albertos and Salt (1990).

A continuous time PID controller can be written as

$$U(s) = K\left(U_c(s) - Y(s) + \frac{1}{sT_i}\left(U_c(s) - Y(s)\right) - \frac{sT_d}{1 + sT_d/N}Y(s)\right)$$

where $U(s)$, $U_c(s)$, and $Y(s)$ are the Laplace transforms of the input, reference signal, and the output, respectively. Notice that the derivative term is only acting on the output. Further there is a filter on the derivative term, where N is in the order of 3–20. Discretizing this controller gives, see Åström and

Wittenmark (1990),

$$
\begin{aligned}
u(kh) &= P(kh) + I(kh) + D(kh) \\
P(kh) &= K\left(u_c(kh) - y(kh)\right) \\
I(kh+h) &= I(kh) + \frac{Kh}{T_i} e(kh) \\
D(kh) &= \frac{T_d}{T_d + Nh} D(kh-h) - \frac{KT_dN}{T_d + Nh}\left(y(kh) - y(kh-h)\right)
\end{aligned}
$$

From this equation it is easily seen how the coefficients of the controller should be changed when the sampling interval h is changed. When h is large the simple discretization will give poor performance of the closed loop system. It is then necessary to use a design method based on sampled-data theory as in the previous example. Also for more complex controllers it is not so easy to find how the regulator parameters should be scheduled with the sampling interval.

6. Summary

In this report we have discussed some of the timing problems in real-time control systems. The influence of the scheduling on the models is discussed together with different interesting problem formulations. The effect of the timing problems are exemplified through some simulated examples.

The future research will concentrate on analysis of the robustness properties with respect to time-delay variations and jitter in sampled-data systems. The following items will be of great interest:

- Studying ways of analyzing time-varying systems, in particular influences of jitter and time-varying delays
- Applicability of robustness theory
- Ways of deriving jitter specifications
- Ways of detecting and compensating for transient errors.

7. References

ALBERTOS, P. and J. SALT (1990): "Digital regulators redesign with irregular sampling." In *IFAC 11th World Congress*, volume IV, pp. 465–469, Tallin, Estonia.

ÅSTRÖM, K. J. and B. WITTENMARK (1990): *Computer Controlled Systems—Theory and Design*. Prentice-Hall, Englewood Cliffs, New Jersey, second edition.

BERNHARDSSON, B. (1992): *Topics in Digital and Robust Control of Linear Systems*. PhD thesis ISRN LUTFD2/TFRT--1039--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

DROUIN, M., H. ABOU-KANDIL, and P. BERTRAND (1985): "Feedback control for linear discrete-time systems with time delays." *Automatica*, **21**, pp. 323–327.

HALANG, W. A. (1992): *Contemporary Computers Considered Inappropriate for Real-Time Control*. Pergamon Press. Proc. IFAC Algorithms and Architectures for Real-Time Control.

HALEVI, Y. and A. RAY (1988): "Integrated communication and control systems: Part I–Analysis." *ASME Journal of Dynamic Systems, Measurements, and Control*, **110**, pp. 367–373.

HIRAI, K. and Y. SATOH (1980): "Stability of a system with variable time delay." *Trans. Automatic Control*, **AC-25**, pp. 552–554.

IKEDA, M. and T. ASHIDA (1979): "Stabilization of linear systems with time-varying delay." *IEEE Trans. Autom. Contr.*, **AC-24**, pp. 369–370.

LIOU, L.-W. and A. RAY (1991a): "A stochastic regulator for integrated communication and control systems: Part I–Formulation of control law." *ASME Journal of Dynamic Systems, Measurements, and Control*, **113**, pp. 604–611.

LIOU, L.-W. and A. RAY (1991b): "A stochastic regulator for integrated communication and control systems: Part II–Numerical analysis and simulation." *ASME Journal of Dynamic Systems, Measurements, and Control*, **113**, pp. 612–619.

NILSSON, K. (1992): *Application Oriented Programming and Control of Industrial Robots.* Lic Tech thesis ISRN LUTFD2/TFRT--3212--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

RAY, A. (1989): "Introduction to networking for integrated control systems." *IEEE Control Systems Magazine*, January, pp. 76–79.

RAY, A. and Y. HALEVI (1988): "Integrated communication and control systems: Part II–Design considerations." *ASME Journal of Dynamic Systems, Measurements, and Control*, **110**, pp. 374–381.

SHIN, K. G. and H. KIM (1992): "Hard deadlines in real-time systems." In *IFAC Symposium on Algorithms and Architectures for Real-Time Control*, pp. 9–14, Seoul, Korea.

TÖRNGREN, M. (1992): "Distributed control of mechanical systems." Licentiate Thesis KTH/MAE/LA–92/6–SE, Department of Machine Elements, KTH.

WITTENMARK, B. (1985): "Sampling of a system with a time-delay." *IEEE Trans. Autom. Contr.*, **AC-30**, pp. 507–510.

WITTENMARK, B. and K. J. ÅSTRÖM (1980): "Simple self-tuning controllers." In UNBEHAUEN, Ed., *Methods and Applications in Adaptive Control*, number 24 in Lecture Notes in Control and Information Sciences, pp. 21–29. Springer-Verlag, Berlin, FRG.

8. Appendix – Glossary

This glossary gives definitions of the terminology used in this project.

Asynchronous execution

With asynchronous execution the distance between related sampling instants can not be guaranteed (i.e. may drift) and depends on the characteristics of the local clocks.

Blackout

Refers to a transient fault which causes the control system to behave in a non predicted way (e.g. no action at all or erroneous action) for a certain period of time.

Computational delay

Term used in control theory to denote the delay in the control algorithm. E.g. the time it takes from the initiation of the control algorithm until the control signal can be sent to the process.

Control delay

The control delay refers the to time between related sampling and actuation actions. Note that it corresponds to the computational delay for a single rate control system. For a multirate system the control delay also depends on the different rates, processing and communication delays, see end-to-end delay. For feedback control it is highly desirable that control delays are constant.

Data delay

The data delay is the time between when a data-item is created, either by sampling or by a computation, and when it is available to control components in other nodes. The data delay thus includes the end-to-end delay.

Data age

The age of a data-item is defined as the time between the sampling instant when the data-item is created, either by sampling or by a computation, and the sampling instant when another component starts processing the data item, or should have unless sample rejection has occurred. Data-items are thus considered to be created and consumed at sampling instants.

End-to-end delay

This delay is the time interval counted from the instant when a process delivers a data item to the communication subsystem/operating system, to the instant when the data item is available for use by the receiver/ receiving processes. The end-to-end delays in general depend on the following four parameters: Synchronism between processes; Communication delays; Processing time; and Run-time system overhead.

Event controlled updating

The communication principle in which data is updated from producer to receiver(s) only when the data has changed, here referred to as an event.

Jitter

Jitter refers to time variations in actual start times of a process, as opposed to the stipulated release time. It is very important for sensor and actuation components that a maximum allowed jitter is guaranteed. In the periodic process model the allowed jitter can be indirectly specified by using the release time and the deadline. Jitter depends on clock accuracy, scheduling algorithms and computer architecture. Input and output jitter can be used to relate to the jitter of sampling and actuation processes respectively.

Multirate control system

A control system which consists of several control loops which may have different sampling frequencies. To facilitate modeling of such systems, it is common that periods are rational numbers (or even integer multiples of each other).

Periodic updating

The communication principle in which data is periodically communicated from producer to receiver(s), regardless of whether data has changed or not.

Sample rejection

Sample rejection refers to the case where more than one sample is obtained by a control component in between two consecutive executions of the component. The reason is typically that the sampling and control component are not synchronized and that the end-to-end delays are time-varying. Sample rejection may then occur more or less frequently.

Scheduling

Scheduling is concerned with the determination of when actions are to take place according to a specific scheduling policy. When one resource is shared by a number of activities the scheduler must determine how the sharing (multiplexing) is to be done. The policy specifies the aim of (e.g. meet deadlines or high average throughput) and rules for scheduling. For implementation of the policy a number of low level mechanisms may be needed. Further characteristics of scheduling policies/algorithms include when the scheduling is done, during run-time (dynamic) or pre run-time (static), where scheduling decisions are taken and whether only local or global actions are considered. Thus, for example, in global static scheduling, the actual scheduling takes place pre run-time and all relevant system resources are considered. The actual algorithms may be centralized or decentralized. A scheduling policy is only valid for one or more specific process models.

Skew (between sampling instants)

The skew is the time between sensor and controller sampling instants. In synchronous execution the skew is constant (possibly zero). In asynchronous execution the skew is a time-varying function. Referring to real-time systems terminology, skew refers to the time between well defined starting points in time of periodic processes.

Synchronization

Synchronous means simultaneous. A synchronization mechanism can therefore be interpreted as a mechanism which ensures that events occur simultaneously according to a common time base. This is contrasted with the use of the word in classical (non real-time) distributed systems where synchronism refers to logical event-ordering. e.g. exemplified by mutual exclusion, logical clocks, rotating privileges, etc. Synchronization is based on message exchange and/or a global clock.

Synchronous execution

A number of periodic processes execute synchronously if the distance in time between related sampling instants always is smaller than a known synchronization accuracy constant. The constant distance between related sampling instants is called skew.

Time-invariant control system

A system is time-invariant if the behaviour of the system relies only on constant parameters. Note that sampled systems are inherently time-dependent and

time-varying. By chosing a small enough sampling interval the systems can be considered to be time-invariant.

Vacant sample

Vacant sample refers to the case where no sample is obtained by a control algorithm in between two consecutive executions of the algorithm. The reason is typically that the sampling and control processes are not synchronized and that end-to-end delays are time-varying. This means that vacant sample may occur more or less frequently.