Tools for Autonomous Process Control

Wallén, Anders

2000

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

Tools for Autonomous Process Control

Tools for Autonomous Process Control

Anders Wallén

Lund 2000

Till Cecilia

Department of Automatic Control
Lund Institute of Technology
Box 118
SE-221 00 LUND
Sweden

ISSN 0280–5316
ISRN LUTFD2/TFRT--1058--SE

Printed in Sweden by Wallin & Dalholm Boktryckeri AB
Lund 2000

Contents

Acknowledgments

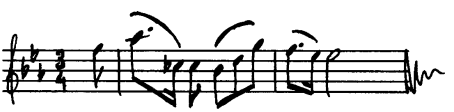
I would like to thank a number of people who have contributed to this thesis in different ways. First of all I would like to thank Professor *emeritus* Karl Johan Åström who has been my supervisor during my PhD studies. You have always been full of ideas, inspiration and enthusiasm. I am grateful for all your support throughout the years, especially when preparing this manuscript. Whereas Karl Johan has been the primary source of ideas, Tore Hägglund has acted as the perfect filter. You have always been an interested listener, always looking at the engineering aspects of my work, and always pointing out improvements and limitations. It has been a great pleasure working with both of you.

I have very much enjoyed my time at the department among all you nice people. In particular, I want to thank my fellow PhD students for many good times during collaborations, travels around the world, AW, and innebandy games. The fifth floor Ladies are always helpful and the greatest contributors to the friendly atmosphere at the department. Many thanks to Leif Andersson and Anders Blomdell for excellent computer support, and to Eva Dagnegård for your last minute efforts to make the thesis printable. Thanks also to Karl-Erik Årzén and Anders Robertsson for reading and giving feedback on parts of this thesis.

It has also been a pleasure to work with people outside the department. The collaborations with Börje Eriksson and Oskar Nordin at MoDo Paper have been very useful in this thesis. My stays with Venkat at Purdue and Ali Ipakchi at ABB Systems Control also gave a lot of new experiences.

During my PhD studies I have received financial support from the Swedish Research Council for Engineering Science (TFR), the Swedish National Board for Industrial and Technical Development (NUTEK), and the Swedish Foundation for Strategic Research (SSF) within the project CPDC.

Finally, but first in my mind, I thank my dear wife Cecilia for being the love of my life, and Viktor for bringing me the most joy. I thank you both for enduring these last months when I was mostly away from home. I shall repent. And remember:



Anders

1

Introduction

1.1 Motivation

One of the major driving forces in the development of new process control systems is to raise the level of automation in the plant. The purpose of a plant is to manufacture some product at a high production rate with a consistent quality, while minimizing the use of resources in terms of energy consumption, raw material and labour. With a higher automation level, it is possible to improve all these aspects, thus increasing the profit of the plant.

The phrase "autonomous process control" may indicate that the goal is to create a plant without human operators. However, humans are still, and will continue to be, needed on all levels of the system, from strategic planning and product development, to on-line operation and equipment maintenance. A more relevant interpretation of autonomous process control, as used in this thesis, is instead the ambition to increase the degree of autonomy in the plant and to provide functions that assist the humans in their tasks.

A fully autonomous controller should be able to govern the execution and performance of its own control functions, see Antsaklis *et al.* (1991). This should be done for long time periods, and with no or little human interaction. The requirements on system hardware and software will of course be immense if the controller is supposed to perform very complex tasks. On the other hand, by just requiring basic set point following and disturbance rejection, a traditional PI controller could function as an autonomous controller for plants with small parameter variations and slowly varying disturbances.

The desired degree of autonomy is of course different in different applications. Autonomous control systems discussed in the literature so far

mostly deals with autonomous vehicles on land or in space. They are often designed to operate in areas not easily accessible to humans, for example the surface of Mars, see Mishkin *et al.* (1998), or hazardous areas such as nuclear waste drums, see Byler *et al.* (1995). Experimental systems have also been running on the German autobahn, Dickmanns and Zapp (1987). Autonomous vehicles are typically given instructions or missions on a fairly high level, and they are supposed to take care of the low-level subtasks themselves. The benefits from this are obvious:

- The amount of communication required to accomplish complex tasks is kept to a minimum by using high-level instructions. This may be important for efficiency reasons.
- The vehicle acts much better as a replacement for a human if it knows how to perform low-level tasks, such as moving 1 meter forward, rotating 90 degrees, or locating neighboring objects. A movement is more likely to fail if all limbs or wheels of the vehicle must be coordinated from the base station each time.
- Robustness to unforeseen circumstances will increase if the vehicle is able to judge if an instruction is feasible or not. Some infeasible instructions might even be altered locally, *e.g.*, a scheme for obstacle avoidance, see Arkin (1998).
- If the communication link to the base station is broken it is desired that the vehicle is able to find out if there might be local causes and then take appropriate actions. This might include moving outside radio shadow, switching to backup hardware, or even perform hardware repair.

What is then a reasonable degree of autonomy? In other words, what operations should be automated and what operations should require human interaction? Naturally, there is no generic answer to these questions. There is a potential risk of setting the level of autonomy too high, since this might lead to destruction of the vehicle or some neighboring object. The risk comes from the fact that you can always make the system operate under conditions which it is not programmed to handle. When designing an autonomous system, safety must thus be the most important objective.

The primary focus in this thesis is on autonomous process control, though some of the ideas may be used in many other applications. The motivation for having autonomy in process control is quite different from vehicle and space applications. Autonomous space vehicles require a high degree of autonomy since they are intended to operate with little or no human interaction. In process control the motivation is instead to assist the operators and process engineers to govern very complex plants with

many control loops. The reason for introducing more autonomy is to make the operators work more efficiently and to extend the region where the plant can be operated satisfactorily without the operator's assistance. In other words, some of the tasks normally performed by an operator or a process engineer, should be transfered to the control system. The transfer should be transparent so that the operator can take over should he so desire. The system should also do new tasks that are currently not done by operators. It is also desirable that the control system is designed so that the operator can increase his knowledge of the system. This thesis will discuss what the new tasks could be, how they should be organized, performed and supervised.

Industrial needs

The main motivation for this thesis is to highlight the need for advanced features in industrial control systems, which are not available in today's systems. The goal is to provide solutions and suggestions for solving problems that do occur in practice.

Many of the ideas presented in the thesis have arisen through discussions with people facing control related problems in their particular plant. Some problems are surprisingly common, and yet there is often no support in the control systems to handle them. This thesis tries to help the operator dealing with the following, rather general questions:

- How to determine the basic properties of the process? Will these properties reduce the chance of achieving the desired control objectives?
- How to use limited process knowledge to select and tune a controller which meet the requirements on the loop?
- How to determine if the control loop performs as intended?
- What is the cause of drastically degraded control performance?

The problems above are today mostly solved by skilled operators or process engineers. The time, effort and skill needed for doing it could be drastically reduced if the control system provides adequate support. This will make it possible for an operator to handle more control loops more efficiently.

Some of the proposed features of an autonomous controller may be implemented in many existing industrial systems, often with large efforts, though. This thesis suggests that the architecture of future control systems must allow new algorithms to be implemented and tested without too much effort. The system should also provide a set of pre-defined solutions to standard tasks. These should be implemented and selected in a way that they may be used regardless of the current level of process knowledge.

1.2 Contribution of the Thesis

The motivation for higher automation levels has been discussed above. In this thesis we are presenting several tools that aid in the process of raising the degree of automation on the local control loop level.

Modeling. In order to do systematic analysis and control design, it is necessary to have a model of the process. A tool for assessment of process dynamics and non-linearities has been developed. It is based on transient response analysis of the process. The emphasis has been on creating a user interface for graphical manipulation of step responses in a natural way. This work has been published in

Wallén, A. (1999): "A tool for rapid system identification." In *Proceedings of the 1999 IEEE International Conference on Control Applications*. Kohala Coast, Hawaii.

Improved controller tuning. Existing auto-tuning methods for PI and PID controllers typically use a simple process model with a few parameters. More advanced design methods require more detailed models, which may be difficult to identify automatically. This thesis presents a method to generate a suitable excitation signal under relay feedback, and an identification scheme which gives a process model that can be used for advanced design of PI and PID controllers. It is shown that the PI design method can be applied iteratively in order to obtain good PID controllers.

Fast grade changes. The PI and PID controllers are mainly suited for regulation problems. When large set point changes are desired, process operators often use manual control until the process output is close to the desired value. This thesis presents an automated procedure which mimics the manual control actions done by the process operator. The method may be applied with limited process knowledge.

Architecture. An autonomous controller contains different types of algorithms. The execution of these algorithms must be organized in a well structured way. This thesis present an architecture based on a high-level graphical language for sequential control. Related publications:

Wallén, A. (1995): "Using Grafcet to structure control algorithms." In *Proceedings of The Third European Control Conference*. Rome, Italy.

Wallén, A. (1997): "Valve diagnostics and automatic tuning." In *Proceedings of the American Control Conference*. Albuquerque, New Mexico.

1.3 Thesis Outline

This thesis is organized as follows.

- Chapter 2 discusses autonomous control in general, with focus on process control. A list of desired features in an autonomous control system is presented.
- A tool for rapid system identification from step response data is presented in Chapter 3.
- Chapter 4 suggests a new auto-tuning procedure for PI and PID controllers. It utilizes relay feedback and spectral estimation to obtain a process model. The model is used for PI and PID design methods based on non-convex optimization.
- Chapter 5 presents a new simple algorithm and implementation structures for fast set point responses.
- Implementation aspects of an autonomous controller is discussed in Chapter 6. A prototype implementation is also presented.
- Chapter 7 summarizes the conclusions in the thesis and points out directions for future work.
- A list of references is given in Chapter 8.

2

Autonomous Process Control

This chapter describes the notion of autonomous process control. The background for autonomous control in general is described in Section 2.1. Section 2.2 contains a discussion on specific issues associated with process control, together with a description of the viewpoint of autonomy taken in this thesis. Desired features of an autonomous controller are reviewed in Sections 2.3 to 2.5.

2.1 Background

There is no commonly accepted formal definition of the term autonomous control. Instead, it is used with slightly different meanings by different authors. A synonym to the word autonomous which is often used in dictionaries is *self-governed*. The purpose of a control system is to solve specific tasks. It is then reasonable to say that a control system is autonomous if it is able to solve its tasks without external intervention.

If there are no uncertainties in the plant, and if the tasks are well specified, even a simple feed-forward algorithm can be fully autonomous. There are however always various types of uncertainties and faults in a plant, for example:

1. Disturbances from the environment, variations in raw material etc.
2. Vaguely specified control tasks. In its simplest form, this means indeterministic set points of a single loop controller.
3. Measurements with bias and noise.
4. Incomplete model of the plant.
5. Failing hardware, for example sensors or actuators.

6. Total power loss.

The last item is mainly included to illustrate that no control system is fully autonomous under *any* uncertainty or fault. It is thus necessary to define both the tasks and the admissible uncertainties when discussing autonomous control systems. In fact, the whole field of automatic control has always been occupied with finding methods to deal with uncertainties. Traditionally, most attention has been paid to items 1 to 4 above.

- Classical control theory mainly considered load disturbances, set point changes, and process uncertainties.
- The optimal control theory made it possible to formulate and solve problems with well-defined criteria.
- The stochastic control theory presented a framework for dealing with load disturbances and measurement noise in a systematic manner.
- Adaptive and robust control increased the tolerance to model imperfections.

Despite the large differences between these methodologies, they share a fundamental property: They all seek to define a controller based on algebraic, differential or difference equations. In terms of autonomy, they are all able to solve their tasks within a certain range of uncertainties. Robust control is the field where the focus on the uncertainties is particularly emphasized. You could thus argue that a controller designed to handle some uncertainty Δ using robust control methods is autonomous with respect to Δ. This is however not the traditional way of using the term autonomous control.

One drawback with the traditional control paradigms is that they can only deal with quantitative representation of control tasks, systems, signals, and uncertainties. In reality, the performance and behavior of a control system is often judged in words such as fast, oscillatory, sluggish, nice, bad, broken etc. These qualitative or *symbolic* measures are more difficult to incorporate into the frameworks of the traditional control methods. This is the motivation for various kinds of methods, often grouped into the term *intelligent control*. This group includes numerous techniques, where the following are most frequently used:

- Expert systems are rule based systems, where the rules may represent the combined knowledge of experienced operators, plant developers, chemists etc. The rules are combined through logical reasoning using an inference engine to produce conclusions of various kinds. The inputs and the outputs from an expert system may be any combination of numerical and symbolic values. See for example Åström and Årzén (1993).

- Fuzzy logic is also used for formulating and combining qualitative rules. Instead of using logical reasoning, a fuzzy inference engine combines the rules using some kind of interpolation. When needed, the quantitative variables in the physical world are interfaced to the fuzzy logic by the fuzzyfication and defuzzyfication procedures. See for example Passino and Yurkovich (1997).
- Neural networks use previously recorded plant data in order to tune weights in a network. The goal is to build a black box model which is able to reproduce a behavior that may be difficult to describe in mathematics. Both the inputs and the outputs are originally numerical values. However, by using enumeration and rounding, they may represent symbolic values as well. See for example Brown and Harris (1994).

A fundamental property that unifies these methods is that they are *not* based on equations for process models and control algorithms. Still, they are often used as alternatives to the traditional control paradigms listed above. They are all non-linear multi-dimensional mappings from the inputs to the outputs. Furthermore, since discrete decisions are natural elements in the methods, the resulting control systems will often be hybrid systems. Analysis and synthesis of hybrid systems is still only done for relatively small examples, Krogh and Chutinan (1999). All this make them more powerful than linear controllers, but unfortunately it is almost impossible to give any formal proof of stability, performance etc except for very simple cases. This actually makes it difficult to show that an "intelligent" control system is autonomous with respect to any reasonable uncertainties.

Despite the lack of formal capabilities, intelligent methods may be used for describing uncertainties on any of the levels 1–6 above. In fact, it is mostly much more natural to describe equipment faults using qualitative terms than trying to capture the faulty behavior in a mathematical model.

Even if the term autonomous control has not been formally defined, it is commonly used for control systems that try to adapt to new situations automatically. The distinction mostly made between autonomous control and traditional adaptive control is that the former should contain both algorithmic/numeric methods and decision-making/symbolic methods, see Antsaklis *et al.* (1991). This distinction is somewhat unclear, however, since any adaptive controller that is supposed to work in practice must have some kind of decision-making capabilities. For example, it should turn off adaptation when the signals do not provide sufficient excitation. An additional requirement that most authors put on an autonomous system is that it should involve decisions on different hierarchical levels. The hierarchies used in this work will be described in Section 2.2.

An application area where the degree of autonomy is perhaps higher than elsewhere is autonomous vehicles. These vehicles are typically programmed to perform well-defined tasks in an uncertain environment. A nice property of a mechanical system, such as a vehicle, is that it can be modeled accurately with a small set of equations. This is an enormous help when programming functions into the autonomous vehicle. Unfortunately, plants in the process industry do not share this nice property. Thus, a fully autonomous plant is not likely in the foreseeable future, if ever. For this to be true, the plant should even include automated maintenance, for example using autonomous vehicles. However, the degree of autonomy may be raised on all levels in a process control system. The view on autonomy used in this thesis will be discussed in Section 2.2.

2.2 A Process Control View

A fully autonomous process control system lies very far into the future. Human interaction is needed today on every level of abstraction. The fundamental reason for this is the difficulty to describe the uncertainties so well that they can be dealt with in a computer program without jeopardizing safety. If a hazardous situation occurs which was not foreseen when the control system was programmed, the human experience may be needed to avoid accidents. On the other hand, if the control system contains almost *all* the human knowledge of the plant, it is more likely to make the correct decisions in a stressful situation. It thus seems reasonable to believe that it will be possible in the future to replace human experience by having better models for the uncertainties. Once the information is available, the computer is superior in handling complexity. Creativity is a human quality which is more difficult to replace. No computer program is even close to the human brain when it comes to inventing solutions to new problems. Given a set of tools, a computer may suggest a better combination in order to solve a problem, but it will not be able to provide a new tool, see Boden (1998).

The goal of this thesis is not to create a fully autonomous control system, not even on a low level. Instead, the goal is to provide functions for an increased degree of autonomy. These functions should be parts of the control system, and to put these into a context, a discussion about the organization of a complex control system is required.

There are numerous suggestions on how to describe the functional structure of a complex control system. In fact, most authors use their own schematic of the control system. In Åström and Wittenmark (1997) the focus is on how people on different levels of the plant interact with the control system. Antsaklis *et al.* (1991) proposes an hierarchical architecture

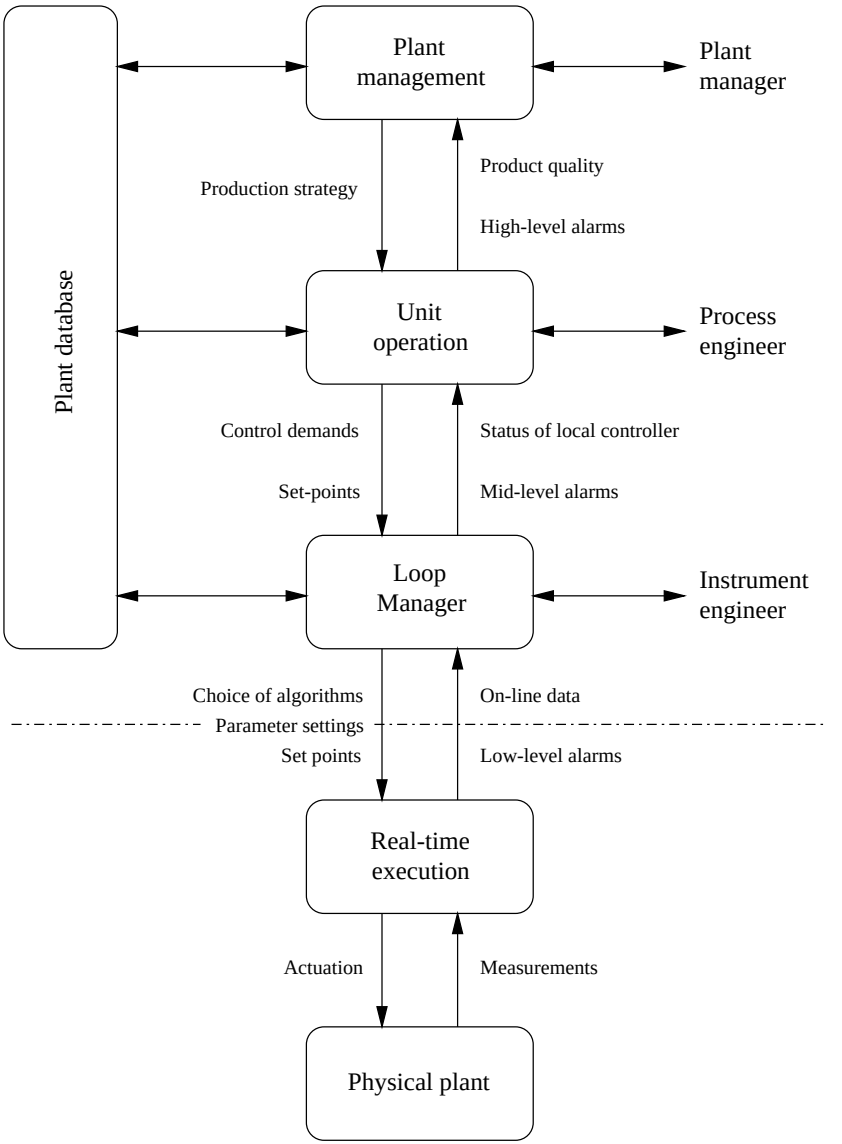


Figure 2.1 Functional architecture of a complex process control system. The texts at the vertical arrows give examples of information exchange between the layers. Traditional control systems only contain the parts below the horizontal line.

for autonomous control where the human interaction is restricted to occur at the top-most level. The methodology of fault tolerant control systems in Blanke *et al.* (1997) suggests that issues concerning fault detection and fault recovery should be considered already during plant design. During execution, a three-layer architecture for feedback control, fault diagnosis, and supervision is used.

There are also industrial standards that give guidelines to how a control system should be structured. For example, ISA (1995), includes a functional structure for a batch control system. It is similar to the functional model that will be used in this work, see Figure 2.1. The different layers correspond to subsystems working on different levels of abstraction. The descriptions next to the vertical arrows give examples of exchanged information between the different layers. The purpose of the different layers will now be explained.

1. ***Plant management.*** This is where the long-term production planning and scheduling take place. The time scales range from one or a few days to weeks or months. The plant management layer sends production orders to the different production units in the plant, and receives status about the execution of the orders. This information may be some achieved quality measure of the product, the amount of used raw material and energy, etc. Increased autonomy on this level could be tools for market analysis and schedule optimization in order to run the plant more effectively. The plant should be rescheduled dynamically, for example when a production unit fails to deliver the desired order properly.
2. ***Unit operation.*** This layer receives production orders from the plant management at a rate between a few hours and a few days. These orders typically contain specifications on what to produce, along with the desired quantity and quality. The control system on this level should contain information on how to obtain the desired product. This "recipe" is basically a set point profile for all the local controllers in the production unit. The control system should be able to recover automatically from faults resulting in production loss. If human interaction at a fault is required, the control system should point out necessary actions. This layer is also responsible for improving the recipes with respect to increased production rate, reduced mean-time between failures and reduced use of raw material.
3. ***Loop Manager.*** This layer is responsible for the local control loop operation, with the set point given by the unit operation layer. Its main task is to govern the execution of the real-time control algorithms. For example, new parameters to a PID controller are entered on this level. It thus corresponds to an operator's panel in a

traditional control system. For some plants, this is still the highest level of automation in the control system. Increased autonomy on this level may include better tuning methods, fault detection in the control loop, dynamic reconfiguration of the real-time control algorithms. This will be discussed in much more detail in the following sections.

4. ***Real-time execution.*** This corresponds to the computers that are actually running the PID controllers etc in real-time. It should contain only those parts of the control algorithms that need to run in hard real-time. This typically implies that this layer contains more "pure" control algorithms and less analyzing functions. An algorithm for performance monitoring may for example execute some recursive model estimation on the real-time level, but analysis and decisions can be made at a higher level.

The horizontal line divides the complex control system into two parts. Traditional control systems consist only of the lower part, and the upper part has to be done by humans, possibly supported by computers separated from the control system. There is an ongoing trend to integrate more and more of the higher levels into the control system. However, this integration does not in itself lead to higher autonomy. In order to achieve this, each level must be extended with functions that increase the range of operating conditions that can be handled by the control system.

In this thesis, only methods intended for the lower half of Figure 2.1 are considered. The main motivation for increasing the autonomy on these levels is that the vast majority of the single loop controllers in process industry are not performing satisfactorily, see Bialkowski (1993) and Ender (1993). The main reason is that it would be to costly to optimize the performance of all the control loops in a plant manually. Much work may thus be done in order to increase the autonomy on the local control loop level. The benefits from doing this is primarily improved control, which hopefully pays off in terms of higher production quality and production rate. The control system should provide a bank of algorithms for doing various tasks on the local control loop level. It should also contain suggestions for how these algorithms should be executed in different cases. The process operator or the instrument engineer is supposed to interact with the methods: set some limits, provide extra information, accept conclusions etc.

It is also desirable that the control system allows the user to define new tailor-made algorithms. This requires an open architecture with well specified software interfaces. It should be easy to implement, simulate and verify the new algorithms. This must of course be done while maintaining a high safety level. One framework which provides automatic safety

handling while testing new algorithms is the simplex architecture, see Seto *et al.* (1998). If the new algorithm brings the plant into a potentially dangerous region, a safety controller would be switched in automatically. Another system which has very flexible procedures for on-line reconfiguration is described in Eker (1999).

The following sections will discuss some functionality that is useful in order to achieve a higher degree of autonomy at the local control loop level. Most of the listed extra functions are not invented in this thesis. Some are taken from classical control research areas, others from the intelligent control field. They are all supposed to fit into the framework outlined in the previous section. The functions are grouped into the following categories:

- ***Loop assessment*** for extracting knowledge about the plant, mainly before continuous on-line operation.
- ***Controller selection and tuning***, where knowledge from the loop assessment and possibly additional experiments are used to find a suitable controller for the current control task.
- ***Loop monitoring and diagnosis*** for assessing the performance of the closed loop, and find causes for bad control.

They will now be addressed one by one.

2.3 Loop Assessment

Loop assessment is performed in order to extract basic features of the plant to be controlled. There is no or little support for this in today's control systems. It is supposed to be done manually by plant operators and instrument engineers. If they neglect to do some of the loop assessment tasks, there is an increased risk that the on-line continuous operation will not be satisfactory.

The main motivation for doing loop assessment is to determine if the most fundamental prerequisites for control are fulfilled, and to find out basic characteristics of the plant. This type of information may be useful both in an initial phase, before tuning and running the on-line controller, and later on, when some kind of problems has occurred. Åström (1993) gives a list of useful information and suggested experiments to obtain this. First, a number of qualitative measures should be known, for example:

- Does the controller output at all influence the measured value?
- Is the process self regulating (asymptotically stable), integrating, or unstable?

- Does the process have a monotonous, oscillatory or inverse response?
- Are the plant dynamics fairly linear over the operating range?
- Are there any local non-linearities, such as valve friction and hysteresis?
- What kinds of disturbances are affecting the process?
- Is the controller mainly supposed to work as a servo or a regulator?

Secondly, it is desirable to have some approximate quantitative process knowledge, for example:

- Noise level of the measurement signal.
- Expected operating range of the control loop.
- Allowed operating range during experiments.
- Static gain k_p, possibly varying over the operating range.
- Dominant time scale in terms of average residence time T_{ar}, dead time L and time constant T.
- Amount of friction and hysteresis in the actuator.
- Requirements on the quality of the control.

These pieces of information are important in order to make correct decisions for controller selection and tuning. They will also help to understand the behavior of an automatic tuning procedure or the performance of the on-line control.

Information may come from different sources. The process flow-sheet may provide estimates already in the process design phase. Instrument and process engineers may know time constants etc from experience. Other estimates may require plant experiments. So far the operators and instrument engineers have been responsible for performing and drawing conclusions from these experiments. It is however desired that the control systems are modified to include this kind of support. There are several reasons for this:

- The awareness and understanding of these issues varies substantially among different operators.
- Standardized methods that are programmed into the control system will be less error prone than manually performed experiments.
- When defining a collection of methods, it is possible do design them such that the information produced by one method is compatible with the information needed by other methods.

As mentioned in Sections 2.1 and 2.2, it is not necessary, or even desirable, that these experiments are performed completely autonomously by the control system.

The loop assessment can take place at any time, but should preferably be done already at the startup of the plant. In this way, the control loop may be tuned optimally from the start and potential problems can be avoided. For example, if extensive valve friction is detected already at startup, valve maintenance may be performed before continuous on-line operation, thus avoiding expensive production losses. Loop assessment may be done repeatedly as soon as some new behavior of the control loop is encountered. This situation will be further discussed in Section 2.5.

The rest of this section will give some examples of how to design experiments in order to find some of the desired process knowledge. The presented methods are all operating in open loop on the real-time execution level. Since there is no hard real-time demands except for data logging, the analysis of the experiments may be performed by the Loop Manager, see Figure 2.1. However, in order to increase the degree of autonomy, it may be desirable to have some kind of feedback also during the loop assessment experiments. This kind of supervisory feedback may take place either on the real-time execution level, or on a higher level, depending on the time criticality.

Assessment of disturbances

There are always different types of disturbances present in a control loop. They are typically divided into two categories:

- Low frequency load disturbances, which the on-line controller is supposed to compensate for.
- High frequency measurement noise, which the on-line controller ideally should disregard.

The measurement noise is mainly caused by the sensor electronics, and thus its characteristics does not change dramatically with time during normal operation. Load disturbances, on the other hand, may be introduced in many ways. For example, changes in plant configuration or other operating conditions may cause sporadic disturbances, whereas bad performance in other control loops may cause persistent oscillatory disturbances. This makes it harder to characterize load disturbances. In this work, no explicit modeling of load disturbances will be done.

Information about the disturbances will be used by different methods both for loop assessment, controller tuning and loop monitoring. These methods will have different requirements on the level of detail. The disturbances may for example be characterized by

- The standard deviation σ_e or maximum amplitude e_{max} of the measurement noise.
- The disturbance spectrum.
- A parametric noise model in terms of for example an AR, MA or ARMA model.

This thesis only deals with methods that require only a crude estimation of the noise characteristics in terms of the standard deviation or maximum amplitude.

In Åström and Hägglund (1984), a simple method for estimating the noise level is suggested. A constant control signal is applied, and when the output has reached stationarity, a large number of data points are recorded, and the standard deviation or the maximum noise amplitude may be calculated. This method of course requires that the process is stable. More detailed noise descriptions can be determined using tools from time series analysis and system identification, see Ljung (1999), Johansson (1993), Söderström and Stoica (1989).

Assessment of local actuator non-linearities

Bad valves is one of the most frequent sources of bad control performance in process control, see Bialkowski (1993) and Hägglund (1995). Static friction will for example often induce oscillations in a control loop with integral action. Figure 2.2 shows one characteristic example taken from a flow control loop in a paper mill. A PI controller with $k = 0.2$ and $T_i = 1$ is used. When the valve gets stuck in some position, the flow will assume a constant value. If this value is different from the set point, the PI controller will integrate the error until the control signal produces a force which overcomes the static friction. The valve then moves to a new position corresponding to a new flow, which typically is on the other side of the set point. This will make the control signal grow in the other direction, and a persistent oscillation may build up. It is clear from the figure that the oscillations need not be symmetric or even constant. However, the shapes of the control signal and the process value are characteristic for friction induced oscillations.

Another common non-linearity in control valves is backlash, or hysteresis, see Figure 2.3. This also induces oscillations in the control loop, see for example the simulations in Figure 2.4. The simulation shows the unit load disturbance response for the plant $G(s) = e^{-s}/(5s + 1)$ with the PI parameters $k = 2.5$ and $T_i = 2.15$. The width of the hysteresis is 0.5 for the full line and 0 for the dashed line. With less aggressive control, the oscillation amplitude would decrease gradually until it finally became zero. Thus, the effects caused by hysteresis are normally less severe than

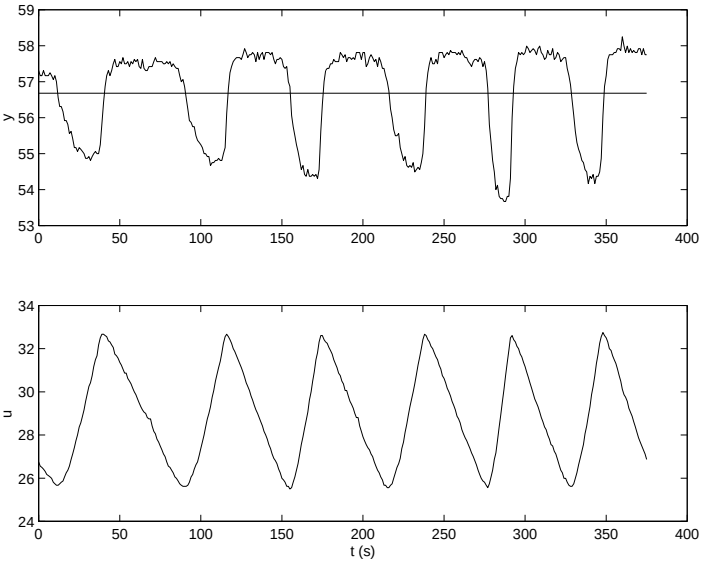


Figure 2.2 Friction induced oscillations in a flow control loop. The PI controller parameters are $k = 0.2$ and $T_i = 1$, and the set point is 56.7.

those caused by friction. However, the transients after a set point change or a load disturbance may be significant.

Hysteresis and friction may not only cause problems during on-line control, but also when performing and interpreting experiments on the plant. Thus, it is useful to assess the amount of friction and hysteresis in

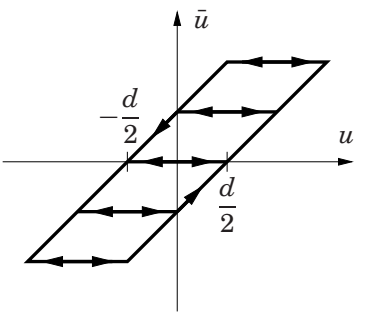


Figure 2.3 Characteristic of the backlash non-linearity with hysteresis width d. u is the applied control signal, and $\bar{u}$ is the effective control signal.

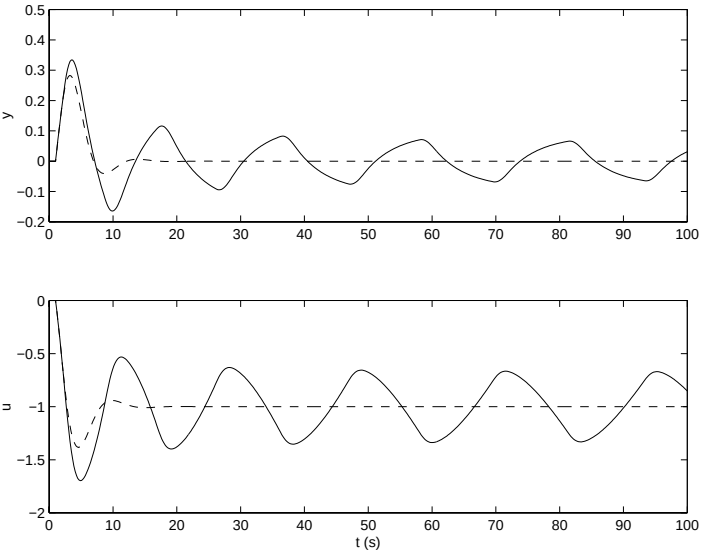


Figure 2.4 Simulation of hysteresis induced oscillations after a unit load disturbance. The plant is $G(s) = e^{-s}/(5s + 1)$, the PI controller parameters are $k = 2.5$ and $T_i = 2.15$, and the width d of the backlash is 0.5 (full line) and 0 (dashed line).

the control valve before the control loop is tuned and put in on-line operation. The assessment experiments may be carried out in many different ways. The experiments outlined here resemble what many instrument engineers perform manually, either at startup or when problems are suspected.

Hysteresis detection The idea behind the algorithm for hysteresis is very simple. The responses of the system with and without the hysteresis can be compared by applying a few open-loop steps in a specific order. One such experiment is shown in Figure 2.5.

Before the experiment is started, it is useful to have crude estimates of the noise level, the gain and the time scale of the process. This information is needed in order to have some apprehension about suitable input magnitudes and necessary times to wait for the process to respond. The required accuracy of the gain and timing estimates could be very low, say within an order of magnitude. They may come from earlier experiments or from user input. If they are missing, the user may still run the method interactively.

An estimate of the amount of backlash in the valve can easily be extracted from the experiments. The first step downwards is performed in

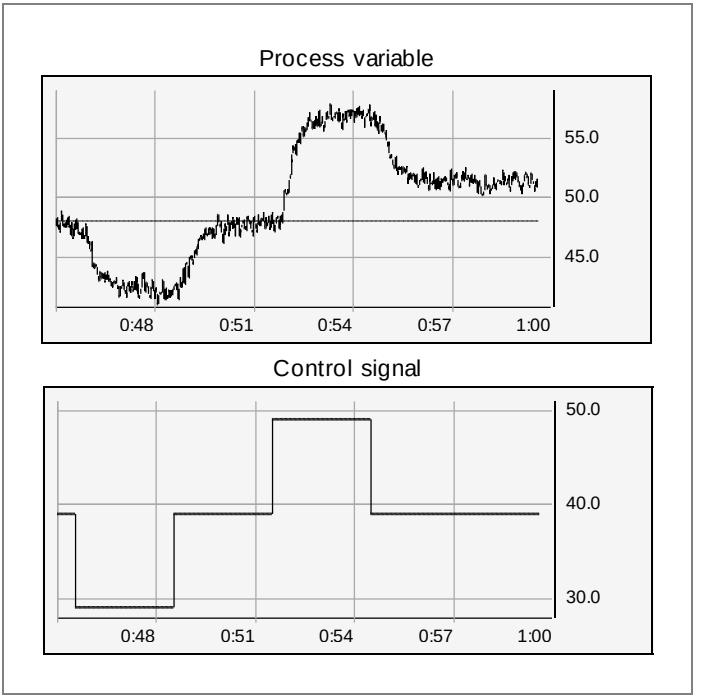


Figure 2.5 A sequence of step responses for detection of hysteresis.

order to find suitable step magnitudes and possibly to set the approximate response time. The step should be large enough to make the output move a distance corresponding to a factor N times the noise amplitude e_{max}. It must for example be large enough to overcome possible static friction. However, there might also be limits on the desirable range Δy_{max} for the experiment. An initial estimate of the required input magnitude may thus be taken as

$$\Delta u = \frac{1}{\hat{k}_p} \min(N e_{max}, \Delta y_{max}) \tag{2.1}$$

where $\hat{k}_p$ is an estimate of the static gain. The step magnitude may have to be changed if the output moves too little or too far. The second step ensures that the possible backlash in the valve is closed when the third step is performed. The third step will thus give a fairly reliable estimate of the static gain of the process. When the fourth step is applied, the output should hopefully go back to the same level as after the second

step. However, the actuator must first travel across the backlash, and therefore the output may exhibit hysteresis. The estimate of d may thus be taken as

$$\hat{d} = \frac{y_4 - y_2}{\hat{k}_p} \tag{2.2}$$

where y_2 and y_4 are the stationary levels of the output after the second and the fourth step, respectively.

Friction detection Many attempts have been made to find good models for describing phenomena caused by friction. Olsson *et al.* (1998) and Olsson (1996) provide a very detailed model, and the latter also includes a survey. However, in this thesis a very simple static friction model for the control valve will be used. It is characterized by one number only. This number u_{fric} is loosely defined as the minimum increment of the control signal that is needed for the valve to move when starting from a constant value. This is a simplistic model because:

- The amount of friction is assumed to be constant over the whole operating range. This is however easily overcome by letting u_{fric} depend on the control signal.
- The size of the friction force often depends of the direction of the movement.
- When u changes continuously, the model is more or less equivalent to quantization. It will thus not capture the effect that the valve stiction becomes less prominent the faster the control signal varies.
- Real friction is random in the sense that it does not stick and slip in exactly the same manner on different occasions, see Olsson *et al.* (1998). This phenomenon is clearly seen in Figure 2.2.

Even if the model is overly simple, it can be used to answer the question: Is there any substantial static friction present, and if so, is it expected to affect the control loop performance drastically?

The static friction u_{fric} may be estimated by applying small steps to the control signal. It is then possible to detect when the control valve actually moves by looking if the process output changes. Figure 2.6 shows a simulation using the simple friction model of such an experiment. Since u moves in steps of 1 unit, and y moves every second or third step, it can be concluded that u_{fric} lies between 2 and 3 units. If higher resolution is needed, u must be increased in smaller steps.

Since the model does not describe the true behavior, the response in reality will not be as distinct as in Figure 2.6. Figure 2.7 shows the result

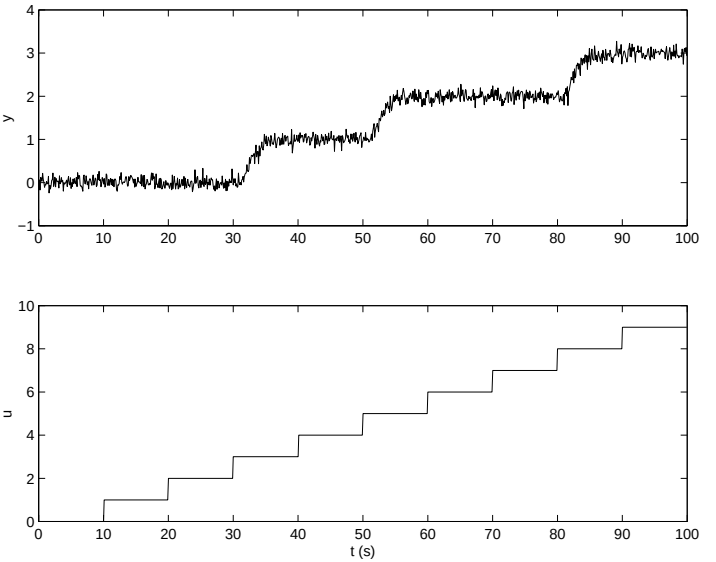


Figure 2.6 A simulated friction detection experiment. The output moves for every second or third step in the control signal.

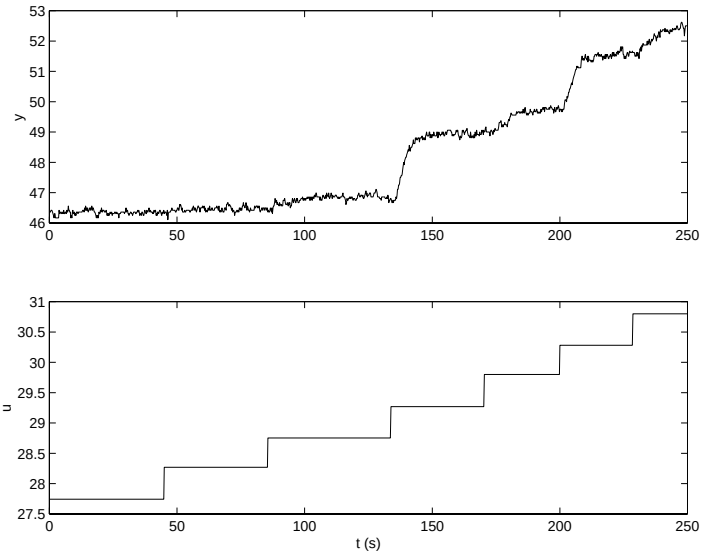


Figure 2.7 Friction detection experiment for the flow control loop from Figure 2.2. The output moves slightly for every step in u, but seem to slip more after the steps at 130 and 200 seconds.

when the same type of experiment is done for the valve in the oscillating control loop in Figure 2.2. The output moves slightly for every step in u, but more for the steps around times 130 and 200 seconds. Since u is changed in steps of 0.5, the estimated friction becomes 1–1.5.

Assessment of dynamics and non-linear characteristics

The purpose of the methods in the previous section was to identify local non-linearities in the actuators. However, they will also provide some assessment of dynamics and "global" non-linear characteristics as a side effect. For example, in the hysteresis test, one true open-loop step response is applied, and may thus be analyzed. Simple measures such as static gain k_p, time constant T, dead time L and average residence time T_{ar} may be extracted from the data. A tool for doing this is described in Chapter 3. The tool may also produce higher order parametric models based on step response analysis.

Apart from a dynamic model, it is also desirable to have at least a crude estimation of the process non-linearities over the whole operating range. The simplest, and often the most important, non-linearity to consider is the static characteristics of the process. The ramp experiment for detecting friction will actually give an estimate of the static input-output map. However, this experiment is usually done in a small range due to the required accuracy. Thus, the same experiment may be repeated for the whole operating range using larger input steps. It will then be possible to get an estimate of the static non-linearity. If this is known, the controller gain may be gain scheduled, and the controller may need to be tuned only for one operating point. This is further discussed in Chapter 3.

Assessment of interactions

Process control systems are complex in the sense that they are non-linear, multivariable, and time-varying. Despite this, most of the sensors and actuators are paired in simple, fixed single-input single-output (SISO) control configurations. The reason for this is of course simplicity, since the modeling and the control design become much more cumbersome if the full problem is considered. A consequence of the single loop control configuration of a multivariable system is that the loops are interacting in a complex manner, and in different ways. Variations in one loop may for example show up as load disturbances in other loops. Depending on the magnitude and frequency content of these disturbances, they may be easily compensated for, or they may actually constrain the performance of the disturbed loop.

Loops that are tightly connected should also be studied jointly. In terms of control design, this can either be done with multivariable control using a few sensors and actuators, or with decentralized control, see Bryant and

Yeung (1996) and Palmor *et al.* (1995). Another interesting issue is to explore which signals can be used for feed-forward and cascade control. One method of assessing this is described in Johansson and Hägglund (1999). Di-graphs are used for describing causal relationships between different variables in the plant. Suitable control structures may be concluded automatically from these graphs.

This thesis will henceforth treat loop interactions only as load disturbances acting on a SISO process. This may of course give erroneous results in some cases where the loop interactions must be looked upon from a multivariable perspective. However, there are still many control loops which are readily treated as SISO loops. This motivates why it is interesting to continue to study autonomous control of SISO processes.

2.4 Controller Selection and Tuning

PI controllers are by far the most common controllers in the process industry. The reason for this is that they are simple, yet able to solve most control problems as long as the performance requirements are modest. The structure of PID controllers is almost as simple, but they do require more effort when tuning the controllers by hand. This has made autotuning procedures desirable features in modern control systems.

More advanced controllers are not used very frequently yet in practice. Adaptive controllers are used occasionally, see Åström *et al.* (1993), and Model Predictive Controllers (MPC) become more and more common, see Morari and Lee (1999). This thesis mainly deals with PI and PID control, but the point of having an autonomous control system is that it should be able to replace them as soon as other controller structures are believed to solve the control problem better.

PI and PID control and tuning

Various tuning methods for PI and PID controllers exist, Åström and Hägglund (1995). The classical empirical methods are the Ziegler-Nichols methods. Their fundamental idea to characterize the process with a few parameters and to determine controller parameters from a table is frequently used.

One of the most frequently used methods in process industry today is the Lambda tuning, see Rivera *et al.* (1986). The fundamental idea is that it should be possible to select the time constant λ of the closed-loop system. This is done by finding a first order delayed model of the process

$$G(s) = \frac{k_p}{sT + 1} e^{-sL} \tag{2.3}$$

The controller parameters are chosen as

$$\begin{aligned} k &= \frac{T}{k_p(\lambda + L)} \\ T_i &= T \end{aligned}$$

The integral time is thus used for cancelling the process pole. There is a potential danger in doing this, since the controllability or observability of the plant is lost. This may for example cause the load disturbance response to be unnecessarily slow. The controller gain is used for setting the closed-loop time constant approximately to λ. The approximation is valid only if λ is significantly greater than L. Controllers designed with Lambda tuning in process industry mostly use $\lambda > T + L = T_{ar}$. In other words, the controller actually makes the closed loop slower than the open loop. The drawback with potentially slow load disturbance response due to cancellation is then not critical. A perhaps more serious limitation with Lambda tuning is that it does not naturally extend to PID control.

The PI design method in Åström *et al.* (1998) takes a different approach. Here, robustness is of primary interest and not the response times. The fundamental idea is to minimize the integrated error after a step load disturbance, subject to the constraint that the sensitivity function is always less than a specified value. To be applied exactly, this method requires knowledge of the full process model. More precisely, it uses knowledge of the frequency response of the plant for frequencies with approximately -90° to -270° phase shift. It is thus sufficient to have a good estimate of the frequency response in this limited frequency range. A drawback with the method is that there is no simple table lookup to find the parameters. Instead, a non-linear equation needs to be solved. With computer support, this is not a severe drawback, though. Panagopoulos (1998) extends the method to PID control. An new, alternative PID design method based on the PI design method in Åström *et al.* (1998) is presented in Section 4.4 in this thesis.

The design methods based on models with a few, easily estimated, parameters are very tractable because of their simplicity. The Kappa-Tau method, Åström and Hägglund (1995), attempts to combine this simplicity with the advanced tuning methods from Åström *et al.* (1998). There is one frequency domain version and one time domain version of the Kappa-Tau method. Both are based on three-parameter models of the plant. The frequency domain version uses the static gain k_p, the ultimate gain k_u and the ultimate period T_u. The time domain version uses the static gain k_p, the apparent lag T and the apparent dead time L. It turns out that it is useful to let the controller parameters depend on the gain ratio $\kappa = 1/k_p k_u$ and the normalized dead time $\tau = L/(L + T) = L/T_{ar}$, which explains

the name of the method. Details on modeling issues are found in Åström and Hägglund (1995). The Kappa-Tau method thus uses the same information as the Ziegler-Nichols methods plus the static gain, which is easily estimated. The method was constructed by designing PI and PID controllers using the sensitivity-based methods in Åström *et al.* (1998) and Panagopoulos (2000) for a large number of plants. The controller parameters were then plotted in diagrams as functions of the model parameters. "Average" curves were then calculated for each controller parameter. These curves thus give controller parameters that "on average" correspond to the sensitivity-based design methods. The parameters may be found either looking in the graphs or by the analytical expressions for the curves.

Automatic tuning

There are two main approaches to automatic tuning in today's commercial control systems. One is based on open-loop step response analysis, and the other is based on relay feedback. Åström *et al.* (1993) presents the basic techniques and a survey of automatic tuning in commercial systems.

Wallén (1995) suggested an extension to the relay feedback method in order to get an estimate of the static gain of the process. This provided an automatic tuning procedure for Kappa-Tau design in the frequency domain. Implementation aspects of this auto-tuning procedure is further discussed in Section 6.2. The method has recently been implemented in SattLine from ABB Automation Products, see Norberg (1999). An autotuner for the time domain version of the Kappa-Tau method has been implemented for the Mitsubishi PLC system at Beijer Electronics, see Bannura (1994).

This thesis will present a new auto-tuning procedure for PI design according to Åström *et al.* (1998). It is based on relay feedback using time-varying hysteresis. The data is used for estimation of the frequency response of the process using spectral methods. The method is described in detail in Chapter 4.

The automatic tuning procedures typically consist of one experiment phase, and one design calculation phase. The experiments must of course be executed on the real-time level, but the experiment data may be sent to the immediately higher level for design calculations. This way, the computational load on the real-time level is very modest. Normally, the design calculations are not time-critical. Thus, rather complex design methods may be used without disturbing the execution on the hard real-time level.

Selection of controller structure

So far, only PI and PID control have been discussed. These controller structures are able to solve most of the SISO control problems occurring

in process control. However, due to the simple structures, the performance that can be achieved is limited. Åström (1997) and Åström (2000) discuss fundamental limitations on achievable control performance given by the dynamics of the plant. Other factors that limit the control performance are the disturbances and possible non-linearities.

If both the desired and the maximum achievable performance is much higher than the one obtained by PI or PID control, it may be worthwhile to consider other structures. For example, for processes dominated by long dead times, the PI and PID controllers will perform far from the fundamental limitations. A few examples:

- For non-linear processes, PID controllers typically give different performance in different regions. This is often successfully solved using gain scheduling. It is very convenient to use auto-tuning to generate the schedules automatically.
- For time-varying dynamics, some adaptive technique may be needed. The survey in Åström *et al.* (1993) lists a number of commercial products with adaptation of the parameters in a PID controller.
- For processes dominated by long dead times, some kind of dead-time compensation may be used in order to increase the bandwidth of the closed loop while retaining the stability margins. One example is the Predictive PI controller in Hägglund (1992).
- Model predictive control (MPC), see is a controller structure that can be used in many situations with, for example, non-standard control objectives and miscellaneous limitations and constraints. See for example Morari and Lee (1999).

Derivative action is not commonly used in PID controllers in process industry. Since the control performance may increase with the use of derivative action, it would be interesting to have some measure and assessment of the expected improvement. Using the design criteria in Panagopoulos (1998), the performance is always improved when derivative action is used. However, evaluating other criteria such as integrated absolute error and amplification of measurement noise, it is not always true that the PID controller outperforms the PI controller. A crude classification of processes showing most benefit of PID control when considering the dynamics only, is when the normalized dead time $\tau = L/T_{ar}$ lies in the range 0.2–0.6. Derivative action is also very beneficial for processes with integral action.

Filtering is another issue related to the controller structure. The normal use of filtering is to attenuate high frequency measurement noise. The effects of the filtering should preferably be negligible around the closed-loop bandwidth from the controller design. If this is not the case, the

filters must be taken into account in the controller design, see Panagopoulos (2000).

Filtering is also used in order to avoid aliasing effects in sampled data systems. The cut-off frequency of the anti-aliasing filter is coupled to the sampling interval. This implies that the filter should be altered when the sampling interval of the controller is changed. However, this is normally not possible, since the anti-aliasing filter is an analog filter just outside the IO board of the computer. This can be solved by having fast sampling of all signals with a fixed anti-aliasing filter, and then use decimation in order to achieve sampling intervals that match each control loop.

2.5 Loop Monitoring and Fault Diagnosis

The purpose of loop monitoring is to detect degraded behavior during on-line control. Some possible faults and types of degraded behavior that can be detected are:

- Sensor and actuator faults.
- Increased disturbance level.
- Badly tuned controller, for example due to changed dynamics.

All methods that do loop monitoring send status signals to the Loop Manager. The type of information carried by these status signals varies between different types of monitoring algorithms. It is reasonable to divide the loop monitoring into three categories that reflect these differences:

- Low-level alarms, which simply detect crossings of levels etc.
- Performance assessment, which calculates some measure of the control performance on-line and sends alarms when this measure is not acceptable.
- Fault diagnosis, which tries to detect faults, not only symptoms, in the control loop.

The different categories will now be discussed briefly.

Low-level alarms

The simplest form of loop monitoring is alarm handling on the signal level. Some possible tasks in the alarm handling are:

- Range check of the control signal and/or process value. The action taken after a triggered alarm could be anything from a warning presented to the operator, to an immediate production shutdown.

- Noise level monitoring. If the noise level is increased dramatically, or if it becomes very small, the sensor or some wiring is probably broken.

These alarms typically use very little computing power, and operate on a very basic level. It is up to the higher levels in the control system to decide which alarms are actually useful, and only implement those. There should thus be some supervisory function that uses the alarms in some way. If, for example, the noise level has become very small, the Loop Manager should do at least one of the following:

- Warn the operator that the sensor may be broken.
- Perform some simple experiment, for example a set point change, to see if the sensor value changes. Before such an experiment is performed, it might have to be accepted on the unit operation level.
- Pass the alarm to the unit operation level, which may use the alarm to explain errors in neighboring control loops and confirm the fault to the Loop Manager.

If an experiment or some higher level reasoning confirms that the sensor or wiring is broken, the instrument engineer should be notified, and the hardware should be repaired.

Performance assessment

The alarms discussed in the previous section provide low-level information about the status of the control loop. They may for example cover the most severe errors when the control loop has more or less stopped functioning. It is, however, more difficult to have simple alarms that give a more detailed status of the quality of the control. This is the motivation for performance assessment methods. The normal use of these methods is to constantly update the performance measure and compare it with the acceptable level which is defined somehow. If bad control performance is detected, an alarm is sent to the Loop Manager. In this respect, performance assessment algorithms do not differ from the low-level alarms discussed above. There is thus not a clear distinction between alarm generation and performance assessment.

There are different classes of methods within the performance assessment category, for example:

- Variance-based methods according to Harris (1989) and numerous followers.
- Detection of oscillations, for example Hägglund (1995).
- Methods for detecting overdamped control, see Hägglund (1997a).

The first class is the one that has drawn most attention. The field was started by Harris (1989), who proposed that the control performance should be measured by comparing the current variance of the output with the one obtained by a minimum variance control law, Åström (1967). Harris also showed that this minimum variance can be estimated irrespective of currently used control law, as long as the dead time of the process is known. Several authors have suggested improvements and modifications to the original algorithm. Lynch and Dumont (1996) use a Laguerre network for estimating the coefficients in the noise description, and an on-line estimation of the dead time. Tyler and Morari (1995) take the effect of unstable poles and non-minimum phase zeros into account. Horch and Isaksson (1999) replace the implicit dead-beat assumption in the minimum variance control law by a more realistic pole placement. Harris *et al.* (1996) extends the measure to multivariable plants. Some of the methods have been implemented in large-scale plants, with reported success.

The other methods presented above are not based on stochastic control theory, but use a more pragmatic view. The oscillation detection algorithm in Hägglund (1995) repeatedly calculates the integrated absolute error (IAE) between two consecutive zero crossings of the control error. If this sequence contains large values of the IAE during a limited time, this is interpreted as an oscillation of the control loop. The method is implemented in commercial controllers from ABB Automation Products.

The performance assessment methods typically have most of the calculations executing in hard real-time. The variance-based methods use recursive estimation of the noise model in order to estimate the minimum achievable variance. The oscillation detection algorithm calculates the IEA sample by sample. However, it is mostly not critical that the bad performance is actually detected exactly when it occurs for the first time. This is especially true since performance typically deteriorates gradually, and there is probably a long time when the methods "almost" signal for bad performance. It should thus mostly be sufficient to send batches of on-line data to the Loop Manager on some regular basis and then perform the calculations without timing constraints.

Control loop diagnosis

The performance assessment algorithms discussed above are supposed to detect unsatisfactory control. However, none of them try to find any causes for the bad control. This is instead a task for fault detection and isolation (FDI) methods. When a control loop is performing badly, without being totally out of order, it is normally caused by either of the following reasons:

- The controller parameters are not set properly.
- External disturbances cause large variations which cannot be taken care of by the controller.
- Non-linearities such as valve friction induce oscillations in the control loop.
- The current controller structure is not able to control the process with acceptable performance.

Many different approaches to FDI exist. Traditional model-based methods are mostly based on residual generation and analytical redundancy, see Frank (1990) for a survey. This kind of methods requires a fairly accurate quantitative model of the nominal plant behavior, as well as the behavior when faults are present. This is mostly not available in a typical control loop in process industry. Neural networks and knowledge-based methods are other approaches that are often used, see for example Frank and Koeppen-Seliger (1997).

In most cases, the traditional FDI methods use multiple sensor readings to distinguish between different faults. Here, we would instead like to do diagnosis on the local control loop level using only the control signal and the process value. The rather specific nature of this FDI problem has inspired some tailored methods. For example, Thornhill and Hägglund (1997) use harmonics analysis in order to find a characteristic signature of an oscillating control loop. Horch (1999) shows that correlation analysis can be used to distinguish valve induced oscillations from other ones. These methods use only the on-line data for the diagnosis. In Wallén (1997) a sequence of off-line experiments, including renewed loop assessment and controller tuning, is suggested in order to distinguish between different causes for the bad control. It should not be considered a drawback to temporarily stop the PI(D) control, as long as the experiments are monitored properly.

2.6 Summary

This chapter has discussed different aspects of autonomy in process control. A fully autonomous control system will not be realistic for many years. The most important reason is that it is difficult to guarantee safe operation in a large-scale plant. However, many things may be done in order to increase the degree of autonomy on any level of the plant. This thesis focuses on increased autonomy on the local control loop level. Some desired features in order to do this have been reviewed.

3

Dynamics Assessment

The purpose of loop assessment as discussed in Section 2.3 is to extract basic process knowledge. Most of this knowledge is formulated as scattered chunks of information, both quantitative and qualitative. In order to do controller design, monitoring and diagnosis etc., it is useful to instead have a process model parameterized as a transfer function or a state space representation.

First, it is necessary to decide what kind of model is needed:

- Should it be in discrete time or continuous time?
- Should it be a white, gray, or black box model?
- What model structure and order is desired?
- Will it be necessary to include non-linearities in the model?

The identification procedure must also be determined:

- What data should be used for identification? Do we need data sets with special input signals or can we use available on-line or off-line data? Are we at all allowed to subject the plant to the input signals we want?
- What optimization methods and criteria should be used for selecting the best model?

There are of course no generic answers to the questions above. What is best in each situation depends very much on the process and the purpose of the process model.

System identification has found its way into master thesis programs worldwide, with several textbooks covering the subject, *e.g.* Ljung (1999), Johansson (1993), Söderström and Stoica (1989). There are also standard commercial tools available for people with some identification knowledge and experience, *e.g.* the Identification Toolbox in MATLAB. Before you can do system identification you need to find suitable excitation signals, model

structure, sampling interval, and so on. In order to decide these, you must know some details about the process, for example in terms of a coarse process model.

Normally, no "proper" system identification is ever done for most control loops in process industry. There are many reasons for this:

- Most control problems are solved with PI or PID controllers, which may be tuned using simple models only, or no process model at all.
- It is mostly practical issues such as disturbances and measurement noise that is important for control, rather than complex dynamics.
- The number of control loops is often so high that it would take too much time to do a full identification procedure for all loops.
- The understanding for system identification, ARMAX modeling and so on, is mostly not very high among operators and process engineers. This hinders the use of more complex models, even where it could be motivated.
- Processes are frequently non-linear.

Still, a simple form of system identification is often done by process engineers when tuning the controllers. Typically, they estimate dead time, time constants, and so on from open-loop step responses, or bump tests as they are often called. Then, they follow some algorithm to find acceptable controller parameters. Some common tuning schemes are the Ziegler-Nichols method, the Cohen-Coon method, lambda tuning, see Åström and Hägglund (1995) for a comprehensive listing. A slightly more advanced method would be to fit the experimental data to a step response of a selected model structure, and use some design method on the resulting process model. In both approaches there is need for computer support to facilitate the identification.

This chapter describes a tool for analyzing step responses, or sequences of step responses. It provides a user friendly method to fit low order dynamic models to data. In practice, it is more important to have a crude estimation of process non-linearities than a very detailed dynamical model. For this reason, the tool makes it possible to identify a static non-linearity along with the dynamical model. This makes the tool useful in practice, since most control problems occurring in process industry can satisfactorily be solved using PID control with gain scheduling. Even if more advanced models and controllers may be needed for complex problems, the tool may serve as a good starting point.

Section 3.1 describes the used model structures. The interactive graphical user interface, GUI, is developed using MATLAB 5 graphics. The user is allowed to shape the step response of a certain model structure to make

it follow the real step response data. The graphical manipulation is discussed in Section 3.2. Identification of non-linear processes is discussed in Section 3.3. Along with the interactive model manipulation, the tool lets you perform a least squares fit of the model to the data. The graphically manipulated model will then supply the optimization procedure with initial parameter estimates. This is described in Section 3.4.

3.1 Model Classes

It is necessary to decide what model classes to use before any identification may take place. The model classes considered in this thesis are

- Linear continuous-time process models described as a transfer function $G(s)$.
- Linear continuous-time process models with a static non-linear function on the input, so called *Hammerstein* models.
- Linear continuous-time process models with a static non-linear function on the output, so called *Wiener* models.

Most plants in process industry are more or less non-linear. In some cases the process gain may differ by orders of magnitude across the operating range. Thus, it is often not useful to model the linear dynamics very accurately unless the non-linear behavior is also described. If the process is normally operated within a small region, it is often reasonable to use a linearized model around the operating point. However, during startup and shutdown, the process will probably deviate from the normal operating point and the dynamics may change drastically. This behavior may be captured by a non-linear model. If the controller is designed to take the non-linearity into account, the need for special solutions and fixes at abnormal situations may be avoided.

It is very difficult to identify a general non-linear dynamical system. Haber and Unbehauen (1990) and Patra and Unbehauen (1993) are two survey papers discussing identification of non-linear systems with different structures. Hammerstein and Wiener models are reasonable simplifications which cover some properties of the non-linearity, at least the gain variations. Properties of these model structures will be discussed in more detail in Section 3.3.

3.2 Principles for Man-Machine Interaction

The tool is designed to analyze step responses or sequences of step responses. The goal is to have a GUI where the shape of the step response can be edited in a natural way. This is achieved by having a number of *handles* which the user may manipulate. In order to make the man-machine interaction intuitive, it is desirable to have a one-to-one correspondence between a process feature and one specific handle. For example, the dead time is one feature of the process that should be manipulated with a single handle. Basically, you need one handle per parameter in the model, plus one handle for each initial condition you want to edit.

When you decide on process features to manipulate, you should try to accomplish the following:

- The user should be familiar with the way of representing the features.
- The features should be easy to manipulate by dragging the handles.
- The user should be able to foresee how the step response will change when a certain handle is dragged.
- The behavior should be as consistent as possible for different model structures, different parameter values etc.
- If possible, there should be an obvious connection between the handles and the model parameters.

It is difficult to accomplish all these things at the same time. The items above may be the more or less important, depending on what the step response manipulation is going to be used for. The discussion in this chapter is aimed at a tool for fitting a model to experimental data. An education tool for understanding properties of dynamical systems should probably have other handles.

First order process

It is by no means a trivial task how to select the most intuitive process features and handles to manipulate. Even for a simple first order model

$$G(s) = \frac{K}{sT + 1} e^{-sL} \tag{3.1}$$

you have many ways of selecting process features. For example, the time constant T may be calculated from the maximum slope of the step response, or from the time where the response has reached $\approx 63\%$ of its

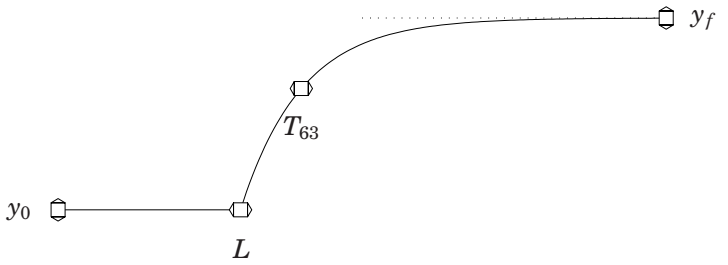


Figure 3.1 Step response and handles for a first order system with time delay.

final value. Actually, any reasonable definition of rise time can be used to calculate T.

One possible set of handles for the model (3.1) is:

- A level y_0 to set the initial value.
- A level y_f to set the final value. y_f and y_0 together give K in (3.1).
- A time L for editing the dead time.
- A time T_{63} where the step response has reached $1 - e^{-1} \approx 63\%$ of its final value. This gives $T = T_{63} - L$.

The step response with its corresponding handles is shown in Figure 3.1. The handles are squares with arrows indicating the direction in which they may be moved. The level handles y_0 and y_f may be dragged vertically, and the time handles L and T_{63} may be dragged horizontally. When one level handle is dragged, the other level handle remains in the same place, and vice versa for the time handles.

Note that this choice of handles does *not* give a one-to-one correspondence between handles and model parameters, since the time constant T will be affected both when L and T_{63} are changed. The reason for using L and T_{63} as handles anyway is that they are intuitive for the user. This is considered more important than perfect decoupling in parameter space.

Figure 3.2 shows how to fit first order model to data, step by step. First, the initial and final values are set (plot 2), next the time delay (plot 3), and finally the rise time (plot 4).

Multiple lags

The transfer function (3.1) is the most frequently used model structure in industrial practice. It is simple, yet it captures a few fundamental properties of the process, and it can be used for simple controller design. However, by using a higher order model with more parameters you will be

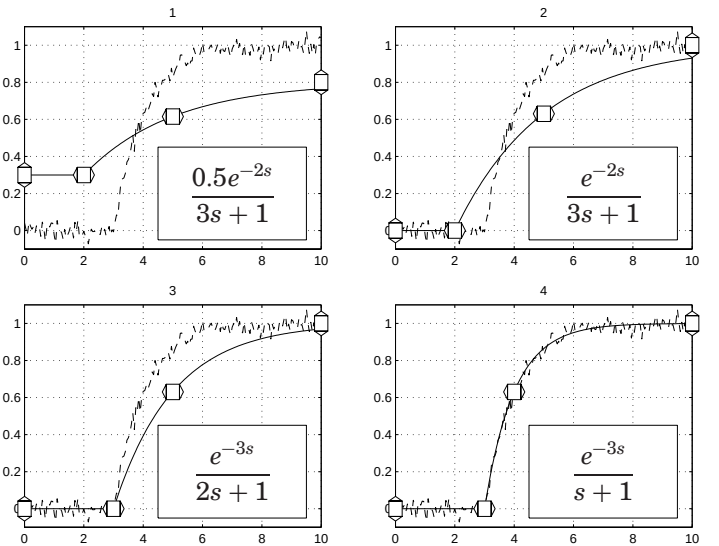


Figure 3.2 Step by step manipulation of a first order system.

able to achieve better model following. At the same time you would need more handles to shape the step response. It becomes more and more difficult to find appropriate step response features and corresponding handles as the number of model parameter increases.

The simplest form of higher order processes is just to have multiple lags, *i.e.*

$$G(s) = \frac{K}{(sT+1)^n}\, e^{-sL} \tag{3.2}$$

where n is a positive integer. The step response of this model is given by

$$y(t) = K\left(1 - e^{-(t-L)/T}\left(\sum_{k=0}^{n-1}\frac{1}{k!}\left(\frac{t-L}{T}\right)^k\right)\right), \qquad t \geq L \tag{3.3}$$

The set of handles for first order systems above will still be possible to use, see Figure 3.3. However, there is no longer an explicit expression for T as a function of L and T_{63}. It will instead be found using simple numerical iteration with initial estimate $T \approx (T_{63} - L)/n$. Other measures may have given explicit formulas for T, but the 63% rise time is well established in practice and easy to modify graphically. Furthermore, the computational burden for finding T from $T_{63} - L$ is negligible.

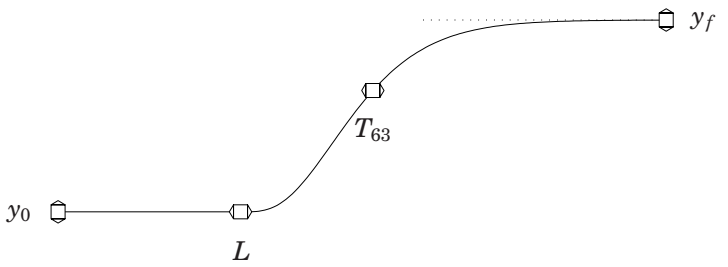


Figure 3.3 Step response and handles for the model (3.2).

Process zero

The next extension is to allow inverse response, or non-minimum phase behavior. This is done by including a process zero in the model:

$$G(s) = K\,\frac{\beta T s + 1}{(sT+1)^n}\,e^{-sL} \tag{3.4}$$

which has the unit step response

$$y(t) = K\left(1 + e^{-(t-L)/T}\left(\frac{\beta}{(n-1)!}\Big(\frac{t-L}{T}\Big)^{n-1} - \sum_{k=0}^{n-1}\frac{1}{k!}\Big(\frac{t-L}{T}\Big)^{k}\right)\right), \qquad t \geq L \tag{3.5}$$

For $\beta < 0$, the process has a non-minimum phase zero. The minimum value of the unit step response is reached at time

$$t_{min} = L + (n-1)\frac{\beta T}{\beta - 1} \tag{3.6}$$

The new model parameter requires one new graphical handle. The basic characteristic in the time domain for a non-minimum phase system is the inverse response. It then makes sense to use the minimum value as a measure of the non-minimum phase behavior. The set of handles for the system (3.4) is shown in Figure 3.4. A drawback with this choice is that neither T nor β can be calculated directly from the handles. However, they are easily found using simple numerical iteration. Again, it is possible to find other handles which would give algebraic expressions for β and T, but they would have been less intuitive for the user.

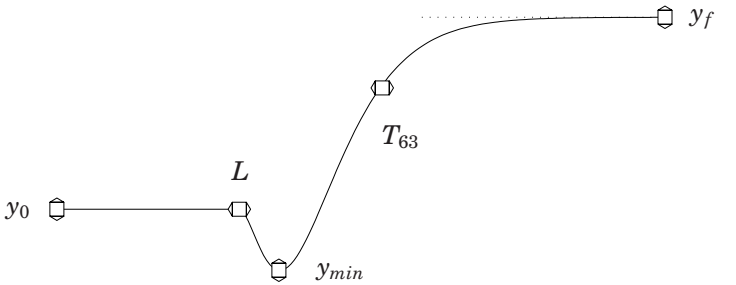


Figure 3.4 Step response and handles for the model (3.4) when $\beta < 0$.

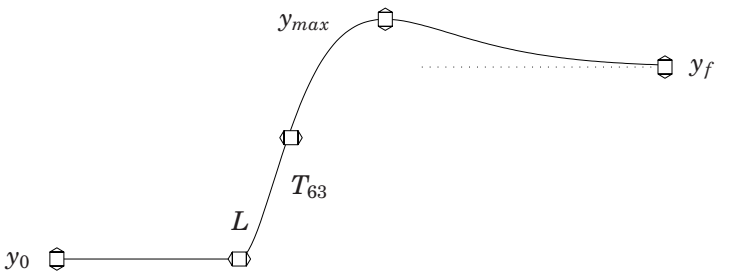


Figure 3.5 Step response and handles for the model (3.4) when $\beta > 1$.

By parameterizing the process zero as $-1/(\beta T)$, β alone will set the shape of the step response, and T will just scale it in time. Thus, β can first be calculated from n and $(y_{min} - y_0)/(y_f - y_0)$. Then, T is calculated from β, n and $T_{63} - L$.

The tool also allows left half-plane zeros, *i.e.* $\beta > 0$. For $\beta > 1$, the zero is closer to the origin than the pole is. This will cause the step response to have an overshoot. This may be treated analogous to the non-minimum phase case, see Figure 3.5. The model can be concluded from the handles by first computing β from the relative size of the overshoot, and then T from β, n and $T_{63} - L$.

When $0 < \beta < 1$, the zero will not cause any minimum or maximum level of the step response. Thus, there is no way of making the manipulation consistent with the previous cases. In the tool, the handle for minimum and maximum level is still used to manipulate β. This is done by interpolating β between 0 and 1 when the handle is moved from y_0 to y_f. This behavior is far from intuitive, since the user no longer directly manipulates a fundamental property of the step response. Despite this, it

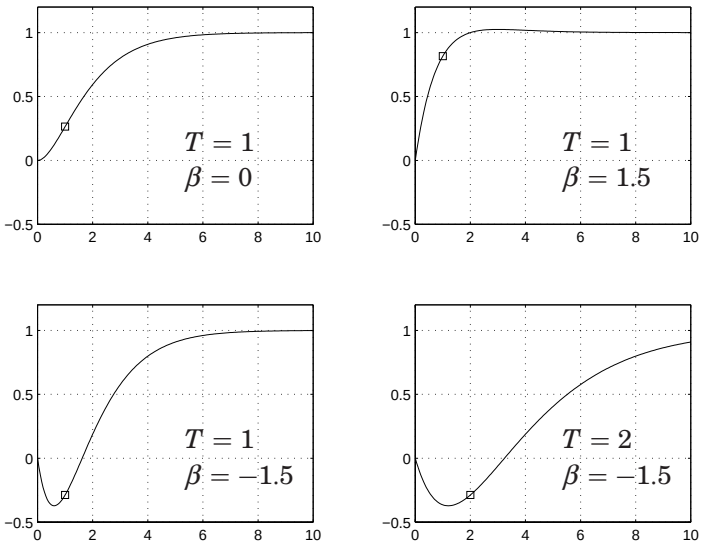


Figure 3.6 Editing T and β using a single handle (the square mark) for the model (3.4) with $n = 2$. Horizontal movement of the handle changes T and vertical movement changes β.

was selected in order to get a smooth transition between the cases $\beta < 0$ and $\beta > 1$.

Just to give an example of alternative ways of selecting handles, we will show how to edit T and β simultaneously using one handle only. If the level of the point is not fixed at 63%, you can use the two degrees of freedom to uniquely determine both T and β. A natural way of doing this is to let T equal the horizontal coordinate of the handle minus the time delay. If the vertical coordinate then is used as $y(T)$, Equation (3.4) gives β. A major advantage of this is that it works for all $T > 0$ and all values of β, see Figure 3.6. One severe drawback is that the user feels that he is no longer editing the "fundamental properties" of the step response when there is an overshoot or undershoot. The reason is that β, and consequently the maximum or minimum value, is sensitive to small vertical movements of the handle, especially with n large. The method is therefore rejected, since the primary motivation for the model structure (3.4) is to handle these cases.

Integrating processes

Integrating processes is another class of processes that is frequent in process industry. The standard example is the level in a tank where the

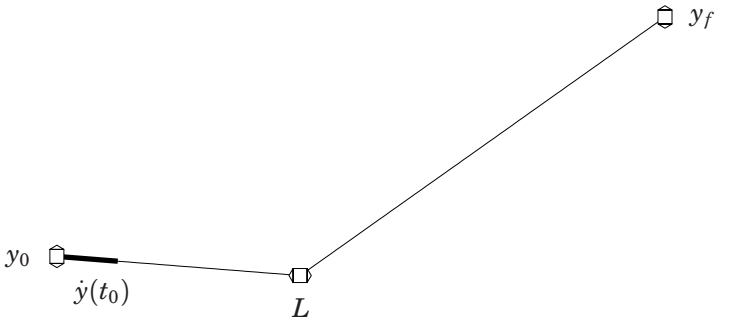


Figure 3.7 Step response and graphical support for the process (3.7).

inflow and outflow are controlled by pumps or valves. The simplest form considered in this tool is a pure integrator, possibly with a time delay:

$$G(s) = \frac{K}{s}\, e^{-sL} \tag{3.7}$$

which has the unit step response

$$y(t) = K\ (t - L)\,, \qquad t \geq L \tag{3.8}$$

We will need handles for manipulating

- The initial value y_0,
- The initial *slope*, $s_0 = \left.\dfrac{dy}{dt}\right|_{t=0}$,
- The time delay L, and
- The gain K, given by the difference between final and initial slope.

The set of handles in Figure 3.7 gives one possible way of manipulating the response. The handles represented by squares with arrows behave in the same way as before, namely translation of one point of the response horizontally or vertically. The initial slope is edited by rotating the thick bar marked $\dot{y}(t_0)$ around the initial point y_0. The final slope, and consequently the gain K, is edited by defining one point y_f on the final asymptote of the step response.

To get a smoother response than for model (3.7), you may have a lag in series with the integrator:

$$G(s) = \frac{K}{s(sT + 1)}\, e^{-sL} \tag{3.9}$$

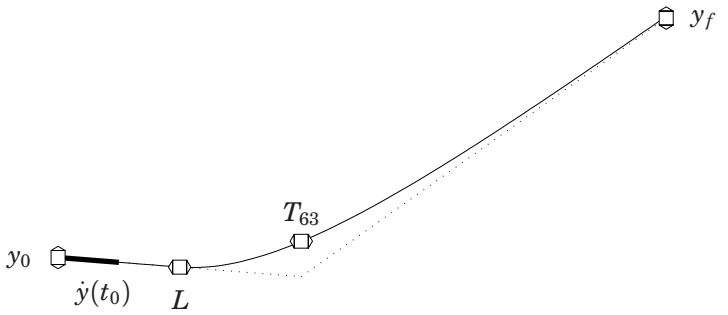


Figure 3.8 Step response and handles for the process (3.9).

with the unit step response

$$y(t) = K\left(t - L - T\left(1 - e^{-(t-L)/T}\right)\right), \qquad t \geq L \tag{3.10}$$

In addition to the handles for (3.7), we also need a way of editing T. One way is to rewrite (3.9) as

$$G(s) = \left(\frac{K_i}{s} + \frac{K_p}{sT+1}\right) e^{-sL} \tag{3.11}$$

with $K_i = K$ and $K_p = -T \cdot K$. Since the second term in (3.11) is a first order system, it makes sense to use T_{63} as a handle in this case too, see Figure 3.8. It may thus be interpreted as the rise time of the first order step that is superimposed on the true integrating response. Alternatively, it may be viewed as the time when the *slope* of the step response has changed 63%.

If you allow all parameters in (3.11) to vary independently, you may describe other behaviors which can be found in process industry. K_i represents difference between final and initial slope, K_p represents the "jump" between the initial and final asymptotes at $t = L$, and T is the time constant of the proportional part. When you vary K_p by moving the handle marked y_1 in Figure 3.9, you may actually see this as moving a process zero. With $K_p < -TK$ you will see non-minimum phase behavior, and with $K_p > 0$ you will have a left half-plane zero close to the origin.

More complex model structures

By adding more poles and zeros you would need more and more degrees of freedom to edit the step response. It will be increasingly difficult to find handles which mirror the visible behavior in the step response, and

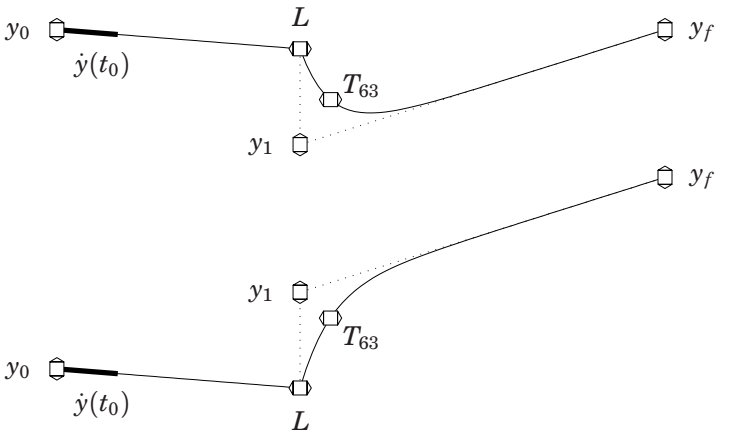


Figure 3.9 Step responses and handles for the process (3.11) with $K_i > 0$. K_p is negative in the upper plot and positive in the lower.

the graphical tool would inevitably be more complex to use. A solution would be to let the user specify a number of points that the step response should to through. This way, the tool will resemble traditional curve fitting programs. However, the model structure imposes very hard constraints on possible curve forms. This implies that if the user tries to move one of the points on the curve, the set of allowed values for this point is very limited. It will then be difficult to drag the response into the desired shape.

Since the tool should be used for preliminary system identification only, it does not make very much sense to add more features than necessary. If a more complex model is needed, the user must first fit one of the model structures above to the data. Then, least squares optimization can be used for finding a complex model, using the simple model for initial parameter estimates.

Additional features in the tool

Apart from the graphical manipulation of the step response of the model described above, the tool includes some features which increase its usefulness.

- It is possible to obtain the process model which minimizes the mean square error between the model output and the experimental data. Properties of this optimization problem are further discussed in Section 3.4.

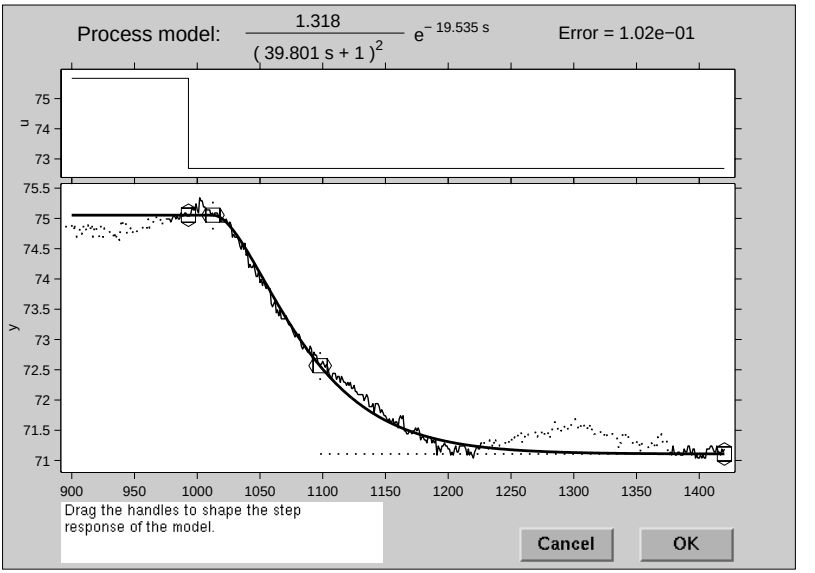


Figure 3.10 Fitting a second order delayed model to data from the temperature control loop in Example 3.1. The dotted data points are not used in the optimization.

- Static input and output non-linearities may be included in the model. This is described in Section 3.3.
- When the data series consists of several step responses, the user may select which step response to manipulate graphically. The optimization is performed on all of the selected data set, though.
- The tool includes routines for PI and PID designs for the current process model. The design methods are taken from Åström *et al.* (1998) and Panagopoulos (1998). It is also possible to get a closed-loop simulation of the current controller and process model.
- Outliers and other disturbances may be removed from the experimental data interactively.

Examples

To conclude this section, a few examples will be given that demonstrates some features of the tool.

EXAMPLE 3.1—TEMPERATURE CONTROL LOOP

This example is taken from Panagopoulos *et al.* (2000). Figure 3.10 shows the graphical user interface of the tool when fitting a model to data from

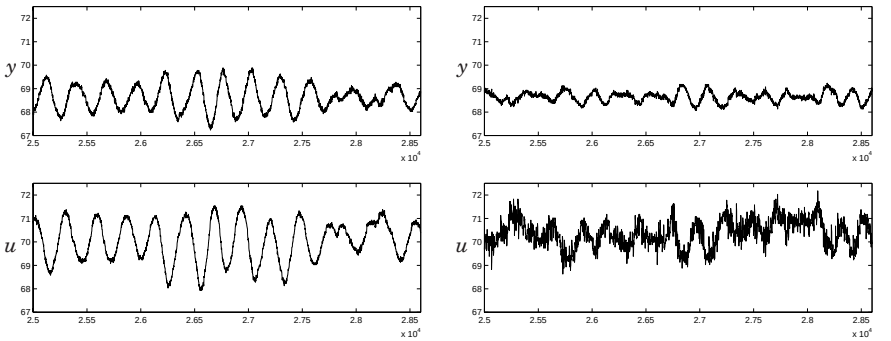


Figure 3.11 Comparison between the old PI controller (left) and the new PID controller (right) for the temperature control loop in Example 3.1.

a temperature control loop in a Swedish paper mill. A second order model was chosen in order to capture some of the higher order dynamics which is often present in temperature control loops. The dotted parts of the experimental data in the figure correspond to load disturbances during the experiment, and have thus been deselected.

After least squares fit, the transfer function

$$G(s) = \frac{1.341e^{-12.9s}}{(43.3s+1)^2} \tag{3.12}$$

was obtained. This model was used for calculating a PID controller in order to improve the control compared to the existing PI controller with parameters $K = 0.80$ and $T_i = 60$. Especially the large variations of the output during regulatory control seen to the left in Figure 3.11 were undesirable. They were caused by a combination of periodic load disturbances, valve hysteresis and detuned controller. The new PID controller with parameters $K = 1.35$, $T_i = 40$ and $T_d = 19$ reduced the variance of the output drastically. A drawback with the new controller was that the increased gain and the derivative action introduced more amplification of the measurement noise. This was reduced by inserting an additional low-pass filter of the measurement signal.

The process model and new controller parameter settings were found with very little effort. This example clearly shows the usefulness of the tool when designing new controllers for a process. □

EXAMPLE 3.2—PRESSURE CONTROL LOOP

The data in Figure 3.12 is collected from a pressure control loop in the same mill as in Example 3.1. Since the plant is moving fast at first and

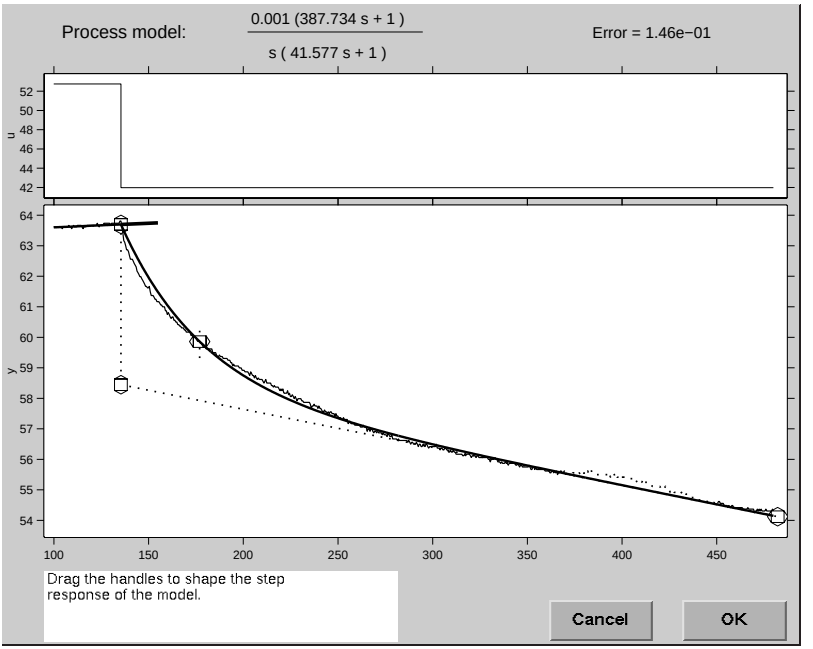


Figure 3.12 Fitting an integrating process model with one pole and one zero to data from a pressure control loop in Example 3.2.

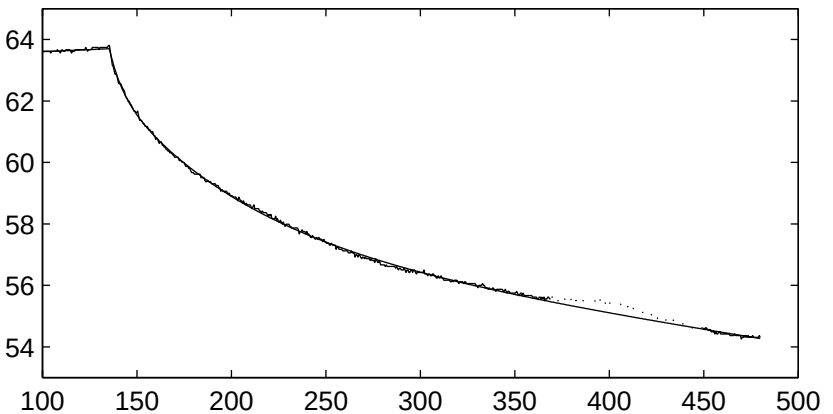


Figure 3.13 Data from the pressure control loop and output from an integrating process model with two poles and two zeros.

then starts to drift, it seems reasonable to model the plant as a sum of an integrator and a first order system. Least squares fitting of the data to this model structure gives:

$$G_1(s) = \frac{0.0014\,(387.7s + 1)}{s\,(41.6s + 1)}$$

The root mean square error between the data and the model output is 0.146. As seen from the figure, the model does not give a perfect fit. If the model structure is augmented with another first order system, the least squares fit will result in the transfer function

$$G_2(s) \quad = \quad \frac{0.0011\,(573.5s + 1)\,(18.4s + 1)}{s\,(71.9s + 1)\,(6.64s + 1)}$$

with the root mean square error reduced to 0.048. Figure 3.13 shows the model output from $G_2(s)$ together with the experimental data. The graphical user interface does *not* support transfer functions with this structure, so the optimization has been carried out using the more general method described in Section 3.4. □

EXAMPLE 3.3—NON-LINEAR PROCESS

When the process is non-linear one may still be interested in a linear model close to the operating point. The tool lets the user define different models in different regions by selecting and deselecting data points to optimize over. The double tank process in Figure 3.14 has free outflow from the tanks, which makes it proportional to the square root of the tank level. A process model is thus given by

$$\begin{cases} \dot{h}_1 &= -\alpha_1\sqrt{h_1} + \beta u \\ \dot{h}_2 &= \alpha_1\sqrt{h_1} - \alpha_2\sqrt{h_2} \end{cases} \tag{3.13}$$

where α_1, α_2 and β are constants, u is the input flow, and h_1 and h_2 are the levels of the upper and lower tank, respectively. Figures 3.15 and 3.16 show experimental data from step responses from u to h_2.

In Figure 3.15 the whole data set is considered at once, with poor result. In Figure 3.16 one step response at a time is identified. A model of second order has been matched to each step response. Note that both the gain and the time constant differs between the two responses. This can be seen if Equation (3.13) is linearized.

Next section will describe how a process non-linearity can be identified together with a linear transfer function. □

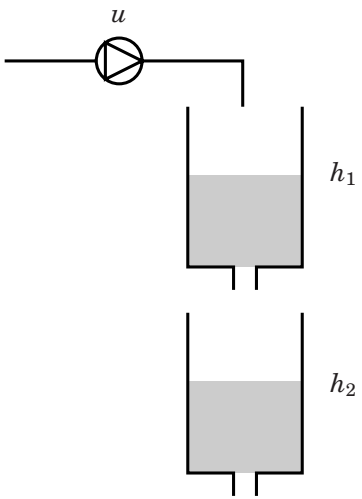


Figure 3.14 Sketch of the double tank process in Example 3.3.

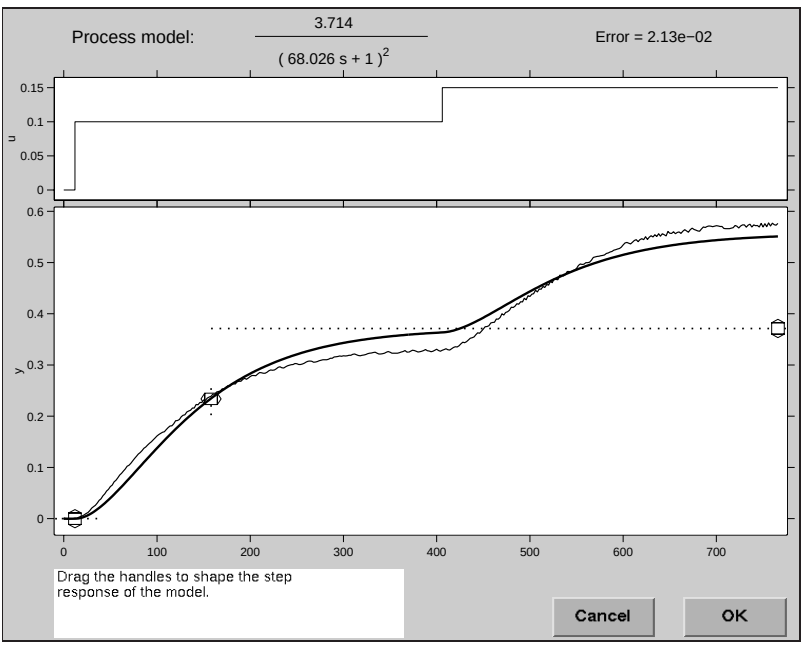


Figure 3.15 Least squares fit of both step responses for the double tank process.

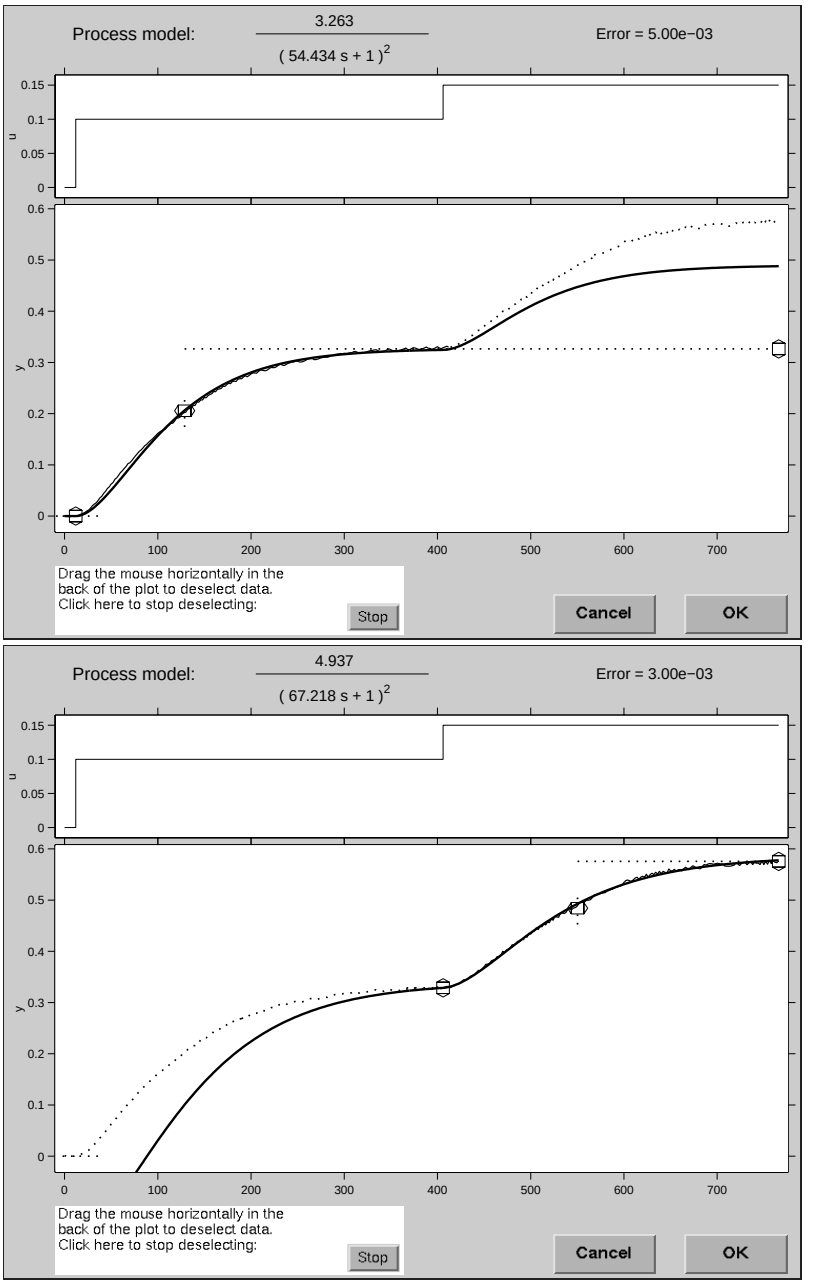


Figure 3.16 Separate least squares fit of the step responses for the double tank process. The dotted data in each plot is discarded.

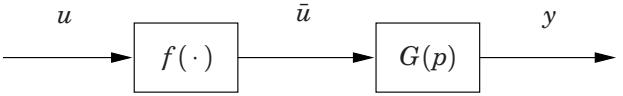


Figure 3.17 A Hammerstein non-linear model.

3.3 Identification of Process Non-linearities

A vast majority of the processes in process industry are non-linear in one way or another. Non-linearities associated with friction and hysteresis are discussed in Section 2.3. These should be regarded as process malfunctions which, hopefully, may be corrected by valve maintenance. However, even with perfect actuators and sensors, the process model usually varies over the operating range.

Process identification of a general non-linear dynamical system is a very complex problem. It is therefore necessary to restrict the complexity of the model, and use tailor-made methods for the chosen model structure. A common restriction is to consider only models with linear dynamics with static non-linearities on the input (Hammerstein models) and/or the output (Wiener models). Several approaches exist to identify both the dynamics and the non-linearities, for example the iterative output error method in Narendra and Gallman (1966), subspace methods in Haverkamp *et al.* (1998), and correlation techniques in Billings and Fakhouri (1979). The non-linearity is often a polynomial of fixed order, but it may also be represented as, *e.g.*, a neural network, see Schram *et al.* (1997), or a series expansion, see Pawlak (1991).

The use of static non-linearities may seem a severe restriction. However, the most dominating non-linearity is often a non-linear static characteristic from process input to process output. As an example, the steady-state flow through a control valve is often not proportional to the control input.

Hammerstein models

A Hammerstein model may be described by

$$y(t) = G(p)\, f(u(t)) \tag{3.14}$$

where $G(p)$ is a linear system and $f(\cdot)$ is an arbitrary non-linear function, see Figure 3.17. In order to get a unique representation of the Hammerstein model, the static gain of the linear part is set to 1. The Hammerstein model has the property that two step responses taken at different input levels will only vary in gain, but have the same shape.

The Hammerstein model seems reasonable to use when the dominant non-linearity is located near the actuator end of the plant, for example a non-linear valve characteristic. Even when this is not the case, the Hammerstein model will at least pick up gain variations in the operating range. The gain non-linearity is the easiest one to compensate for, and in most cases the most important one as well. When $f(\cdot)$ has been identified, it is possible to include its inverse in the control system. This way, the process will be linear as seen by the controller, which may then use fixed gains.

Another reason for considering Hammerstein models is that we are mainly doing step response analysis. This makes it natural to represent the non-linearity as pairs $(u_k, \bar{u}_k)$, for each value u_k of the input signal in the current data set. The non-linear function f is thus just defined by

$$\bar{u}_k = f(u_k), \quad k = 1 \dots N_u \tag{3.15}$$

where N_u typically is the number of values u takes on in the data set. This way there is no need for any functional parameterization of the non-linearity. One drawback is that the number of parameters $\bar{u}_k$ to find increases as the number of input levels increases. This should not be a severe drawback, since typical data series do not include too many step responses.

One might want to evaluate $f(u)$ between the values of u_k, *e.g.*, for doing gain scheduling. If the non-linearity is given by (3.15), this is naturally done with for example linear or spline interpolation. When $f(u)$ is expressed using basis functions, this is not necessary, since f may be evaluated at any point. On the other hand, with only a few levels of the control signal in the data series, $f(u)$ may vary a lot between the u_k:s.

In the tool, you may graphically edit the non-linearity by dragging the values of $\bar{u}_k$, see Figure 3.18. u is on the horizontal axis and $\bar{u}$ is on the vertical axis. Note that the values of u_k are fixed, and only the values of $\bar{u}_k$ can be altered.

Since the static gain of the linear dynamics is fixed to 1, the values of $\bar{u}_k$ equal the stationary levels of the process output y in the data series. Thus, when a level $\bar{u}_k$ is dragged, the gain for all step responses taken from or to the level u_k will change, but the rest will remain the same.

The tool also allows least squares fit of a Hammerstein model to data. The linear dynamics part is parameterized as in the previous section, except that the gain K is now fixed to 1:

$$G(s) = \frac{\beta T s + 1}{(sT + 1)^n} e^{-sL} \tag{3.16}$$

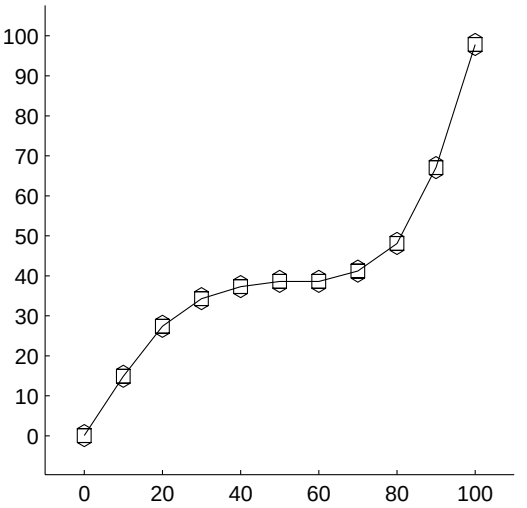


Figure 3.18 A graphical representation of the non-linearity $\bar{u} = f(u)$ for a Hammerstein model, and $y = f(\bar{y})$ for a Wiener model. The handles may be moved in vertical direction.

The order n must be chosen, the other parameters may vary. With the non-linearity given by (3.15), the least squares problem will be to find the parameter vector

$$[T \quad L \quad \beta \quad \bar{u}_1 \quad \dots \quad \bar{u}_{N_u}]^T \tag{3.17}$$

which minimizes

$$\sum_t \left(y(t) - G(p) f(u(t))\right)^2 \tag{3.18}$$

Wiener models

A Wiener model may be described by

$$y(t) = f(G(p) u(t)) \tag{3.19}$$

where $G(p)$ is a linear system and $f(\cdot)$ is an arbitrary non-linear function, see Figure 3.19. To get a unique representation, the static gain of the linear part is again set to 1. As opposed to a Hammerstein model, the Wiener model will have step responses where the shape as well as the

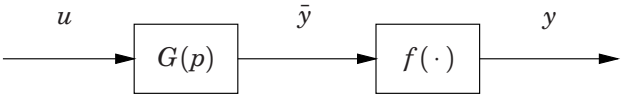


Figure 3.19 A Wiener non-linear model.

gain may vary for different input levels. To see this clearly, consider for example the non-linearity

$$y = f(\bar{y}) = \text{sign}(\bar{y})$$

with a step response crossing the zero level. Since $\bar{y}$ varies continuously, the time where it changes sign depends heavily on the initial and final levels of the input signal.

The definition of the non-linearity is analogous to the one for the Hammerstein model. It is defined by pairs

$$y_k = f(\bar{y}_k), \quad k = 1 \ldots N_y \tag{3.20}$$

with linear or spline interpolation for values between the $\bar{y}_k$:s, see Figure 3.18. The big difference from the Hammerstein case is that there are no obvious ways to select the values of $\bar{y}_k$. A simplistic approach would then be to spread the $\bar{y}_k$:s evenly over the current range. There is however no guarantee that this will give the best representation of the non-linearity. This is further discussed in Example 3.5 below.

Other parameterizations

As pointed out in Haber and Unbehauen (1990), static non-linearities plus linear transfer functions is just one class of non-linear dynamical system. Wiener and Hammerstein models have been chosen here for the sake of simplicity. By allowing the process gain to vary over the operating range, it is possible to treat many problems that occur in practice. However, there are of course many other cases with more complex behavior, where other parameterizations would be required. For example, it is very common that the dynamic behavior depends on an auxiliary signal such as the current production level. This may be solved by letting the parameters in the SISO transfer function depend on this auxiliary variable. An alternative would be to use a MIMO model where all the interesting variables are included. This would however require much more modeling effort.

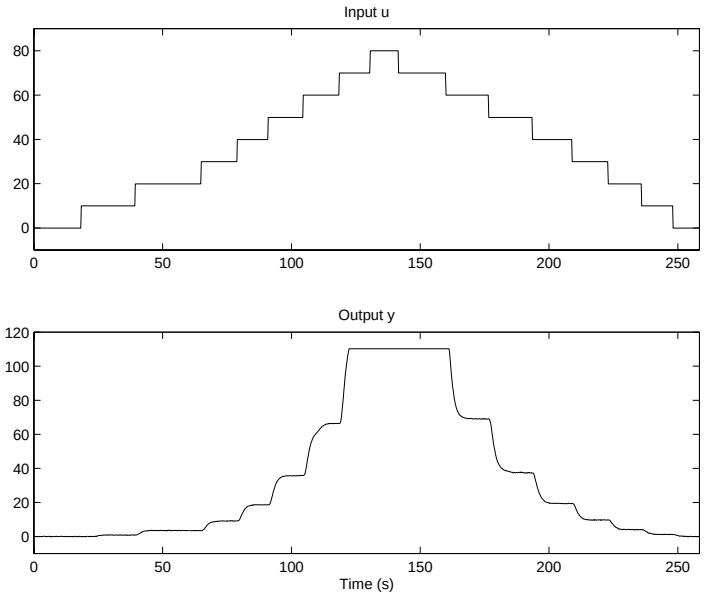


Figure 3.20 A series of step responses from a flow control loop.

Examples

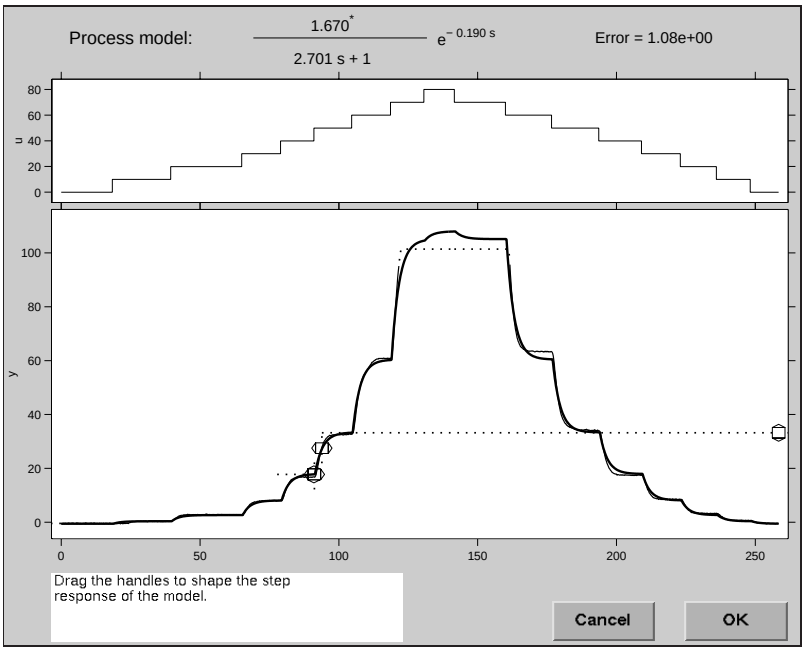
To conclude this section, two examples will be given, where Hammerstein and Wiener models have been used to describe non-linear systems.

EXAMPLE 3.4—FLOW CONTROL LOOP

The data set in Figure 3.20 is from a flow control loop in the same paper mill as in Examples 3.1 and 3.2. u is the control signal sent to the control valve. The valve is closed at $u = 0$ and fully open at $u = 100$. The flow is measured by a magnetic flow transmitter. The flat part of the curve in the lower plot is due to saturation of the flow measurements. These data points will thus be neglected when doing the least squares fit.

The sequence of step responses indicates very large gain variations over the operating range. However, the dynamics are fairly constant. It is thus reasonable to use a Hammerstein model for this example. It also makes sense to model the linear part using a delayed first order model.

The lower plot in Figure 3.21 shows the non-linear function $\bar{u} = f(u)$ after least squares fit of the non-linear model to data. The value of f at $u \approx 80$ should be neglected, since it does not correspond to any data used in the optimization. There may be large errors in $f(70)$ as well, since only the transients have been used.



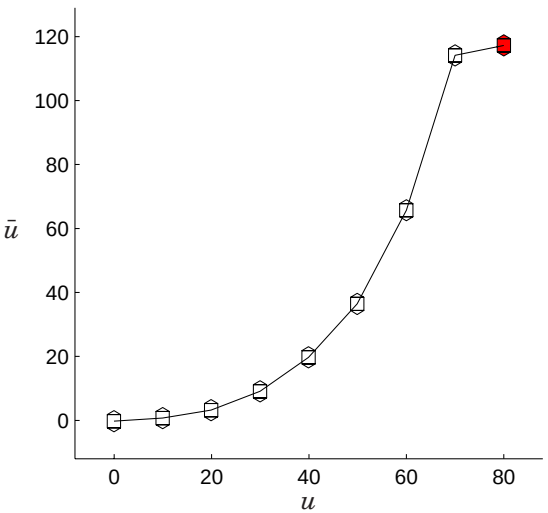


Figure 3.21 Identification of non-linear model to data from flow control loop. The non-linear function $\bar{u} = f(u)$ is shown in the lower plot. $u = 80$ corresponds to the saturated data, and was not used for optimization.

With this non-linearity and the linear model

$$G(s) = \frac{1}{(2.77s + 1)} e^{-0.20s}$$

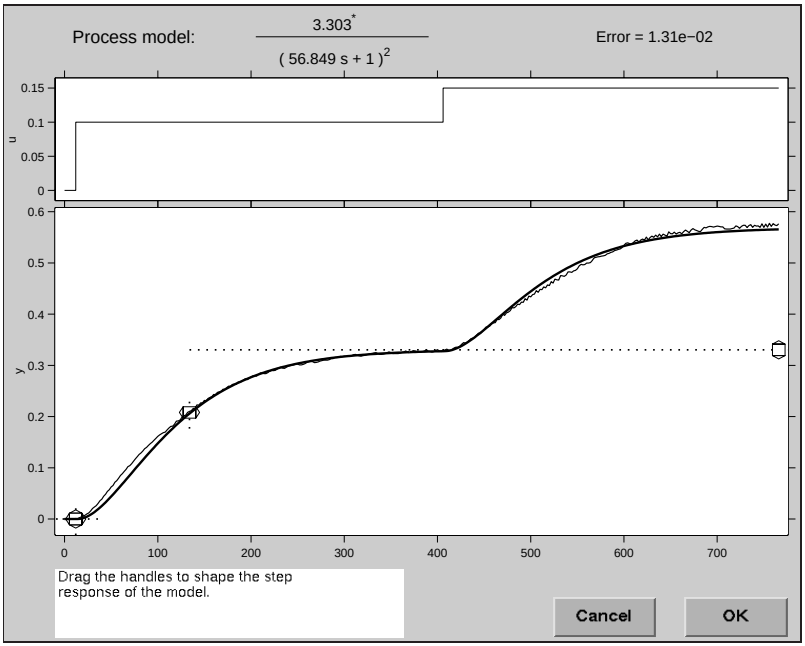
the RMS error becomes 1.14, see Figure 3.21. The value for the gain is 1.66, which is the slope of f at the levels corresponding to the currently active step response. The dashed data from $t \approx 120$ s to $t \approx 160$ s have not been used in the optimization.

In this example, there is clearly no linear model which is valid outside a very small operating region. The process gain varies substantially across the normal operation range. It is thus crucial to take the non-linearity into consideration in order to achieve reasonable behavior when using a process model for control design or process supervision. □

EXAMPLE 3.5—DOUBLE TANK PROCESS

The double tank process discussed in Example 3.3 may also be identified using the non-linear modeling tool. First, a Hammerstein model is used, with results shown in Figure 3.22. When the tank model given in Equation (3.13) is linearized, both the gain and the time constants will be a function of the operating point. Consequently, the double tank process can clearly not be captured by a Hammerstein model. This causes the model fitting less accurate than in Example 3.3 where each step was identified separately. On the other hand, it is advantageous to be able to identify a complete model at once. Whether the loss of model accuracy is important or not depends on what the model will be used for.

If instead a Wiener model is used, it is possible to achieve a much better fit, see Figure 3.23. The non-linearity used in this example consists of five pairs $(\bar{y}_k, y_k)$ with spline interpolation between them. In fact, by introducing more and more complex non-linearities, it is possible to achieve arbitrarily close fit to the model. The extreme would be to have $G(s) = 1/s$, making $\bar{y}(t)$ a linear function of time, and to have a non-linear function which is exactly the data points $y(t)$. Figure 3.24 shows this idea, where the dynamic behavior is captured by the (static) non-linear function instead of the linear transfer function. This is of course not a good model, since neither the static nor the dynamic behavior would be described well for other input signals. To avoid this kind of deteriorated behavior, it would be desirable to have step responses in both directions between many levels in the region of interest. The bad behavior is also avoided if the non-linear function is simple enough. In fact, since it is supposed to take care of the static gain variations only, the non-linear function should not contain much more information than the steady-state levels used in the experiments. □



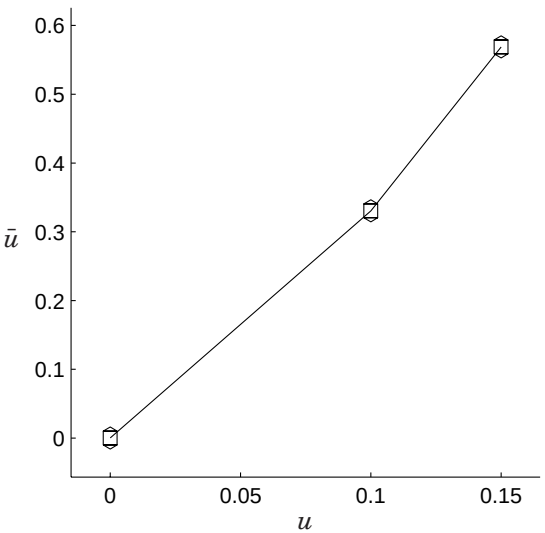
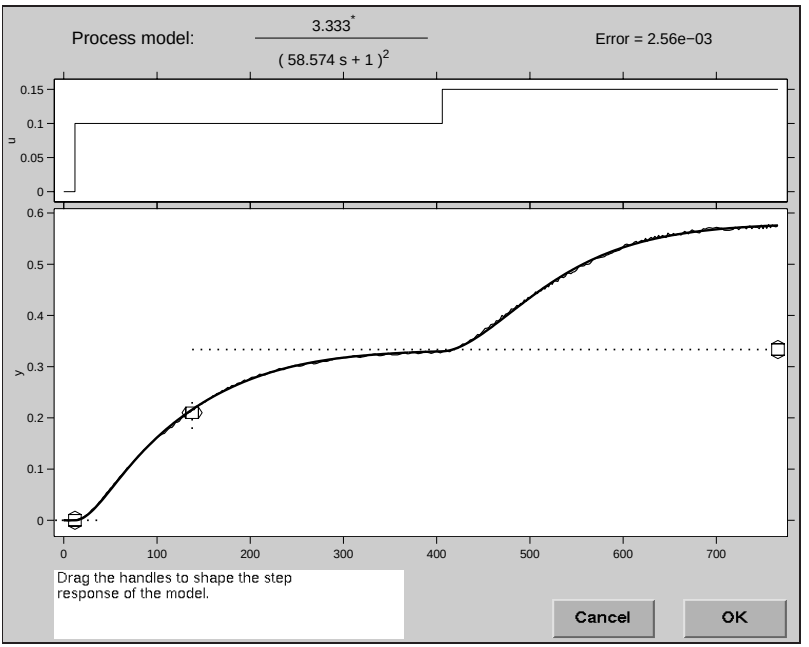


Figure 3.22 Identification of a Hammerstein model for the double tank process in Example 3.5.



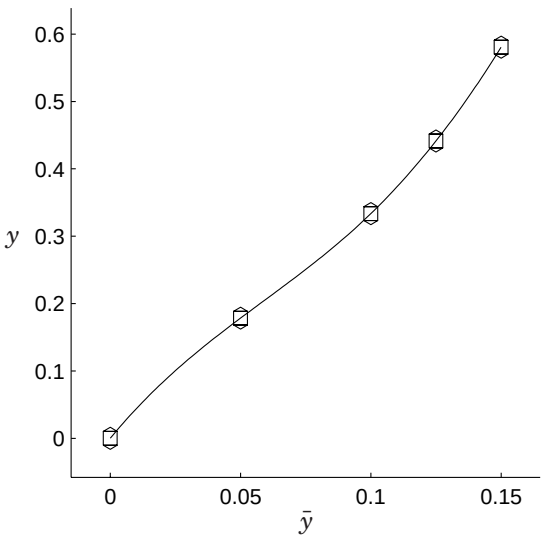
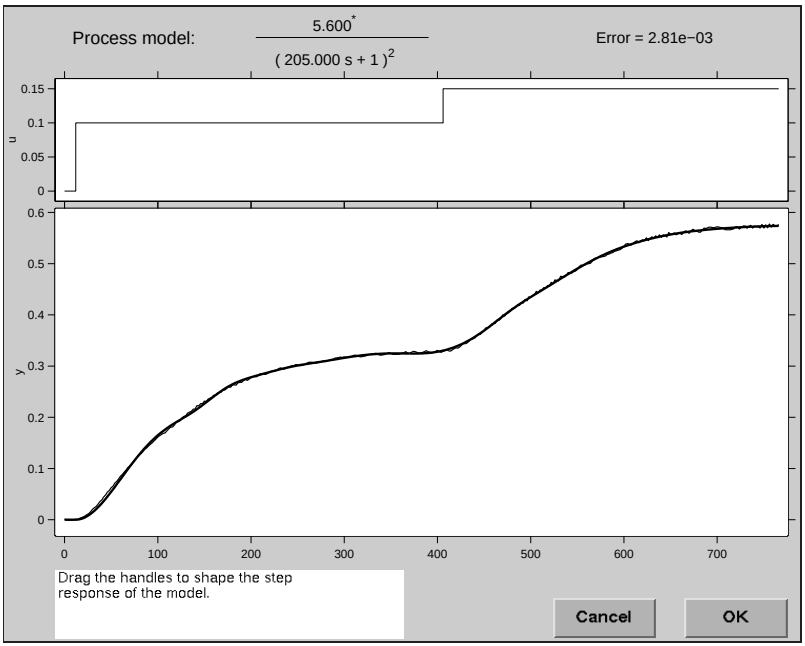


Figure 3.23 Identification of a Wiener model for the double tank process.



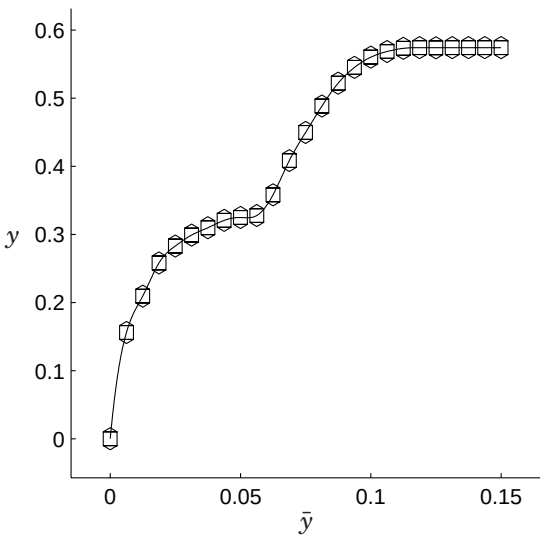


Figure 3.24 Example of a deteriorated Wiener model for the double tank process using over-parameterized non-linearity. The non-linearity does not reflect the static characteristics of the process.

3.4 Least Squares Fit of Step Response Data

Along with the graphical manipulation described in the previous sections, the tool allows least squares optimization of the model to the step response data. In order to do this we must have an analytical expression for the step response of the selected model structure. The interactive tool described above has a number of pre-defined model structures that can be fitted to data. This implies that the step response for each model structure may be calculated in advance.

Sometimes, you might want some other model parameterization which is not supported. This may be the case when you have many poles or zeros, or when you have a partly known model structure, for example when a physical parameter shows up in different coefficients in the transfer function.

The approach taken here is described by the following steps:

1. Collect data $y(t)$ corresponding to a piecewise constant input signal $u(t)$ for the time points t.
2. Select a desired model structure in terms of a linear transfer function $G(s,\theta)$, where θ is the vector of unknown parameters. The numerator and denominator should be formulated as products of first and second order polynomials. It is also possible to include a known or unknown delay in the model. The unknown parameters may show up non-linearly, and in any number of positions in the coefficients of the transfer function.
3. Formulate inequality constraints on the parameters, for example to ensure stability.
4. Calculate the step response $S(t,\theta)$ *symbolically* in, *e.g.*, Maple for the selected $G(s)$. This is straightforward when $G(s,\theta)$ only contains factors of first and second order. Some care must, however, be taken to handle multiple poles and/or zeros.
5. Write a function for, *e.g.*, MATLAB, which computes the step response *numerically* for a specific vector θ, supplied as an input argument. Rewrite the input signal as a sum of delayed steps: $u(t) = c_1 H(t - d_1) + \ldots + c_N H(t - d_N)$, where $H(t)$ is the Heaviside unit step function. Compute the model output as $y_m(t,\theta) = c_1 S(t - d_1,\theta) + \ldots + c_N S(t - d_N,\theta)$.
6. Use standard optimization methods to obtain the least squares estimate of the parameters

$$\hat{\theta} = \arg\min_{\theta} V(\theta) = \arg\min_{\theta} \sum_{t} (y(t) - y_m(t,\theta))^2 \qquad (3.21)$$

such that the constraints in step 3 are fulfilled.

The steps 4–6 are done automatically by the optimization tool. The stability conditions in step 3 may also be obtained automatically. Computer advice and support can be given for the remaining steps.

Remark 1: In order to solve the optimization problem (3.21) more efficiently, you may also calculate the gradient vector

$$\frac{\partial V}{\partial \theta} = -2\sum_t \big(y(t) - y_m(t,\theta)\big)\frac{\partial}{\partial \theta} y_m(t,\theta) = -2\sum_t \big(y(t) - y_m(t,\theta)\big) \cdot \left(c_1 \frac{\partial}{\partial \theta} S(t-d_1,\theta) + \ldots + c_N \frac{\partial}{\partial \theta} S(t-d_N,\theta)\right)$$

in steps 4 and 5. The optimization criterion (3.21) will typically have several local minima, since it is non-linear in the parameters θ. It is thus necessary to supply the optimization method with reasonable initial values in order to find the correct local minimum. These initial values may be obtained from the GUI described in the previous section.

Remark 2: The reason for allowing only first and second order factors in the denominator is of course that these give a simple analytic expression for the step response. Second order systems may have either real or complex poles, which leads to different expressions for the step response. However, this is no problem since MATLAB handles complex arguments to the trigonometric and hyperbolic functions. Higher order denominator factors could be handled by calculating the step response *numerically* upon each iteration. However, this is very time consuming, especially if the parameter gradients should also be calculated.

Remark 3: $V(\theta)$ can be minimized using the MATLAB function `constr` from Optimization Toolbox. The skeleton of the loss function to minimize is shown in Listing 3.1. `step_fcn` should contain a string with a function name for calculating the unit step response for a certain model structure. Any of the parameters in the vector `big_theta` may be assigned minimum and maximum values. Additional constraints may also be added as a string `cstr` that should be evaluated to a vector with negative elements.

The procedure is illustrated by a few examples.

EXAMPLE 3.6—SIMPLE EXAMPLE.
Consider again the simple model structure

$$G(s) = \frac{K}{sT+1}\, e^{-sL} \tag{3.22}$$

```
function [V,constraints] = ...
    loss_fcn(big_theta,y_data,u,t,step_fcn,cstr)
%
theta = big_theta(1:end-1);  % model parameters
y_0 = big_theta(end);        % initial output level
y_step = feval(step_fcn,theta,t);
                             % unit step response
y_m = stepsum(y_step,u,t) + y_0;
                             % sum of delayed step responses
y_diff = y_data - y_m;
V = y_diff'*y_diff;          % sum of squares
constraints = eval(cstr)     % additional constraints
```

Listing 3.1 A MATLAB function for calculating the loss function in (3.21).

```
function y_step = lag1d_step(theta,t)
%
K = theta(1);
T = theta(2);
L = theta(3);
y_step = K*(1-exp(-(t-L)/T));
y_step(find(t<L))=0;
```

Listing 3.2 A MATLAB function for calculating the step response of the model (3.22).

Its step response is calculated by the MATLAB function in Listing 3.2. This function can be automatically generated using *e.g.* Maple or the Symbolic Toolbox in MATLAB. □

EXAMPLE 3.7—PARTIALLY KNOWN MODEL STRUCTURE.
For open-loop unstable plants, it is mostly necessary to do closed-loop identification. Suppose that the plant transfer function is parameterized as

$$G_p(s) = \frac{b_0}{s^2 + a_1 s + a_2}$$

with a_1 and/or a_2 negative. It can be stabilized by an ideal PD controller

$$G_r(s) = K\ (1 + sT_d)$$

giving a closed-loop transfer function

$$G_c(s) = \frac{b_0\, K\ (1 + sT_d)}{s^2 + a_1 s + a_2 + b_0\, K\ (1 + sT_d)}$$

from the reference value to the output. If the controller parameters K and T_d are known, you may solve for $\theta = [b_0\, a_1\, a_2]^T$ directly from closed-loop step response data if the model structure $G_c(s)$ is used. □

As hinted in step 4 above, there will be problems with singularities when you are close to multiple poles. To realize this, consider the following example:

EXAMPLE 3.8—MULTIPLE POLE SINGULARITY

$$G_1(s) = \frac{ab}{(s+a)(s+b)}$$

(a, $b > 0$) with the corresponding step response

$$y_1(t) = 1 - \frac{be^{-at}}{b-a} - \frac{ae^{-bt}}{a-b}.$$

You will run into numerical problems when b is close to a. This can happen either when the loss function actually has a local minimum at $a \approx b$, or accidentally during the optimization for certain initial parameter values. One solution is simply to add a constraint, for example

$$(a-b)^2 > \varepsilon^2$$

with a small ε. It is possible to use a tool like Maple to find singularities like the one above and add the constraints automatically. ε is chosen just below the desired parameter accuracy. □

Properties of the identification procedure

The process identification tool described in this chapter is not intended to replace the use of traditional identification tools. Instead, it offers a fast way of obtaining a coarse process model using available step response data. The use of a least squares output error criterion is by no means crucial. Many other methods exist for identifying a delayed first order model from a single step response, for example methods based on area calculations (Åström and Hägglund (1995)) and instrumental variable techniques (Bi *et al.* (1999)). The least squares output error method has been chosen since it is intuitive to interpret the result. More advanced system identification methods and model structures may have to be used if the demands on control performance is high. Still, the tool may be useful as a "pre-modeler" in order to *,e.g.*, get the approximate timing of the process and to detect non-linear behavior.

The proposed method for transient response analysis differs from traditional identification using, say, an ARMAX model in a number of ways:

- The process model is formulated in continuous time instead of discrete time.
- The transfer function may contain known process model structure, whereas ARMAX uses black box modeling. It is, for example, difficult to force all poles of a high order process to be real using ARMAX modeling.
- The time delay may be identified, while in ARMAX modeling, it is assumed to be known. If it is not known, you can identify and compare models with different time delays. An alternative is to identify many numerator coefficients, where the first ones will be close to zero.
- It is not crucial to decide suitable sampling periods, even irregular sampling is feasible.
- The proposed method minimizes the output error, whereas ARMAX methods minimize the prediction error. This tends to give closer output following when the models are under-parameterized. There exist discrete-time output error methods that can be used instead of ARMAX, should this be a problem.
- The loss function (3.21) typically has many local minima, so reasonable initial estimates are crucial for convergence. ARMAX identification is much less sensitive to this.
- Minimizing (3.21) gives a constrained non-linear optimization problem, typically solved with a sequential quadratic programming algorithm. The computational effort is comparable to prediction error and output error identification of ARMAX model structure.
- It is difficult to say anything about parameter consistency with the proposed method.

3.5 Summary and Concluding Remarks

An interactive tool for simple system identification has been presented. The graphical user interface handles a few process parameterizations as well as static input and output non-linearities. The main focus has been on providing an intuitive user interface where the step response of the process model can be edited by hand using graphical handles. This approach gives a quick way to get a simple model of process dynamics and non-linearities. In many cases this information is sufficient for designing a PI(D) controller, possibly with gain scheduling.

A wisely designed tool for process analysis should be possible to use by people with varying background. Control loops that are well described by delayed first order models are normally understood by several people at a plant, and may thus be examined and tuned by these people. For loops with more complex dynamics of higher order, or with zeros and non-linearities, it may be necessary to use external expertise. Open-loop data may then be sent to the central office of a vendor or a process control company with highly qualified control engineers serving as support. Many problems may then be easily sorted out, for example

- Static, as well as dynamic non-linearities.
- Bad controller tuning.
- Time-varying dynamics.
- Unrealistic demands on control performance.

If the tool includes good interactive help functions, it could instead be used by process operators and instrument engineers to solve more advanced problems. The tool would then serve as a means of raising the education level and increase the awareness of control and dynamic behavior.

4

Frequency Domain Identification and Design

In Chapter 3, an estimate of the transfer function of the process was derived from step response experiments. The model may be used for controller design, for example using the methods presented in Panagopoulos (1998). The design procedure was demonstrated in an example taken from the pulp and paper industry. There is unfortunately no guarantee that process models derived as in Chapter 3 are very accurate at the frequencies that are relevant for control. Typically, frequencies around the ultimate frequency ω_u are most important in the control design, whereas step response experiments instead contain most of their information at lower frequencies. It would of course be possible to apply a PRBS-like input signal which excites the relevant frequencies. The response can then be analyzed, for example using the tool in the previous chapter. To do this, the ultimate frequency must first be identified. Furthermore, it may be cumbersome to find a suitable model structure which captures the important part of the frequency response. It thus seems difficult to make an automated tuning procedure based on this approach.

An alternative method will be introduced in this chapter. The key idea is to automatically provide excitation which is well suited for identification for control design. The signal is produced by relay feedback, discussed in Section 4.1. The process response may be analyzed directly using frequency domain estimation. This is described in Section 4.2. Design of PI and PID controllers based on the estimate is discussed in Sections 4.3 and 4.4, respectively.

4.1 Relay Feedback

Relay feedback has proven to be efficient for tuning PI and PID controllers with a minimum amount of *a priori* known process data. The basic idea is to pick one point on the Nyquist curve, and use that for calculating controller parameters. A standard relay feedback experiment gives a point close to the ultimate frequency. When the point has been identified, tuning formulas similar to the Ziegler-Nichols ultimate gain method are used. A review of the methodology may be found in Åström and Hägglund (1995).

The design methods that use only one point on the Nyquist curve work well for a large number of systems, but in many cases they are too simplistic. The performance and/or robustness of the closed-loop system may improve significantly with more process knowledge. One method aiming at this is the Kappa-Tau method, see Åström and Hägglund (1995). In the frequency domain version of this method, both the static gain and the ultimate point are used.

The behavior may be improved further if the full transfer function is known. The design methods in Åström *et al.* (1998) and Panagopoulos (1998) find the PI or PID controller which minimizes the integrated error after a step load disturbance on the plant input. In order to ensure good performance and robustness, additional constraints are put on the maximum value loop M_s of the sensitivity function.

The design methods typically use the frequencies where the phase lag of the plant is between -90° and -240° for checking the sensitivity constraint. Since the transfer function is normally not known, an estimate which is accurate at the interesting frequencies must be used. In traditional relay feedback, the process information is concentrated around the frequency of the limit cycle, and higher harmonics. Here, we suggest a modified relay experiment which excites more relevant frequencies.

A relay with hysteresis

Relay feedback is a common way of forcing an otherwise stable plant to oscillate in a controlled way. Many properties of relay feedback can be understood by describing function analysis. If the input to the relay is a sinusoid with amplitude a, the output is a square wave. The describing function $N(a)$ gives the relation between the amplitudes and the phases of the input and the first harmonic of the output. The intersection between the Nyquist curve of the process and the negative reciprocal of the describing function for the relay will provide approximations of the amplitude and frequency of possible limit cycles. Consequently, an oscillation that occurs under relay feedback provides an estimate of one point on the Nyquist curve of the process. It is important to remember that this estimate is only approximate. Details on describing function analysis are

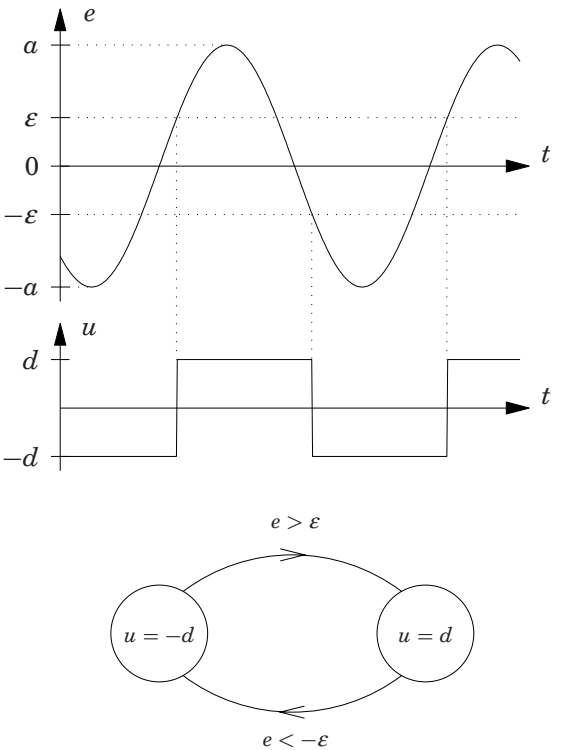


Figure 4.1 Input e and output u for a relay with amplitude $d > 0$ and hysteresis $\varepsilon > 0$. The automaton describes the behavior.

found in textbooks on non-linear control, see for example Atherton (1975) and Khalil (1992).

Figure 4.1 shows the response of a relay with amplitude $d > 0$ and hysteresis $\varepsilon > 0$ to a sinusoid. The behavior is described by the automaton in the same figure. The describing function for the relay is given by

$$N(a) = \frac{4d}{\pi a}\left(\sqrt{1-\left(\frac{\varepsilon}{a}\right)^2} - i\frac{\varepsilon}{a}\right) \tag{4.1}$$

Its negative inverse forms a straight line with negative real part and constant imaginary part $-\pi\varepsilon/(4d)$. Different values of the ratio ε/d give rise to different intersections with the Nyquist curve of the process and thus different frequencies and amplitudes of the limit cycles. The main reason for introducing hysteresis in the relay is traditionally to increase

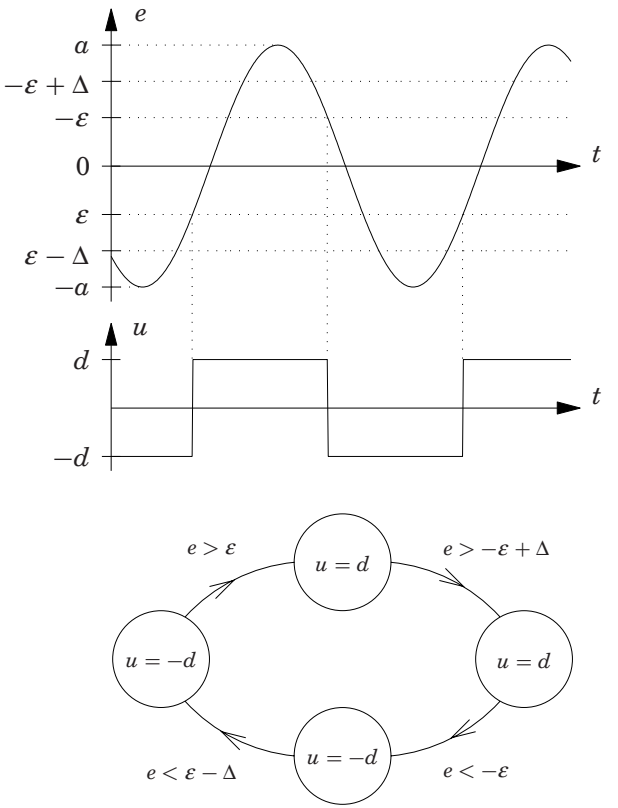


Figure 4.2 Input e and output u for a relay with amplitude $d > 0$ and hysteresis $\varepsilon < 0$. The automaton describes the behavior. Note that the automaton works for $\varepsilon > 0$ as well.

the robustness to noise. A noisy input to the relay will cross the zero level repeatedly, causing undesired chattering of the ideal relay output. It is therefore usually recommended that ε is chosen larger than the amplitude of the measurement noise. As a consequence the relay will stop switching if the input amplitude is less than ε. An alternative method that has been used for avoiding chattering is to neglect relay crossings for a certain time after each switch.

Negative hysteresis

As pointed out in Holmberg (1991), it is possible to use negative values of both d and ε. The describing function may then be located in any of

the four quadrants. If $d > 0$ and $\varepsilon < 0$, $-1/N(a)$ is a straight line in the second quadrant. However, care must be taken when implementing this kind of relay. Since ε is negative, chattering due to noise may still occur. To overcome this, another parameter Δ is introduced. Figure 4.2 shows the behavior for this relay, both described by a time sequence and an automaton. Note that the automaton will describe the behavior also for $\varepsilon > 0$. The extra states compared to Figure 4.1 are required to make sure that the input is outside $\pm|\varepsilon|$ before switching again. If the input is noisy, the relay will not exhibit chattering as long as Δ is chosen larger than the peak-to-peak amplitude of the noise. For $\varepsilon \geq \Delta/2$ the behavior of this relay is equivalent to the one in Figure 4.1.

The formula for $N(a)$ still be given by Equation (4.1). However, since the relay will stop switching if the input amplitude is less than $a_{min} = \max(-\varepsilon + \Delta, \varepsilon)$, $N(a)$ will not be defined for $a < a_{min}$. Thus, the negative inverse of the describing function then starts in the point

$$-\frac{1}{N(a_{min})} = \begin{cases} -i\dfrac{\pi\varepsilon}{4d}, & \varepsilon \geq \Delta/2 \\ -\dfrac{\pi}{4d}\left(\sqrt{\Delta\,(\Delta - 2\varepsilon)} + i\varepsilon\right), & \varepsilon < \Delta/2 \end{cases} \tag{4.2}$$

For the case $\varepsilon < \Delta/2$ you may instead write

$$\begin{cases} \left|-\dfrac{1}{N(a_{min})}\right| = \dfrac{\pi(\Delta - \varepsilon)}{4d} \\ \arg\left(-\dfrac{1}{N(a_{min})}\right) = -\pi + \arcsin\dfrac{\varepsilon}{\Delta - \varepsilon} \end{cases} \tag{4.3}$$

The negative inverse of the describing function is plotted for a fixed value of Δ and different values of ε in Figure 4.3.

The benefit of using negative hysteresis will be demonstrated later in this chapter. The implementation of the relay using the automaton in Figure 4.2 is useful even if negative hysteresis is not needed. For example, it makes it possible to implement a relay with zero hysteresis that works properly also for noisy measurements.

As pointed out previously in this section, the relay with a fixed hysteresis provides most excitation at the fundamental frequency of the limit cycle. The higher harmonics are also be excited, but to a much lesser extent. This may not be sufficient for advanced control design methods, such as the ones in Panagopoulos (1998), which require knowledge of the frequency response over a larger interval.

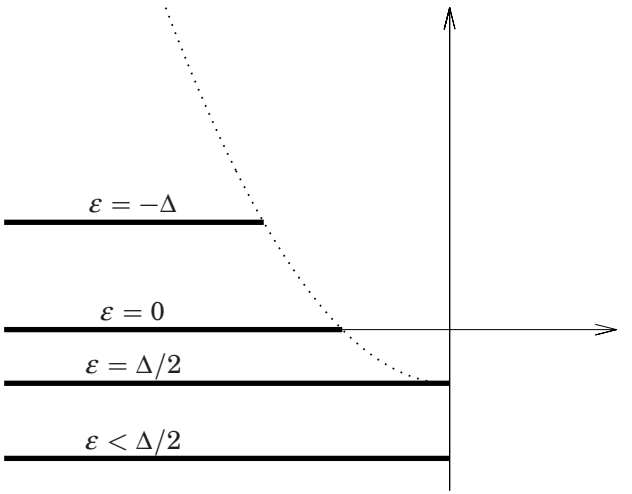


Figure 4.3 The negative inverse of the describing function for fixed Δ with different values of ε. The imaginary part is always $-\pi\varepsilon/(4d)$. The dotted line shows the starting point for different choices of ε.

Several authors have suggested modifications to the relay feedback method to obtain more information. Shen *et al.* (1996) use a biased relay to obtain both the critical point and the static gain. A parasitic relay can be used to estimate three points on the Nyquist curve, see Bi *et al.* (1997). A filter with variable phase shift may be cascaded with the plant to give rise to limit cycles with different frequencies, see for example Schei (1992). With this approach you must perform individual identification experiments for each of the interesting frequencies, since the plant put under relay feedback will be time-varying. This restriction is valid regardless if describing function analysis or any other method is used. Johansson (1997) suggests two different relay experiments, the ordinary one plus one with an integrator in series with the plant. The relay experiments, together with an estimate of the static gain, give the three points on the Nyquist curve with 0, -90° and -180° phase shift. There is then one third order model with one zero which matches these points exactly. However, the frequency response using this model may deviate substantially from the true one.

Time-varying hysteresis

A time-varying hysteresis will now be introduced as a means of achieving excitation for a larger frequency range than a fixed relay would give.

There are two basic experimental setups that can be used:

1. Let the hysteresis level be constant until a stationary limit cycle is obtained for each frequency point that should be estimated. This is required if describing function analysis should be used.

2. Change the hysteresis level upon each relay switch. There will then be no stationary limit cycle. The data series may instead be used for traditional system identification.

The main drawback of the first method is that the time required for the experiment will be very long if a large number of frequency points is needed. The second method will thus be explored.

Next, the experimental conditions such as sampling rates, interval of used hysteresis ε, and length of the experiment must be decided. This will be further discussed later Section 4.2. For now it is necessary to observe that the range of ε must be wide enough to provide excitation in the frequency interval that should be used in the control design. Typically, the plant phase shift should vary between approximately -90° and -240°. In order to ensure excitation at high frequencies, the relay with negative hysteresis may be needed.

4.2 Frequency Domain Identification

The data obtained from the relay experiment in Section 4.1 is used for identification. Two different classes of models can be used:

- Parametric models, where a model structure must be selected and identified, *e.g.* using the tool in Chapter 3, or some prediction error method. If the frequency response is needed, for example in control design, this is of course easily computed from the model.

- Non-parametric models, where for example the frequency response is estimated directly.

In this work, a non-parametric model of the frequency response will be used. The reason is that it may be difficult to select an appropriate model structure, particularly if this should be done automatically in an auto-tuning scheme.

The model used here is based on the *empirical transfer function estimate*, ETFE, of the process, see for example Ljung (1999). Let the Discrete Fourier Transform (DFT) of the output vector $y(k)$ and the input vector

$u(k)$ be given by

$$\begin{aligned} Y_N(\omega) &= \mathrm{DFT}(y(k)) = \frac{1}{\sqrt{N}} \sum_{k=1}^{N} y(k) e^{-i\omega k} \\ U_N(\omega) &= \mathrm{DFT}(u(k)) = \frac{1}{\sqrt{N}} \sum_{k=1}^{N} u(k) e^{-i\omega k} \end{aligned}$$

where N is the number of observations and the frequency ω is normalized to the interval $[-\pi, \pi]$. Note that the DFT is only defined at frequencies $\omega = 2\pi k/N$ with k integer. Furthermore, it is periodic with period N.

The ETFE is given by

$$\hat{\hat{G}}_N(e^{i\omega}) = \frac{Y_N(\omega)}{U_N(\omega)} \tag{4.4}$$

The estimate has the correct expected value for each frequency point. Unfortunately, it is not a smooth function, regardless of the number of data points used in the experiment. The reason for this is that the estimates at two neighboring frequencies have approximately the same variance, but are asymptotically uncorrelated. To overcome this, the Fourier transforms can be smoothed by using a convolution window function $W_\gamma(\omega)$ before forming the ETFE. The transfer function estimate is then given by

$$\begin{aligned} \hat{G}_N(e^{i\omega}) &= \frac{\int_{-\pi}^{\pi} W_\gamma(\xi - \omega) \left|U_N(\xi)\right|^2 \hat{\hat{G}}_N(e^{i\xi}) d\xi}{\int_{-\pi}^{\pi} W_\gamma(\xi - \omega) \left|U_N(\xi)\right|^2 d\xi} \\ &= \frac{\int_{-\pi}^{\pi} W_\gamma(\xi - \omega) Y_N(\xi) \overline{U_N(\xi)} d\xi}{\int_{-\pi}^{\pi} W_\gamma(\xi - \omega) \left|U_N(\xi)\right|^2 d\xi} \end{aligned} \tag{4.5}$$

where γ is a shape parameter, related to the length of the window. More precisely, γ is the length of the *lag* window $w_\gamma(\tau)$, which is the inverse Fourier transform of $W_\gamma(\omega)$. This is a form of averaging and the variance of the estimates will be reduced. A bias is, however, also introduced since the averaging is performed over a range of frequencies. The length of the frequency window is a trade-off between bias and variance. Longer windows result in lower variance but more bias. This bias is small if the true $G(i\omega)$ is fairly constant over the window length.

The functions `spa` and `etfe` in the System Identification Toolbox for MATLAB both do essentially the desired estimation of the frequency response. The function `spa` is more suited for long frequency windows $W_\gamma(\omega)$ and `etfe` for shorter frequency windows. Both functions use Hamming

windows for smoothing the estimate; `spa` uses windows on the estimated covariance function, `etfe` operates directly in frequency domain, as in Equation (4.5).

Other authors have studied the use of DFT on relay feedback data from a relay with fixed hysteresis, see for example Wang *et al.* (1999). However, a time-varying hysteresis gives better estimation over a larger frequency interval.

There are fundamental difficulties with identification in closed loop under linear feedback, see for example Gustavsson *et al.* (1977). Relay feedback is, however, far from linear and it only acts at the brief instants when the output crosses the hysteresis level. We have not encountered any difficulties, but the problem undoubtedly deserves a theoretical investigation.

Experimental conditions

When doing system identification, it is crucial that the experimental conditions are appropriate. Important issues are for example:

1. How should the input signal be designed? Stated differently, between which values should ε vary, and in what way?
2. How many data points N should be collected?
3. What sampling time h should be used?

Each question affects the result in different ways, but they are still coupled to each other. The influence of different design parameters in the identification algorithm will be explored using the plant transfer function

$$G(s) = \frac{1}{(s+1)^7} \tag{4.6}$$

taken from the batch of transfer functions used in Panagopoulos (1998) for PID design. All plants in this batch and several more have been tested. Even if only a few processes are discussed here, the results carry over to other processes, possibly with slight modifications.

Selection of hysteresis interval When evaluating the feasibility of the method, a simple strategy for changing the hysteresis ε has been used. It is initially set to a large value ε_{max} and then decreased linearly to ε_{min}. Finally the input is reset to its initial value until the output settles. This reduces the effect of the implicit assumption of periodic signals when computing the DFT, see for example Oppenheim and Schafer (1989). Wang *et al.* (1999) solve this by decomposition of the output into a periodic and a transient part. This method cannot be used for a relay with

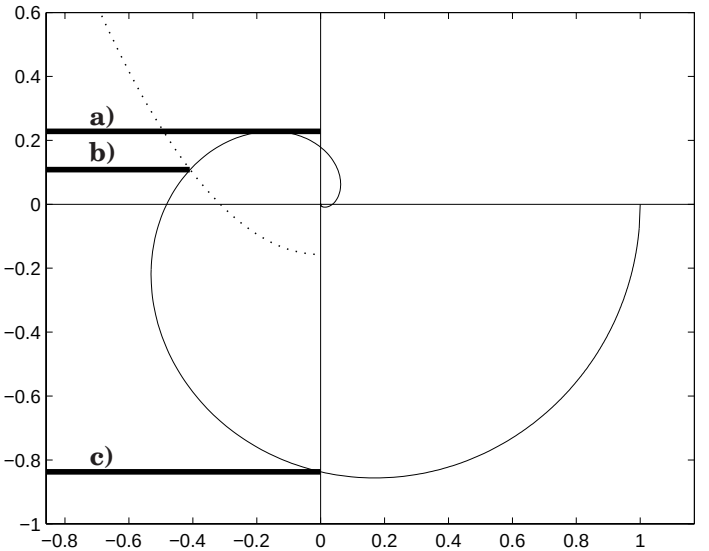


Figure 4.4 The Nyquist curve of $G(s) = 1/(s+1)^7$ together with three describing functions. They correspond to the preliminary choice of **a)** $\varepsilon = \varepsilon_{min}$ with $\Delta = 0$, **b)** $\varepsilon = \varepsilon_{min}$ with $\Delta = 0.4d$, and **c)** $\varepsilon = \varepsilon_{max}$.

time-varying hysteresis since the output will not reach a stationary limit cycle.

Preliminary limits for ε_{min} and ε_{max} will now be assessed. The following discussion is not intended as a practical guideline for choosing limits, but is included in order to provide some insight, and to relate to the standard analysis of relay feedback systems.

The describing function analysis can either be done by looking at the Nyquist curve or the inverse of the Nyquist curve. The latter provides more exact results, see Khalil (1992), but the former is more commonly used. No oscillation may be predicted using describing function arguments if $-1/N(a)$ does not intersect the Nyquist curve of the plant. An estimated upper limit of ε is then given by the point where the plant has -90° phase shift. From Figure 4.4 it is found that this point is $-0.837i$ for the given $G(s)$, and the corresponding $\varepsilon_{max} = 4d \cdot 0.837/\pi = 1.066d$. However, since $G(s)$ has a monotone step response and static gain 1, it is obvious that ε must be less than d, since the output would never cross the hysteresis level otherwise. Rather arbitrarily, ε_{max} is initially set to $0.9d$.

The estimate of ε_{min} is obtained from the intersection between $G(i\omega)$ and the parabola described by Equation (4.2). If there is no measurement noise, Δ may be chosen arbitrarily close to zero, and $-1/N(a)$ can be

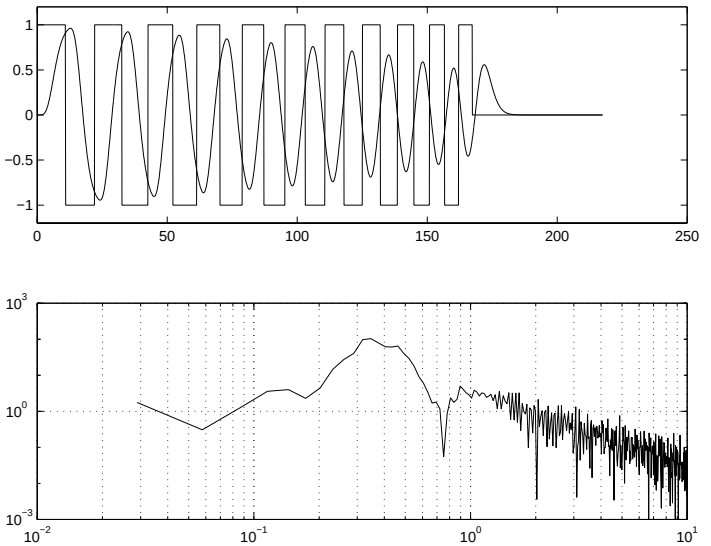


Figure 4.5 Noise-free simulation of a relay experiment with varying hysteresis. The input power spectrum is shown in the lower plot. Note the logarithmic scales.

chosen to intersect $G(i\omega)$ close to its highest point in the third quadrant. This corresponds to $\varepsilon_{min} = -4d \cdot 0.228/\pi = -0.291d$ in this case, see Figure 4.4. In the presence of white measurement noise, Δ should be greater than the maximum peak-to-peak value of the noise. With $\Delta = 0.4d$, the describing function starts at the dotted parabola in Figure 4.4. Consequently, ε_{min} should be higher than $-4d \cdot 0.109/\pi = -0.138d$. Note however that the noise occasionally makes the relay switch too early, and thus the output amplitude may not reach the level $\Delta - \varepsilon$ which is required for the oscillation to persist. The describing function analysis, which is approximate anyway, should thus be used only as a guideline.

The describing function analysis indicates that different plants require different ranges of the hysteresis. For example, plants dominated by a long dead time has a large value of $|G(i\omega)|$ while the phase is dropping from -90° to -270°. It will thus be possible to use approximately as large negative as positive values of ε. On the other hand, processes which are dominated by a first order time constant do not allow any negative values of ε at all, particularly if noise is present.

Noise-free estimation A noise-free simulation of a relay experiment is shown in Figure 4.5. The lower plot shows the power spectrum $|U_N(w)|^2$

of the input signal. The following parameters are used in the experiment: $h = 0.1$, $\varepsilon_{max} = 0.9$, $\varepsilon_{min} = -0.29$ and the number of hysteresis levels $N_\varepsilon = 20$. The number of data points N becomes 2175. The input is zero long enough for the output to return close to zero.

As a comparison, a simulation of a relay with fixed hysteresis $\varepsilon = 0.01$ is shown in Figure 4.6. The number of switches has been increased so the number of data points in each experiment is almost identical. It is clear that the input power spectrum in Figure 4.6 has higher but more narrow peaks. This shows that the excitation is more evenly spread with a time-varying relay.

The exact and the estimated Nyquist curves are shown in Figure 4.7. No smoothing windows are used, so the estimated frequency response is simply obtained by dividing the Fourier transforms of the output and the input. The estimation is perfect within the plotting accuracy. This is actually the case also for fixed relay with no noise present.

Influence of measurement noise Now, introduce additive noise to the process output:

$$y(k) = G(q)u(k) + v(k) \tag{4.7}$$

where $G(q)$ is the true pulse transfer operator of the plant. In this work, $v(k)$ is assumed to be white noise with zero mean and standard deviation σ.

Figure 4.8 shows simulations with two different levels of the measurement noise, $\sigma = 0.01$ and $\sigma = 0.1$, respectively. The ETFE will not be perfect in the presence of disturbances. The amplitude curve of the ETFE with the noise-free situation and with the two different noise sequences is shown in Figure 4.9. For $\sigma = 0.01$ the noise does not give very much error in the estimates up to $\omega = 1$. However, with $\sigma = 0.1$ the estimates for frequencies outside the interval $\omega = 0.2$ and $\omega = 0.6$ have substantial errors. The acceptable interval is of course where the most of the excitation is concentrated during the relay experiment. The corresponding phase shift of the process falls from $\approx -80^\circ$ to $\approx -220^\circ$. This should cover the interesting frequency interval in the PI and PID designs. Even within the interesting interval, the relative error of the amplitude may be as large as a few percents, and the phase error may be a few degrees. Thus, the ETFE should probably not be used directly in this case.

The quality of the estimate may be assessed by plotting the coherency spectrum $\kappa_{yu}(\omega)$ between the input and output. It is defined by

$$\kappa_{yu}(\omega) = \frac{|\Phi_{yu}(\omega)|}{\sqrt{\Phi_y(\omega)\Phi_u(\omega)}} \tag{4.8}$$

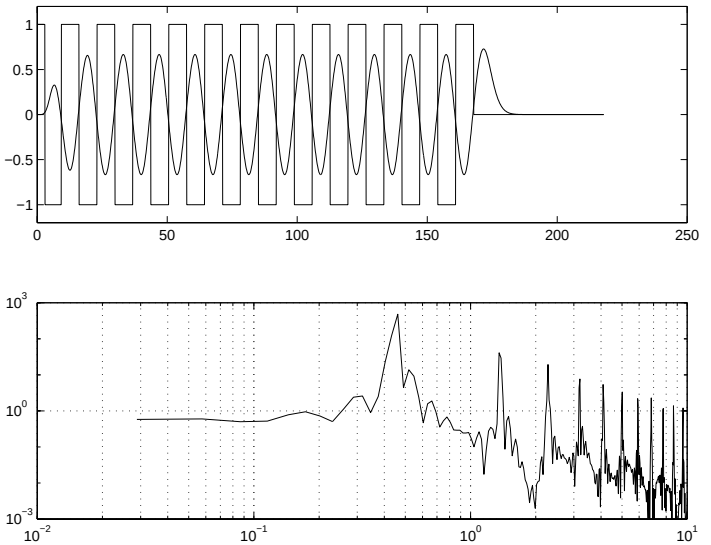


Figure 4.6 Noise-free simulation of a relay experiment with fixed hysteresis.

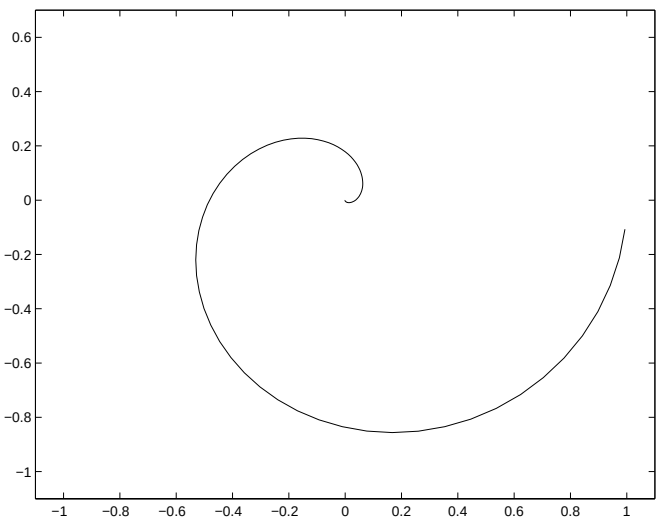


Figure 4.7 Perfect fit of estimated and true Nyquist curves from the noise-free simulation.

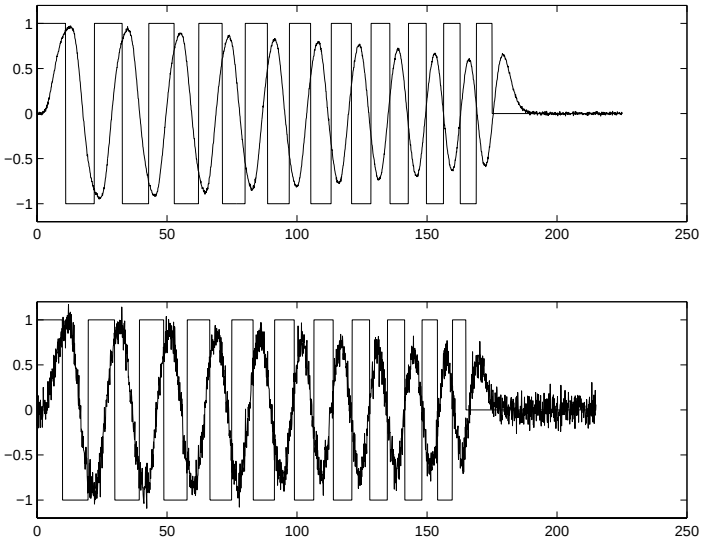


Figure 4.8 Simulations with white measurement noise. The standard deviation of the noise is $\sigma = 0.01$ (upper plot) and $\sigma = 0.1$ (lower plot).

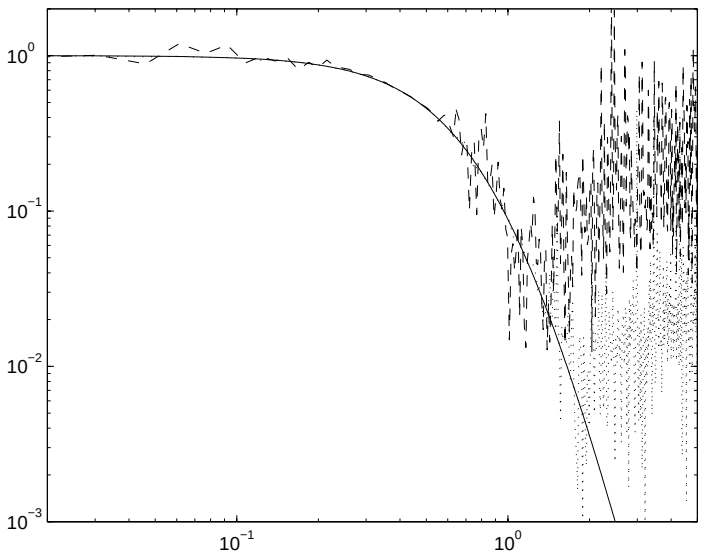


Figure 4.9 Amplitude curve of the ETFE with measurement noise. The curves correspond to $\sigma = 0$ (solid), $\sigma = 0.01$ (dotted) and $\sigma = 0.1$ (dashed).

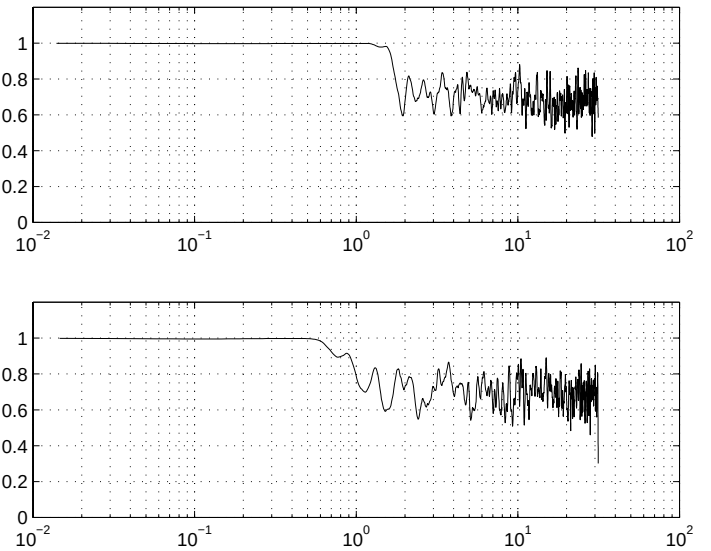


Figure 4.10 Coherency spectrum between y and u for $\sigma = 0.01$ (upper plot) and $\sigma = 0.1$ (lower plot). The curves are smoothed by averaging.

where $\Phi_y(\omega)$ and $\Phi_u(\omega)$ are the autospectra of y and u, respectively, and $\Phi_{yu}(\omega)$ is the cross spectrum between u and y. The spectra can be estimated as the Fourier transform of the corresponding covariance function multiplied by an appropriate window. The coherency spectrum is a measure of the linear correlation between the input and output for different frequencies. A value close to 1 indicates that there is substantial correlation, and a value close to 0 indicates that the signals are almost uncorrelated. The estimated coherency spectra for two noise levels are shown in Figure 4.10. They show that it is reasonable to use the signals for frequencies up to 1 rad/s for $\sigma = 0.01$ and up to approximately 0.6 rad/s for $\sigma = 0.1$. For higher frequencies, the noise term $v(k)$ in (4.7) dominates over the term $G(q)u(k)$.

According to Ljung (1999) the variance for each point of the ETFE is asymptotically given by

$$\frac{\Phi_v(\omega)}{|U_N(\omega)|^2} \tag{4.9}$$

where $\Phi_v(\omega)$ is the spectral density of an additive disturbance. When $v(k)$ is white measurement noise, $\Phi_v(\omega)$ is just a constant equal to the variance σ^2. The variance of the estimate is then proportional to the inverse of

the input spectrum. From Figure 4.5 it can be seen that the variance will be small in a frequency band from approximately 0.2 to 0.6 rad/s. Note that the input spectrum will not change drastically when measurement noise is introduced. The reason for this is that the input signal unaffected except for small variations in the switching instants. The noise typically causes the relay to switch a little earlier, making the oscillation frequencies slightly higher and the total experiment time shorter. The input spectrum is therefore shifted, and the frequency resolution is slightly lower. The deviations from the noise-free case are, however, marginal.

Normally, the noise power spectrum is not known. As pointed out in Ljung (1999), it may be estimated as

$$\hat{\Phi}_v(\omega) = \hat{\Phi}_y(\omega) - \frac{|\hat{\Phi}_{yu}(\omega)|^2}{\hat{\Phi}_u(\omega)} \tag{4.10}$$

Equation (4.9) then gives an estimate of the variance of $\hat{G}_N(e^{i\omega})$.

Length of experiment The number of values N_ε between ε_{max} and ε_{min} determines how the different frequencies are excited. It will also affect the length of the experiment. This in turn determines the frequency resolution in the ETFE. The frequency difference between two consecutive points of the DFT is given by

$$\Delta\omega = \frac{2\pi}{Nh} = \frac{2\pi}{T_f} \tag{4.11}$$

where T_f is the duration of the experiment. If T_f is increased, $\Delta\omega$ becomes smaller, and $G(i\omega)$ at neighboring frequencies become closer to each other.

Even if the frequency resolution is higher with a longer experiment, the variance of $\hat{G}_N(e^{i\omega})$ at a specific frequency point will be approximately the same. A consequence of this is that large deviations of the estimates are *more* likely for long experiments. This is illustrated in Figure 4.11.

The upper plot in Figure 4.12 shows the input power spectrum for the input signal corresponding to the case $N_\varepsilon = 80$ in Figure 4.11. The lower plot shows the inverse of the input power spectrum if a *fixed* relay hysteresis is used, still with 80 switches in the relay and approximately the same number of data points. Compared to the noise-free case with $N_\varepsilon = 20$ in Figures 4.5 and 4.6, the bands of excitation are more distinct, but not drastically different.

Windowing The main reason for introducing a frequency window is to reduce the variance of the points in the estimated frequency response. This is done at the expense of a bias in the estimate. Thus, the average

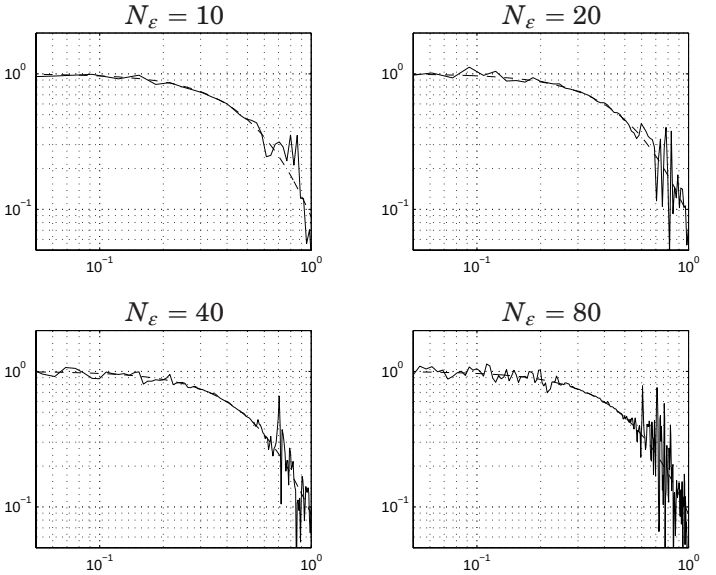


Figure 4.11 Amplitude curves of the ETFE with $\sigma = 0.1$ for different experiment durations. The dashed curves correspond to the true amplitude curves.

errors will inevitably be larger when using a window. However, since the estimates are smoother, there may be smaller errors in the first and second derivatives of the estimated frequency response.

The ratio between the estimated and true amplitude curves is shown for different choices of window lengths in Figure 4.13. The corresponding phase error is shown in Figure 4.14. The same experiment with $N_\varepsilon = 20$ and $\sigma = 0.1$ has been used in all cases. The effects on bias and variance are obvious in the figure. The variance for $M = 1$ may cause trouble when differentiating the frequency response. On the other hand, the bias introduced by the longer windows will cause errors, both in the frequency response and its derivatives. The RMS error will often be larger with a longer window, despite the fact that variance decreases.

According to Ljung (1999), the bias is asymptotically given by

$$E\hat{G}_N(e^{i\omega}) - G(e^{i\omega}) = M(\gamma) \cdot \left(\frac{1}{2} G''(e^{i\omega}) + G'(e^{i\omega}) \frac{\Phi_u'(\omega)}{\Phi_u(\omega)} \right) \tag{4.12}$$

where $M(\gamma)$ is a number which depends on the shape of the window. For a Hamming window it is proportional to the square of the length of the frequency window. It is thus natural that the bias increases with the window

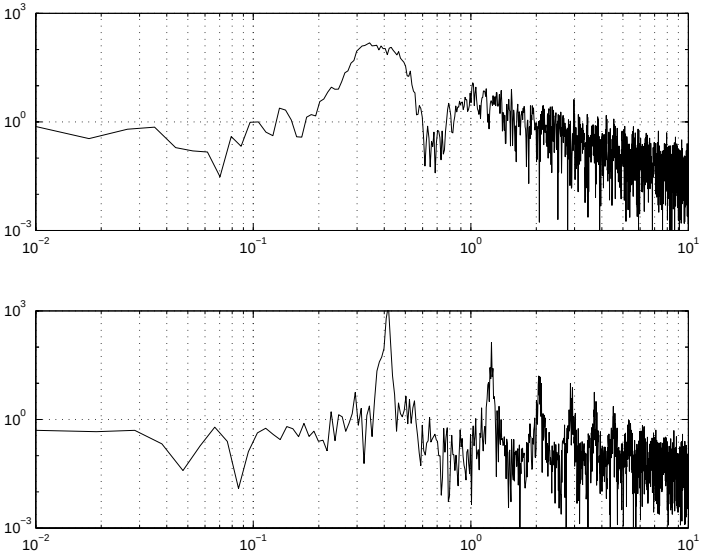


Figure 4.12 Input power spectrum for time-varying (upper plot) and fixed (lower plot) hysteresis with $N_\varepsilon = 80$.

length. Another observation is that the second term in Equation (4.12) is proportional to the derivative of the input spectrum. By comparing with Figure 4.5, this can explain the large phase errors at $\omega = 0.2$ and $\omega = 0.6$.

One way of reducing the bias for long windows is to use longer experiments. According to Equation (4.11), this increases the frequency resolution, and thus makes the frequency response estimates closer for neighboring frequencies.

Choice of sampling interval The PI and PID design methods that will be used are based on continuous-time models. It is thus desired that the effects of the sampling interval on the estimated frequency response are minimized. The most striking effect of the sampling interval h is that it defines the highest frequency for which the DFT is computed. Faster sampling results in more data points for a given T_f, which increases the CPU time and memory requirements in the calculations. When no disturbances are present, the choice of h does not have any dramatic effect on the quality of the transfer function estimate. Slow sampling will introduce some discrepancies between the frequency responses of the continuous-time process and the sampled one. The main difference is the extra phase lag introduced by the sampling. According to Åström and Wittenmark (1997),

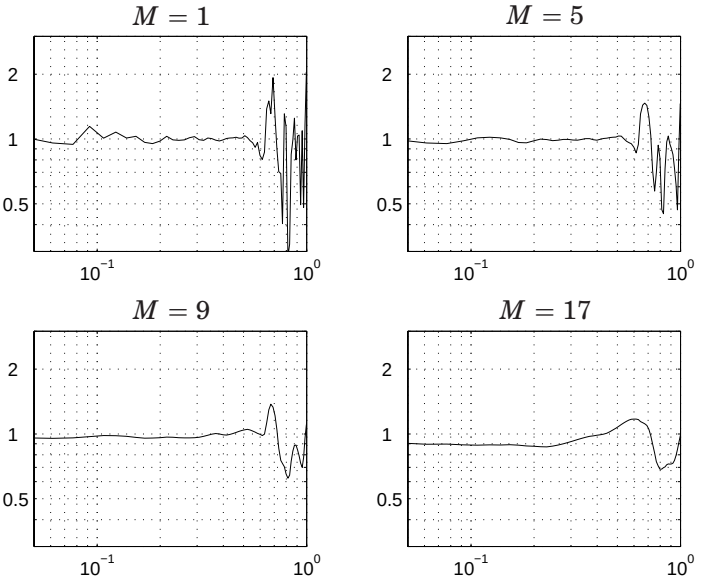


Figure 4.13 $|\hat{G}_N(e^{i\omega})|/|G(e^{i\omega})|$ for different lengths M of the frequency window. $N_\varepsilon = 20$ and $\sigma = 0.1$ in all cases.

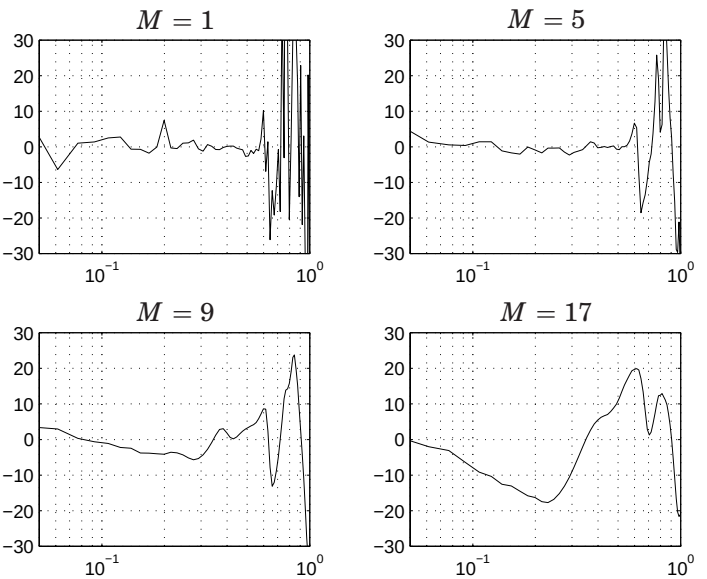


Figure 4.14 $\arg\left(\hat{G}_N(e^{i\omega})\right) - \arg\left(G(e^{i\omega})\right)$ for different lengths M of the frequency window. $N_\varepsilon = 20$ and $\sigma = 0.1$ in all cases.

this is approximately given by $\omega h/2$ radians, which can be added to the estimated frequency response.

When there are substantial disturbances during the experiment, the choice of sampling interval has large influence on the quality of the estimated frequency response. First, assume that an experiment has been conducted with N data points using a sampling interval h. The corresponding input signal is $u(k)$ and its Fourier transform is $U_N(\omega)$. The variance of the transfer function estimate at each frequency is then given by Equation (4.9). When $v(k)$ is white noise with variance σ^2, $\Phi_v(\omega) = \sigma^2$ irrespective of the sampling interval.

Now, the sampling interval will be reduced by a factor L. This results in modified switching instants, corresponding to a new input signal $\bar{u}(k)$ with Fourier transform $\bar{U}_{LN}(\omega)$. In order to simplify the analysis we will however make the assumption here that the switching instants do *not* change. L is then required to be an integer, but this is no serious restriction. $\bar{u}(k)$ then consists of LN points.

Using the assumption above, we have

$$u(k) = \bar{u}(Lk) = \bar{u}(Lk-1) = \ldots = \bar{u}(L(k-1)+1) \tag{4.13}$$

The Fourier transform of $\bar{u}(k)$ is then given by

$$\begin{aligned}
\bar{U}_{LN}(\omega) &= \frac{1}{\sqrt{LN}} \sum_{k=1}^{LN} \bar{u}(k) e^{-i\omega k} \\
&= \frac{1}{\sqrt{LN}} \Bigg(\sum_{k=1}^{N} \bar{u}\,(Lk)\, e^{-i\omega Lk} + \sum_{k=1}^{N} \bar{u}\,(Lk-1)\, e^{-i\omega(Lk-1)} + \\
&\qquad \ldots + \sum_{k=1}^{N} \bar{u}\,(L(k-1)+1)\, e^{-i\omega(L(k-1)+1)} \Bigg)
\end{aligned} \tag{4.14}$$

Using (4.13) and the definition of the DFT we get

$$\begin{aligned}
\bar{U}_{LN}(\omega) &= \frac{1}{\sqrt{L}} \left(1 + e^{i\omega} + \ldots + e^{i(L-1)\omega} \right) \cdot U_N(L\omega) \\
&= \frac{1}{\sqrt{L}} \cdot \frac{e^{iL\omega} - 1}{e^{i\omega} - 1} \cdot U_N(L\omega)
\end{aligned} \tag{4.15}$$

Note that ω is the normalized frequency for each sequence, defined from $-\pi$ to π. $\bar{U}_{LN}(\omega)$ and $U_N(L\omega)$ thus refer to the same absolute frequency. After simplifications, the power spectrum of the new input signal can be written as

$$\left|\bar{U}_{LN}(\omega)\right|^2 = \frac{1}{L} \cdot \frac{\cos(L\omega) - 1}{\cos(\omega) - 1} \left|U_N(L\omega)\right|^2 = c_L(\omega) \left|U_N(L\omega)\right|^2 \tag{4.16}$$

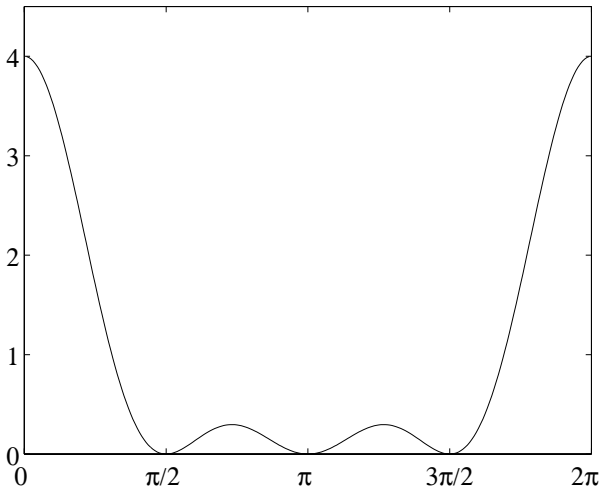


Figure 4.15 The scaling factor $c_L(\omega)$ of the input power spectrum with 4 times higher sampling frequency. Note that the power spectral density is increased for all frequencies below the original Nyquist frequency ($\pi/4$).

The scaling factor $c_L(\omega)$ between the two power spectra is plotted in Figure 4.15 for $L = 4$. For low frequencies we have

$$c_L(\omega) \approx \frac{1}{L} \cdot \frac{1 - \frac{1}{2}L^2\omega^2 - 1}{1 - \frac{1}{2}\omega^2 - 1} = L \tag{4.17}$$

According to Equation (4.9) the variance of the transfer function estimate for low frequencies will thus be reduced by a factor L. For higher frequencies, $c_L(\omega) < 1$, and the corresponding variance would instead increase. On average, the variance in fact remains constant, which is reasonable, since $u(k)$ and $\bar{u}(k)$ correspond to the same continuous-time signal with a certain energy. However, for frequencies below the Nyquist frequency for the original data, we have $c_L(\omega) > 1$ for all $L > 1$. For the frequencies of interest, we typically have $c_L(\omega) \approx L$.

Even if the analysis above is done by resampling a fixed control signal $u(k)$, the result is approximately correct for the actual $\bar{u}(k)$ which is produced by relay feedback using faster sampling. In the interesting frequency interval we thus have that the variance of each point in the transfer function estimate is approximately proportional to the sampling interval h for a fixed length of the experiment. In theory it is then possible to have arbitrarily good estimates by increasing the sampling rate. The price for the benefits is increased CPU and memory requirements.

4.3 PI Control

The main purpose of the process identification in Section 4.2 is to design a controller. The identification procedure was tailored to give good estimates of the frequency response where the phase shift of the process is between approximately $-90°$ and $-240°$. The estimated frequency response of the process will be used in this section for designing robust PI controllers.

Design method

The design procedure for PI controllers is taken from Åström *et al.* (1998), and will be discussed briefly here. The basic idea is to find the controller which minimizes the integrated error

$$IE = \int_0^\infty (y_{sp}(t) - y(t))\, dt$$

after a step load disturbance on the plant input. When the plant is controlled by a PI controller

$$G_c(s) = k\left(1 + \frac{1}{sT_i}\right) = k + \frac{k_i}{s} \tag{4.18}$$

it is straightforward to show that IE is given by

$$IE = \frac{1}{k_i} \tag{4.19}$$

The integrated error is thus minimized by maximizing the integral gain k_i. There is however no guarantee that a large k_i by itself results in good performance. On the contrary, large k_i will mostly make the closed-loop system badly damped, which leads to an oscillating load disturbance response. The integrated error may still be small, whereas the integrated absolute error IAE may be arbitrarily large.

The solution is to maximize k_i subject to constraints on the loop transfer function. For example, it is possible to limit the maximum value of the sensitivity function to a certain value M_s, which may then be used as a design parameter. This measure has nice robustness properties, and may be interpreted geometrically as the inverse of the minimum distance from the Nyquist curve of the loop gain to the point -1. A slightly more general constraint is to force the loop gain to avoid the circle with its center in $-C$ and radius R. The constraint on the sensitivity function is obtained by setting $C = 1$ and $R = 1/M_s$.

It should be noted that it is not the use of IE *per se* that makes it necessary to add constraints on the loop gain. Even if IAE is minimized,

the resulting controller will perform badly and have bad robustness if no additional constraints are used, see Andreas and Åström (1997).

In Åström *et al.* (1998) it is shown that the optimization problem outlined above can be reduced to solving a non-linear equation. Introduce polar coordinates for the frequency response of the process

$$G(i\omega) = r(\omega)e^{i\varphi(\omega)} \tag{4.20}$$

which is assumed to be known. Define the function

$$h(\omega) = 2R\left((R + C\sin\varphi(\omega))\left(\frac{r'(\omega)}{r(\omega)} - \frac{1}{\omega}\right) - C\varphi'(\omega)\cos\varphi(\omega)\right) \tag{4.21}$$

The frequencies ω_0 where the constraint is active are then solutions to the equation $h(\omega_0) = 0$. This equation may have many solutions, but Åström *et al.* (1998) show that only frequencies where the plant has a phase shift between -90° and -180° need to be checked. If the frequency response is monotonous in this interval, there exists exactly one solution in this interval. Furthermore, it should hold that

$$\frac{dh}{d\omega}(\omega_0) < 0 \tag{4.22}$$

at the optimum. The controller gains are given by

$$k = -\frac{C}{r(\omega_0)}\cos\varphi(\omega_0), \tag{4.23}$$

$$k_i = -\frac{\omega_0}{r(\omega_0)}\left(C\sin\varphi(\omega_0) + R\right) \tag{4.24}$$

In order to improve the set point step response, a set point weighting factor b on the proportional part is introduced. b is allowed to vary between 0 and 1, and selected to obtain, if possible,

$$\max|G_{sp}(i\omega)| = 1$$

where $G_{sp}(s)$ is the transfer function from the set point to the plant output. Once the parameters k and k_i are computed, it is straightforward to calculate b, see Åström *et al.* (1998). A more advanced method for obtaining nice set point responses is described in Chapter 5.

The design method outlined here is applicable to all processes with a phase shift of -180° or less for high frequencies. For minimum-phase plants with relative degree one, k and k_i can be increased infinitely without violating the sensitivity constraint. However, since the problem is

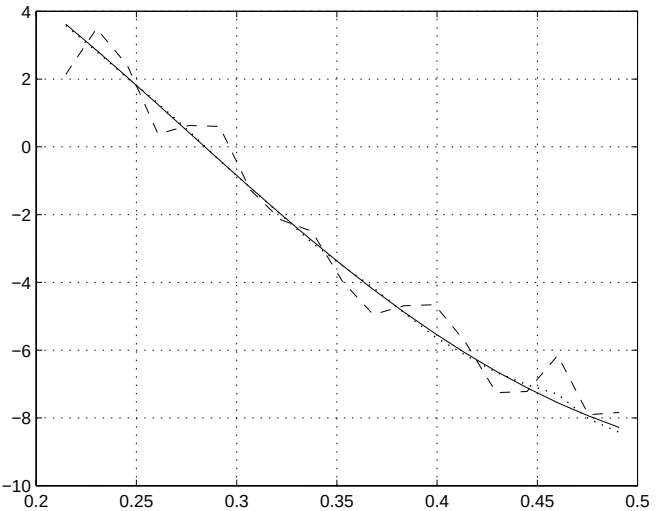


Figure 4.16 The function $h(\omega)$ for a design with $M_s = 2.0$ for $\sigma = 0$ (full), $\sigma = 0.1$ (dashed), and $\sigma = 0.01$ (dotted). $\hat{G}_N(e^{i\omega})$ is taken as the raw spectral estimate.

solved by looking at a frequency interval, there may exist some local maximum to k_i as long as the phase shift is between -90° and -180° in a sufficiently large interval. This situation is probably not very common in practice, since most plants have high frequency roll-off rather than high frequency amplification.

Design based on estimated frequency response

In practice, it is not reasonable to assume that $G(i\omega)$ is known exactly. The estimate $\hat{G}_N(e^{i\omega})$ taken from Equation (4.5) in Section 4.2 will be used instead. The functions $r(\omega)$ and $\varphi(\omega)$ in Equation (4.21) are just vectors of floating point numbers. $r'(\omega)$ and $\varphi'(\omega)$ are obtained through differentiation of these vectors. Since differentiation is a noise sensitive operation, the differentiated vectors may have to be low-pass filtered.

The problem is then solved by forming the real-valued vector $h(\omega)$ from Equation (4.21) and finding zero crossings of this vector in the interval where $-\pi < \varphi(\omega) < -\pi/2$. The condition (4.22) reduces to a sign check of $h(\omega)$ on each side of the zero crossing. ω_0 may be refined slightly by doing linear interpolation to find a better estimate of the zero crossing.

Figure 4.16 shows the function $h(\omega)$ for PI designs with $M_s = 2.0$. The relay experiment was simulated with the parameters $\varepsilon_{max} = 0.9$, $\varepsilon_{min} = -0.1$, $N_\varepsilon = 20$ and $h = 0.1$, using both $\sigma = 0$, $\sigma = 0.01$ and $\sigma = 0.1$. No window or filtering has been used in the estimation of $G(i\omega)$

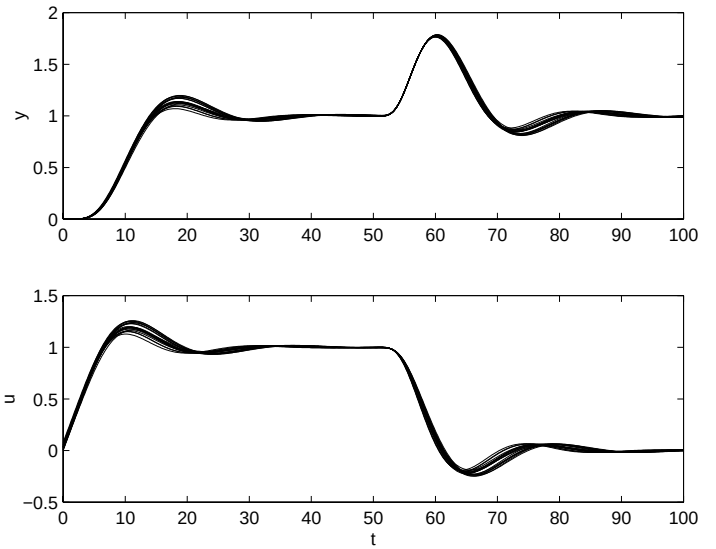


Figure 4.17 Simulations of PI control with $M_s = 2$ using raw spectral estimates. 20 different noise sequences with $\sigma = 0.1$ has been used in the relay experiments.

or its derivative.

The zero crossings for the full and dotted lines, corresponding to $\sigma = 0$ and $\sigma = 0.01$, are very close to each other. After linear interpolation of the numerical vectors, the solution is given by $\omega_0 = 0.284$ in both cases. This corresponds to a phase shift of -111° of the process. From Equations (4.23) and (4.24) the controller parameters are calculated as $k = 0.47$ and $k_i = 0.16$.

Simulations with $\sigma = 0.1$ have more variance in $\hat{G}_N(e^{i\omega})$, which gives a noisy $h(\omega)$, and the solution ω_0 typically varies between 0.27 and 0.30. Consequently, the controller parameters may vary substantially. From Equations (4.23) and (4.24) it may be concluded that k (and T_i) will vary more than k_i in this case, since $\cos(\varphi(\omega))$ varies more than $\sin(\varphi(\omega))$ around ω_0. The actually obtained M_s value may also vary from approximately 1.9 to 2.1. Figure 4.17 shows 20 simulations of set point and load disturbance responses on top of each other. These simulations show that the behavior does not change dramatically even if k and k_i varies as much as 15% and 5%, respectively.

In order to decrease the variations even further, the uncertainties in $h(\omega)$ must be reduced. In Section 4.2 it was found that this may be done either by decreasing the sampling interval h, or by using a frequency window according to Equation (4.5). In Section 4.2 it was found that the

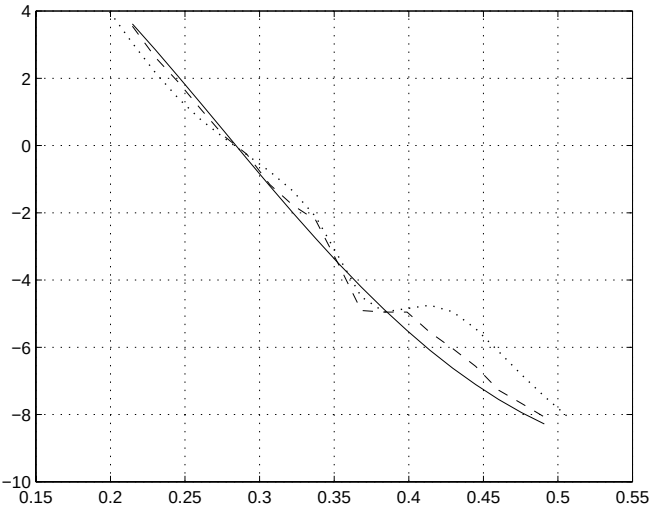


Figure 4.18 The nominal function $h(\omega)$ with no windowing (full), window lengths 5 (dashed), and 9 (dotted). Bias in $\hat{G}_N(e^{i\omega})$ causes large errors in $h(\omega)$.

variance of the estimated frequency response is approximately proportional to the sampling interval. This is transformed in a non-trivial way to the function $h(\omega)$, to the solution ω_0, and to the parameters k and k_i. Simulations have shown that there is a noticeable reduction in variance of the estimated ω_0, k, and k_i when h is decreased. However, the reduction in variance is less than proportional to the reduction in sampling interval, whereas the memory and CPU requirements increase faster than this.

When using a frequency window, the length should be chosen at least as $M = 5$, and preferably much larger, in order to achieve a noticeable reduction of the variance in the estimate. However, the bias actually may introduce even larger errors in $h(\omega)$. The nominal function is plotted in Figure 4.18 together with the distorted function for two different window lengths. The bias will give rise to systematic errors in ω_0. These errors happen to be almost zero for the combination of the process, the M_s value, and the window lengths used in Figure 4.18. With longer experiments, the systematic errors will be reduced. However, again the memory and CPU requirements increase.

Most of the variance in $h(\omega)$ comes from the differentiations. Low-pass filtering of these differentiated vectors is then another way of obtaining improved results. By using non-causal FIR filters with linear phase, it is possible to achieve a smoother function without introducing bias in $h(\omega)$, see Figure 4.19. It is of course important to choose the cut-off frequency

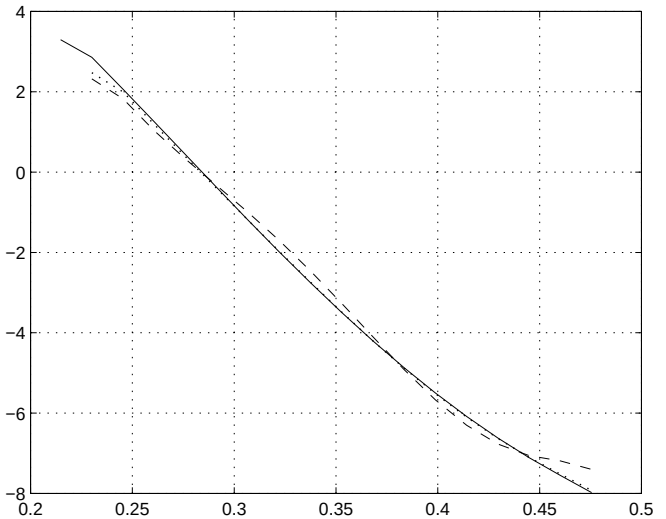


Figure 4.19 The function $h(\omega)$ for $\sigma = 0$ (full), $\sigma = 0.1$ (dashed), and $\sigma = 0.01$ (dotted) after low-pass filtering of $\hat{G}'_N(e^{i\omega})$.

high enough in order to capture the variations of the true frequency response. It turns out that low-pass filtering is the most efficient way of reducing the uncertainties in the control design.

The identification and design procedure outlined here works very well for PI design of all test processes in Panagopoulos (1998), as soon as good values for ε_{min} and ε_{max} are found. The test batch has been evaluated for the noise levels $\sigma = 0.01$ and 0.1 using $h = 0.1$ and $N_\varepsilon = 20$ as default parameters. In most cases a lower N_ε may be used, and some faster processes require lower h.

4.4 PID control

Design method

The basic idea from the PI design carries over to the PID case. The PID controller is given by

$$G_c(s) = k\left(1 + \frac{1}{sT_i} + sT_d\right) = k + \frac{k_i}{s} + k_d s \tag{4.25}$$

The integrated error after a step load disturbance is still minimized by maximizing k_i. By using the derivative action for lifting the phase curve

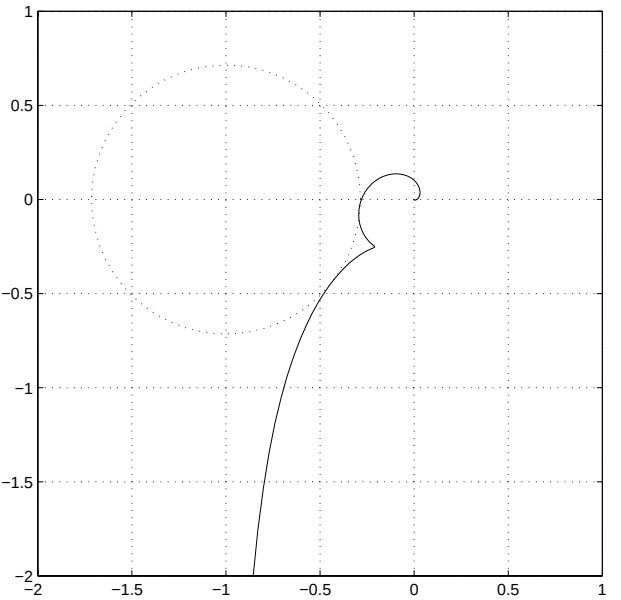


Figure 4.20 The loop transfer function for the PID design which maximizes k_i with $G = 1/(s+1)^7$, $M_s = 1.4$.

it is possible to obtain a higher k_i compared to the PI design without violating the sensitivity constraint. However, too much phase lead may cause undesired shapes of the Nyquist curve of the loop gain. There may for example be more than one point of tangency of the M_s circle, see Figure 4.20. This leads to worse robustness properties. In order to avoid too much phase lead, some additional constraint must be used. In Panagopoulos (1998) the following constraints are used close to the M_s circle:

- The loop transfer function should have decreasing phase curve.
- The curvature of the loop transfer should be negative.

These constraints give a nice shape of the loop gain without being overly conservative. One drawback with is that the computation of the curvature involves two differentiations of the frequency response of the process. Since $G(i\omega)$ is not known exactly, the variance in $\hat{G}_N(e^{i\omega})$ is amplified dramatically by the differentiations. These variations may be reduced by windowing and low-pass filtering. It is however still difficult to obtain a smooth and accurate estimation of the curvature for high noise levels.

Another way of putting limits on the phase lead is to use a fixed ratio between T_i and T_d. For example, Ziegler-Nichols tuning methods use

$T_i/T_d = 4$, which is the lowest ratio to avoid complex zeros of the controller. The constraints on loop transfer phase and curvature result in different ratios depending on the process and the chosen value of M_s. For the test batch in Panagopoulos (1998), the ratio lies in the range 1.2–2.6. This indicates that $T_i/T_d = 4$ may be overly conservative, and there is no fundamental reason for sticking to this ratio. However, the optimization problem will have an elegant solution in this case. With $T_i/T_d = 4$ the controller may be written as

$$G_c = k\left(1 + \frac{1}{sT_i} + \frac{sT_i}{4}\right) = k\frac{\left(\frac{T_i}{2}s + 1\right)^2}{T_i s} = k^\star \frac{(T_i^\star s + 1)^2}{T_i^\star s} \tag{4.26}$$

where $k^\star = k/2$ and $T_i^\star = T_i/2$. Thus, the PID design can be solved by doing a PI design for the modified plant

$$G^\star(s) = (Ts + 1)\, G(s) \tag{4.27}$$

where T should be chosen as the integral time $T_i^\star$. Since the modified plant contains the unknown variable $T_i^\star$, the problem needs to be solved iteratively. A straightforward algorithm is defined by:

$$\begin{aligned}
G_0^\star(s) &= G(s) && \longrightarrow && T_{i1}^\star \\
G_1^\star(s) &= (T_{i1}^\star s + 1)\, G(s) && \longrightarrow && T_{i2}^\star \\
G_2^\star(s) &= (T_{i2}^\star s + 1)\, G(s) && \longrightarrow && T_{i3}^\star \\
&\cdots
\end{aligned}$$

It turns out that this simple scheme mostly converges to the desired solution. The iteration is illustrated for the plant $G(s) = 1/(s+1)^7$ in Figure 4.21. The full line shows the optimal T_i for the modified plant $(1 + sT)G(s)$. The algorithm will converge if $|T_i'(T)| < 1$ in a neighborhood of the intersection with the straight line with unit slope. This is the case for all plants in the test batch used in Panagopoulos (1998). $T_i(T)$ normally is a monotonically decreasing function, since higher T gives a faster plant, which mostly has a lower optimal T_i. Furthermore, the curve starts at the optimal T_i for the original plant $G(s)$ and ends at the optimal T_i for the differentiated plant $sG(s)$.

As mentioned in Section 4.3, PI controllers for minimum-phase plants with pole excess one can be made arbitrarily fast and the integral gain can thus be made arbitrarily large. The design problem that maximizes the integral gain is thus not well posed. Analogously, the PID design problem is not well posed for minimum-phase plants with pole excess two. From Åström *et al.* (1998) it is also known that a monotonous frequency

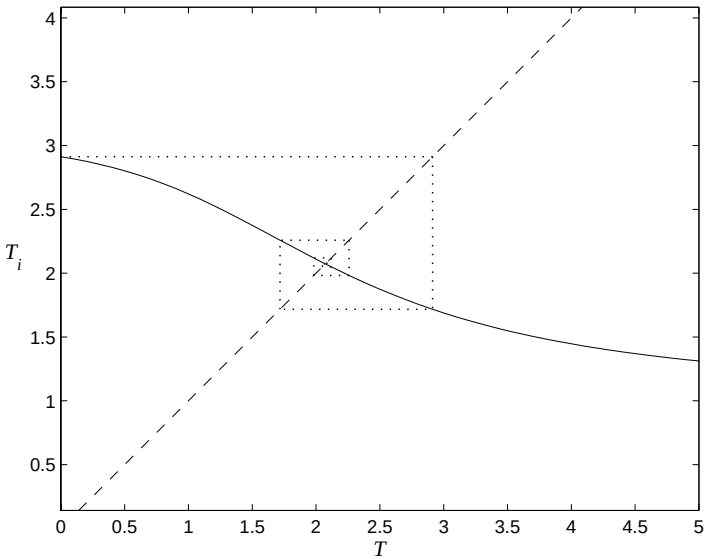


Figure 4.21 Iterative PI design of modified plant. The full line shows the integration time T_i from a PI design of the plant $(Ts+1)/(s+1)^7$.

response of the plant facilitates the PI design problem because suitable initial values for the optimization can always be found. Since the iterative scheme for PID design introduces phase lead, it is not obvious that the modified plant has a monotonous frequency response in the interesting interval. Extra care must thus be taken when solving the PI subproblems.

The different constraints will now be illustrated in an example. Figures 4.22 and 4.23 show the behavior for PID control of the process $G(s) = 1/(s+1)^7$ with two different values of M_s, and when using different constraints:

A. Maximum sensitivity M_s and fixed ratio $T_i/T_d = 4$ (full lines),

B. Maximum sensitivity M_s, decreasing phase curve and negative curvature (dashed lines), and

C. Maximum sensitivity M_s only (dotted lines).

The IE and IAE after a step load disturbance is summarized in Table 4.1. As might be expected, IE is slightly higher when a fixed ratio is used and the corresponding time responses are overdamped for $M_s = 1.4$. On the other hand, the underdamped behavior for $M_s = 2.0$ with the other designs are avoided with the fixed ratio, so the IAE is actually lowest for this design.

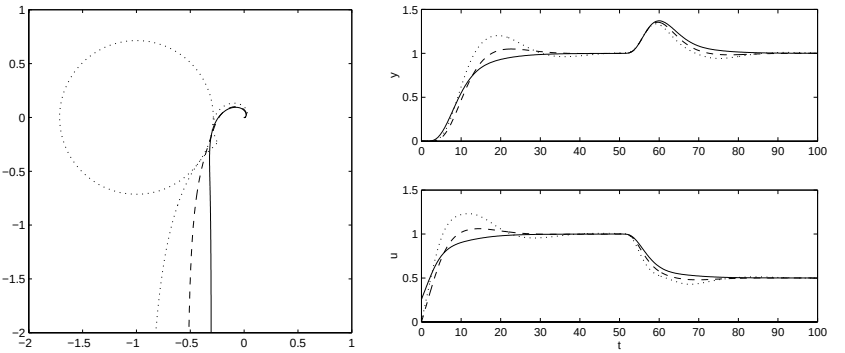


Figure 4.22 Loop transfer functions and time responses for PID designs with $M_s = 1.4$ using constraint combinations **A.** (full), **B.** (dashed), and **C.** (dotted).

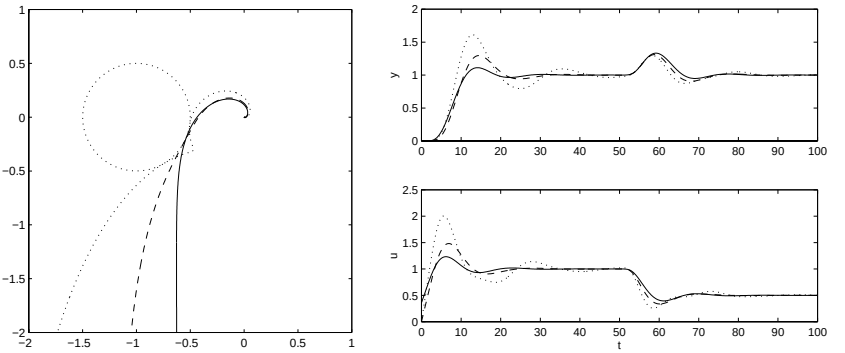


Figure 4.23 Loop transfer functions and time responses for PID designs with $M_s = 2.0$ using constraint combinations **A.** (full), **B.** (dashed), and **C.** (dotted).

	$M_s = 1.4$					$M_s = 2.0$				
	k	T_i	T_d	IE	IAE	k	T_i	T_d	IE	IAE
A.	0.52	4.40	1.10	8.55	8.55	0.91	4.15	1.04	4.58	5.80
B.	0.55	3.55	1.69	6.45	7.21	0.98	3.14	1.73	3.19	5.87
C.	0.56	2.62	3.06	4.69	7.41	0.99	2.01	3.36	2.03	7.21

Table 4.1 Comparison of load disturbance errors for PID control using **A.** $T_i = 4T_d$, **B.** curvature and phase constraints, and **C.** only sensitivity constraint.

In conclusion, the design constraint $T_i = 4T_d$ is a feasible alternative to the curvature and phase constraints in Panagopoulos (1998). Both methods have been tested on estimated frequency response data. This is further discussed below.

Design based on estimated frequency response

The PID design methods give stricter demands on the quality of the estimated frequency response than the PI design method does. First of all, the range of interesting frequencies will be larger, since the bandwidth generally is higher for the optimal PID controller than for the optimal PI controller. Furthermore, if constraints on the curvature are going to be used, two differentiations of the frequency response estimate are required. This operation amplifies high frequency noise dramatically.

The difficulties in estimating the curvature appear already at low noise levels. By using fast sampling, long experiments, windowing and low-pass filtering of the differentiated vectors, it is possible to obtain feasible results for some plants. However, the computation of the curvature is extremely sensitive to correct estimation of both the first and second derivative of the frequency response. No method has been found that reproduces the true curvature in a consistent and robust way. This design method has therefore been abandoned here.

The iterative scheme for obtaining a PID design with $T_i = 4T_d$ is possible to use even if the data is noisy. However, the function $T_i(T)$ plotted in Figure 4.21 is no longer guaranteed to be a smooth function if the raw spectral estimates are used when calculating $h(\omega)$. This effect is not so severe for low noise levels, but for $\sigma = 0.1$ it becomes more prominent. Therefore, any kind of periodic or chaotic behavior can be expected from the iteration. One example with a 2-periodic limit cycle is shown in Figure 4.24. The discontinuous and non-monotonic shape of the function $T_i(T)$ can be explained by looking at the function $h(\omega)$ for the modified plant. Figure 4.25 shows the function $h(\omega)$ for the modified plant for different values of T with the noise level $\sigma = 0.1$. The solutions ω_0, marked with crosses in the figure, will not be spread evenly due to the uncertainties in $h(\omega)$. For $T \approx 2.55$ there will even be a discontinuity in the solutions around $\omega_0 = 0.4$ due to a non-monotonic $h(\omega)$. The errors in ω_0 carry over to T_i through Equations (4.23) and (4.24).

The function $h(\omega)$ must be smoothed in order to be used iteratively. This may for example be done by low-pass filtering of the differentiated frequency response of the process. Using this technique, the iteration will produce consistent results, see Figure 4.26. This is very close to the exact iteration in Figure 4.21. The parameter values vary a few percents between different simulations. Figure 4.27 shows 20 simulations of set point and load disturbance responses on top of each other. The varia-

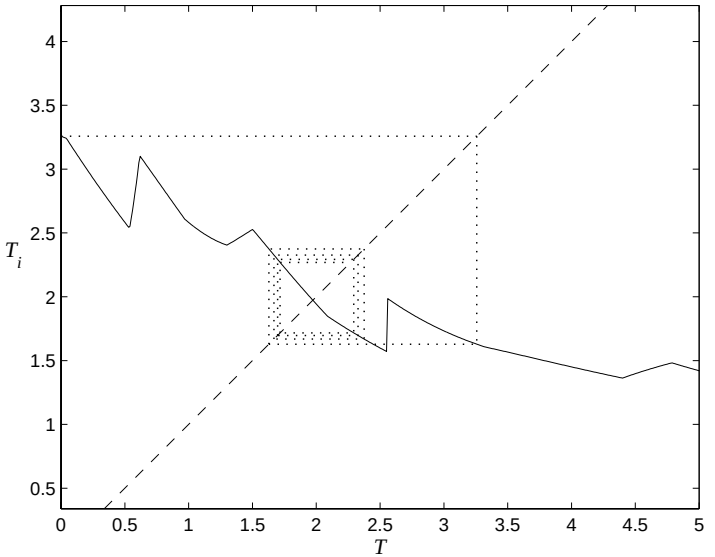


Figure 4.24 Iterative PI design based on raw spectral estimates with $\sigma = 0.1$. In this case the iteration converges to a 2-periodic limit cycle.

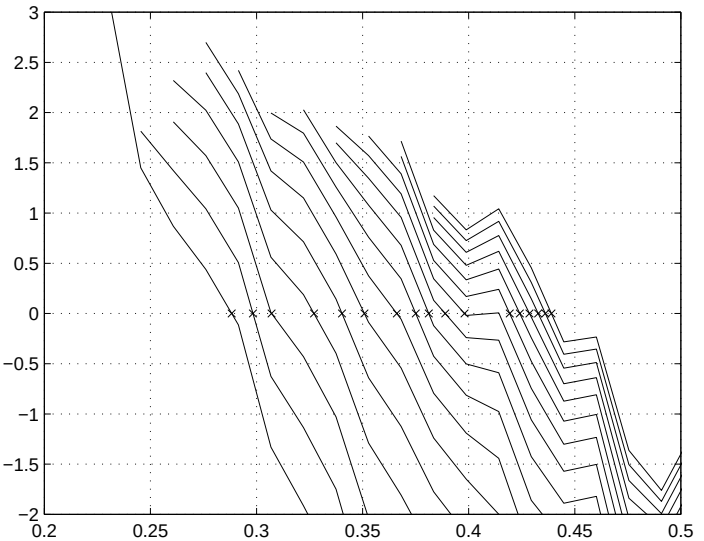


Figure 4.25 The estimated function $h(\omega)$ for the modified plant $G^\star(s) = (Ts + 1)\, G(s)$, with $T = 0,\ 0.25,\ 0.5,\ \ldots 4$.

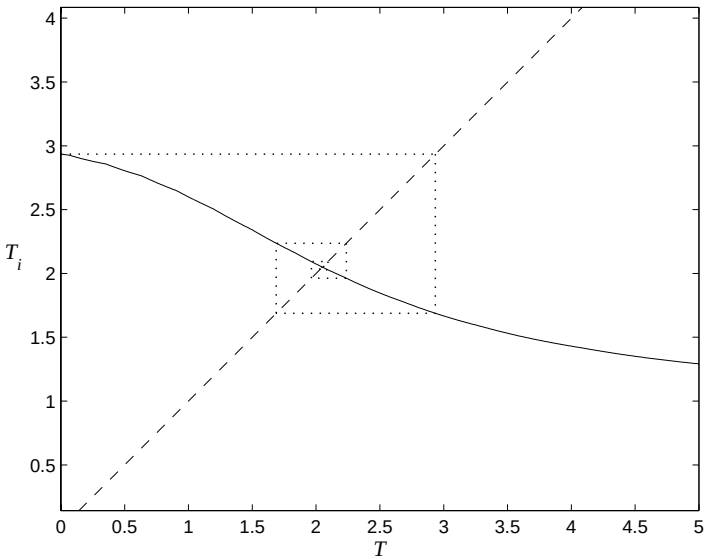


Figure 4.26 Iterative PI design based on low-pass filtered derivatives of the spectral estimates.

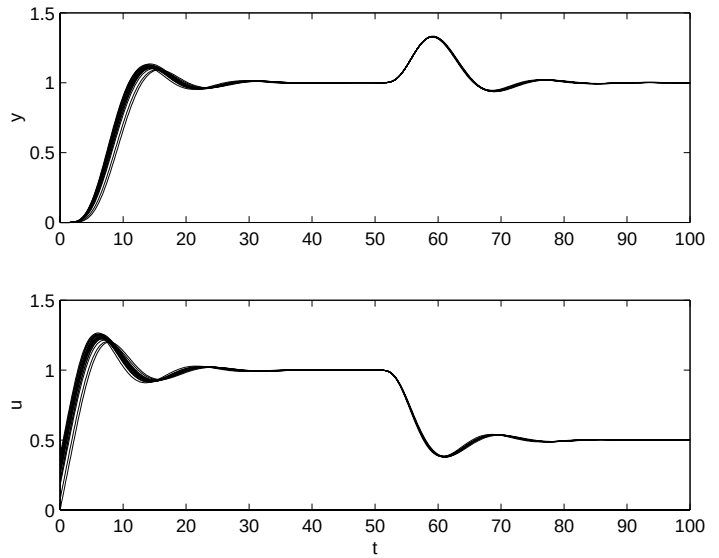


Figure 4.27 Simulations of PID control using iterative PI design for $M_s = 2$ using filtered spectral estimates. 20 different noise sequences with $\sigma = 0.1$ has been used in the relay experiments. The load disturbance response is fairly consistent.

tion in the set point response is caused by the weighting factor b, which is very sensitive both to errors in the controller parameters and the estimated frequency response. No effort has been spent on improving this, since another method for fast set point changes is presented in Chapter 5.

4.5 Summary and Concluding Remarks

A design method for PI and PID control using relay feedback has been developed. The relay uses time-varying hysteresis in order to provide excitation for the interesting frequency interval. The frequency response of the process is estimated using spectral analysis. This is used for PI design according to Åström *et al.* (1998). It was shown that a PID controller can be achieved by solving PI design problems iteratively.

The proposed method for doing controller design seems to be promising. The goal is to be able to use advanced PI and PID design methods based on frequency domain identification of relay experiments. The cost of using more advanced design methods is that the experiments and identification also need to be more advanced. Ziegler-Nichols-like tuning schemes need one relay experiment with approximately 5–10 switches. The method proposed here will typically need more switches using the time-varying relay. Thus, we have to pay for the improved design both in computation requirements and in time used for experiments. The former should impose no problem, but the latter may in many cases not be worth the cost compared to the simpler design methods.

This chapter has shown that it is feasible to use relay experiments with advanced design methods. However, there are of course many practical considerations to take into account before it is possible to use it as a fully automatic tuning procedure. Some of these issues will be discussed here.

Since the plant is not known in practice, the assessment of possible ranges for the hysteresis ε must be done with some kind of trail-and-error approach. One way could be to let the user decide. In any case, an intelligent scheme for changing ε has to be developed in order to ensure that unsuitable values are avoided. This can be done by having logic supervising the oscillation and take action when it stops. It will certainly not be sufficient to use the automaton in Figure 4.2 only, together with the simple scheme used here for changing the hysteresis. Today's commercial auto-tuners based on relay experiments already have an extensive amount of logic supervising the tuning experiments. This is discussed in more detail in Section 6.2.

In fact, the effects of all the experimental conditions discussed in Section 4.2 must be investigated further in order to handle all types of plants. When testing the feasibility of the method, different sets of parameters

have been used for different plants, both in the experiments, in the identification, and in the control design. By using some kind of adaptive parameter settings, it might be possible to define a robust automatic tuning procedure, similar to the standard relay auto-tuners available today.

When designing PID controllers, the constraint $T_i = 4T_d$ has been used in order to limit the derivative action. The main obstacle for using PID design methods based on constrained curvature of the loop transfer function is the sensitivity to noise. A parametric model would not have these problems, since the differentiations are computed analytically. One way of obtaining a model could be to use the interesting parts of the estimated frequency response as input to the methods described in Lilja (1989). However, the design methodology may then seem a little too complicated for tuning a PID controller. It is probably better to reformulate the design constraints not to involve computation of second order derivatives.

5

Fast Set Point Response

Most of the PID controllers in process industry work as regulators most of the time, *i.e.*, they try to keep a variable at a constant value. However, fast grade changes are often very important in many applications. This may be the case when a plant switches between two different production modes. The time and wasted material spent on these changes will reduce the profit of the plant. It is thus interesting to make the grade changes as fast and smooth as possible.

Unfortunately, PID control is not very well suited for handling step set point changes. Typically, the response is unnecessarily slow *and* has an overshoot. A common way of dealing with the overshoot is to ramp the set point between the two levels. Another possible way is to introduce set point weighting on the proportional part, see for example Åström and Hägglund (1995). A drawback with both these approaches is that the response is slowed down even further. Hang and Cao (1996) uses a time-varying set point weight to achieve both faster step response and less overshoot.

Another way to solve the problem is to split the control problem into a servo problem and a regulation problem. A simple method which has sometimes been used in practice, is to use only proportional control in the transient phase and turn on the integrating control when the error has become small. More advanced switching strategies may drastically improve the behavior. In Malmborg (1998) it was shown that very good results can be obtained by a hybrid strategy that combines a conventional PID controller with a minimum-time control strategy. The minimum-time strategy was based on a second order model. Methods for safe mixing of the strategies were also developed. A drawback with the minimum-time strategy used in Malmborg (1998) is that it requires a fairly accurate process model of second order.

An alternative sub-optimal solution is proposed in this chapter. The method tries to automate what experienced operators often do to make

fast set point changes. They typically switch to manual control and increase the control signal to a high constant value. When the output starts to move they have to decrease the control signal again in order to catch the process output so it approaches the new desired value in a nice way. Finally, they set the control signal to what they think is a good level and switch to automatic mode. Switching conditions for the proposed method will be derived in this chapter. An advantage is that the conditions may be adjusted to the current level of process knowledge.

5.1 Preliminaries

The solution which takes the system between two stationary levels in minimum time is the well known bang-bang control strategy. The number of switches between the extreme values in the optimal control law is less or equal to $n-1$, with n being the order of the system. The problem is very easy for a first order system

$$\frac{dy}{dt} = -ay + bu \tag{5.1}$$

We simply apply the maximum admissible control signal until the output reaches the desired value y_f. The control signal is then switched to the value that gives the desired steady state. If $u = ay_f/b$ is applied when $y = y_f$, Equation (5.1) gives that y will be constant for all future times.

For a second order system the time-optimal strategy is also very intuitive: start by "accelerating" as fast as possible, and then "break" just in time to make $\dot{y}(t) = 0$ when reaching the new desired stationary level. At the new level, $u(t)$ is set to the constant value which keeps the system at this level. One simulated example is found in Figure 5.1. It is also reasonably simple to find the conditions when to switch for $n = 2$. For $n > 2$ the time-optimal strategy consists of more switches, and the problem becomes increasingly difficult to solve.

Intuitively, the strategy with full acceleration and full break seems reasonable to use even for systems of higher order than two. The strategy will in general *not* bring the system to stationarity at the set point level in finite time. Still, it can be useful for set point changes in a mixed control strategy. As long as the new set point is approached smoothly enough, a PID controller can be switched in to achieve the desired steady state.

The underlying idea is that we could approximate the control signal with the one in Figure 5.2. The control signal is given by

$$u(t) = a\delta(t) + b\theta(t-T) \tag{5.2}$$

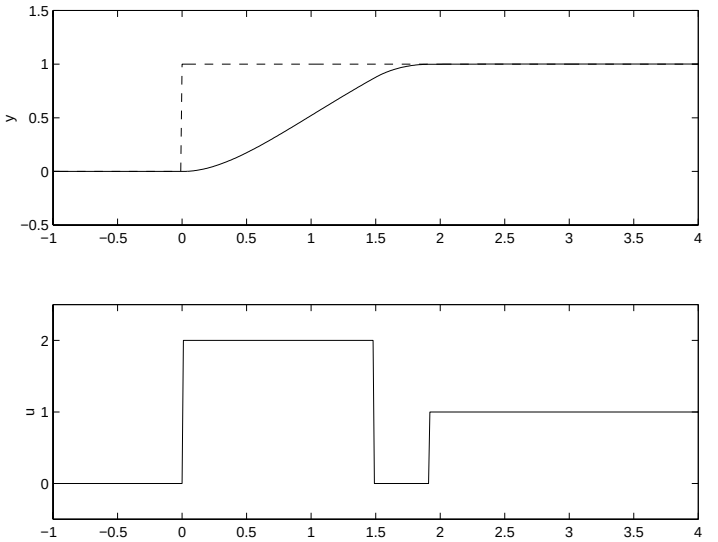


Figure 5.1 Time-optimal set point response for $G(s) = 1/(s+1)^2$ when $u \in [0, 2]$.

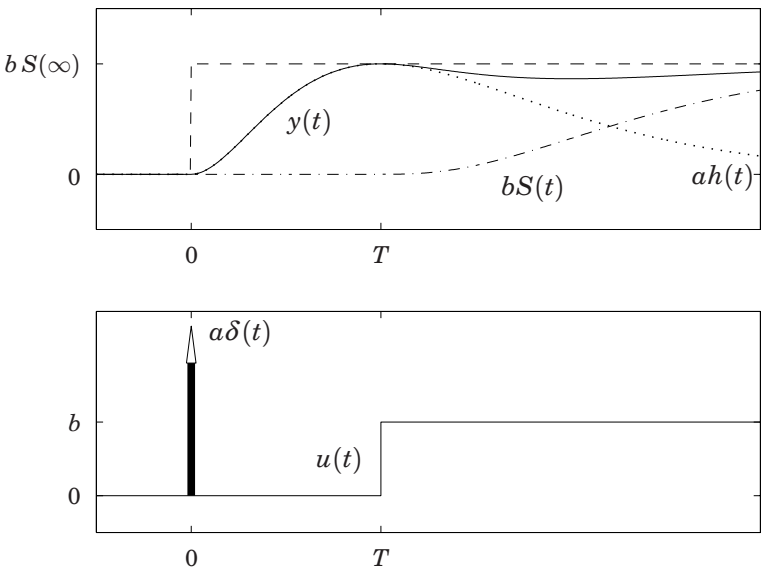


Figure 5.2 Response to an impulse and a delayed step.

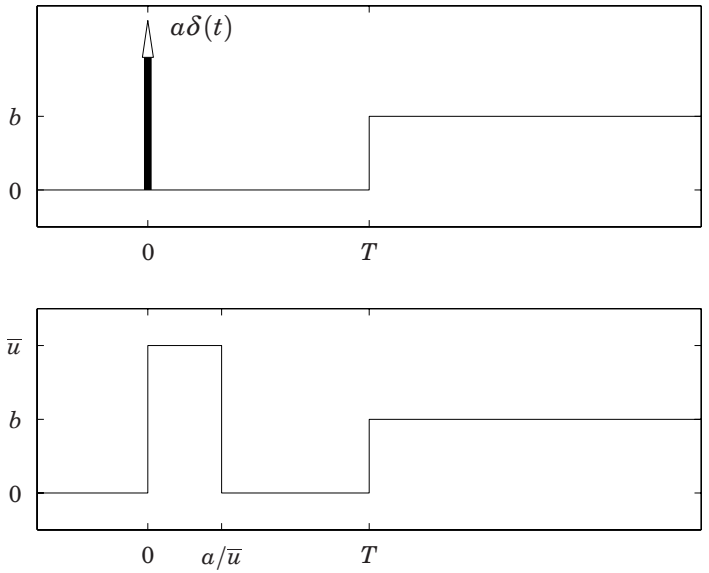


Figure 5.3 The ideal pulse-step control signal in the upper plot, and its realizable approximation below.

where $\delta(t)$ is the Dirac impulse function, $\theta(t)$ is the Heaviside step function, and a, b and T are constants. The output then becomes

$$y(t) = ah(t) + bS(t - T) \tag{5.3}$$

with $h(t)$ and $S(t)$ being the impulse response and step response, respectively. Due to the shape of the control signal in Figure 5.2 we will from now on use the name *pulse-step method* for the proposed algorithm. If the parameters a, b and T are chosen optimally, the settling time is approximately equal to the time where the impulse response has its maximum. The response time is thus matched to the system dynamics. The parameters a, b and T may be chosen as follows:

- b should be chosen to obtain a correct steady state.
- a should be chosen such that the impulse response $ah(t)$ has a maximum value close to the new set point.
- T should be chosen such that the step response and the trailing edge of the impulse response approximately add up to the new set point.

In practice, the impulse of course has to be replaced by a pulse of finite amplitude. The pulse width should be adjusted to give approximately the

same area as the impulse has. The ideal pulse-step signal and the approximated, realizable, control signal are shown in Figure 5.3.

The approximation with one impulse and one step is reasonable when the control signal has the same value immediately before and after the impulse. The response will however be faster if the control signal is allowed to go below the initial level. In the problem formulation below, we will thus not use the approximation but a more direct approach. However, the idea of the pulse-step method is still tractable. The impulse feeds the system with enough energy to reach the final value. The system then responds with its "natural" speed. A minimum-time strategy may use more pulses to inject more energy into the system. The response is faster than the natural response time, but it will be less robust due to the additional energy that is built up inside the system. This relates to the result that a minimum-energy strategy may be preferred to a minimum-time strategy, for example when swinging up pendulums, see Åström and Furuta (2000).

5.2 Problem Formulation

The problem setup for the pulse-step method will now be defined. Here, we study self regulating, or exponentially stable, linear time-invariant processes. Consider the set point change from y_0 to $y_0 + \Delta y$ for a process $G(s)$. If $G(0) = K$, the stationary control signal will then change from u_0 to $u_0 + \Delta y/K$. With Δy and K positive the control strategy will then be as shown in Figure 5.4. If $\Delta y/K < 0$, the control strategy should of course be reversed.

By scaling the signals and removing bias levels we may consider a normalized problem without loss of generality. The transfer function is

$$G_0(s) = \frac{B(s)}{A(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \ldots + b_{m-1}s + 1}{a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1}s + 1}, \tag{5.4}$$

the set point change is

$$y_{sp}(t) = \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases} \tag{5.5}$$

and the control signal is

$$u(t) = \begin{cases} 0, & t < 0 \\ \overline{u}, & 0 \leq t < T_1 \\ \underline{u}, & T_1 \leq t < T_1 + T_2 \\ 1, & t \geq T_1 + T_2 \end{cases} \tag{5.6}$$

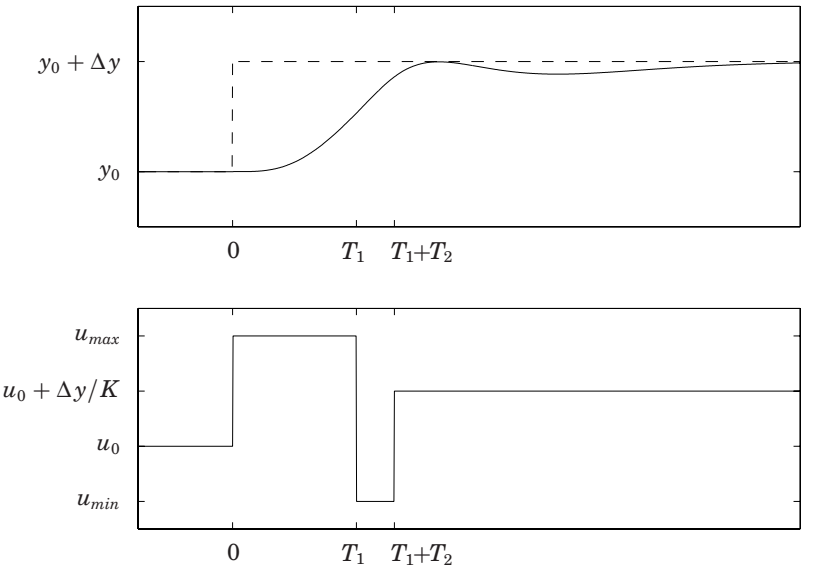


Figure 5.4 Control strategy for fast set point response.

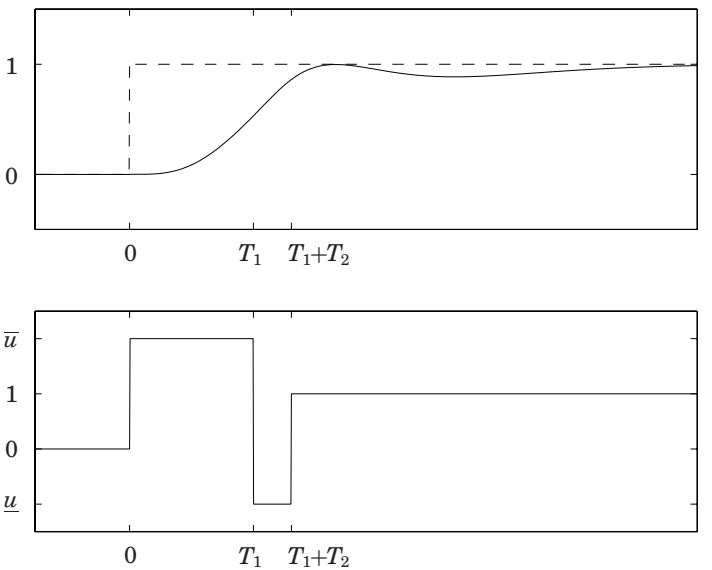


Figure 5.5 Control strategy for the normalized problem setup.

where $\overline{u}$ and $\underline{u}$ are the maximum and minimum values of the control signal in the normalized problem, see Figure 5.5. Note that the transfer function does not include any dead time. This is not necessary, since the optimal switching times for a time-delayed process is the same as for the one without delay. Also note that the problem formulation is meaningful only if $\overline{u} \geq 1$, $\underline{u} \leq 0$, $T_1 \geq 0$ and $T_2 \geq 0$. When translating the original problem to the normalized setup, the maximum and minimum levels of the control signal are transformed as

$$\begin{aligned} \overline{u} &= \frac{K}{\Delta y}\left(u_{max} - u_0\right) \\ \underline{u} &= \frac{K}{\Delta y}\left(u_{min} - u_0\right) \end{aligned}$$

if $K/\Delta y > 0$. Otherwise, $\overline{u}$ and $\underline{u}$ will just change place in the formulas above.

The chosen criterion to minimize is the integrated error

$$IE = \min_{T1,T2} \int_0^\infty \Big(y_{sp}(t) - y(t)\Big)dt \tag{5.7}$$

subject to the constraint

$$y(t) \leq 1, \qquad \forall t \tag{5.8}$$

i.e., no overshoot in the set point step response.

With $S(t)$ being the unit step response for $G_0(s)$, the output $y(t)$ can be written as

$$y(t) = \overline{u}S(t) + (\underline{u} - \overline{u})S(t - T_1) + (1 - \underline{u})S(t - (T_1 + T_2)) \tag{5.9}$$

If $G_0(s)$ is asymptotically stable, IE can be calculated using the final value theorem:

$$\begin{aligned} IE &= \lim_{t\to\infty} \int_0^t \Big(y_{sp}(\tau) - y(\tau)\Big)d\tau = \lim_{s\to 0}\Big(Y_{sp}(s) - Y(s)\Big) = \\ &= \lim_{s\to 0}\left(\frac{1}{s} - G_0(s)U(s)\right) \end{aligned}$$

A series expansion for small s gives

$$\begin{aligned} U(s) &= \frac{\overline{u}}{s} + \frac{\underline{u} - \overline{u}}{s}e^{-sT_1} + \frac{1 - \underline{u}}{s}e^{-s(T_1+T_2)} = \\ &\approx \frac{1}{s}\Big(\overline{u} + (\underline{u} - \overline{u})\left(1 - sT_1\right) + (1 - \underline{u})\left(1 - s\left(T_1 + T_2\right)\right)\Big) = \\ &= \frac{1}{s} - (1 - \overline{u})\,T_1 - (1 - \underline{u})\,T_2 \end{aligned}$$

and

$$G_0(s) \approx 1 + s\,(b_{m-1} - a_{n-1}) = 1 - sT_{ar}$$

where T_{ar} is the *average residence time* of the system. Substituting this into the expression for *IE* gives

$$\begin{aligned} IE &= \lim_{s\to 0}\left(\frac{1-G_0(s)}{s} + G_0(s)\big((1-\overline{u})\,T_1 + (1-\underline{u})\,T_2\big)\right) \\ &= T_{ar} + \underbrace{(1-\overline{u})}_{<0}\,T_1 + \underbrace{(1-\underline{u})}_{>0}\,T_2 \end{aligned} \tag{5.10}$$

The integrated error thus consists of one part T_{ar} which depends on the process model, and one part $(1-\overline{u})T_1 + (1-\underline{u})T_2$ which depends on the control strategy. Since $(1-\overline{u}) < 0$ and $(1-\underline{u}) > 0$, T_1 should be large and T_2 small in order to make *IE* small. If T_1 is made too large, though, the zero overshoot constraint will clearly not be met. Optimal T_1 and T_2 may be found numerically.

Optimality conditions

Since the objective function is linear in the parameters T_1 and T_2, it is clear that the constraint must be active at the optimal solution $(T_1{}^\star,\, T_2{}^\star)$. Thus, there exists a time $t = t^\star$ with $y(t^\star) = 1$. Note that $t^\star$ may be infinite in some cases, for example if $\overline{u} = 1$. Furthermore, if $\deg A(s) - \deg B(s) \geq 2$, both $y(t)$ and $\dot{y}(t)$ will be continuous, so it will also hold that $\dot{y}(t^\star) = 0$. Introducing

$$f_1(t, T_1, T_2) = \overline{u}\,S\,(t) + (\underline{u} - \overline{u})\,S\,(t - T_1) + (1-\underline{u})\,S\,(t - (T_1 + T_2)) - 1 \tag{5.11}$$

and

$$f_2(t, T_1, T_2) = \overline{u}\,h\,(t) + (\underline{u} - \overline{u})\;h\,(t - T_1) + (1-\underline{u})\;h\,(t - (T_1 + T_2)) \tag{5.12}$$

we thus have two conditions for optimality

$$\begin{cases} f_1(t^\star, T_1{}^\star, T_2{}^\star) = 0 \\ f_2(t^\star, T_1{}^\star, T_2{}^\star) = 0 \end{cases} \tag{5.13}$$

The pairs $(T_1,\, T_2)$ for which $\max y(t) = 1$ implicitly define a curve in the $(T_1,\, T_2)$ -plane, say $F(T_1,\, T_2) = 0$. F can be seen as the envelope of

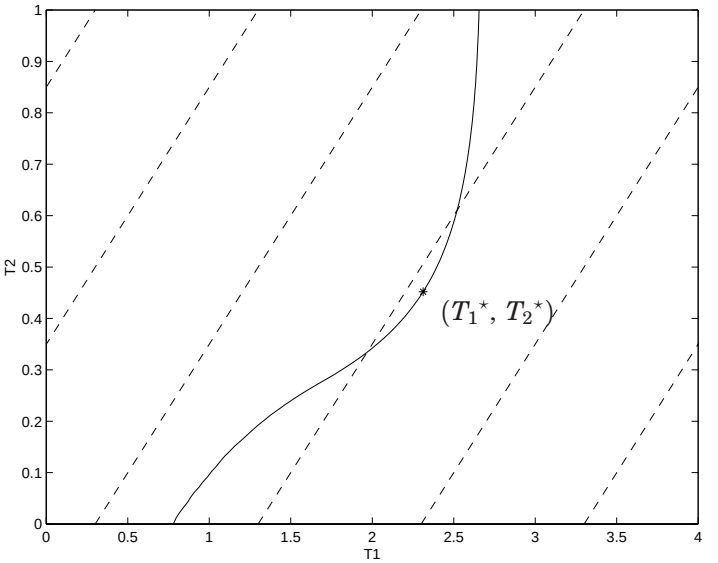


Figure 5.6 (T_1, T_2) on the solid line will give $\max y(t) = 1$ for $G_0(s) = 1/(s+1)^4$ with $\overline{u} = 2$ and $\underline{u} = -1$. The dashed lines are level curves of *IE*.

$f_1 = 0$ when t is varying. Thus, $F(T_1, T_2) = f_1(\hat{t}, T_1, T_2)$ where $\hat{t}$ satisfies

$$\begin{cases} f_1(\hat{t}, T_1, T_2) = 0 \\ f_2(\hat{t}, T_1, T_2) = 0 \end{cases}$$

The optimal solution will lie on this curve at the point where *IE* is minimized. Figure 5.6 shows one example of such a curve. Points to the left of the curve correspond to zero overshoot responses. Notice that this region defines a non-convex set. The value of *IE* decreases from the top left to the bottom right corner. Provided that the curve F is smooth at the optimum, it is clear that the tangent to F at the optimum will be parallel to the level curves of *IE*. This will give a third condition for the optimal solution. By setting *IE* constant we obtain

$$\frac{dT_2}{dT_1} = \frac{\overline{u} - 1}{1 - \underline{u}}$$

for the level curves. Using the implicit function theorem, the slope of F

is given by

$$\begin{aligned}\frac{dT_2}{dT_1} &= -\frac{\partial F(T_1,\,T_2)}{\partial T_1}\Big/\frac{\partial F(T_1,\,T_2)}{\partial T_2} = -\frac{\partial f_1(\hat{t},T_1,T_2)}{\partial T_1}\Big/\frac{\partial f_1(\hat{t},T_1,T_2)}{\partial T_2} = \\ &= -\frac{-(\underline{u}-\overline{u})\;h\left(\hat{t}-T_1\right)-(1-\underline{u})\;h\left(\hat{t}-(T_1+T_2)\right)}{-(1-\underline{u})\;h\left(\hat{t}-(T_1+T_2)\right)} = \\ &= -\frac{(\underline{u}-\overline{u})\;h\left(\hat{t}-T_1\right)}{(1-\underline{u})\;h\left(\hat{t}-(T_1+T_2)\right)}-1\end{aligned}$$

Equating the expressions for the slope at the optimal solution we get

$$\frac{\overline{u}-1}{1-\underline{u}} = -\frac{(\underline{u}-\overline{u})\;h\left(t^\star-{T_1}^\star\right)}{(1-\underline{u})\;h\left(t^\star-({T_1}^\star+{T_2}^\star)\right)}-1$$

which gives the condition

$$h\left(t^\star-{T_1}^\star\right) = h\left(t^\star-({T_1}^\star+{T_2}^\star)\right) \tag{5.14}$$

With $f_3(t,T_1,T_2) = h\left(t^\star-{T_1}^\star\right)-h\left(t^\star-({T_1}^\star+{T_2}^\star)\right)$ the optimal solution will thus satisfy

$$\begin{cases} f_1(t^\star,{T_1}^\star,{T_2}^\star)=0 \\ f_2(t^\star,{T_1}^\star,{T_2}^\star)=0 \\ f_3(t^\star,{T_1}^\star,{T_2}^\star)=0 \end{cases} \tag{5.15}$$

If we know $G_0(s)$, we may derive analytical expressions for f_1, f_2 and f_3. Equation (5.15) then reduces to a system of non-linear equations.

Remark 1: A sufficient condition for the existence of an optimal solution is that $S(t) \leq 1$, $\forall t$. $(T_1,\,T_2) = (0,\,0)$ will then be a feasible solution with $IE = a_{n-1} - b_{m-1}$. *IE* may be smaller than this by selecting $T_1 > 0$. However, if $\overline{u} > 1$, there exists a time T_{1max} such that $\overline{u}S(t)$ becomes greater than 1 at $t = T_{1max}$. This gives a lower (mostly unachievable) bound on $IE = a_{n-1} - b_{m-1} + T_{1max}\,(1-\overline{u})$. When $\overline{u} = 1$, any $(T_1,\,T_2) = (T_1,\,0)$ will of course be optimal.

Remark 2: Alternative criteria such as the integrated absolute error (*IAE*) or the integrated squared error (*ISE*) could of course also be used. A drawback is that it is harder to obtain explicit formulas for these. A solution strategy could then be to start with a large vector S containing the unit step response, and then find optimal T_1 and T_2 by shifting and superposing S. This will typically require more operations than the proposed method for a given accuracy of the solution.

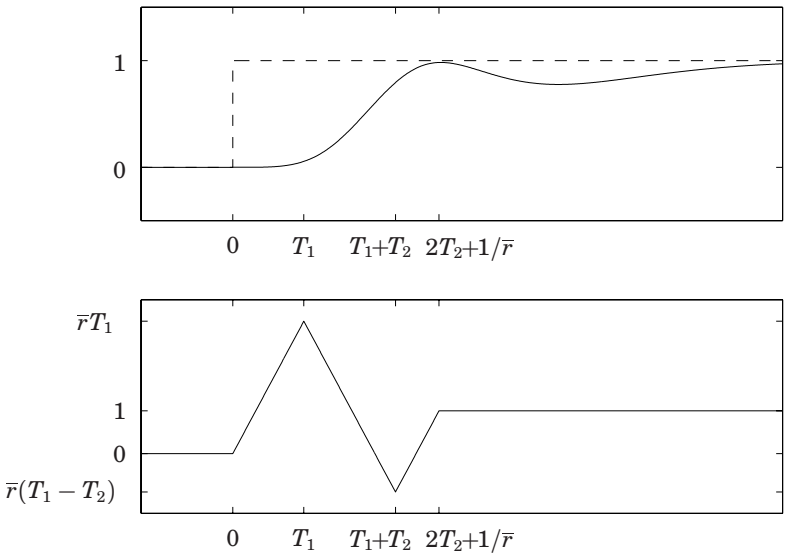


Figure 5.7 Control strategy with rate limitations on the control signal.

Rate limitations

So far, only limitations on the size of the control signal have been discussed. It is also common to have limitations on the maximum slope of the control signal. The pulse-step method may be modified in order to handle these rate limitations.

First, we consider the case where the control signal is limited by rate constraints only. Assume that the control signal should satisfy the constraint

$$|\dot{u}(t)| \leq \overline{r} \tag{5.16}$$

for all values of t, but there is no constraint on the size of the control signal. A natural modification of the pulse-step method is then to use the control strategy in Figure 5.7. The expression for $u(t)$ is given by

$$u(t) = \begin{cases} 0, & t < 0 \\ \overline{r}t, & 0 \leq t < T_1 \\ \overline{r}(2T_1 - t), & T_1 \leq t < T_1 + T_2 \\ \overline{r}(1 - 2T_2 - 1/\overline{r} + t), & T_1 + T_2 \leq t < 2T_2 + 1/\overline{r} \\ 1, & t \geq 2T_2 + 1/\overline{r} \end{cases} \tag{5.17}$$

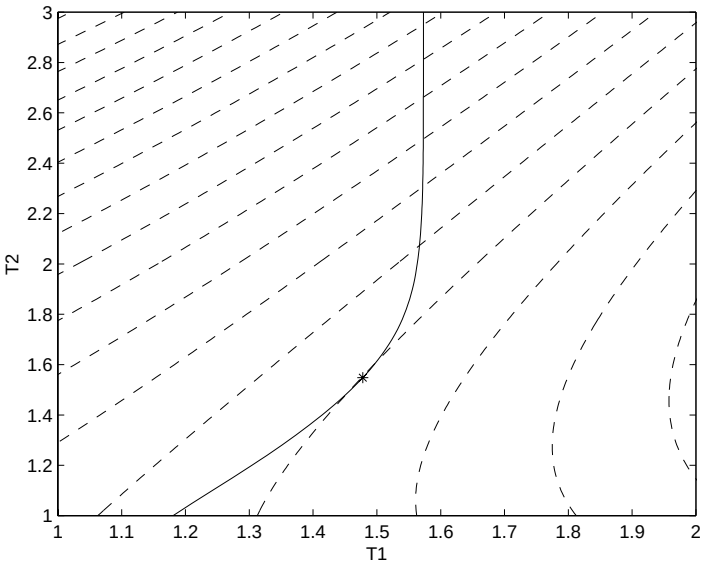


Figure 5.8 (T_1, T_2) on the solid line give $\max_t y(t) = 1$ under the rate constraint $|\dot{u}(t)| \leq 2$ for $G(s) = 1/(s+1)^4$. The dashed lines are level curves of *IE*.

There is an implicit requirement $T_2 \geq T_1 - 1/\overline{r}$ in order to have the ramp in the negative direction to pass the final value. Similar to the pulse-step strategy presented before, this strategy tries to achieve maximum acceleration followed by maximum retardation. Furthermore, the strategy coincides with the time-optimal strategy under rate limitations for a second order system. This can be seen by rewriting the problem such that $\dot{u}(t)$ is the control signal and $G(s)/s$ is the process.

It is possible to solve for T_1 and T_2 in a similar way as for the pulse-step method. The optimality conditions will however be different since the expressions for the output $y(t)$ and *IE* are different. The output is given by

$$y(t) = \overline{r}\,(R(t) - 2R(t - T_1) + 2R(t - T_1 + T_2) - R(t - 2T_2 - 1/\overline{r})) \tag{5.18}$$

where $R(t)$ is the unit ramp response of the process. Using the final value theorem it is straightforward to calculate the integrated error as

$$IE = r{T_2}^2 - 2rT_2T_1 + 2T_2 - b_1 + a_1 + \frac{1}{2r} \tag{5.19}$$

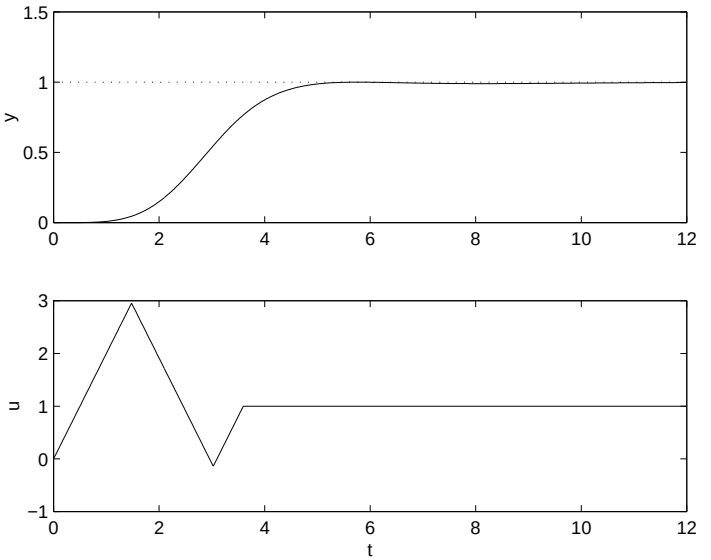


Figure 5.9 Control strategy with the rate constraint $|\dot{u}(t)| \leq 2$ for the process $G(s) = 1/(s+1)^4$. $T_1 = 1.48$ and $T_2 = 1.55$.

Since *IE* is not linear in T_1 and T_2, the level curves will not be straight lines, as was the case for the pulse-step method. However, it is still possible to form and solve the system of three equations representing the zero overshoot constraint, the zero derivative condition and the tangency condition, respectively. The envelope of the zero overshoot constraint together with level curves of *IE* are shown in Figure 5.8 for $G = 1/(s+1)^4$ and $\overline{r} = 2$. The pair (T_1, T_2) marked by a star fulfills the optimality conditions. The corresponding optimal control strategy is shown in Figure 5.9.

The above strategy takes only rate limitations into consideration. This may cause the control signal to become very large. To overcome this, it may be necessary to combine the rate and level constraints on the control signal. When this is done, the shape of the control signal will be different, depending on which constraints are actually dominant, see Figure 5.10. It is possible to find the optimality conditions for all of the different shapes, but this is not shown here. The strategy in the upper left plot corresponds to the case where the original pulse-step method is modified by a moderate rate constraint. If the control signal is allowed to change fast compared to the time scale of the process, the rate constraints may be neglected, and the result from the pulse-step method may be applied directly. A minor modification can be made by applying the ramps in advance such that they reach half of the ramp interval just in time when the steps computed

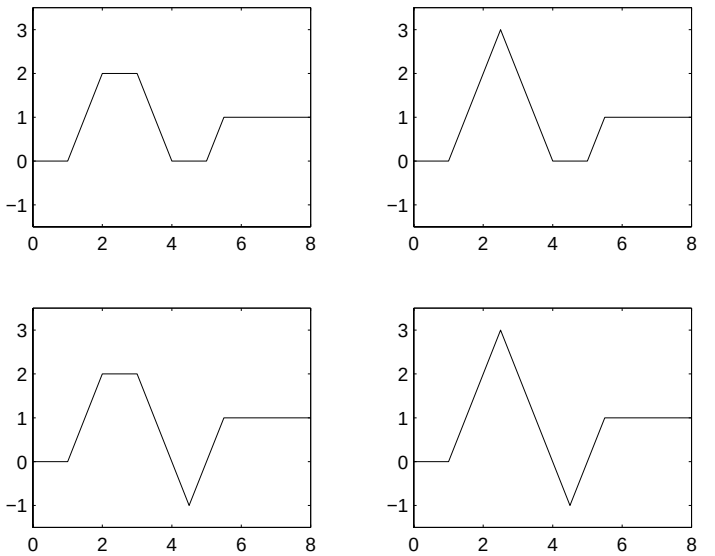


Figure 5.10 Different shapes of the control signal when rate and level constraints are combined.

by the pulse-step method should have been applied. However, if the rate limitations are more dominant, they should be taken into account when calculating the control strategy.

5.3 Evaluation

In this section the feasibility of the pulse-step method will be explored. The proposed method may be applied to many types of transfer functions, particularly those where PID control is normally used. Especially higher order systems are of interest, including applications involving heating. The method will first be compared with both true time-optimal control, and to PID control using set point weighting. A few examples where the method does not perform very good will also be given.

Comparison with time-optimal control

Since the control signal generated by the pulse-step method is similar to time-optimal control, it is interesting to compare the two methods. For a system with two real poles and no zeros, the proposed method will actually result in a time-optimal strategy. To see this, first note that the

time-optimal strategy is a valid control strategy, since it has no overshoot. Furthermore, it reaches its final value exactly at the last switching point, *i.e.*, $t^\star = T_1 + T_2$. The only way to reduce *IE* further would be to increase T_1 and/or decrease T_2. However, both these actions would inevitably violate the zero overshoot constraint.

For higher order systems without zeros, the time-optimal strategy will perform better than the sub-optimal strategy, both with respect to the time measure and to the *IE* measure. For systems with equal lags, the difference between the methods increases with the order of the system. Just to give an example, we will compare the two strategies for the process

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

Since it is a fourth order system, the time-optimal strategy will have two more switches than the pulse-step strategy.

Figure 5.11 shows comparisons with time-optimal control. Each group of three plots shows the behavior for one pair $(\underline{u}, \overline{u})$. The upper plot in each group shows the output for the time-optimal strategy (full) and the pulse-step method (dashed). The middle plot shows the time-optimal control signal, and the lower plot shows the pulse-step control signal. The figure indicates that the relative merit of the time-optimal strategy increases as the size of $\underline{u}$ and $\overline{u}$ increase. The reason for this is that much more energy is injected into the system, see for example the case with $\overline{u} = -\underline{u} = 16$. As one might expect, Figure 5.11 also shows that $\overline{u}$ is the more important parameter. Not very much is gained from a large negative $\underline{u}$, especially not with $\overline{u}$ small.

The rise time, here defined as the first time when $y(t) = 1$ and $\dot{y}(t) = 0$, as a function of $\overline{u}$ is shown in Figures 5.12 and 5.13. Figure 5.12 shows the rise time when the input range is $[-\overline{u}, \overline{u}]$. Asymptotically, the rise time goes to zero for the time-optimal strategy, but to positive constant value for the pulse-step strategy. If the input range is instead $[0, \overline{u}]$, the rise time converges to a positive constant for both strategies as seen in Figure 5.13. The integrated error depends on the sizes of the control signal in a similar way.

These asymptotic comparisons are, however, of no or little practical relevance, since you should probably not use all the available input range for a relatively small set point change. In practice one would instead limit the used input range in some way. This will be further discussed on page 132.

Comparison with PID control

In this section we will compare the proposed method with the set point step response using PID control. Since the main task for a PID controller

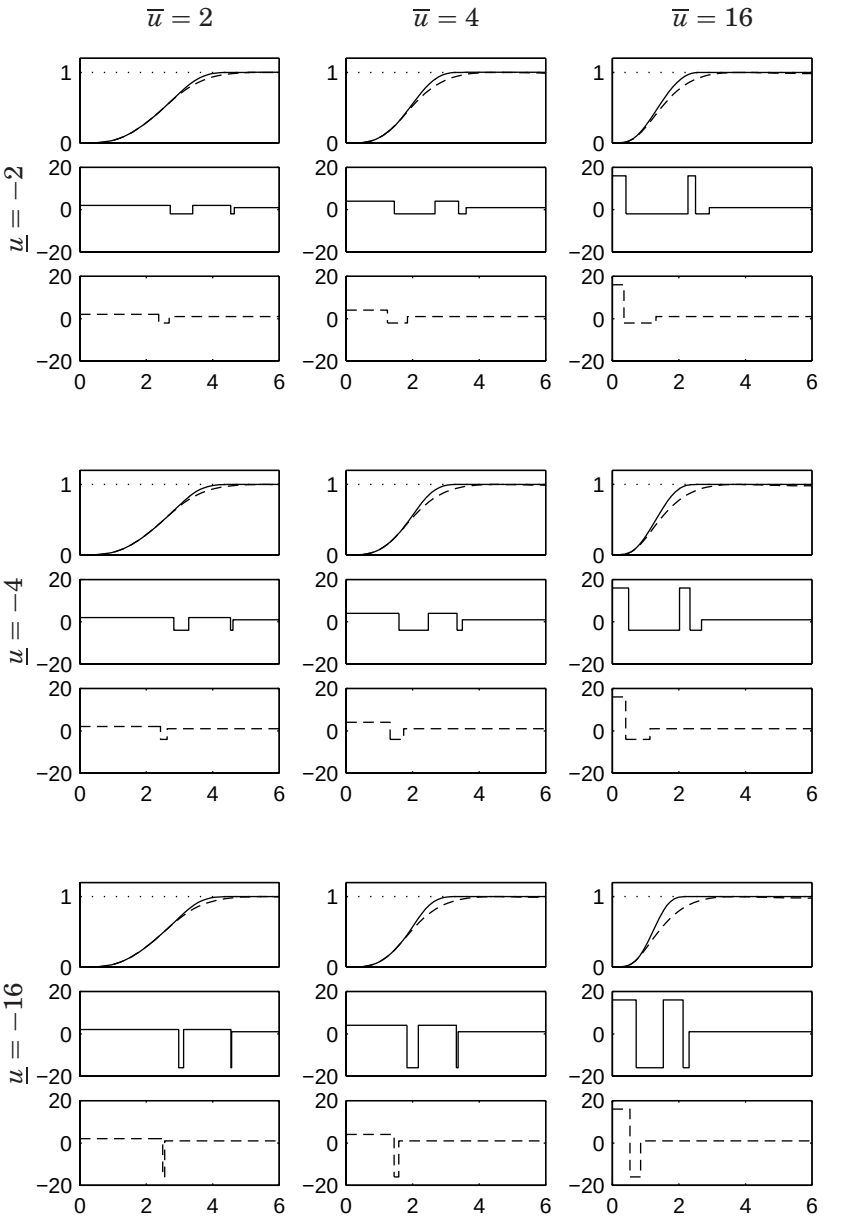


Figure 5.11 Comparison between the time-optimal controller (solid lines) and pulse-step strategy (dashed lines) for $G(s) = 1/(s+1)^4$. The benefit of large negative control signals is marginal, especially when combined with small positive control signals.

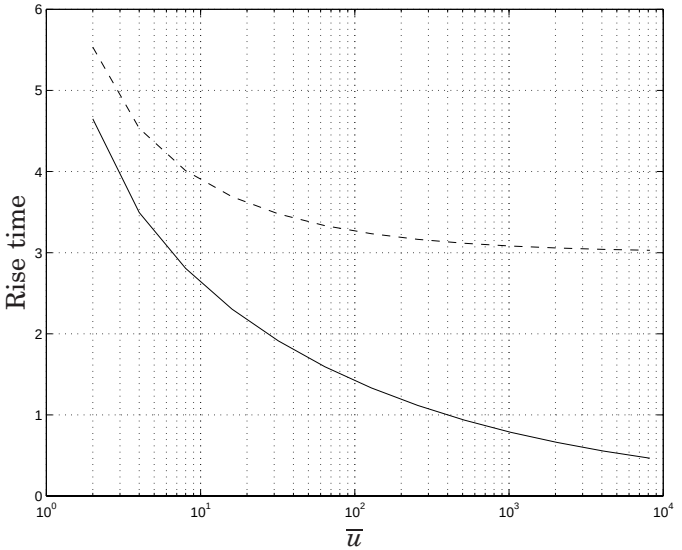


Figure 5.12 Rise time as a function of the magnitude of the control signal ($\underline{u} = -\overline{u}$) with time-optimal controller (solid line) and the pulse-step strategy (dashed line) for $G(s) = 1/(s+1)^4$.

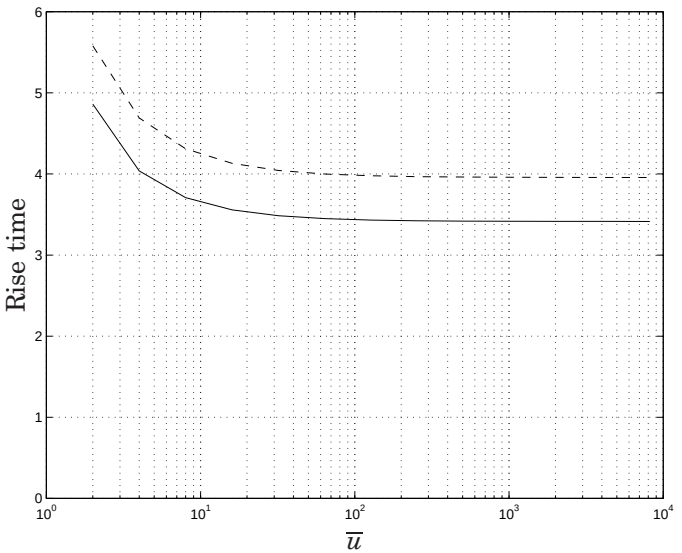


Figure 5.13 Rise time as a function of the magnitude of the control signal ($\underline{u} = 0$) with time-optimal controller (solid line) and the pulse-step strategy (dashed line) for $G(s) = 1/(s+1)^4$.

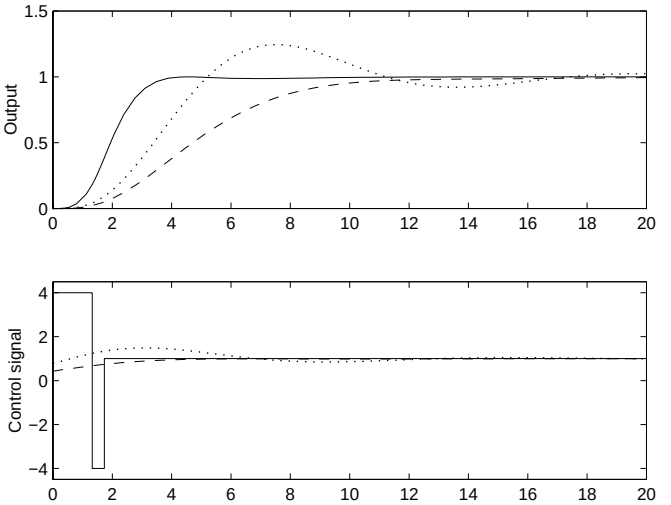


Figure 5.14 Comparison between the fast set point response strategy (full) and PI control with $M_s = 1.4$ (dashed) and $M_s = 2.0$ (dotted) for $G(s) = 1/(s+1)^4$.

is regulation, its parameters K, T_i and T_d are typically tuned to give nice response to load disturbances, see for example Åström *et al.* (1998) and Ho *et al.* (1992). Generally, the PID controller will produce a slow set point step response, often also with an overshoot. If set point weighting is introduced, the overshoot may be reduced, at the expense of even slower response. In Åström *et al.* (1998), the set point weighting factor b is chosen to set the maximum gain from set point to output close to 1, with the constraint $0 \leq b \leq 1$.

Figure 5.14 shows a comparison between the fast set point response method and two different PI settings. The fast set point response has been computed with $\overline{u} = 4$ and $\underline{u} = -4$, and the resulting rise time and settling time are approximately 4 time units. The two PI designs have been designed according to the method in Åström *et al.* (1998) with maximum sensitivity $M_s = 1.4$ and $M_s = 2.0$, respectively. The corresponding controller parameters are $K = 0.43$, $T_i = 2.25$ and $b = 1$ for $M_s = 1.4$, and $K = 0.78$, $T_i = 2.05$ and $b = 0.23$ for $M_s = 2.0$. Both PI designs are clearly outperformed by the pulse-step method. The rise times are a factor 2–3 higher and the settling times approximately 3 times higher. The reason is of course that much less of the available control authority is used. If b and/or M_s is increased, the size of the control signal will increase. This leads to a faster rise time, but at the expense of larger overshoot, so the

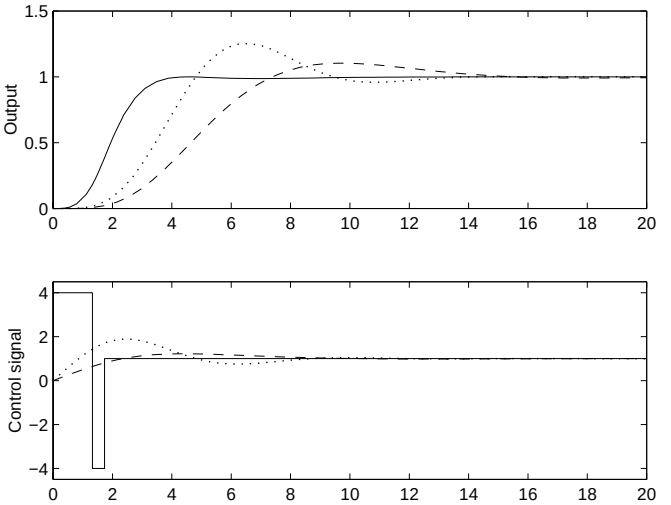


Figure 5.15 Comparison between the fast set point response strategy (dashed) and PID control with $M_s = 1.4$ (full) and $M_s = 2.0$ (dotted) for $G(s) = 1/(s+1)^4$.

settling time may actually be even higher.

As shown in Panagopoulos (1998), this example uses one of the types of processes where you get a noticeable improvement of the load disturbance response by introducing derivative action. However, the set point response is not improved. Figure 5.15 shows the step responses with PID design according to Panagopoulos (2000). The parameters are $K = 1.14$, $T_i = 2.23$, $T_d = 1.00$ and $b = 0$ for $M_s = 1.4$ and $K = 2.29$, $T_i = 1.92$, $T_d = 0.98$ and $b = 0$ for $M_s = 2.0$. Both responses have higher overshoot and longer settling times than the corresponding PI designs. The relative benefits of using the fast set point response is thus even larger for the PID case.

Comparisons done for other processes show similar results. They all indicate that you may actually gain much in performance if you would combine the PI(D) regulatory control with a strategy for fast set point changes. Possible implementation structures for this are discussed later.

Infeasible transfer functions

The pulse-step method is applicable for many processes, but unfortunately not for all. The method is inspired by the time-optimal control strategy. When that is not performing well, neither will the pulse-step method. A few examples of systems where the method performs badly will now be given.

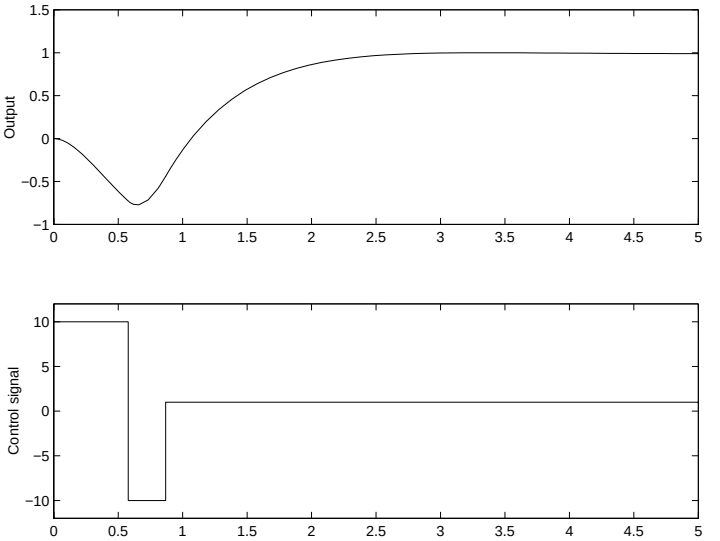


Figure 5.16 Performance of the pulse-step method for the non-minimum phase process in Example 5.1.

EXAMPLE 5.1—NON-MINIMUM PHASE PROCESS
Consider the non-minimum phase transfer function

$$G(s) = \frac{1-s}{(s+1)^3}$$

with $\overline{u} = 10$ and $\underline{u} = -10$. The resulting response of the pulse-step method is shown in Figure 5.16. The large undershoot, -0.77, is of course not desirable. The reason for the bad performance is the non-minimum phase behavior combined with the large input signal. If the control signal is limited to a lower value, the response will be nicer but also slower. □

EXAMPLE 5.2—PROCESS WITH STABLE ZERO
Consider a system with the transfer function

$$G(s) = \frac{2s+1}{(s+1)^2} = \frac{1}{s+1} \cdot \frac{2s+1}{s+1}$$

with stable zeros and pole excess 1. The transfer function can be factored as seen above where the second factor containing the zero can be inverted. The system is essentially of first order. The optimal strategy for a first

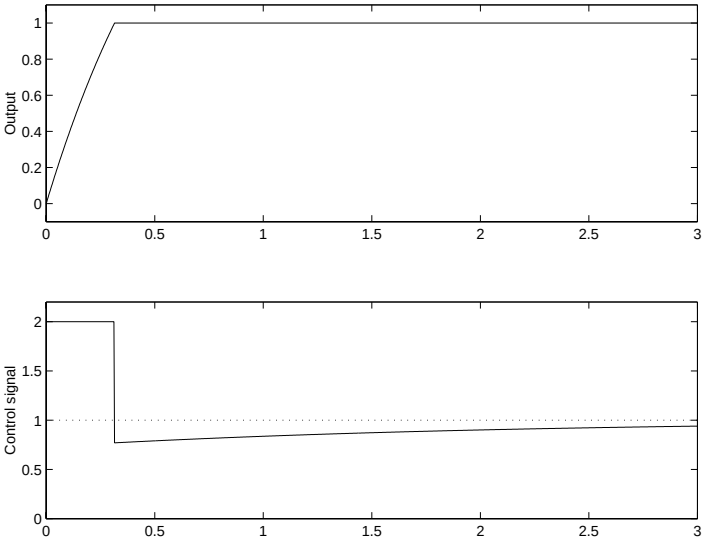


Figure 5.17 Alternative fast set point response strategy used in for the plant with overshoot in Example 5.2.

order system may thus be used, with a slight modification because of the need to invert the transfer function $(2s+1)/(s+1)$. This strategy is shown in Figure 5.17.

An attempt to apply the pulse-step method directly on the second order system $G(s)$ results in responses with overshoot. From Equation (5.9), we may write the output and its derivative as

$$\begin{aligned} y(t) &= \overline{u}\, S(t) + (\underline{u} - \overline{u})\; S(t - T_1) + (1 - \underline{u})\; S(t - (T_1 + T_2)) \\ \dot{y}(t) &= \overline{u}\, h(t) + (\underline{u} - \overline{u})\; h(t - T_1) + (1 - \underline{u})\; h(t - (T_1 + T_2)) \end{aligned}$$

where the step and impulse responses are

$$\begin{aligned} S(t) &= 1 + (t-1)\, e^{-t} \\ h(t) &= (2-t)\, e^{-t} \end{aligned}$$

for $t \geq 0$. It is clear that the step response will be greater than 1 for $t > 1$. We will now show that there is no way to choose $\overline{u}$, $\underline{u}$, T_1 and T_2 such that y does not have an overshoot.

If $y(t)$ should not have an overshoot, it must approach 1 from below,

i.e. $\dot{y}(t)$ must be positive as t goes to infinity. Thus we examine

$$\begin{aligned}\lim_{t\to\infty}\left(\dot{y}(t)\frac{e^t}{t}\right) &= -\overline{u}-(\underline{u}-\overline{u})\,e^{T_1}-(1-\underline{u})\,e^{T_1+T_2}= \\ &= -\overline{u}-e^{T_1}\left(-\overline{u}+\underline{u}+e^{T_2}\,(1-\underline{u})\right)\end{aligned}$$

For this expression to be positive, the factor multiplied by e^{T_1} must be negative and T_1 must be sufficiently large. This leads to the conditions

$$\begin{aligned}T_2 &< \ln\frac{\overline{u}-\underline{u}}{1-\underline{u}} \\ T_1 &> \ln\frac{\overline{u}}{\overline{u}-\underline{u}-e^{T_2}\,(1-\underline{u})}\end{aligned}$$

However, choosing T_1 too large will make $y(T_1)>1$. T_1 will be as low as possible for $T_2=0$, which gives

$$T_1 > \ln\frac{\overline{u}}{\overline{u}-1}\equiv T_{1min}$$

This gives after simplifications

$$y(T_{1min})=\overline{u}S(T_{1min})=1+\ln\left(\frac{\overline{u}}{\overline{u}-1}\right)(\overline{u}-1)$$

which is greater than 1 for all $\overline{u}>1$. Thus there exists no feasible solution to the optimization problem for this process.

One way to deal with this could be to relax the zero overshoot constraint. This is easily done by replacing the condition $y\le 1$ with $y\le y_{max}$ in the derivations in Section 5.2. For y_{max} close to 1, it will still give good results, but it is less obvious that it is good idea to minimize *IE*.

□

EXAMPLE 5.3—SLUGGISH STEP RESPONSE

Heat conduction in an infinite rod where one endpoint is heated and the temperature at a fixed distance is measured can be modeled by a partial differential equation, see for example Åström (1969). It corresponds to the non-rational transfer function

$$G(s)=e^{-\sqrt{s}}$$

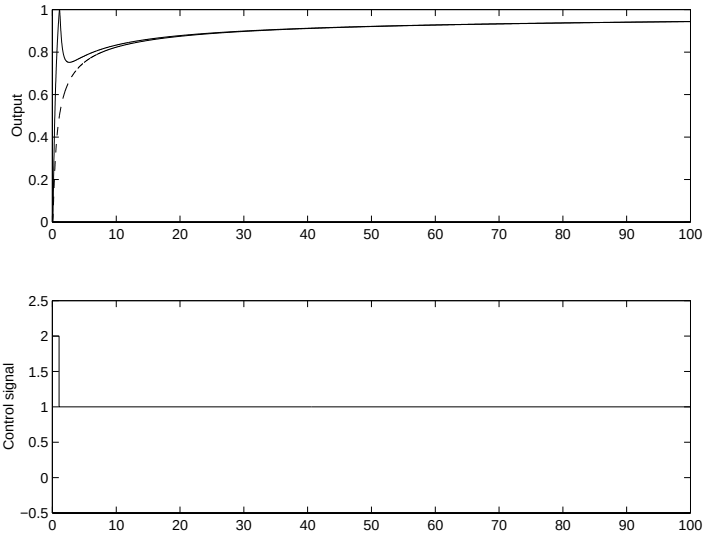


Figure 5.18 Open-loop step response (dashed) and pulse-step set point response (full) for the process in Example 5.3.

where all coefficients have been normalized. The impulse and step responses of $G(s)$ for $t \geq 0$ are given by

$$\begin{aligned} h(t) &= \frac{e^{-1/(4t)}}{2\sqrt{\pi}t^{3/2}} \\ S(t) &= 1 - \operatorname{erf}\left(\frac{1}{2\sqrt{t}}\right) = 1 - \frac{2}{\sqrt{\pi}} \int_0^{1/(2\sqrt{t})} e^{-\tau^2} d\tau \end{aligned}$$

The step response is the dashed curve in Figure 5.18. The initial slope is rather steep, but it approaches 1 extremely slowly. The sluggish step response carries over to the response for the pulse-step method, see the full lines in Figure 5.18. The fast initial slope of the step response forces T_1 to be comparatively small, approximately 1.05 for $\overline{u} = 2$ and $\underline{u} = 0$. The step in the negative direction will then make the output go away from the set point and approach the open-loop step response.

A similar strategy to the one proposed in the previous example could be used here as well. A finite-dimensional approximation of $G(s)$ gives a transfer function with interlaced zeros and poles on the negative real axis. By having $u(t)$ governed by these zero dynamics, it is possible to get both fast initial response and immediately close following of the set point. □

It seems that the process zeros impose the most severe limitations, at least when they are closer to the origin than the dominant poles are. For such processes much better performance may be achieved by letting u be generated by some (approximate) inverse of the zero dynamics. As pointed out in Section 5.2, systems with monotonous step responses will at least have a feasible solution to the pulse-step method. In most cases the optimal solution performs reasonable, with the heat conduction example above being one exception.

Selection of input range

The full range of the control signal should probably not be used for each set point change. In the comparison with time-optimal control it was noted that the benefits from increasing the size of the control signal magnitude was marginal. The essential thing is to "accelerate" the system using a constant value of the control signal. It was further shown that the benefit from having $\underline{u} < 0$ is hardly noticeable unless $\overline{u}$ is large. Therefore we will recommend in the following that $\underline{u} = 0$, and that a fixed value of $\overline{u}$ is used.

An advantage with having fixed values of $\overline{u}$ and $\underline{u}$ for the normalized setup is that all set point changes will have the same shape. The corresponding switching times need also be computed only once. Close to the limits of the operating range it may be necessary to use a smaller $\overline{u}$ due to the physical limitations of the actuators. However, this occurs less frequently the lower the default value of $\overline{u}$ is.

Numerical solution

Equation (5.15) may be written as

$$\begin{cases} \overline{u}\, S\left(t^\star\right) + \left(\underline{u} - \overline{u}\right) S\left(t^\star - {T_1}^\star\right) + \left(1 - \underline{u}\right) S\left(t^\star - \left({T_1}^\star + {T_2}^\star\right)\right) - 1 = 0 \\ \overline{u}\, h\left(t^\star\right) + \left(\underline{u} - \overline{u}\right) h\left(t^\star - {T_1}^\star\right) + \left(1 - \underline{u}\right) h\left(t^\star - \left({T_1}^\star + {T_2}^\star\right)\right) = 0 \\ h\left(t^\star - {T_1}^\star\right) - h\left(t^\star - \left({T_1}^\star + {T_2}^\star\right)\right) = 0 \end{cases} \tag{5.20}$$

This non-linear system of equations can be solved using some Newton-Raphson-like method. Initial guesses for the unknowns may be derived from the impulse and step approximation. The height of the impulse should be $1/\max_t h(t) = 1/h_{max}$ to make it reach $y = 1$. This implies that the area of the pulse equals that of the impulse if T_1 is chosen as

$$T_1 = \frac{1}{h_{max}\overline{u}}$$

The effect of having a finite pulse is that the impulse response is "spread out" so that the peak of the response is somewhat smaller, and delayed approximately $T_1/2$ compared to the impulse response. If the impulse response has its maximum at $t = t_{max}$, the pulse response will thus have its maximum at $t \approx t_{max} + T_1/2$.

Next, T_2 should be chosen such that the step response and the trailing edge of the impulse response approximately add up to 1. An initial guess may be to apply the step between the end of the pulse and the maximum of the pulse response. This gives $T_2 \approx 0.5t_{max} - 0.25T_1$. An initial guess for $t^\star$ may then be obtained either by simulating the process using the initial values for T_1 and T_2, or by using $f_3(t^\star, T_1, T_2) = 0$ from Equation (5.15).

The suggested initial values has worked for all tested processes. However, they must be modified if $\underline{u}$ is large negative, since the pulse-step approximation is not very good in that case. This will not be treated here, since the recommendation in the previous section was to use $\underline{u} = 0$.

5.4 Implementation Structure

The pulse-step method is a simple strategy expressed by two time instants where the control signal should change its value. In other words, it leads to a feed-forward strategy, where the shape of the control signal $u(t)$ depends on the current set point change and the available input range. There following questions must then be addressed:

- How can we make sure that the feed-forward strategy performs reasonably well despite disturbances and modeling errors?
- How should we switch from the transient set point change strategy to the stationary regulatory control, typically solved with PID control?

The sensitivity to disturbances and modeling errors is inherent to all feed-forward strategies. A common way to deal with this is to reformulate the feed-forward strategy as a feedback control law. In the time-optimal "bang-bang" control, this is often solved by finding switching curves or switching surfaces, which will tell what is the optimal control signal for all points in the state space. In practice, the measured (or estimated) state variables will never follow the pre-computed trajectories exactly. The effect of this is that the control signal will have many more switches than the theoretical value $n - 1$. Another drawback with the approach is that it is mostly difficult to find expressions for the switching surfaces. A solution is to use Model Predictive Control (MPC) instead, see for example Camacho and Bordons (1995). However, this imposes high on-line computational demands.

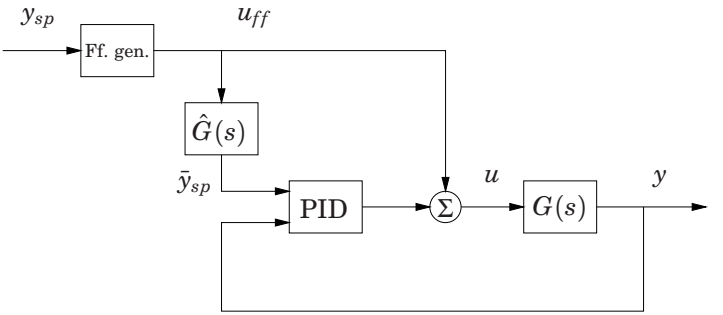


Figure 5.19 Suggested implementation structure.

The other question is typically solved simply by switching to PID control when the output, and possibly some of the state variables, are close enough to the desired set point. As pointed out in Malmborg (1998), this hybrid controller may cause undesired behavior such as limit cycling, if the switching strategy is not carefully formulated. To overcome this, Malmborg (1998) suggests a safe Lyapunov-based switching between different controllers.

Here, we will suggest another controller structure as shown in Figure 5.19. This structure will actually address both problems above at once. The top left block generates the feed-forward control signal u_{ff} identical to the pulse-step control signal. The system $\hat{G}(s)$ contains the process model that was used when computing u_{ff}. The output $\bar{y}_{sp}$ of this block will then be the desired output trajectory. If we have a perfect model and no disturbances, we get $\bar{y}_{sp} \equiv y_{sp}$, which makes $u_{PID} = 0$ [1]. Hence, the pre-computed feed-forward signal alone will take care of the set point change, and no feedback action is used. However, when the output does not follow the desired trajectory, the PID block will try to compensate by adding a component to the control signal.

The proposed controller structure will behave as follows:

- Load disturbances and set point changes are completely separated. This makes sense, since the PID controller typically is tuned for optimal load disturbance response. Whenever a load disturbance occurs, the output signal will be a superposition of an optimal set point response and an optimal load disturbance response. This is in

[1] In order to achieve this, the simplified controller structure in Figure 5.19 must be extended with some anti-windup mechanism. Furthermore, the PID block must operate on the error $\bar{y}_{sp} - y$ only. This causes no problem in this case since $\bar{y}_{sp}$ is a smooth signal, and may thus be differentiated.

contrast to implementation structures using switching surfaces for the optimal set point response. In this case, a disturbance will affect the states, and consequently the switching times. Load disturbances and set point changes will thus interfere with each other.

- Modeling errors may cause a deviation between the desired and the actual output. Since these deviations are taken care of by the PID block, it may be natural to treat them as effects of a load disturbance instead. In fact, you may rewrite the effects of modeling errors as an equivalent input load disturbance $v = G_v u_{ff}$, where

$$G_v = \frac{G - \hat{G}}{G}\left(1 + GG_{PID}\right) \tag{5.21}$$

The two features above rely on the fact that the control signal may actually move in both directions around the constant levels used in the set point strategy. This is one good reason why not all the available control signal should be used when designing the fast step response.

Figure 5.20 shows an example with $G(s) = 1/(s+1)^4$ where an input step load disturbance is applied at the same time as the set point change. The full line shows the behavior without the load disturbance, the dashed and dotted curves have load disturbance steps of size 0.5 in positive and negative direction, respectively. $\overline{u} = 4$ and $\underline{u} = -4$ have been used in the fast step response design, and the PID controller has been designed to give $M_s = 1.4$, see page 127. Due to the load disturbance, the maximum magnitude of the control signal is 4.09 instead of 4. The actual deviation from the nominal value will of course depend on the relative size of the load disturbance, as well as the timing.

Next, we will examine what happens if $G(s)$ and $\hat{G}(s)$ differ. It is reasonable to assume that the same model is used for design of both the set point response and the PID controller. We assume that the process model used in the calculations is the same as before, namely $\hat{G}(s) = 1/(s+1)^4$. The following true processes are then examined:

$$\begin{aligned} G_1(s) &= \frac{1}{(s+1)^4}e^{-sL}, & L &= \{0.5,\ 1\} \\ G_2(s) &= \frac{1}{(4/n\,s+1)^n}, & n &= \{3,\ 5\} \\ G_3(s) &= \frac{K}{(s+1)^4}, & K &= \{0.8,\ 1.2\} \end{aligned}$$

The resulting responses are compared with the nominal response in Figures 5.21–5.23.

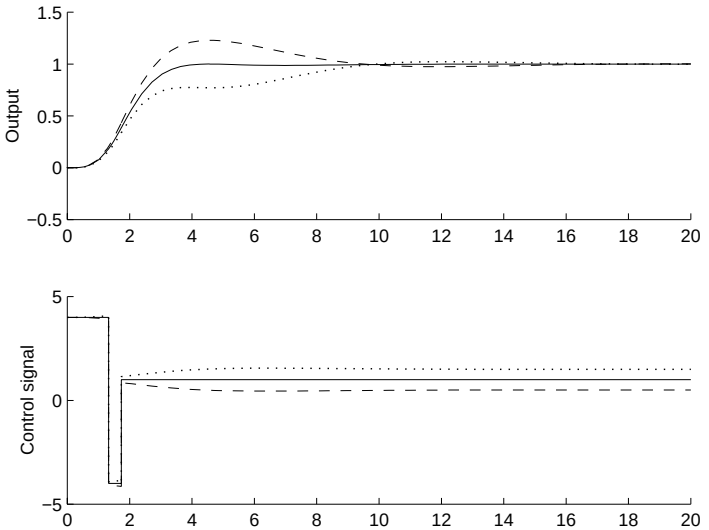


Figure 5.20 Fast step response (full line) with positive (dashed) and negative (dotted) load response at time 0.

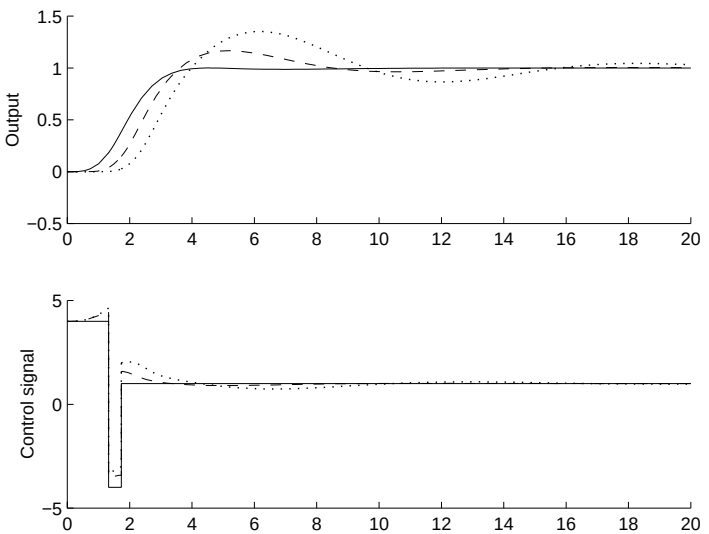


Figure 5.21 Nominal set point response (full line) with unmodeled delay of 0.5 (dashed) and 1 (dotted) time units.

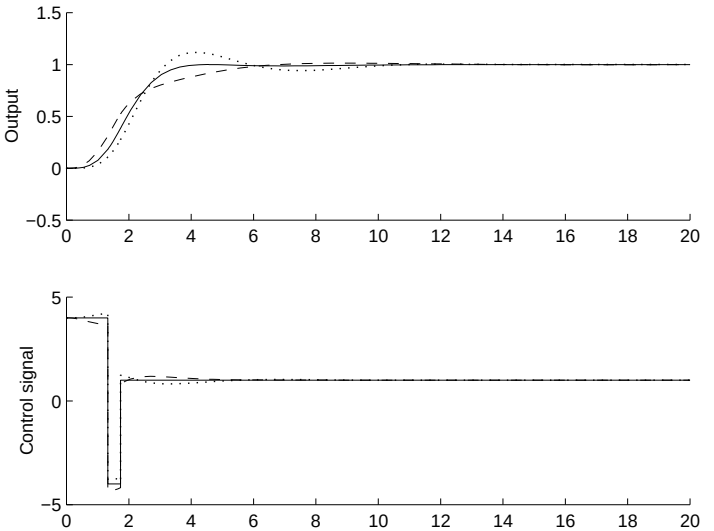


Figure 5.22 Nominal set point response (full line) with third (dashed) and fifth (dotted) order process.

$G_1(s)$ contains an unmodeled time delay. This implies that the nominal feed-forward signal is still optimal, but the optimal response will be delayed. The responses shown in Figure 5.21 are obviously delayed, but there is also an overshoot. The deterioration mainly comes from the fact that y is delayed, whereas $\bar{y}_{sp}$ is not. The PID controller then tries to compensate for the difference instead of just awaiting the delayed response. The behavior is also slightly affected by the additional phase lag which is not accounted for in the PID design. This results in a higher value of M_s (1.7 and 2.1 for a delay of 0.5 and 1 time units, respectively), and consequently less well-damped behavior. The deviation from the nominal response will actually be approximately the same regardless of which $\overline{u}$ and $\underline{u}$ is used for the fast set point response.

Figure 5.22 shows the response for $G_2(s)$ where the order of the process is wrong. The time constants in $G_2(s)$ have been changed such that the step response of the process has the correct average residence time and (approximately) the correct rise time, defined as the time when it reaches $1 - e^{-1} \approx 0.63$. Compared to G_1 there is an additional sources of the deviation from the nominal response. Here, the pre-computed u_{ff} is no longer optimal. For example, the large control signal should have been applied for a longer time when $n = 3$. This is the main reason for the sluggish response of the dashed curve in Figure 5.22.

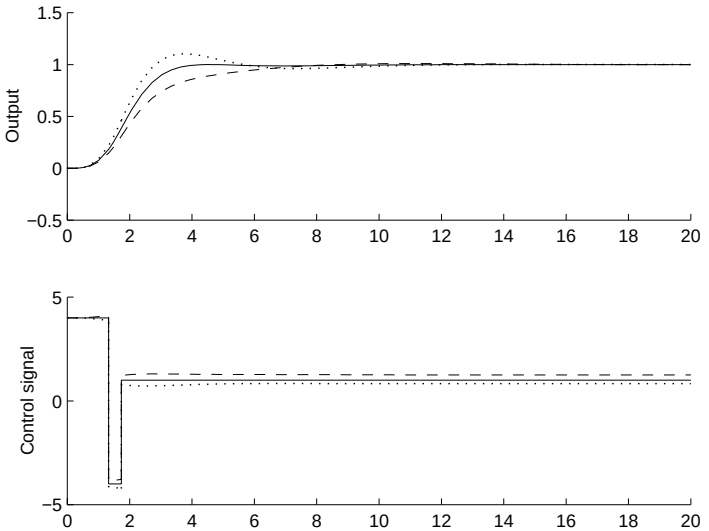


Figure 5.23 Nominal set point response (full line) with decreased (dashed) and increased (dotted) process gain.

Figure 5.23 shows the response when the process gain is incorrect. u_{ff} will not be correct in this case either. When $K = 0.8$, the magnitude of $\overline{u}$ and $\underline{u}$ should have been decreased when solving the normalized problem setup. Thus, the switching times T_1 and T_2 should have been larger. The resulting response will thus be too slow.

To summarize the robustness evaluations in this section, the proposed controller structure behaves reasonably well for the rather large disturbances and model errors simulated here. It has nice stability properties, since the PID controller has been designed with robustness in mind, and the switching scheme is not part of the feedback loop. In all the simulations above, it outperforms the PID controller with set point weighting.

To achieve closer following of the nominal response, a higher order robust controller could be designed, but that is outside the scope of this thesis. Furthermore, a strategy where the switching times are allowed to change may be better at compensating for deviations, but such a strategy would inevitably be more complicated. The proposed structure is tractable in its simplicity. A nice property is that it has very low demands on real-time computing power, as opposed to for example Model Predictive Control, which can used to solve similar problems.

A drawback with the proposed controller structure is when the set point change causes the control signal to saturate. There is then no room

for adjustments to compensate for disturbances or model mismatch. A strategy based on switching curves and feedback from the states may be truly optimal in all cases.

5.5 Summary and Concluding Remarks

A method for fast set point changes has been developed. The basic idea is to use a pulse in the control signal to make the process move fast in the desired direction. The pulse width is adjusted to make the output reach the desired value. A step is then applied at a suitable time in order to catch the output before it would return towards the initial level. This control strategy mimics what is often done manually by a process operator. The strategy has been compared with time-optimal control and PID control.

An advantage with the method is that it can be used with different levels of process knowledge:

1. Initial guesses for the switching times were given in this chapter using only the maximum slope of the step response, and the time when this maximum occurs. These guesses can be used as initial parameters which may be adjusted manually when testing the method on the real plant.
2. If a recorded step response of the process is available, it can be used for finding optimal switching conditions by solving a constrained optimization problem.
3. Conditions have been derived in this chapter for the case where analytical expressions for the step and impulse responses of the process are available. This leads to a non-linear system of equations.

The method has only been developed for asymptotically stable processes. The problem may be solved for integrating processes in a similar way by having an initial positive pulse and a final negative pulse. This corresponds to what is often done in motor control. For unstable processes, the approach with open-loop steps is less tractable. It would still be possible to use the same control strategy as long as it is done using a feedback control structure. However, the switching times must be calculated in a different way.

6

System Architecture

In Chapter 2, autonomous control was discussed from a user's point of view. Since this thesis focuses on process control, the typical user is a process engineer or a plant operator. This chapter discusses implementation aspects of an autonomous control system. System architecture for autonomous control is a subject for a thesis on its own. A few prototype implementations have been made in order to get some insight into the difficulties involved. This chapter summarizes the experiences gained. Section 6.1 points out some of the basic requirements on a control system that should be able to provide the functionality described in the previous chapters. Section 6.2 describes how the graphical language Grafchart can be used for structuring the execution of an autonomous single loop controller.

6.1 Architectural Requirements

The discussion in Chapter 2 outlined some of the desired functionality on the local control loop level in an autonomous control system. In order to accomplish all this, the control system consists of algorithms that use different formalisms. Using a coarse classification, they contain elements from the following computational domains:

- ***Signal processing***, which contains numerical algorithms described using differential, difference and algebraic equations.
- ***Symbolic processing***, which contains logic, sequencing, planning and reasoning.

Since methods from both these domains are mixed in a complex control system, this inevitably results in a hybrid system. Systematic analysis and design of hybrid systems can still be done only on small examples, see for example Krogh and Chutinan (1999). It is therefore not realistic to apply

the theory for hybrid systems on a full control system. However, it can be applied on small sub-problems. One such example is found in Malmborg (1998) and Eker and Malmborg (1999), where the switching between an time-optimal controller and a PID controller is studied. Another example is systems which are piecewise linear, see Johansson (1999). For more complex cases, the interactions between the processing domains must be carried out with care using *ad hoc* methods and "common sense". As an example, on-line controllers should be implemented with anti-windup schemes, and mechanisms for bumpless parameter and mode changes. The supervisory logic should be implemented such that a new control algorithm is not switched in until a safe transfer is expected. These mutual design criteria are intended to avoid undesired influences from the supervisory logic on the control loop. A well-behaving control loop will in turn give less alarms for the supervisory logic to handle.

Another important issue concerning the interactions between the signal and the symbolic processing domains is how the signal-to-symbol and symbol-to-signal conversions are carried out. Whereas the former deals with numerical measures such as $75°C$ and 0.4%, the latter instead uses symbols like `hot` and `negligible`. It is then reasonable to represent different symbolic measures with intervals of the corresponding continuous variables. These intervals, which of course may vary substantially in different cases, must then be set using engineering decisions. Fuzzy logic is often used to make these choices less decisive, but still it may be necessary to spend much effort on tuning the membership functions in order to achieve the desired behavior. In this thesis, only crisp logic will be used in the conversions between the domains.

So far, the different methods have been characterized based on what processing domain they belong to. They may also be characterized by the time-criticality of the execution of the methods:

- Time-critical filtering and control, where it is important that the computational delay from the input to the output is minimized, and that all calculations are performed in time each sample.

- Less time-critical on-line computations, such as recursive identification, controller parameter calculations, performance assessment and monitoring. Here, it is less severe if the calculations are delayed temporarily due to heavy load on the computer.

- Computations with only moderate timing constraints, for example off-line calculations for identification, analysis of experiments and planning. These calculations should not use resources needed by the hard real-time algorithms, and they should preferably be performed on different computers.

All levels contain algorithms from both the signal processing domain and the symbolic processing domain. The time-critical algorithms are typically dominated by numerical algorithms, with a few discrete events interacting with them. Conversely, the off-line calculations are typically governed by algorithms for planning and sequencing, which may be supported by numerical algorithms. On the intermediate level there is typically a mix of sequencing, logic and numerical algorithms.

In order to handle this broad spectrum of methods and methodologies there must be some intelligent way of controlling the execution of the algorithms. There are some major challenges involved with this:

- Choosing a suitable software architecture which allows a flexible representation of the different types of algorithms.
- Defining the structure of the logic controller which plans, runs and monitors the different algorithms. This is the core of the autonomous controller.
- Representing the process and control knowledge in a good way.

This chapter will focus on the second topic.

In early computer control systems the logic was mixed with the control algorithms in a rather unstructured way. As the complexity of the control system grows, the logic tends to hide the actual control algorithm. Even if the pure algorithm may be written in just a few lines, the final computer code will be very complex and unclear. More complex control systems require more structured ways of implementing the control logic and sequencing is.

There are numerous ways of describing sequences and logic. The canonical concept is the Finite State Machine (FSM), see Mealy (1955) and Moore (1956). The FSM describes a discrete event dynamical system (DEDS), which is characterized by a set of discrete states, a set of discrete input signals which causes changes in the states, and a set of discrete output signals generated by the current state and/or input signals. A discrete variable is one which can have a countable number of values, for example a boolean variable. Most of the work concerning analysis and synthesis of DEDS uses the FSM model due to its generic nature.

An FSM suffers from exponential growth, *i.e.*, if the state space contains n discrete variables, the number of combinations of these variables grows as c^n, where c is the size of the symbol set. Each combination represents one state of the FSM. The FSM can thus be represented by a graph containing all states, where the current state is indicated by a boolean flag. As a system becomes more complex, the FSM model thus becomes very large.

Petri nets is a way of condensing a large state space into a more compact representation, see David and Alla (1992). This is done by having so called places, which may contain an integer number of tokens. In addition, more than one place is allowed to contain tokens at the same time. If the number of tokens are bounded, each combination of possible token markings corresponds to one state in an FSM model. However, it is easy to construct a Petri net where the number of tokens is not guaranteed to be bounded. In this case, the Petri net may no longer be transformed into a *finite* state machine. Many analysis methods have been developed for Petri nets in order to check for example this boundedness property.

Standards play a significant role in industrial process control. Systems that build upon established standards are much easier to implement, commission and maintain. The standard IEC 1131 specifies how to implement control systems in programmable logic controllers, PLCs, see IEC (1993). The standard covers many parts of the implementation, for example technical details such as hardware configuration, multi-processor architectures and network communication, but also more general topics such as service, storage, transportation etc. The part that is most interesting from the perspective of this thesis is IEC 1131-3, which specifies programming languages in PLC systems.

The standard includes four different languages, namely structured text (ST), ladder diagrams (LD), function block diagrams (FBD) and instruction lists (IL). These languages can be used for implementing basic algorithms and functions, for example PID controllers, relays, analog and digital IO etc. It is possible to use abstract data types along with a handful standard primitive data types. There is however no support for object orientation. Furthermore, a fifth language, sequential function charts (SFC), can be used for logic and sequential control. More importantly, it can be used to structure and govern the execution of functions written in the other languages. Sequential function charts are closely related to Grafcet and Petri Nets, see IEC (1988) and David and Alla (1992).

The lack of object orientation and the limited support for hierarchical abstraction in IEC 1131-3 may seem as a severe restriction from a software engineering point of view. Furthermore, some methodologies, such as expert systems, are not readily programmed using the languages provided by the standard. On the other hand, it is desirable to use established industrial standards. One solution is then to use layers on top of the standard, where the high-level representations can be translated or compiled into valid code in the different languages in IEC 1131-3. This chapter will make use of one such tool for structuring algorithms, namely Grafchart, see Årzén (1994). It is a high-level graphical sequential language based on Grafcet with influences from object oriented programming and Coloured Petri Nets, see Jensen (1992). Johnsson (1999) also shows

that a Grafchart diagram can be automatically transformed into a Grafcet, and thus into an FSM model. Therefore it is relevant to explore how it can be used for structuring an autonomous control system. Grafcet and Grafchart are briefly described in Appendix A.

Software architectures

There are several architectures discussed in the literature that can be used for complex control systems. Some are specialized to a certain application such as robot control, see for example Nilsson (1996) and Albus *et al.* (1989). Others, like Moore *et al.* (1999), IEC (1993) and Object Management Group (1995), are much more general in their setting. The methodology in fault tolerant control systems also uses an architecture that can be used in various applications, see for example Blanke *et al.* (1997). Architectures focusing on the real-time execution are described in Seto *et al.* (1998) and Eker (1999). More references on software requirements on complex controllers can be found in Åström *et al.* (2000).

In this thesis, a simple object oriented architecture has been used in a prototype implementation. This is briefly described in Section 6.2.

6.2 A Prototype Implementation

A prototype implementation of an autonomous controller will now be described. It is implemented in G2, which has evolved from an object oriented expert system shell in the start, to a general programming environment in later versions. The data structures are built using classes and objects (instances of classes). The objects may have attributes of a number of more or less simple data types. They may also have other objects as attributes. Objects may have a graphical representation, and different objects can be connected to each other graphically. A G2 program, or knowledge-base, consists of rules (`if <logical-expression> then <action>`) and procedures in a pascal-like notation. There are many ways of establishing relations between different objects, and reasoning about these relations. There are for example language constructs to reason directly about the graphical representations and interconnections between objects. Furthermore, there is a simple built-in dynamical simulator for difference and differential equations. All these factors make G2 a rather fast prototyping environment, where you relatively easy can build graphical user interfaces. One drawback is that G2 is not efficient for numerical computations. To overcome this, a bridge has been implemented between G2 as a client and the MATLAB Engine as a compute server. Apart from the increased efficiency, another advantage is that it is possible to use the

extensive amount of existing MATLAB code for, *e.g.*, system identification from G2.

The implementation is based on a Grafchart toolbox implemented in G2, Årzén (1994). Grafchart admits a high-level description of the sequential logic. It is also a reasonably realistic approach, since Grafchart can be automatically translated into the sequential function charts of the industrial standard IEC 1131. Grafcet and Grafchart is briefly described in Appendix A.

The prototype is intended to operate only on the local control loop level. As pointed out in Section 2.2, some of the functionality is executed in the hard real-time environment, and some may be executed in a supervisory control system, *i.e.* the Loop Manager level in Figure 2.1. This partitioning is partly reflected in the prototype implementation, but not completely. The different layers for example have separate data structures, but the communication channel between the layers is not modeled at all. Furthermore, the Loop Manager and the real-time execution as well as the process simulation run in the same G2 knowledge-base.

Data structures on the real-time level

The real-time level and the Loop Manager level will have different data structures. On the real-time level there are various numerical processing blocks such as analog to digital conversion, filters, PID blocks, limiters etc. These blocks are naturally modeled as objects. A base class `signal-flow-object` has been defined, and the different block types are sub-classes derived from this base class. Parameters such as controller gain, sampling time etc. are defined as attributes of the objects. The sampling time `h` is a generic attribute which is defined for the base class, and therefore is inherited by all sub-classes. Parameters that are specific for each block type are defined as attributes of the corresponding sub-class.

Each object should have at least these methods:

- `init` for initialization of internal data structures.
- `calculate-output` for computing the outputs of the block given the inputs and the current internal state.
- `update-state` for updating the internal state of the block.
- `set` for assigning new values to parameter attributes in the block.
- `get` for obtaining parameter values from a block.

The methods are all defined for the `signal-flow-object` class, but they may be empty. Normally, the methods are specialized for the derived classes, otherwise the method belonging to the base class is used.

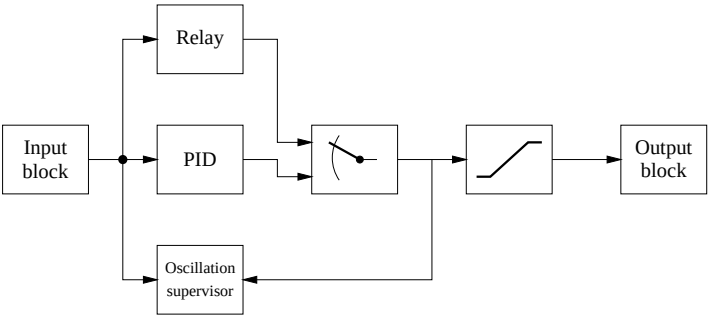


Figure 6.1 A simple controller configuration. The blocks should be executed in an order which minimizes the time delay from the input to the output.

The instances of the sub-classes of `signal-flow-object` are passive components, *i.e.* they do not perform any calculations spontaneously, as opposed to more data-driven architectures. Instead, the methods must be invoked by a `real-time-execution` object. Its main task is to make sure that the methods for real-time execution are called in the correct order corresponding to the signal flow of the controller. Each sample, the `calculate-output` methods should be executed in a way that minimizes the computation time from sampling in an `AD-in` block to actuation in a `DA-out` block. Then, the `calculate-output` methods for objects that lie outside the control loop should be executed, for example loop monitoring blocks. Controllers that are not currently in the loop, but are intended to be so later, should also be included here in order to facilitate bumpless mode changes. Finally, the `update-state` methods should be executed in the reverse order. This is to ensure that limitations are propagated backwards in the controller in order to provide anti-windup mechanisms. This is a sort of internal feedback in the controller which may be necessary when the continuous feedback loop has been cut due to actuator limitations. Each block may decide if it should compensate for the whole limitation on its output, or back-calculate an equivalent limitation on its input, or a combination of the two.

Figure 6.1 shows a simple example which can be used to illustrate the scheduling principle. The blocks that are currently in the feedback loop are the input block, the relay, the switch, the limiter and the output block. Upon each new sampling instant, the `calculate-output` methods of these blocks should be executed in the given order. After that, the PID block and the oscillation supervisor can be executed in any order. Finally, the signals should be back-propagated in reversed order by calling the `update-state` methods, which *may* use modified input signals to succeeding blocks. For

example, the updated value of all input signals to the switch should be back-propagated all the way from output block. If an anti-windup scheme is implemented in the PID block, it should use the updated value at the output of the block. This way, the control signal will not make large bumps when switching to PID mode. Since the relay does not have any internal states that can be unbounded, it is not necessary to implement any anti-windup feature for this block.

Structure of the Loop Manager

The core of the autonomous controller is the Loop Manager, which is responsible for the supervisory control of the local control loop. It is represented by an object of the class `loop-manager`. This object typically resides in a different computer than the hard real-time execution. It communicates with the `real-time-execution` object over a computer network. The Loop Manager sends set points, parameters and reconfiguration commands to the real-time system. The real-time system sends on-line data and alarms to the Loop Manager.

The main task for the `loop-manager` object task is to govern the execution of the different algorithms described in Chapter 2. In the prototype, this is done using Grafchart, see Appendix A. The algorithms are represented using Grafchart procedures. The process information and results from performed experiments are stored in a central process data-base.

In this prototype, the execution strategy is defined by a fixed logic structure in Grafchart. This structure makes extensive use of Grafchart procedures and procedure steps, which actually make the system flexible, since the procedures may be exchanged to achieve different behaviors. In the following, one example of an execution sequence will be described. It consists of two main steps: loop initialization and continuous operation, see Figure 6.2. The loop initialization is naturally described as a sequence of steps that are performed before starting regulatory control. It is less obvious that the continuous operation is sequential in its nature. However, it is possible to define a fixed execution strategy which determines what to do when the regulatory control does not perform satisfactorily. Both the initialization and continuous operation strategies my be exchanged; either fully, or by changing some of the internal procedure calls.

A simple execution sequence for loop initialization could for example look like this:

1. Before startup, ask the user for some information, typically the type of control loop, actuator and sensor information, scaling of the signals, constraints on input and output, desired control objectives, known process characteristics such as crude classifications, approximate time delays and time constants etc.

2. Loop assessment experiments: Find out basic characteristics of the process (non-linearities etc), verify and use the user-supplied information. This could possibly be done for different operating points.
3. Tune a controller, for example by using relay feedback.
4. Switch to on-line control.

The Grafchart implementation of the suggested execution structure above is shown in Figure 6.3. Each of the procedure steps calls a specific Grafchart procedure in order to obtain the desired information by performing some experiment. The first four steps all belong to the loop assessment procedure. The step labeled Setup experiment will for example launch a dialog window, where the operator is supposed to enter some data, and to move the process manually to the operating point for the experiments. An example of the dialog window is shown in Figure 6.4. The system will be able to do a crude analysis of the data produced while going to the operating point. This can be used if the user does not enter any information. If the user enters information which has large discrepancies compared to the observed data, a dialog will appear where the user is asked to confirm either of the approximate sets of process information.

Loop assessment The noise listening phase consists of a call to the Grafchart procedure `noise-listen`. This procedure follows the description in Section 2.3 and will not be discussed further here. However, the hysteresis test may deserve some more attention. The hysteresis test consists of a number of open-loop step responses. In Section 2.3 an initial guess of a suitable input step size was given as

$$\Delta u = \frac{1}{\hat{k}_p} \min(N e_{max}, \Delta y_{max})$$

where $\hat{k}_p$ is the crude estimate of the static gain, e_{max} is the noise amplitude, Δy_{max} is the maximum allowed deviation from the operating point during the experiments, and N is the desired ratio between the amplitudes of the output and the noise. Typical values of N could be 5–15.

Figure 6.5 shows a part of the Grafchart procedure `step-test` that performs the experiment for the hysteresis test. Each of the steps is performed by a call to the Grafchart procedure `single-step`, shown in Figure 6.6. The `single-step` procedure applies a step of a size which is passed as an input argument. After a wait corresponding to the crude estimate of the time scales of the process, the procedure `noise-listen` is called. If the output variation during this listening phase is not dramatically

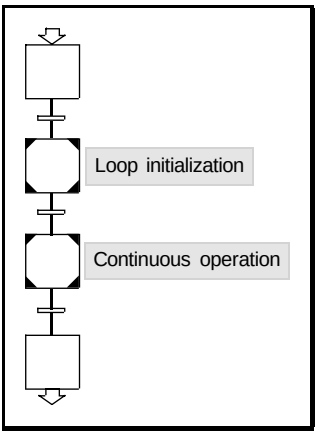


Figure 6.2 The main Grafchart procedure consisting of loop initialization and continuous operation.

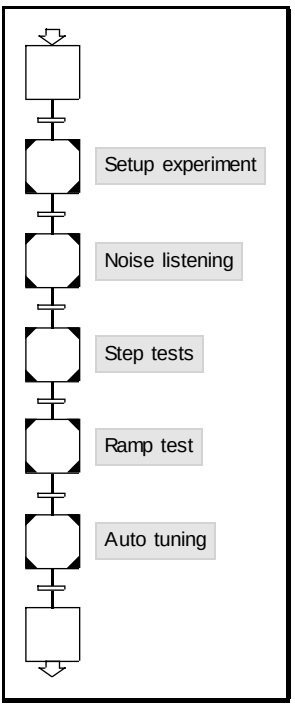


Figure 6.3 A loop initialization sequence for a simple controller.

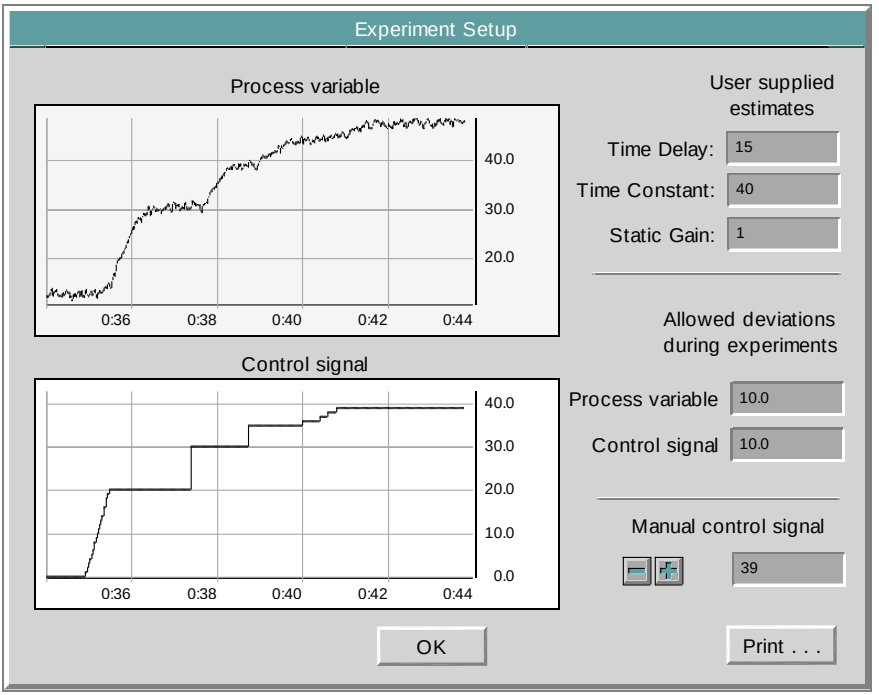


Figure 6.4 A dialog window for experiment setup. The user may provide crude estimates of the time scale and gain of the process.

larger than the recorded noise level during previous noise tests, it is assumed that stationarity is reached, and the `single-step` procedure may finish. Otherwise, new listening phases are repeated until stationarity is reached.

When the `single-step` procedure has finished, the execution returns to the `step-test` procedure in Figure 6.5. If the output moved significantly during the step, the execution will proceed to the next call to `single-step`, otherwise the first step has to be repeated with a larger input amplitude. The other possible failure is that the output moves outside the allowed region during the step. This should however be handled directly when it is detected, and not after the step response is completed. This is a typical situation where exception transitions should be used. The abnormal behavior will interrupt the execution of `single-step` regardless of its current state. The exception handling is a call to `single-step` again, now with the initial level of the control signal as input argument. The `single-step` procedure also records the maximum and minimum value of the output, which may be useful when calculating new amplitudes for the

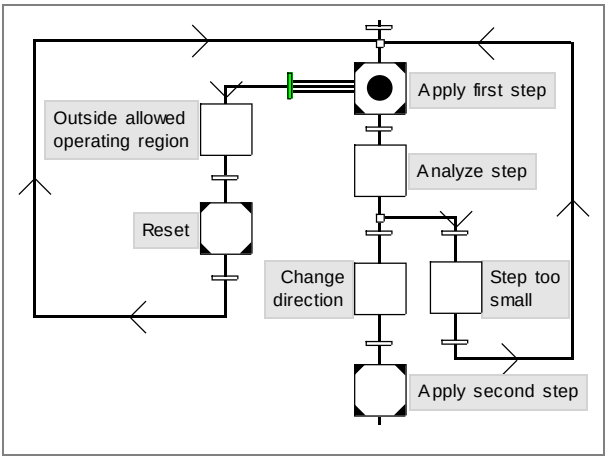


Figure 6.5 A part of the Grafchart procedure `step-test` for the hysteresis test.

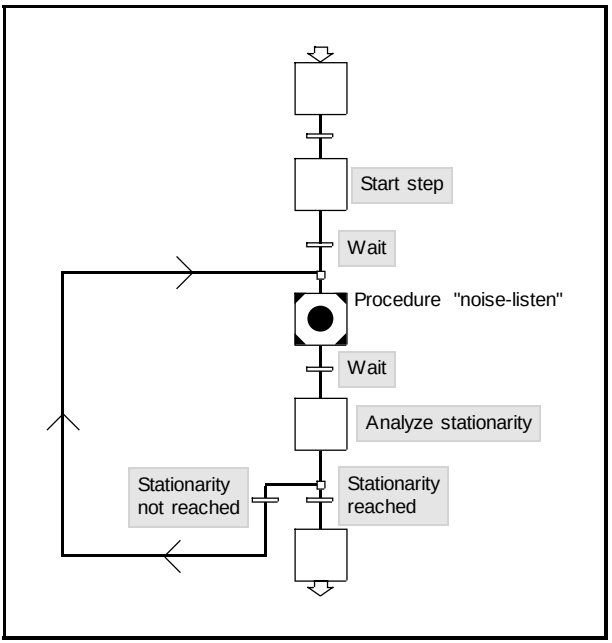


Figure 6.6 The contents of the Grafchart procedure `single-step`.

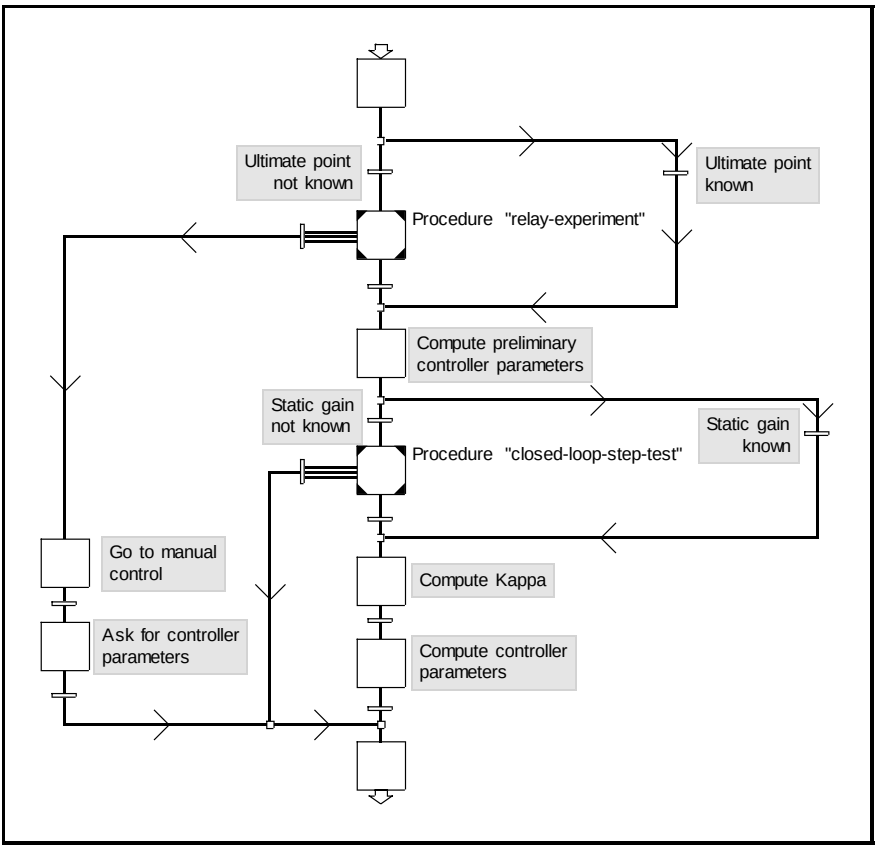


Figure 6.7 The Grafchart procedure `kappa-tau`.

step tests. When the output has returned to stationarity, the step test is repeated, now with lower amplitudes.

This fairly detailed example indicates the kind of supervisory logic that is needed even for a comparatively simple experiment. The use of exception transitions proved very useful here, as it does in many cases.

Automatic tuning The automatic tuning procedure will also be outlined briefly. It is based on the Kappa-Tau method described in Section 2.4. The necessary input data is obtained by a relay experiment followed by a step response in closed loop. The Grafchart procedure `kappa-tau` is shown in Figure 6.7. It consists of a relay feedback experiment to obtain the ultimate gain and frequency of the process. Using this information, a preliminary PI or PID controller tuning is calculated. Then, a closed-loop

step response test is performed with this preliminary PID setting in order to find a good estimate of the static gain of the process. When this information is obtained, a Kappa-Tau PID tuning can be calculated, and the tuning procedure has finished. If something goes wrong during the relay experiment, the controller is forced into manual control, and the user may enter some controller parameters, or restart the installation procedure at some stage. This mechanism is not discussed here. If something goes wrong during the closed-loop step response experiment, the preliminary parameters are kept, and the tuning procedure is finished.

The Grafchart procedure `relay-experiment` is shown in Figure 6.8. The upper part consists of initializations and a wait for the first switch. The lower part consists of the loop that is run through every time the relay switches. After a switch has occurred, there are two major decisions that need to be made. During the first oscillation cycles, the relay amplitude may be adjusted in order to achieve an acceptable oscillation amplitude on the output. When the amplitudes will not be adjusted any more, it will instead be decided if a steady limit cycle has been built up. If this is the case, execution exits the loop, and the ultimate point may be computed. There are two different situations that may cause the relay experiment to fail. First, the oscillations may stop, or never start, even if the relay amplitude is set to its maximum value. The reason for this is either large input non-linearities or gross sensor faults. Secondly, the oscillations may never come sufficiently close to a stable limit cycle. This means that the ultimate point cannot be computed, and the relay experiment must be aborted after a certain number of switches.

Continuous operation When the initial loop assessment and tuning have been completed, the sequence in Figure 6.2 will start the Grafchart procedure for continuous operation. An example of one such procedure is shown in Figure 6.9. Initially it starts a Grafchart procedure for loop monitoring in a process step. By using a process step, it is possible, but not necessary, to have some part of the monitoring algorithm running on the supervisory level. Advanced monitoring algorithms may for example use both on-line data and historical data for reasoning. Simple monitoring algorithms will typically execute only on the real-time system. The procedure that is invoked by the process step will then only start the monitoring algorithm, and then terminate. In this example we assume that the monitoring algorithm is the oscillation detection algorithm described in Hägglund (1995).

When the loop monitoring algorithm has started, a PID controller based on the Kappa-Tau design method is switched in. This is done in the step labeled Normal operation in Figure 6.9. The PID controller is now supposed to work on-line until an operator or some supervisory control

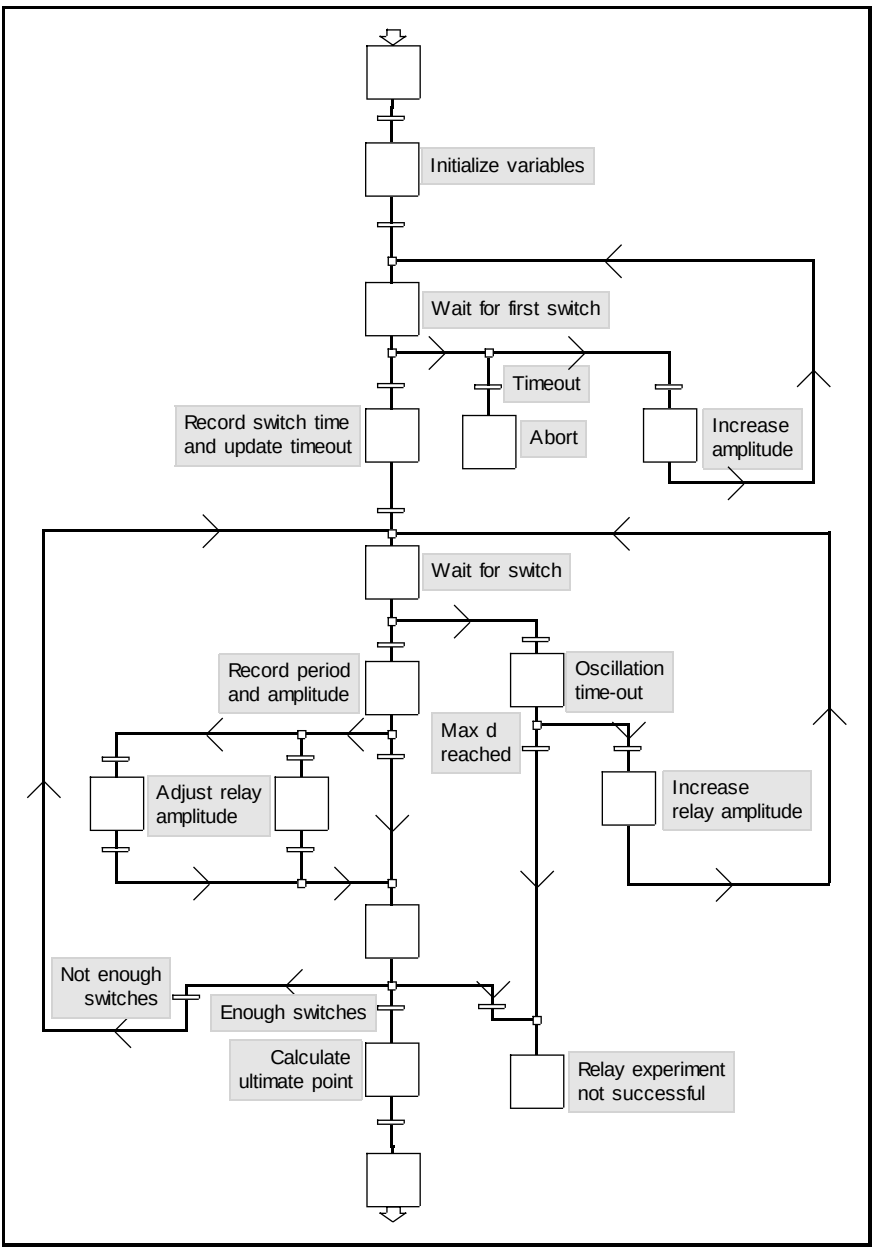


Figure 6.8 The Grafchart procedure `relay-experiment`.

function decides otherwise. When bad control performance is detected, in this case a persistent oscillation, the loop is switched to manual control. The manual control signal should be set to some average of the control during closed-loop control.

The purpose of the manual control is to determine if the oscillations are caused by feedback, or by external disturbances. If the oscillations have not disappeared after some time, it is assumed that the oscillations are caused by external disturbances. The tolerance of the monitoring algorithm is adjusted in order not to trigger for the same disturbance again. Normal operation is then resumed, and the incident is reported to the operator, and to possible superior levels of the control system. The time to wait for the oscillations to disappear can be determined using knowledge of the time scale of the closed-loop system.

If the oscillations disappear during manual control, it is assumed that the bad performance is caused either by friction in the actuator or a badly tuned controller. Thus, a method for friction detection is started, for example the one outlined in Section 2.3. If no friction is detected, a new auto-tuning procedure is performed, and normal operation is resumed. If, on the other hand, experiments show that there is friction present, the user is notified and recommended to maintain the valve. While waiting for this to happen, some kind of friction compensation can be activated before returning to PID control using the old parameters. One method that is frequently used to avoid limit cycling due to friction is to add a dead-band in the controller such that the integration of the control error is stopped if it is small enough. This will however allow the control error to be either positive or negative for very long time periods. This may be undesired if the control loop is used for, *e.g.*, consistency control. Another friction compensation is presented in Hägglund (1997b). It is based on injections of short pulses in order to avoid the stick-slip motion.

Comments on the execution sequence The execution scheme presented above is simple, and resembles what operators sometimes are recommended to do when they face bad control performance, see Åström and Hägglund (1995). The description of the sequence is by no means complete. However, it gives some hints to how the logic in an autonomous single-loop controller can be implemented. The sequence has a seemingly fixed structure, but it is still flexible since any Grafchart procedure may be exchanged for other methods. It is for example possible to let the system use new tuning schemes, and even new controller structures, each time the controller is found to work unsatisfactorily. This can for example be achieved by having an expert system select the controller structure that seems most appropriate in the current situation.

If fixed supervisory logic is still desired, the described execution se-

quence may be changed in several ways.

- For simplicity the proposed sequence has very little exception handling. In order to increase the robustness, each Grafchart procedure should have some kind of error handling.
- The recovery from bad control performance may by performed in many different ways. For example, it is possible to start a friction compensation method as soon as bad control performance is detected. If the performance is improved, friction is the likely reason for the problems. If the performance is not improved, it may be interesting to find other causes for the bad performance.
- When external disturbances is the likely reason for bad performance, it would be interesting to investigate if some signal could be used for feed-forward control.
- Loop interactions are not at all considered in the described sequence.
- In a real system, the recovery sequence must be approved, either by an operator or by some function at the unit operation level, see Figure 2.1.

6.3 Summary

This chapter has discussed implementation issues for an autonomous controller. The use of a high-level graphical language for structuring the control algorithms has been motivated. A prototype implementation in G2 using the Grafchart toolbox has been presented. Grafchart is tractable since it is a high-level sequential description language that can be automatically translated into an industrial standard. Some examples of detailed supervisory logic have been shown.

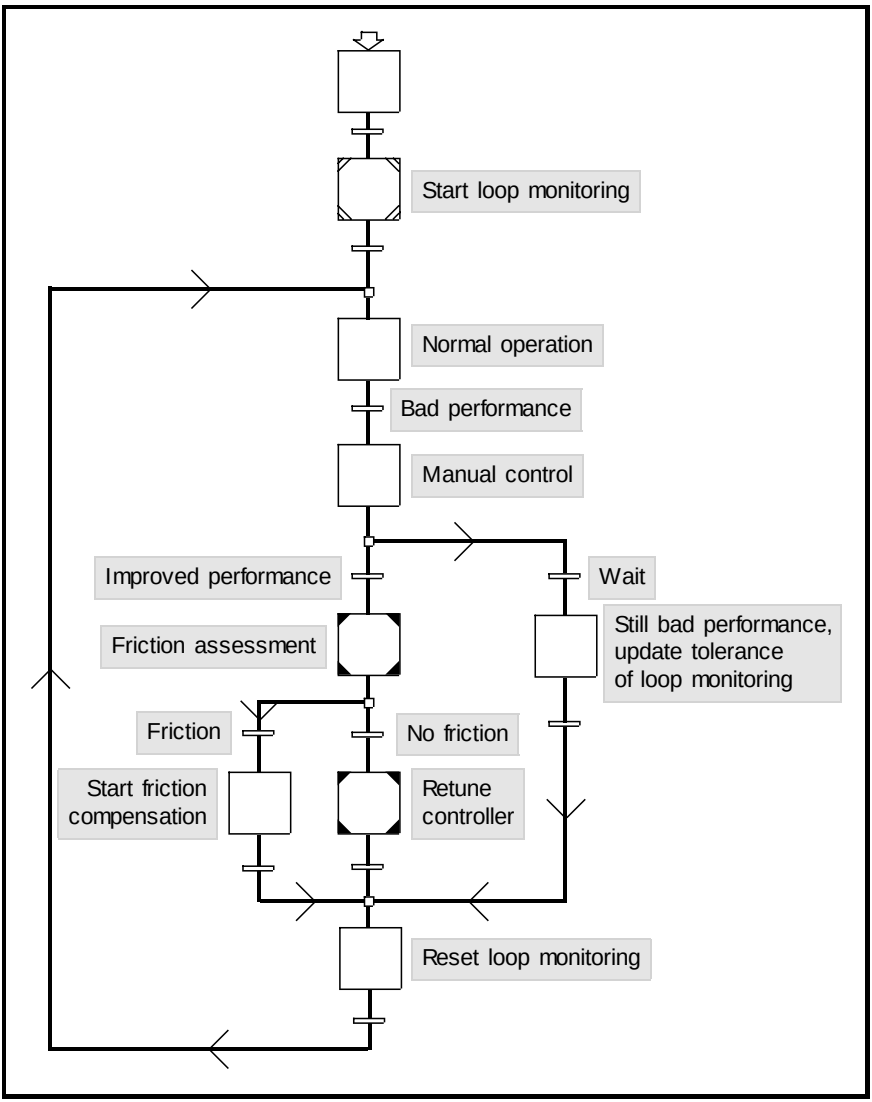


Figure 6.9 The interior of the Grafchart procedure for continuous operation.

7

Conclusions

This thesis has treated different aspects of autonomous process control. The topics can roughly be divided into two parts: one part dealing with desired functionality and implementation of an autonomous single loop controller, the other part describing a set of new tools and algorithms in more detail.

The viewpoint taken on autonomy in the thesis is based on the assumption that human operators are active parts of the control system. The purpose of introducing more autonomy is to extend the region where the control system can run the plant without human interaction and to provide assistance to the operator. A fully autonomous control system would impose extremely high demands on safety, that are not possible to meet today. Instead, the level of autonomy should be increased gradually as improved algorithms are developed.

The thesis has presented a list of desirable features and algorithms that could be parts of an autonomous control system. These algorithms must be executed and analyzed in a structured way. It is shown that this can be successfully done using Grafchart, a high-level graphical language for sequential and logic control which can be automatically translated to sequential function charts.

Detailed descriptions of new tools and algorithms have been given. These components fit into the general framework outlined above. They deal with identification and control design; issues that belong to the classical control field.

The tool for step response analysis provides a simple environment for process identification. Since step response data is very commonly used by operators, it is relevant to provide a tool for analyzing the experiments. The tool actually permits any type of piecewise constant input signal. It should not be regarded as an alternative to advanced process identification tools, but rather as a tool for preliminary assessment of process dynamics. The possibility to manipulate the output from the process model graphi-

cally makes it easy to obtain a crude dynamical model of the process. The parameters of the chosen model structure may also be computed from a least squares fit of the model output to the experimental data. The tool may be used as an integrated part of an autonomous control system, or as a stand-alone application for off-line analysis.

The inclusion of simple process non-linearities in the identification tool is also very useful, since the static characteristics of a plant may be just as important as the dynamical behavior. With a static non-linearity on the input or the output of the system, it is possible to linearize the process by inverting the non-linearity. The combination of a simple dynamical model of the system, a static process non-linearity, good PI(D) tuning rules and gain scheduling gives a simple way of obtaining a non-linear controller. With good computer tools, this may be an alternative to other simple non-linear controllers, for example fuzzy controllers.

A new automatic tuning procedure for PI and PID control has been presented. The use of a time-varying hysteresis during relay feedback gives better excitation over the frequencies relevant for PI and PID control, than relay feedback using fixed hysteresis does. It has been shown that the spectral estimate from this experiment can be used for PI design. It has also been shown that a PID design can be obtained by iterative PI design.

A simple strategy for fast set point response has been developed. It mimics what experienced process operators often do manually to obtain fast and set point step responses with no overshoot. The strategy is to do a short sequence of steps in the control signal. First, a large step is taken to make the process output move fast in the correct direction. After some time, the control signal should return to the original value, or even below it, in order for the process output to approach the new set point smoothly. Finally, a step corresponding to the correct steady-state value of the control signal should be applied in time to "catch" the process output before it moves away from the new set point again. The sizes and lengths of the steps should be adjusted in order to get a fast and smooth set point response. Methods for finding good values for the step sizes and the switching times has been given. The method may be applied with varying degrees of process knowledge, though with different qualities of the resulting response.

Future work

Higher autonomy at every level of a control system will continue to be a subject for further research for years to come. Concerning the specific problems treated in this thesis, there are also a large number of future research directions.

The list of desired functionality can be made much longer. The thesis

only deals with problems related to SISO processes. Some of the ideas carry over to simple multivariable structures such as cascade control. It would be useful to have a similar set of tools for multivariable processes.

The experiments presented here for assessment of local non-linearities such as friction and hysteresis may take unduly long time to perform. It should be possible to do a combined experiment which finds out both friction and hysteresis using a minimum experiment duration.

The scheduling of algorithms is currently done in a flexible, but static manner. A desirable feature is to plan the experiments dynamically in order to do exactly the right experiment in every instant. One way of approaching this is to use the concepts of pre-conditions and post-conditions from expert system technology, and use planning algorithms for scheduling the experiments.

The step response analysis tool may be improved in several ways. One interesting feature would be to include other representations of non-linearities, for example non-linearities that depend on exogenous signals. A good experiment strategy for that kind of system should also be developed.

The auto-tuning scheme based on relay feedback has to be made more robust before it is possible to apply on all kinds of practical systems. The selection of suitable hysteresis levels should be done automatically during the experiment.

The implementation structure for the fast set point response method should perhaps be modified. For example, it would be nice to have an implementation where the switching times can be calculated using feedback. It would also be interesting to choose the switching times based on simple observations that gives more consistent results than the current method does.

8

References

Albus, J. S., H. McCain, and R. Lumia (1989): "NSA/NBS standard reference model for telerobot control system architecture (NASREM)." Technical Report 1235, National Bureau of Standards.

Andreas, H. and K. J. Åström (1997): "Design of PI controller by minimization of IAE." Report ISRN LUTFD2/TFRT--7565--SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Antsaklis, P. J., K. M. Passino, and S. J. Wang (1991): "An introduction to autonomous control systems." *IEEE Control Systems Magazine*, **11:4**, pp. 5–13.

Arkin, R. (1998): *Behavior-based robotics*. MIT Press.

Årzén, K.-E. (1994): "Grafcet for intelligent supervisory control applications." *Automatica*, **30:10**, pp. 1513–1526.

Åström, K. J. (1967): "Computer control of a paper machine—An application of linear stochastic control theory." *IBM Journal of Research and Development*, **11:4**, pp. 389–405.

Åström, K. J. (1969): "Optimal control of Markov processes with incomplete state information II." *Journal of Mathematical Analysis and Applications*, **26**, pp. 403–406.

Åström, K. J. (1993): "Autonomous process control." In *Proceedings of The Second IEEE Conference on Control Applications*. Vancouver, British Columbia.

Åström, K. J. (1997): "Limitations on control system performance." In *European Control Conference*. Brussels, Belgium.

Åström, K. J. (2000): "Limitations on control system performance." *European Journal of Control*. To appear.

Åström, K. J., P. Albertos, M. Blanke, A. Isidori, W. Schaufelberger, and R. Sanz (2000): *Control of Complex Systems*. Springer. To appear.

Åström, K. J. and K.-E. Årzén (1993): "Expert control." In Antsaklis and Passino, Eds., *An Introduction to Intelligent and Autonomous Control*, pp. 163–189. Kluwer Academic Publishers.

Åström, K. J. and K. Furuta (2000): "Swinging up a pendulum by energy control." *Automatica*, **36:2**, pp. 287–295.

Åström, K. J. and T. Hägglund (1984): "Automatic tuning of simple regulators with specifications on phase and amplitude margins." *Automatica*, **20**, pp. 645–651.

Åström, K. J. and T. Hägglund (1995): *PID Controllers: Theory, Design, and Tuning*, second edition. Instrument Society of America, Research Triangle Park, North Carolina.

Åström, K. J., T. Hägglund, C. C. Hang, and W. K. Ho (1993): "Automatic tuning and adaptation for PID controllers—A survey." *Control Engineering Practice*, **1:4**, pp. 699–714.

Åström, K. J., H. Panagopoulos, and T. Hägglund (1998): "Design of PI controllers based on non-convex optimization." *Automatica*, **35:5**.

Åström, K. J. and B. Wittenmark (1997): *Computer-Controlled Systems*, third edition. Prentice Hall.

Atherton, D. P. (1975): *Nonlinear Control Engineering—Describing Function Analysis and Design*. Van Nostrand Reinhold Co., London, UK.

Bannura, P. (1994): "Programpaket för uppsättning och intrimning av PID-regulatorer," (Program package for initiating and tuning of PID controllers). Master thesis ISRN LUTFD2/TFRT--5508--SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Bi, Q., W. Cai, E. Lee, Q. Wang, C. Hang, and Y. Zhang (1999): "Robust identification of first-order plus dead-time model from step response." *Control Engineering Practice*, **7:1**, pp. 71–77.

Bi, Q., Q. G. Wang, and C. C. Hang (1997): "Relay-based estimation of multiple points on process frequency response." *Automatica*, **33:9**, pp. 1753–1757.

Bialkowski, W. L. (1993): "Dreams versus reality: A view from both sides of the gap." *Pulp and Paper Canada*, **94:11**.

Billings, S. and S. Fakhouri (1979): "Identification of a class of nonlinear systems using correlation analysis." *Proc. IEE*, **125:7**, pp. 691–697.

Blanke, M., R. Izadi-Zamanabadi, S. Bøgh, and C. Lunau (1997): "Fault-tolerant control systems - a holistic view." *Control Engineering Practice*, **5:5**, pp. 693–702.

Boden, M. (1998): "Creativity and artificial intelligence." *Artificial Intelligence*, **103:1–2**, pp. 347–356.

Brown, M. and C. Harris (1994): *Neurofuzzy Adaptive Modelling and Control*. Prentice Hall.

Bryant, G. F. and L. F. Yeung (1996): *Multivariable Control System Design Techniques: Dominance and Direct Methods*. Wiley.

Byler, E., W. Chun, W. Hoff, and D. Layne (1995): "Autonomous hazardous waste drum inspection vehicle." *IEEE Robotics & Automation Magazine*, **2:1**, pp. 6–17.

Camacho, E. F. and C. Bordons (1995): *Model Prediction Control in the Process Industry*. Advances in Industrial Control. Springer-Verlag, Berlin.

David, R. and H. Alla (1992): *Petri Nets and Grafcet: Tools for modelling discrete events systems*. Prentice-Hall.

Dickmanns, E. D. and A. Zapp (1987): "Autonomous high speed road vehicle guidance by computer vision." In Isermann, Ed., *Automatic Control—World Congress, 1987: Selected Papers from the 10th Triennial World Congress of the International Federation of Automatic Control*, pp. 221–226. Pergamon, Munich, Germany.

Eker, J. (1999): *Flexible Embedded Control Systems. Design and Implementation*. PhD thesis ISRN LUTFD2/TFRT--1055--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Eker, J. and J. Malmborg (1999): "Design and implementation of a hybrid control strategy." *IEEE Control Systems Magazine*, **19:4**.

Ender, D. B. (1993): "Process control performance: Not as good as you think." *Control Engineering*, **40:10**, pp. 180–190.

Frank, P. and B. Koeppen-Seliger (1997): "New developments using AI in fault diagnosis,." *Engineering Applications of Artificial Intelligence*, **10:3**, pp. 3–14.

Frank, P. M. (1990): "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy—A survey and some new results." *Automatica*, **26:3**, pp. 459–474.

Gustavsson, I., L. Ljung, and T. Söderström (1977): "Identification of processes in closed loop–Identifiability and accuracy aspects." *Automatica*, **13:1**, pp. 59–75.

Haber, R. and H. Unbehauen (1990): "Structure identification of nonlinear dynamic systems – a survey on input/output approaches." *Automatica*, **26**, pp. 651–677.

Hägglund, T. (1992): "A predictive PI controller for processes with long dead times." *IEEE Control Systems Magazine*, **12:1**, pp. 57–60.

Hägglund, T. (1995): "A control-loop performance monitor." *Control Engineering Practice*, **3**, pp. 1543–1551.

Hägglund, T. (1997a): "The idle index." Report ISRN LUTFD2/TFRT--7568--SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Hägglund, T. (1997b): "Stiction compensation in control valves." In *European Control Conference*. Brussels, Belgium.

Hang, C. C. and L. S. Cao (1996): "Improvement of transient response by means of variable set point weighting." *IEEE Transactions on Industrial Electronics*, **43:4**, pp. 477–484.

Harris, T., F. Boudreau, and J. MacGregor (1996): "Performance assessment of multivariable feedback controllers." *Automatica*, **32:11**, pp. 1505–1518.

Harris, T. J. (1989): "Assessment of control loop performance." *Canadian Journal of Chemical Engineering*, **67**, pp. 856–861.

Haverkamp, B., C. Chou, and M. Verhaegen (1998): "Subspace identification of continuous-time Wiener models." In *Proceedings of the 37th IEEE Conference on Decision and Control, 1998.*, vol. 2, pp. 1846–1847.

Ho, W. K., C. C. Hang, and L. S. Cao (1992): "Tuning of PI controllers based on gain and phase margin specifications." In *Preprints IEEE Int. Symposium on Industrial Electronics*. P. R. China.

Holmberg, U. (1991): *Relay Feedback of Simple Systems*. PhD thesis TFRT-1034, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Horch, A. (1999): "A simple method for oscillation diagnosis in process control loops." In *1999 Conference on Control Applications*. Kohala Coast, Island of Hawaii, Hawaii.

Horch, A. and A. Isaksson (1999): "A modified index for control performance assessment." *Journal of Process Control*, **9:6**, pp. 475–483.

IEC (1988): "IEC 60848: Preparation of function charts for control systems." Technical Report. International Electrotechnical Commission.

IEC (1993): "IEC 61131 programmable controllers - part 3: Programming languages." Technical Report. International Electrotechnical Commission.

ISA (1995): "ISA S88.01 batch control." Instrument Society of America.

Jensen, K. (1992): *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use.*, vol. 1, Basic Concepts. Springer-Verlag.

Johansson, K. H. (1997): *Relay Feedback and Multivariable Control*. PhD thesis ISRN LUTFD2/TFRT--1048--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Johansson, K. H. and T. Hägglund (1999): "Control structure design in process control systems." Report ISRN LUTFD2/TFRT--7585--SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Johansson, M. (1999): *Piecewise Linear Control Systems*. PhD thesis ISRN LUTFD2/TFRT--1052--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Johansson, R. (1993): *System Modeling and Identification*. Prentice Hall, Englewood Cliffs, New Jersey.

Johnsson, C. (1999): *A Graphical Language for Batch Control*. PhD thesis ISRN LUTFD2/TFRT--1051--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Khalil, H. K. (1992): *Nonlinear Systems*. MacMillan, New York.

Krogh, B. and A. Chutinan (1999): "Hybrid systems: Modeling and supervisory control." In Frank, Ed., *Advances in control: highlights of ECC 99*, pp. 227–246. Springer.

Lilja, M. (1989): *Controller Design by Frequency Domain Approximation*. PhD thesis TFRT-1031, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Ljung, L. (1999): *System Identification—Theory for the User*, second edition. Prentice Hall, Englewood Cliffs, New Jersey.

Lynch, C. B. and G. A. Dumont (1996): "Control loop performance monitoring." *IEEE Transactions on Control Systems Technology*, **4**, pp. 185–192.

Malmborg, J. (1998): *Analysis and Design of Hybrid Control Systems*. PhD thesis ISRN LUTFD2/TFRT--1050--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Mealy, G. (1955): "A method for synthesizing sequential circuits." *Bell System Technical Journal*, **5:34**, pp. 1045–1079.

Mishkin, A., J. Morrison, T. Nguyen, H. Stone, B. Cooper, and B. Wilcox (1998): "Experiences with operations and autonomy of the Mars pathfinder microrover." In *1998 IEEE Aerospace Conference*, vol. 2, pp. 337–351.

Moore, E. (1956): "Gedanken experiments on sequential machines." *Automata Studies*, pp. 129–153. Princeton University Press.

Moore, M., V. Gazi, K. Passino, W. Shackleford, and F. Proctor (1999): "Complex control system design and implementation using the nist-rcs software library." *IEEE Control Systems Magazine*, **19:6**, pp. 12–28.

Morari, M. and J. Lee (1999): "Model predictive control: past, present and future." *Computers and Chemical Engineering*, **23**, pp. 667–682.

Narendra, K. and P. Gallman (1966): "An iterative method for the identification of nonlinear systems using the Hammerstein model." *IEEE Transactions on Automatic Control*, **11**, pp. 546–550.

Nilsson, K. (1996): *Industrial Robot Programming*. PhD thesis ISRN LUTFD2/TFRT--1046--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Norberg, A. (1999): "Kappa tuning – Improved relay auto-tuning for PID controllers." Master thesis ISRN LUTFD2/TFRT--5621--SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Object Management Group (1995): *Common Object Request Broker Architecture and Specification*, 2.0 edition.

Olsson, H. (1996): *Control Systems with Friction*. PhD thesis ISRN LUTFD2/TFRT--1045--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Olsson, H., K. J. Åström, C. Canudas de Wit, M. Gäfvert, and P. Lischinsky (1998): "Friction models and friction compensation." *European Journal of Control*.

Oppenheim, A. V. and R. W. Schafer (1989): *Discrete-Time Signal Processing*. Prentice-Hall, Englewood Cliffs, New Jersey.

Palmor, Z. J., Y. Halevi, and N. Krasney (1995): "Automatic tuning of decentralized PID controllers for TITO processes." *Automatica*, **31:7**.

Panagopoulos, H. (1998): *PID Controller Design*. Lic Tech thesis ISRN LUTFD2/TFRT--3224--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Panagopoulos, H. (2000): *PID Control; Design, Extension, Application*. PhD thesis ISRN LUTFD2/TFRT--1059--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Panagopoulos, H., A. Wallén, O. Nordin, and B. Eriksson (2000): "A new tuning method with industrial application." Report. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Passino, K. M. and S. Yurkovich (1997): *Fuzzy Control*. Prentice Hall, Englewood Cliffs, NJ.

Patra, A. and H. Unbehauen (1993): "Nonlinear modelling and identification." In *International Conference on Systems, Man and Cybernetics, 1993*, vol. 3, pp. 441–446.

Pawlak, M. (1991): "On the series expansion approach to the identification of Hammerstein systems." *IEEE Transactions on Automatic Control*, **36:6**, pp. 763–767.

Rivera, D. E., M. Morari, and S. Skogestad (1986): "Internal model control—4. PID controller design." *Ind. Eng. Chem. Proc. Des. Dev.*, **25**, pp. 252–265.

Schei, T. S. (1992): "A method for closed loop automatic tuning of PID controllers." *Automatica*, **28:3**, pp. 587–591.

Schram, G., M. Verhaegen, and A. Krijgsman (1997): "System identification with orthogonal basis functions and neural networks." In *Proceedings of the 13th World Congress International Federation of Automatic Control*, vol. 1, pp. 221–226.

Seto, D., B. H. Krogh, L. Sha, and A. Chutinan (1998): "Dynamic control system upgrade using the simplex architechture." *IEEE Control Systems*, **18:4**, pp. 72–80.

Shen, S. H., J. S. Wu, and C. C. Yu (1996): "Use of biased-relay feedback for system identification." *AIChE*, **42**, pp. 1174–1180.

Söderström, T. and P. Stoica (1989): *System Identification*. Prentice-Hall, London, UK.

Thornhill, N. F. and T. Hägglund (1997): "Detection and diagnosis of oscillation in control loops." *Control Engineering Practice*, **5**, pp. 1343–1354.

Tyler, M. L. and M. Morari (1995): "Performance assessment for unstable and nonminimum-phase systems." In *IFAC Workshop on On-Line Fault Detection and Supervision in the Chemical Process Industries*, pp. 187–92.

Wallén, A. (1995): "Using Grafcet to structure control algorithms." In *Proceedings of The Third European Control Conference*. Rome, Italy.

Wallén, A. (1997): "Valve diagnostics and automatic tuning." In *Proceedings of the American Control Conference*. Albuquerque, New Mexico.

Wallén, A. (1999): "A tool for rapid system identification." In *Proceedings of the 1999 IEEE International Conference on Control Applications*. Kohala Coast, Hawaii.

Wang, Q. G., C. C. Hang, and Q. Bi (1999): "A technique for frequency response identification from relay feedback." *IEEE Trans. Control System Tech.*, **7:1**, pp. 122–128.

A

Graphical Languages for Sequential Control

The purpose of this appendix is to give a short description of the graphical language Grafchart that was used in Section 6.2. Grafchart is an extension of Grafcet, and therefore Grafcet is also described briefly. For more detailed descriptions of the syntax and semantics of Grafcet, refer to David and Alla (1992) and IEC (1988). Grafchart is described in Årzén (1994) and Johnsson (1999).

A.1 Grafcet

Grafcet, or sequential function charts (SFC), can be viewed as a specialization of Petri Nets. The specialization lies in the fact that each place, called step in Grafcet, is allowed to contain only one marker. This means that a Grafcet is easily transformed into a Petri net. Thus, the existing formal methods for analyzing Petri Nets may be applied to Grafcet and SFC. Since a Grafcet contains a bounded number of markers it is also possible to transform it to a Finite State Machine.

An example containing the basic Grafcet building blocks is shown in Figure A.1.

The Grafcet consists of interconnected steps and transitions. The steps represent the states of the Grafcet. A filled marker indicates that a step is currently active. Steps drawn with double squares are called initial steps, implying that they are initially active. Associated with each step is a number of actions to be performed while the step is active. The box containing **A** in Figure A.1 represents an action.

The steps are connected via transitions, each having a boolean condition, an event, or both. A condition (**C** in Figure A.1) is tested while

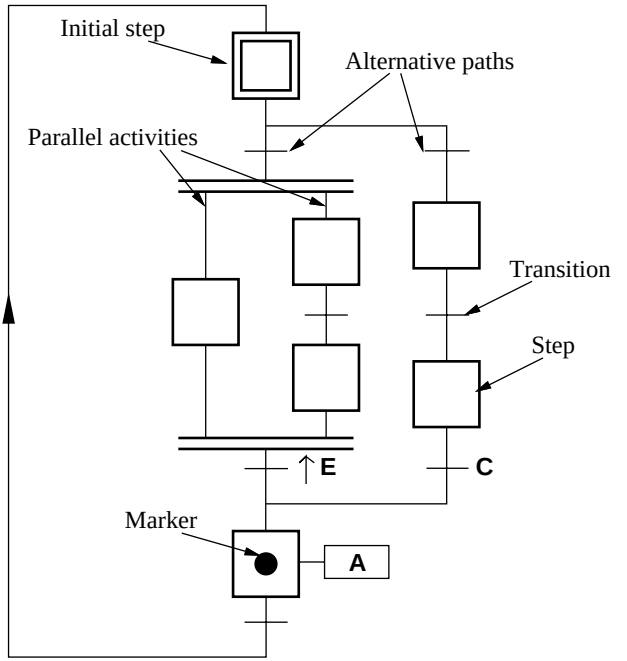


Figure A.1 Basic Grafcet building blocks. **A** is an action, **C** is a condition, and ↑**E** is an event.

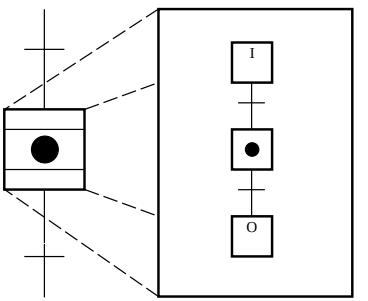


Figure A.2 A Grafcet macro step with its internal structure.

the transition is enabled, *i.e.*, when the preceding step is active. If the condition is true, the transition fires, making succeeding steps active, and preceding steps inactive. An event (↑**E** in Figure A.1) does the same thing, except that it is required to *become* true while the transition is enabled for the transition to fire. Grafcet is only specified for simple boolean conditions and actions, whereas SFC allows complex language elements.

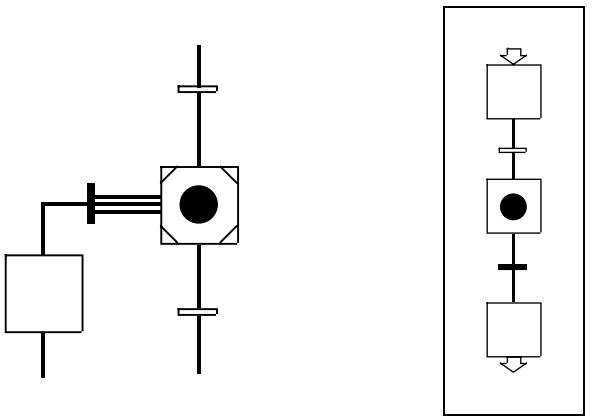


Figure A.3 A Grafchart macro step with its internal structure to the right. The transition connected to the left side of the macro step is an exception transition. Enabled transitions are filled.

Grafcet supports alternative and parallel paths. The syntax for these is shown in Figure A.1. In order to structure the Grafcet hierarchically, it is possible to use macro steps, see Figure A.2. A macro step is a way of condensing an arbitrary number of steps and transitions into just one step. It has one enter step (I) and one exit step (O), and general Grafcet structures in between. The enter step is activated as soon as the macro step becomes active. The transitions following the macro step may not be fired until the exit step is active.

A.2 Grafchart

When building complex control systems, the Petri net or Grafcet models of it soon become big and difficult to get an overview of. Various high-level nets have been suggested in order to cope with this, see for example the brief survey in Johnsson (1999). In this work, Grafchart has been used for structuring control algorithms. Grafchart is an extension of Grafcet, and has been implemented in the real-time expert system environment G2. We will now give a brief description of the elements of Grafchart that has been used here. For a more complete description, see Årzén (1994) and Johnsson (1999), where a formal definition with complete syntax and semantics is also presented.

The basic components in Grafchart are the same as in Grafcet, with slight differences in notation. Figure A.3 shows the graphical notation in Grafchart, along with a new language element called exception transition.

The exception transition is connected to a macro step via a wide line. It is enabled the whole time when the macro step is active. Other transitions connected to the macro step are only enabled when the exit step of the macro step is active. This makes the exception transition suitable for handling failures, abortion and other types of exception handling. The macro step has an internal structure shown to the right in Figure A.3 with an enter step at the top, an exit step at the bottom, and general Grafchart structures in between. In the G2 implementation, the internal structure of a macro step resides on its subworkspace.

There are several other language elements introduced in Grafchart. A procedure step is similar to a macro step, though it does not have a fixed internal structure. Instead, it invokes the Grafchart procedure named by the `procedure` attribute. A Grafchart procedure is an object with an associated Grafchart sequence, which has the same syntax as the internal structure of a macro step. Similarly, this Grafchart sequence resides on the subworkspace of the Grafchart procedure. When a procedure step becomes active, the corresponding Grafchart procedure is instantiated and activated. It then executes similar to a macro step. Since the `procedure` attribute may contain different procedure names at different times, this becomes a more flexible way of organizing sequential logic in a compact manner. It is also possible to define input and output parameters of a Grafchart procedure. These are passed to the procedure via the `parameters` attribute, which is used to set attributes of the Grafchart procedure. In the example in Figure A.4, the attribute `n` of `proc-1` is set to 5 when the Grafchart procedure is invoked.

Inside the procedure, the notation `sup.<attribute-name>` is used to access the parameters in actions, conditions and attributes to steps. With the notation `sup.<attribute-name>^` it is possible to have reference parameters. The expression refers to the object that is named by the contents of the attribute `<attribute-name>` of the Grafchart procedure. Attributes of this object may in turn be referred to by expressions like `sup.attr1^.attr2` etc. This is illustrated in Figure A.5.

In addition to the procedure step, Grafchart also provides a process step, see Figure A.6. Both call a Grafchart procedure, possibly with parameters. When a procedure step becomes active, the invoked Grafchart procedure must finish before any transitions after the procedure step become enabled. Conversely, the transitions after a process step become enabled as soon as the Grafchart procedure has been invoked. Thus, the sequence is split into two parallel execution threads, one in the original sequence, and another inside the invoked procedure.

A condition in the G2 implementation of Grafchart is an attribute of its corresponding transition. It is allowed to contain any boolean-valued G2 expression, with the possibility to use dot notation and `sup` references,

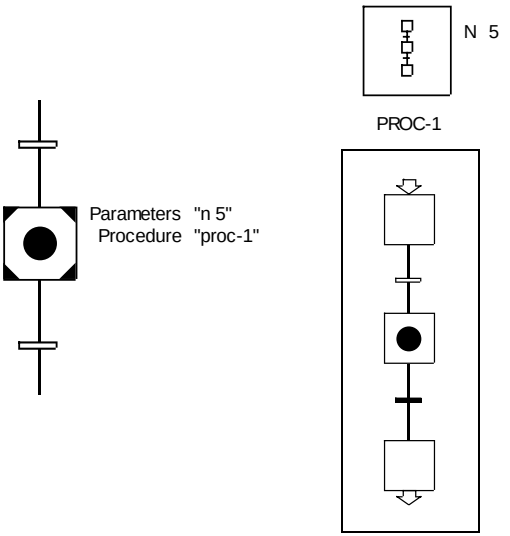


Figure A.4 A procedure step which calls the Grafchart procedure `proc-1` with the attribute `n` set to 5. The parameter may be used within the internal structure of the procedure to the right. Note that the syntax is case insensitive.

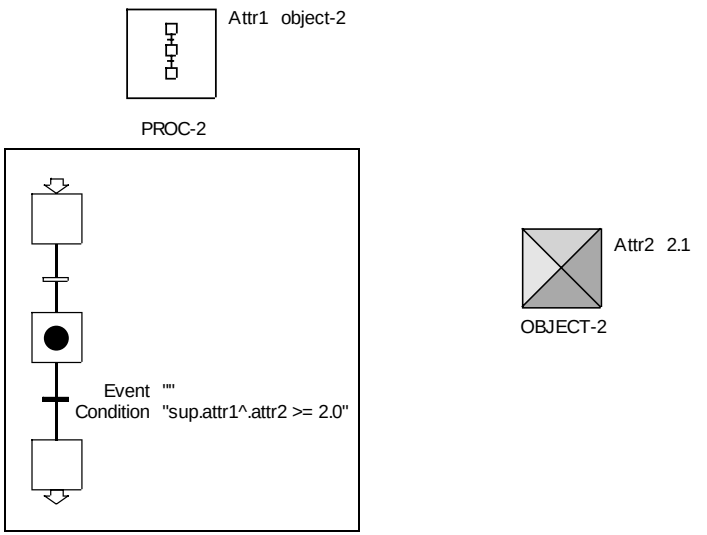


Figure A.5 An example of reference notation. The condition of the enabled transition evaluates the `attr2` attribute of the object that is named by the `attr1` attribute of the Grafchart procedure.

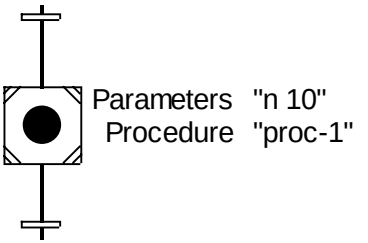


Figure A.6 A process step which calls the Grafchart procedure `proc-1` with the attribute `n` set to 10. The transition after the process step is enabled as soon as `proc-1` has been invoked.

as in Figure A.5. Furthermore, a transition has an attribute `event` which may contain a reference to a variable. The transition may fire whenever this variable receives a new value, and the condition is satisfied.

There are five types of actions in Grafchart:

- ***Initially*** actions, which are executed once when the step becomes active.
- ***Always*** actions, which are executed periodically when the step is active.
- ***Finally*** actions, which are executed once just before the step becomes inactive.
- ***Abortive*** actions, which are executed once when an exception transition is making the step inactive. This type of action is thus only possible in macro steps and Grafchart procedures.
- ***Finally, abortive*** actions, which is a combination of the last two types.

The syntax of an action in the G2 implementation is one of the action categories above, followed by a general G2 action, possibly extended with the dot notation used above. The actions associated with a step reside on its subworkspace. Before the Grafchart sequence containing the step becomes active, the actions must be compiled into valid G2 rules. This may be done statically when the Grafchart sequence has been constructed, or dynamically at runtime, when for example a Grafchart procedure is invoked. The same applies to the events and conditions of transitions as well.

Grafchart has been used in several application areas both in industry and universities, for example batch control (Johnsson (1999)) and hydrogen balance advisory control (Årzén (1994)).