94_09I

Advanced Control Theory and Digital Controllers

Bart De Moor

Christiaan Moons

Katholieke Universiteit Leuven
Department of Electrical Engineering (ESAT)
Research Group SISTA
Systems, Identification, System Theory and Automation

Kardinaal Mercierlaan 94 B-3001 Leuven, Belgium
tel: 32/16/220931 fax: 32/16/221855
email: bart.demoor@esat.kuleuven.ac.be
christiaan.moons@esat.kuleuven.ac.be

ESAT-SISTA/TR 1994-09

Vijfdaagse Cursus Geavanceerde Elektrische Aandrijftechniek
KVIV - KU Leuven - TU Delft - TU Eindhoven
Woensdag 9 maart 1994

Contents

1 Introduction and motivation 5
- 1.1 Introduction . . . 5
- 1.2 The control system design loop . . . 7
- 1.3 Conclusion . . . 7
- 1.4 Acknowledgments . . . 10

2 Mathematical models of systems 11
- 2.1 Introduction . . . 11
- 2.2 Models and how to obtain them ? . . . 13
  - 2.2.1 Linear time-invariant systems . . . 13
  - 2.2.2 Black box system identification, a case study using a Graphical User Interface . . . 15
  - 2.2.3 Continuous time and discrete time models . . . 24
- 2.3 Grey Box identification, parameter estimation . . . 24
  - 2.3.1 Grey box identification of an induction motor . . . 24
  - 2.3.2 Derivation of a first identification scheme . . . 25
  - 2.3.3 It's better when also the rotor speed is measured ! . . . 28
  - 2.3.4 Some simulations . . . 28
- 2.4 Physical modelling with Bond Graphs . . . 31
- 2.5 Nonlinear system identification using neural state space models . . . 31
- 2.6 Conclusions . . . 32

3 Control System Design Specifications 35
- 3.1 Control configurations . . . 35
  - 3.1.1 One-degree-of-freedom control (1DOF) . . . 35
  - 3.1.2 Two-degree-of-freedom control (2DOF) . . . 36
  - 3.1.3 Transfer functions of interest in the 1 DOF control strategy . . . 37
- 3.2 Control specifications . . . 39
  - 3.2.1 Closed-loop stability . . . 39
  - 3.2.2 Disturbance rejection and tracking . . . 39
  - 3.2.3 Decoupling . . . 40
  - 3.2.4 Time-domain spec's . . . 43
  - 3.2.5 Sensitivity . . . 43
  - 3.2.6 Robustness specifications . . . 48
  - 3.2.7 Minimum variance control . . . 49

- 3.3 Conclusion . . . 49

4 PID controllers 51
- 4.1 PID-control laws . . . 51
- 4.2 PID Tuning . . . 52
- 4.3 Model-based PID-tuning . . . 54
  - 4.3.1 Tuning for stabilization . . . 54
  - 4.3.2 Performance-based tuning . . . 56
  - 4.3.3 Classical PID tunning with Xmath . . . 60
  - 4.3.4 PID-Tuning with Optimization . . . 64
- 4.4 Implementation . . . 65
  - 4.4.1 Analog or digital? . . . 65
  - 4.4.2 Anti-windup . . . 66
  - 4.4.3 Incremental PID . . . 67
  - 4.4.4 Leadlag, a special class of PID-controllers . . . 68
- 4.5 Example: The glas tube manufacturing plant . . . 68
- 4.6 Conclusions . . . 69

5 Model-based control 71
- 5.1 Pole placement - state feedback . . . 71
  - 5.1.1 The pole placement problem . . . 72
  - 5.1.2 Solution of the pole placement problem . . . 72
  - 5.1.3 A numerical example . . . 73
  - 5.1.4 Continuous time example . . . 74
- 5.2 How do we find the state: Observers . . . 75
- 5.3 Dynamics of estimated state feedback controllers . . . 77
- 5.4 The linear quadratic regulator . . . 78
  - 5.4.1 The problem . . . 78
  - 5.4.2 An example . . . 79
  - 5.4.3 The general case . . . 80
  - 5.4.4 An example . . . 81
  - 5.4.5 The steady state solution . . . 83
  - 5.4.6 Solving the algebraic Riccati equation . . . 83
  - 5.4.7 Another example . . . 85
  - 5.4.8 Elimination of cross terms . . . 86
- 5.5 Neural optimal control design . . . 87
- 5.6 Conclusions . . . 87

Bibliography 89

Chapter 1

Introduction and motivation

In this Chapter, we briefly discuss the philosophy of model-based computer-aided control system design and its main benefits (and possible drawbacks). Section 1.1 is the introduction, while Section 1.2. describes the typical control design loop, which has become feasible ever since the computational power of both 'numerical' machines (workstations, PCs) and control system hardware has been growing. General conclusions can be found in Section 1.3.

1.1 Introduction

Control system engineering is concerned with modifying the behavior of dynamical systems to achieve certain prespecified goals. The way this is usually done is to start from a given system, complete it with certain measurement devices (called *sensors*) that assess the general state of the system. Based on this information control inputs are determined in order to achieve the goals. They are applied to the system via *actuators*. Of course, *feedback* is one of the crucial central themes in this story. With the advent of *mechatronics*, systems are designed in an integrated way, using elements of mechanics, electronics and computer science. Contrary to the traditional approach, where controllers were only used as a last resource (if everything else failed), the new philosophy concentrates on the incorporation of control system design from the very beginning. A nice example is the manufacturing of *flexible* robots, in which the lack of stiffness of new, ultralight materials such as composites, is compensated by sophisticated control strategies.
Therefore, in contrast with the classical approach, control system design has become an *active* discipline, which is taken into account *ab initio* in the design of new products, such as compact discs, flexible robots, video recorders, etc...

For any design of a controller, four items seem crucial.

1. **Process model:** Central in the new control system design philosophy is the concept of a *mathematical model*. The experience of the operator who 'knows' ('fingerspitzengefühl') how the plant will respond to certain inputs is the simplest kind of model. At the other end, a sophisticated model may consist of a set of coupled nonlinear partial differential equations. The modelling issue in control is very different from its counterparts in other areas of science (despite obvious similarities of course). It centers on finding robust mathematically

parsimonious models that are consistent with the main dynamical features of the systems behavior. Such models may be constructed from measured data by a process called *system identification*. In many cases, some equations representing the system's behavior can be written down based on physical laws and more than often, they will contain some unknown parameters. These parameters can then be estimated (using techniques that are grouped under the name *parameter estimation*) from available measurements.
It has been claimed that one of the big advantages of the classical control tools such as PID-controllers or more recent (post-)modern approaches such as fuzzy controllers, etc..., is that they don't need mathematical models to be operational. This statement is not completely true. Indeed, before such a PID-controller is installed, extensive *tuning* is needed as to find the right values of the 3 tunable parameters. This is nothing else than obtaining a rough model of the system. In addition, PID-controllers or fuzzy controllers in many cases do nothing more than just stabilizing the system (a fact that even for most fuzzy controllers can not be guaranteed) and nothing can be said about *optimality* with respect to performance and or robustness.
All of these requirements can however be tackled when a (not necessarily accurate) model of the system is available. One of the central messages is that more than often an extremely accurate model of the system is not really needed (though it might be helpful of course). Nevertheless, there is an issue of *robustness* involved, which brings us to the following point.

2. **Model uncertainty:** Neglecting "model uncertainty" leads to controllers which are 'too tight' and likely to become unstable in a real environment. Such model uncertainty is however always present. Recall that we are not interested in a precise mathematical model of the real system, but that we only want to capture the most relevant dynamical features in a (by preference simple) model. This always leads to simplifying approximations (for instance, the model works fine for low frequencies but gives considerable deviations for high frequencies..), which must be taken into account in the model based design of the controller (model uncertainty in the high frequencies may for instance reduce the robustness).

3. **Type of inputs:** It is impossible to design controllers that work well for all kinds of inputs. The designer has to decide which inputs are most important and frequent and accept inferior performance for other inputs. An important point is that any list of inputs must include possible disturbances and the way they will interact with the system.

4. **Performance and robustness objectives:** Once a model has been obtained, a *model based controller* has to be designed taking into account certain *performance*, *sensitivity* and *robustness specifications*. These are requirements that describe how the system has to perform in closed-loop, meaning *with the controller operating*. Some examples are:

   - The closed-loop system is stable.
   - The actuator effort is small.
   - The effect of disturbances (noises) on the output or internal variables is small (*disturbance rejection*).
   - The outputs of the system respond in particular ways to reference inputs. A particular example is *tracking* of a reference input. Another example is *decoupling*, when we want to control a specific output with one or more inputs without affecting the other

outputs. For example, an autopilot system is to track simultaneously both airspeed and altitude rate commands. A change in commanded speed should not severely affect the commanded altitude rate.

- Variables must stay within prespecified ranges. For instance, there may be an upper bound on the overshoot of a step response.
- The closed-loop system must keep its performance in the face of *model uncertainty*.

Not all control specifications result in *linear* controllers. For instance, the minimum time control problem with a limit on actuator effort, results in a highly nonlinear controller, known as a *bang-bang* controller. Another example are the so called self-tuning or adaptive controllers, which lead to sophisticated nonlinear control strategies, in which even *chaotic* behavior might be observed. A third class of nonlinear controllers consists of *neural networks*, which we will briefly treat in Chapter 5.

1.2 The control system design loop

So far we have pointed out that model-based control system design has several benefits, which, up to this day, are largely ignored in the industrial environment. The rationales behind model-based control system design are summarized in Figure 1.1. Maybe one of the reasons is that the design of multivariable controllers that are optimal with respect to certain performance and robustness specifications is not straightforward at all. Typically, the design of a model-based controller requires several iterations, the elements of which are illustrated in Figure 1.2. All of these elements are treated (not necessarily in full detail) in the course.
In Chapter 2, we briefly summarize possible (mathematical) models of systems that are suited for control system design. We also describe in some length how such models can be obtained via multivariable system identification techniques. Performance and robustness specifications, together with control configurations are the subject of Chapter 3. In Chapter 4, we elaborate extensively upon the oldest existing control strategy, namely PID controllers, which has widespread use in industry. More advanced model-based control strategies are treated in Chapter 5. Everywhere, we have aimed at illustrating the points we make with real or artificial (numerical) examples. One may wonder whether it is computationally feasible to achieve good models and controllers. Without doubt, much has changed the last decade or so. The available data processing capacities of present day PCs and workstations, combined with the availibility of user-friendly software packages (both for system identification and control systemn design, such as in e.g. Matlab [21] or Xmath [2]) makes that the user can now concentrate on the really important issues (such as performance and robustness and implementational aspects) without bothering too much about the fine mathematical detail.

1.3 Conclusion

Research on system identification, model-based control and implementations of the resulting algorithms into graphical user interfaces, when applied to real industrial processes and systems, will potentially result in products and production strategies that *are cleaner, safer and faster, are more user and cost efficient, allow for just-in-time and 'zero-defect' production, have narrower*

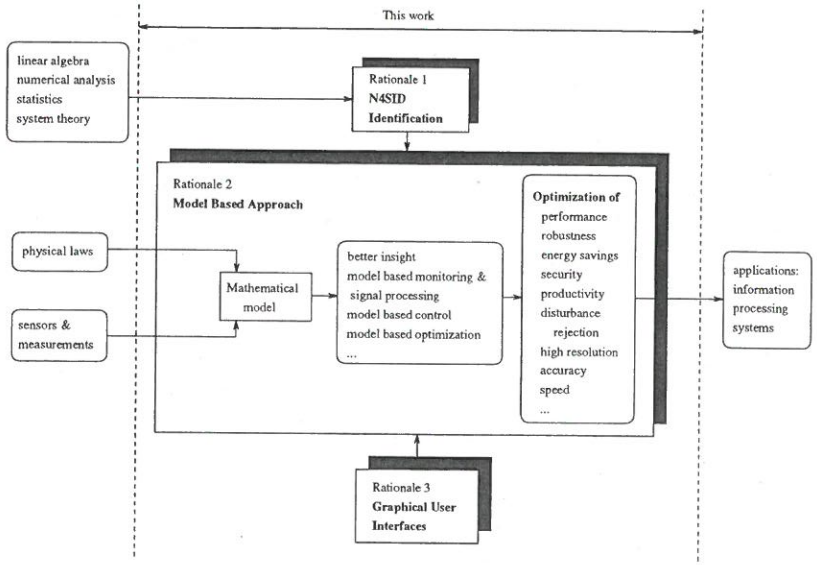


Figure 1.1: Rationales behind model-based control system design: Mathematical models of systems are obtained in several ways (see Chapter 2). They allow us to design controllers and control strategies that satisfy performance and robustness specifications (Chapter 3, 4 and 5). The design of both models and controllers is greatly facilitated by the availibility of user-friendly software packages with Graphical User Interfaces.

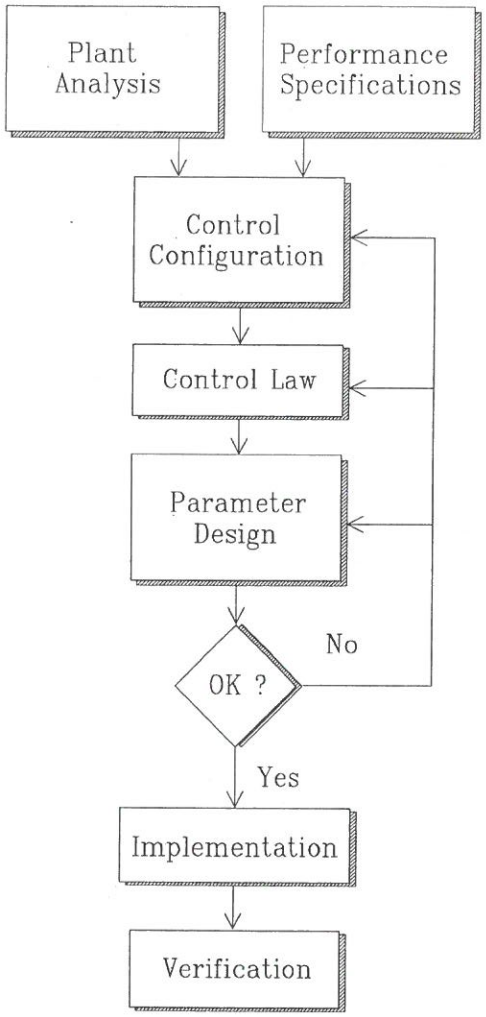


Figure 1.2: Computer-aided control system design is typically an iterative process in which models, control configurations and control laws may change according to performance and robustness specifications. In this course, we mainly concentrate on the design procedures while we say little to nothing about the real-life hardware implementations or requirements (such as sensors and actuators).

tolerances, are more flexible and responsive to user's specifications, have an improved performance, etc ....

The crucial new ingredient here is that systems can be optimized when a model is available. Modeling allows to incorporate and exploit a priori information, ultimately leading to improved performance. In combination with a user friendly tool such as the Graphical User Interface, it allows to implement Rapid Prototyping Methodologies, either on a real life prototype plant or within a *virtual engineering* setting, in which the real plant is replaced by a(n) (approximate) model.

1.4 Acknowledgments

We would like to thank all of our colleagues in our research group SISTA of the Department of Electrical Engineering of the Katholieke Universiteit Leuven. Especially Peter Van Overschee for the parts on system identification, Willem Minten for the bondgraph material, Paul Van Vuchelen for the Chapter on PID control, Johan Suykens for the sections on Neural Networks and Johan Buelens, who in brought our computer infrastructure in such a good shape, that it is still up and running without much trouble several months after he had to leave us.

Parts of the research reported in this memorandum were made possible by Grants from the Belgian Programme on Interuniversity Attraction Poles (IUAP-17 and IUAP-50) initiated by the Belgian State, Prime Minister's Office, Science Policy Programming and by the Concerted Research Action on Applicable Neural Networks supported by the Flemish Government. The support and funding by the NFWO (Nationaal Fonds voor Wetenschappelijk Onderzoek) is also acknowledged.

Chapter 2

Mathematical models of systems

In this Chapter, we present an non-exhaustive survey of models and modelling techniques which are illustrated by some real-life examples. In Section 2.1. we elaborate upon the basic differences between an engineer and a physicist or mathematician concerning the modelling problem. In Section 2.2. we explain how mathematical models can be obtained for linear systems (Sections 2.2.1, 2.2.2) and we illustrate by a case study how a Graphical User Interface is a helpful tool to find state space models. In Section 2.2.4. we analyze the relation between discrete time and continuous time models while a grey box modelling example is elaborated upon in Section 2.3. In Section 2.4 we spend some words on bond graphs while in Section 2.5, we briefly touch upon Neural Net modelling. Finally, some conclusions are found in Section 2.6.

2.1 Introduction

A dynamic system can conceptually be described as in Figure 2.1, which covers almost all physical, economical, biological, industrial, etc ... systems. One could distinguish between mental, intuitive or verbal models, or graphically oriented approaches such as graphs and tables, but we will mainly be interested in mathematical models. Such models are described as differential (continuous time) or difference (discrete time) equations. They describe the dynamic behavior of a system as a

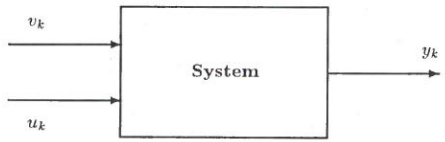


Figure 2.1: A dynamic system with deterministic inputs u_k, outputs y_k and disturbances v_k (see below). All arrows represent vector signals and k is the discrete time index. The user can control u_k but not v_k. In some applications, either u_k, v_k or w_k can be missing. The measured output signals provide useful information about the unknown system.

function of time. Mathematical models exist in all scientific disciplines, and, as a matter of fact, form the heart of scientific research itself. They are used for simulation, operator training, analysis, monitoring, fault detection, prediction, optimization, control system designs, quality control, *etc.* Most typically, models are highly useful in those situations in which experimenting with the real system is too expensive, too dangerous, too difficult or merely impossible. Last but not least, mathematical models are used for *control* and *feedback*, which, by the way, is one of the major engineering inventions.
The underlying motivation of this course is the claim that

model-based solutions are superior in performance and robustness

compared to plain heuristic approaches. This fundamental insight is not new, and has been at the heart of modern control theory ever since the work of Kalman in the beginning of the sixties. In this Chapter, we analyze how a mathematical model can be obtained. Surprisingly enough, this question has been neglected for a long time, even if all the work on control system design explicitly *assumes* that there is a model available. It is only the last decade or so that modeling of industrial processes has become a serious mathematical engineering discipline.

Basically, there are two main roads to construct a mathematical model of a dynamic system. Physicists will be interested in models (physical laws) that carefully explain the underlying essential mechanisms of observed phenomena and that are not *falsified* by the available experiments. The necessary mathematical equipment is that of *partial differential equations*. This is the analytic approach, which rigorously develops the model from *first principles*.
For engineers however, this framework is often much too involved to be really *useful*. The reason is that engineers are not really interested in the exact model *as such*, but more in the potential engineering applications of models. In this perspective, a mathematical model is only one step in the global design of a system. The quality of a model is dictated by the ultimate goal it serves. *Model uncertainty is allowed as long as the robustness of the overall system is ensured.* Engineers -in contrast with mathematical physicists- are prepared to *trade-off model complexity versus accuracy*. A complex model will lead to a complex design, while a simplistic model will deteriorate overall performance and robustness of the final implementation. As an example, the best model for simulation (for instance a set of partial differential equations which accurately models the system's behavior) is not the best one for control, because, as a generic property of control system design, the complexity of the controller and the degree of difficulty associated with its implementation, are proportional to the model complexity. Therefore engineers will typically use *system identification* techniques to build their models. This is the field of modeling dynamical systems from experimental data: Experiments are performed on a system, a certain parameterized model class is predefined by the user and suitable numerical values are assigned to the parameters so as to fit as as closely as possible the recorded data. In this sense, system identification is the dynamic extension of curve fitting. Finally there is a validation step, in which the model is tried out on experimental data which were not used in the system identification experiment.

The message is that system identification provides a meaningful engineering alternative to physical modeling. Compared to models obtained from physics, system identification models have a limited validity and working range and in some cases have no direct physical meaning. But, they are relatively easy to obtain and use and even more importantly, these models are simple enough

to make model-based control system design mathematically (and also practically) tractable. Of course, there are still problems such as the choice of an appropriate model structure, the fact that many systems are time-varying and often largely underestimated measurement problems (appropriate sensors, sampling times, filters, outlier detection, etc...).
Let us conclude this section by saying that system identification, being a typical engineering discipline, borrows many of its concepts and techniques from other mathematical and engineering fields (as was already suggested in Figure 1.1), such as optimization, numerical analysis, linear algebra, complex function theory, statistics, sensors and physical devices, experimental design, software engineering, etc ...and therefore is in many respects an interdisciplinary activity.

2.2 Models and how to obtain them ?

In this section, we explore how discrete-time linear time-invariant systems can be obtained from input-output experiments. First we treat models that describe such systems, followed by a case study in which we show how a model for a glass-tube production process can be obtained.
One may wonder why we are only dealing with discrete time systems and not with continuous time ones, which are ubiquitous in nature and in an industrial environment. The reason is simple since in most cases, modelling is done based on sampled data. There exists however a one-to-one relation between discrete time and continuous time models, which is briefly treated in this section also.

2.2.1 Linear time-invariant systems

While most processes and systems are nonlinear, surprisingly, linear approximations (for instance around a working point) often provide good engineering models for which, in addition, there exist many well developed control system design methodologies (to be treated in Chapter 3, 4 and 5). Such linear state space models are treated in this section. For more details we refer to the appropriate literature (see e.g. the reference list at the end of the text).

Mathematically, linear time-invariant discrete time models are described by the following set of difference equations[1]:

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{2.1}$$
$$y_k = Cx_k + Du_k + v_k \tag{2.2}$$

with

$$\mathbf{E}[\begin{pmatrix} w_p \\ v_p \end{pmatrix} \begin{pmatrix} w_q^t & v_q^t \end{pmatrix}] = \begin{pmatrix} Q_s & S_s \\ S_s^t & R_s \end{pmatrix} \delta_{pq} \geq 0 \tag{2.3}$$

In this model, we have

- **vectors:** The vectors $u_k \in \mathbb{R}^{m\times 1}$ and $y_k \in \mathbb{R}^{l\times 1}$ are the measurements at time instant k of respectively the m inputs and l outputs of the process. The vector x_k is the state vector of the process at discrete time instant k and contains the numerical values of n states. These states do not necessarily have a direct physical interpretation but they have a conceptual relevance. Of course, if the system states would have some physical meaning, one could always find a similarity transformation of the state space model to convert the states to physically meaningful ones. $v_k \in \mathbb{R}^{l\times 1}$ and $w_k \in \mathbb{R}^{m\times 1}$ are unmeasurable vector signals. It is assumed that they are normally distributed, zero mean, stationary white noise vector sequences.

- **matrices:** $A \in \mathbb{R}^{n\times n}$ is called the system matrix. It describes the dynamics of the system (as completely characterized by its eigenvalues). $B \in \mathbb{R}^{n\times m}$ is the input matrix which represents the linear transformation by which the deterministic inputs influence the next state. $C \in \mathbb{R}^{l\times n}$ is the output matrix which describes how the internal state is transferred to the outside world in the measurements y_k. The term with the matrix D is called the direct feedthrough term. In continuous time systems this term is most often 0, which is not the case in discrete time systems due to the sampling [2].
  The matrices $Q_s \in \mathbb{R}^{n\times n}$, $S_s \in \mathbb{R}^{n\times l}$ and $R_s \in \mathbb{R}^{l\times l}$ are the covariance matrices of the noise sequence w_k and v_k.

A graphical representation of the system can be found in Figure 2.2.

Let us comment in some detail why it is often a good idea to try to fit experimental (industrial) process data to the model just described.

- First of all, for multiple-input, multiple output systems, the state space representation is the only model that is convenient to work with in *computer aided control system design* (CACSD). Most optimal controllers can be effectively computed in terms of the state space

[1] $\mathbf{E}$ denotes the expected value operator and δ_{pq} the Kronecker delta.

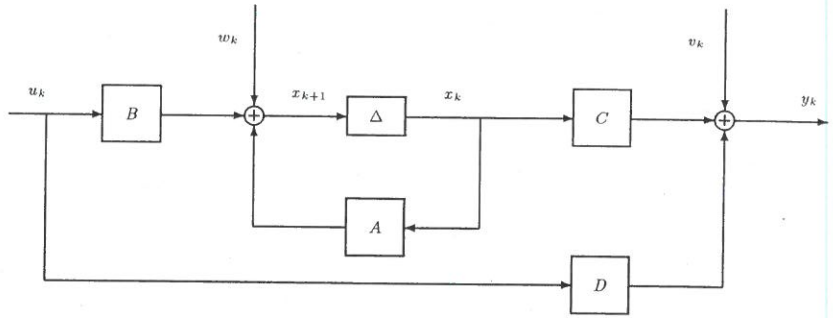


Figure 2.2: This picture is the same as the one in Figure 2.1. But here, we have restricted ourselves to finite dimensional linear time invariant systems to be identified. The vector signals u_k and y_k are available (measured) while v_k, w_k are unknown disturbances. The symbol Δ represents a delay. Note the inherent feedback via the matrix A (which represents the dynamics). Sensor or actuator dynamics are completely contained in A too. It is assumed that u_k is available without measurement noise.

[2] There are additional technical assumptions on the matrices A, B, C, Q_s, S_s, R_s in terms of controllability and observability, but we won't go into too much technical detail in this presentation.

model, while for other system representations (such as e.g. matrix fractional forms) the calculations are not so elegant.

- Observe that we have collected *all* dynamics in one matrix A, that is to say that the eigenvalues of the matrix A will describe all the dynamical modes that have been measured, whether they come from the real system, from stochastic dynamic disturbances, from measurement sensors or the dynamics of the input actuators. This is quite unusual as compared to approaches that are described in the literature, in which one always carefully distinguishes between e.g. deterministic models (such as models for the 'real' system and sensor and actuator dynamics) and noise models for stochastic disturbances (as is for instance the case in the Box-Jenkins approach [9]). The point here is that more often than not, we don't care about the precise origin of the dynamic modes, since, if they are important, they will certainly influence the controller action, independent of their origin. There is a modern trend in CACSD to define what is called a *standard plant* (see e.g. [10]), which contains the model of all disturbances, all sensors and the system model in one general state space description, which exactly corresponds to the model we will use.

- The assumption that the noise sequences v_k and w_k are Gaussian is quite natural for many applications, due to the central limit theorem (which here acts as an important engineering simplification). Of course, it can not always be made uncriticaly. Nevertheless, the approximation is often very satisfactory.
  Measurement noise (which is ubiquitous in industrial environments) is included in the stochastic white noise sequence v_k, while it is assumed that the input is applied to the system without distortion (in other words, we do not assume that the sequence u_k is corrupted by noise).

- A crucial question is of course why linearity would apply to everyday processes, since we all know that most phenomena are intrinsically non-linear. One reason is the experience that many industrial processes are really well approximated by linear finite dimensional systems and that sometimes, complex behavior can be captured by choosing the order n high enough. In order to cope with non-linearities, two measures are possible: Either the non-linearity is dealt with by identifying a time-varying system using a recursive updating of the model. This corresponds to a local linearization of the nonlinear system. A second possibility is provided by the observation that (mild) nonlinearities do not matter as they can be incorporated in the control design (robustness for dynamic uncertainties). Moreover, it is well known that a controller effectively linearizes the behavior of a system around a working point. Finally, we recall that the design of a controller is relatively easy for linear finite dimensional systems. As a matter of fact, this is the only class of systems for which CACSD is actually tractable in full generality and for which there is a complete rigorous theory available.

2.2.2 Black box system identification, a case study using a Graphical User Interface

The black box system identification problem is the following:

Given input and output measurements $u_1, \ldots, u_N$ and $y_1, \ldots, y_N$, $(N \to \infty)$.
Find an appropriate order n and the matrices $A, B, C, D, Q_s, R_s, S_s$.

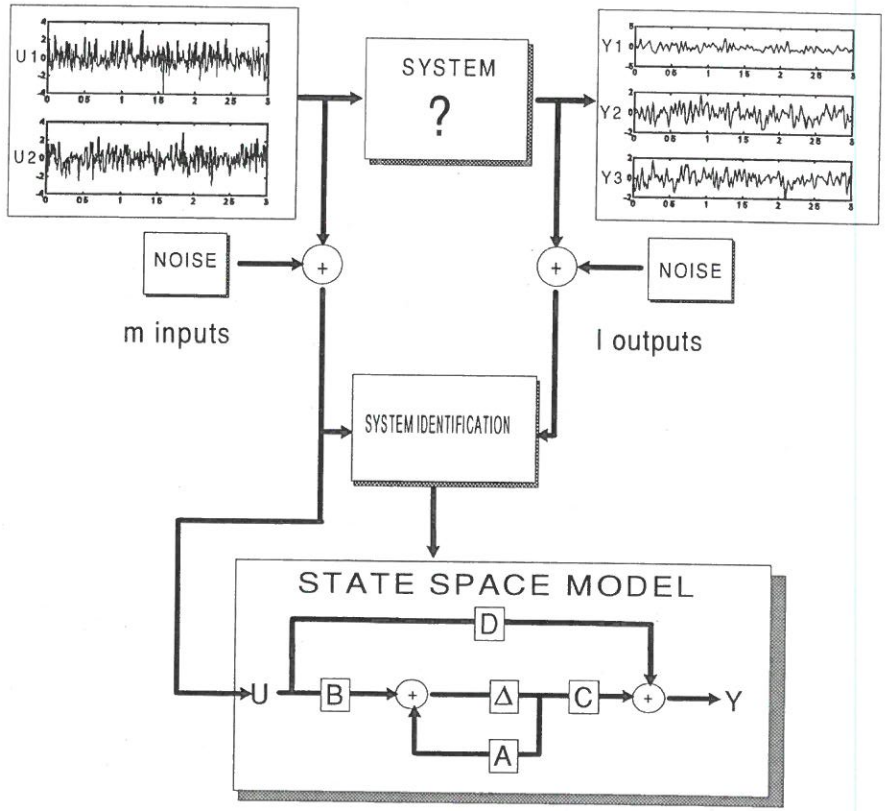


Figure 2.3: *System identification problem: How to obtain a state space model from input-output data?*

The problem is ilustrated in Figure 2.3. We will not treat here the full mathematical details of the required algorithms and concepts, for which we refer to [12] [13] and the references therein. Instead, we will illustrate the results on a glass-tube manufacturing problem. These results are a nice illustration of the statements we made at the beginning of this Chapter. The model of the process could in principle be characterized completely using the laws of physics (in this case the laws that govern the behavior of solidifying glass). But, not only would this be a formidable task -if practically possible at all-, but even if there was such a model, it would be impossible to derive an appropriate control action to regulate the system, because of the complexity of the model. Here, it will be shown how a relatively simple state space model, obtained by mathematical identification methods implemented within a Graphical User Interface can be used to design controllers.
In a later Chapter, we will illustrate how the obtained state space model leads to the design of a minimum variance controller.

Figure 2.4 shows a schematic description of the glass tube production process. Quartz sand is fed to the machine at the top. The sand is melted to glass inside the furnace. The glass tubes are drawn at the bottom of the machine.

Inputs: The inputs are drawing speed and mandrill pressure. The drawing speed is the speed at which the tubes are pulled out at the bottom of the machine. The mandrill pressure is the pressure applied to the mandrill at the top of the machine. Through the mandrill, this pressure is then applied to the inside of the tubes when they are pulled out of the machine.

Outputs: The outputs to be controlled are the geometrical parameters of the tube which are its mean diameter and its thickness.

Two input-output data sequences were measured (sampling frequency 2 Hz). The first one (1355 points, see Figure 2.5) is used for the identification of the process. The second one (893 points) is used for the validation of the results. For both input signals a pseudo random binary noise sequence was used as input [3].

Using the Graphical User Interface (GUI) **ISID II**, a full description of which can be found in [13] [14] one can obtain different models from these input-output data. One of the main tools in this GUI is the so-called Chain Window, which graphicaly represents the user's requirements on how to obtain the state space models for these data.

Figure 2.6 shows the chain that was used to identify the glass tube manufacturing process. **ISID II** allows for *graphical programming* of the chain, which means that only the manipulation of mouse buttons is required to build a chain and tune its parameters. The algorithms (the building blocks) can be easily selected from the menus. They can be connected via a simple graphical interaction. The user is guided in making these connections since she can only connect compatible blocks (the output has to be compatible with the input).
Each of the algorithm building blocks contains a sophisticated algorithm, the parameters of which are automatically set to a meaningful default value. The user doesn't have to know the details

[3] The data we use here are real industrial measurements on a real industrial commercial production plant. However, due to reasons of confidentiality, we are not allowed to reveal in detail exact physical production parameters.

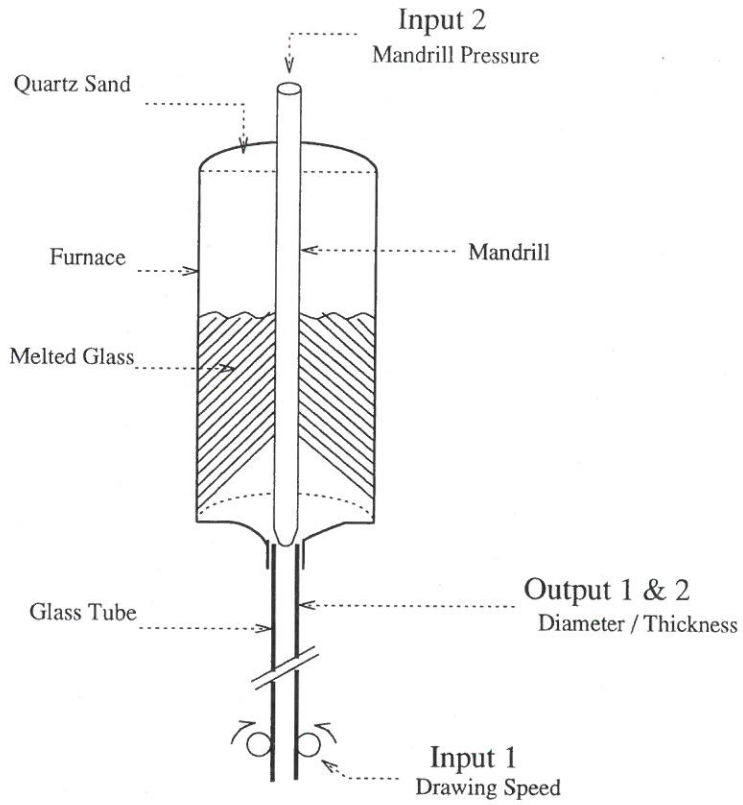


Figure 2.4: The glass tube manufacturing process. Inputs are drawing speed and mandrill pressure. Outputs are tube diameter and thickness. The input signals were pseudo-binary noise sequences. These inputs are sufficiently 'wild' to excite all the dynamic modes of the system (which is necessary since we want to control these modes afterwards). The tubes that are produced from these inputs are worthless since they are too irregular due to the wild inputs. This situation is typical for industrial system identification: There is a basic trade-off between the production loss that goes together with experimenting and the production quality enhancement as a result of the identification/model-based control design.

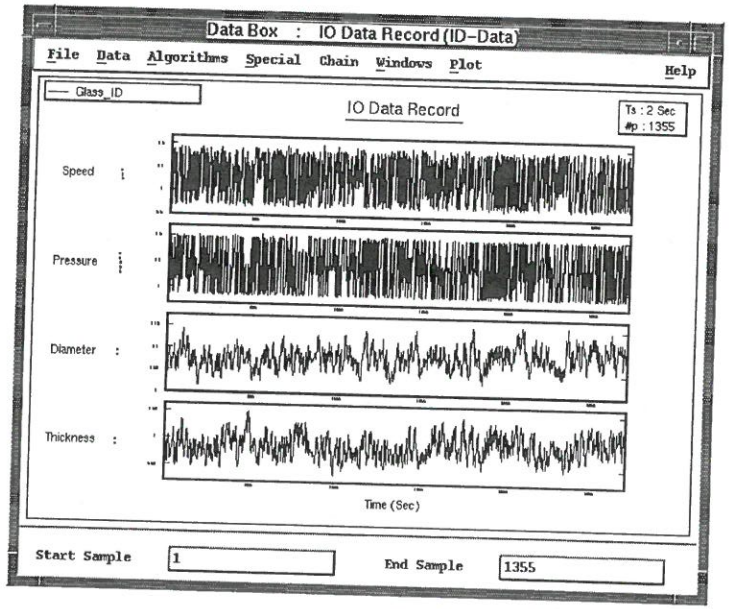


Figure 2.5: Data set used for the identification of the glass tube production process. The inputs (top two signals) are drawing speed and mandrill pressure. They are excited using pseudo-binary noise sequences. The outputs (bottom two signals) are tube diameter and thickness.

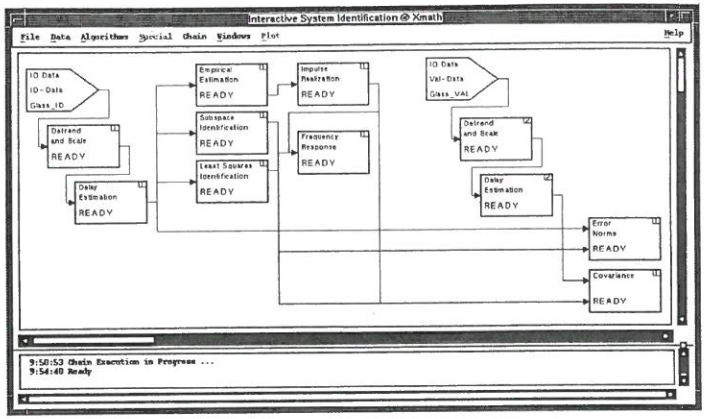


Figure 2.6: A typical **ISID II** chain representing a full identification process. Each block represents an algorithm. The blocks to the left are the input-output data and preprocessing blocks, followed by the identification blocks. To the left, we also see some validation blocks. Behind each block there is an algorithm window that visualizes the algorithm specific data and allows for adjustment of the parameters. This chain shown here in this Figure was used for the identification of a glass tube manufacturing process.

of the algorithm, of which only the most important parameters are readily available for direct manipulation.

ISID II contains several *hard baked* chains, which represent programs that perform the most common identification procedures. For instance, the identification from filtered input output data using a least squares algorithm, followed by a cross validation of the obtained model. In this way, the novice user is given a guideline of how to use the identification algorithms, without the requirement of having any identification expertise and without having to read through the manual[4]. Figure 2.7 shows the full computer screen during the **ISID II** session, with all the algorithm windows open.

Processing: Both input-output data records are first detrended to remove the mean and the linear trends in the data. Since the measurements of the outputs can only be done when the tubes are sufficiently cooled down, there are significant time delays. The two delay estimation blocks estimate these delays and compensate for them by shifting the input and output signals in the appropriate direction. Figure 2.8 shows the algorithm window behind the delay estimation algorithm block.

Identification: Three different identification algorithms are applied to the data. By comparing the resulting models, we can enhance our confidence in them.

- A first model is obtained by realizing the impulse response obtained from the empirical transfer function.

[4] Many more functionalities of **ISID II** are described in [14] [13].

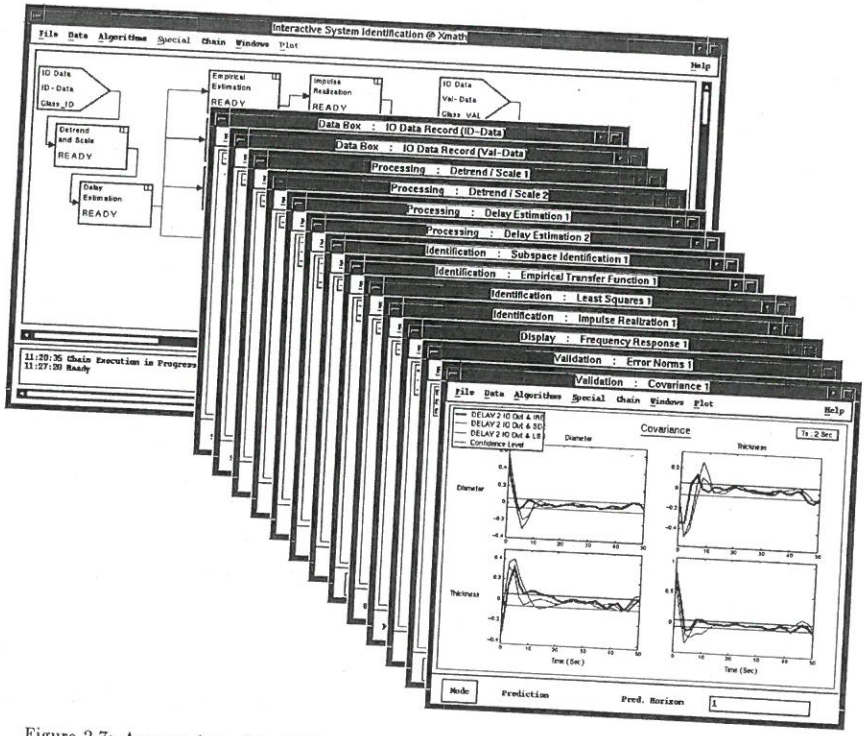


Figure 2.7: An overview of the **ISID II** session for the identification of the glass tube manufacturing process. The Chain Window is displayed in the top left corner (partially covered). All algorithm windows are organized in a "card system". **ISID II** contains several features to manipulate and reorder this "card system" of windows. For instance, one can 'pull out' the delay estimation window (see next figure), analyse it in detail and then just 'put it back'.

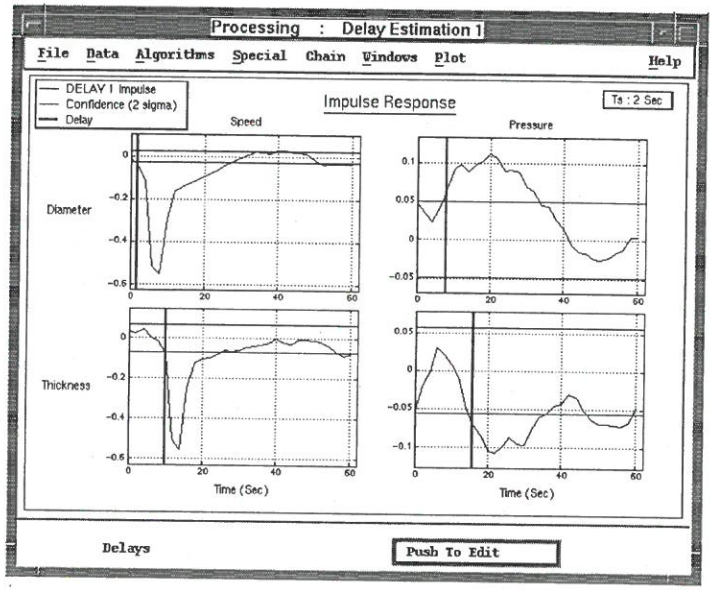


Figure 2.8: Behind each algorithm building block there is an algorithm window. These **ISID II** algorithm windows display algorithm specific data. These algorithm windows allow the user to change parameters graphically. Shown here is the window that goes with the delay estimation algorithm. The delays are indicated by the vertical lines. They can be changed by clicking on the vertical lines and dragging them to the desired value i.e. the intersection of the impulse response and the confidence bounds (horizontal lines).

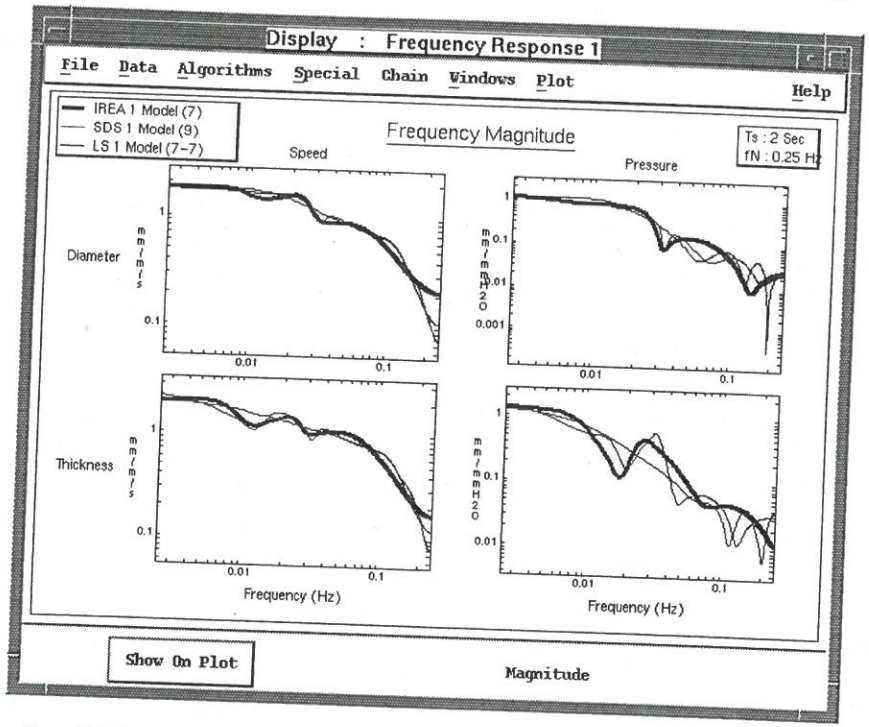


Figure 2.9: Frequency response of the three identified models, superimposed on each other. One can highlight one specific plot, corresponding to one specific model, by clicking with the left mouse button on the model's name. There are also several zoom in features available.

- A second model is obtained from a subspace (**N4SID**) identification algorithm (full details can be found in [12]). Interactive decision about the order n of the model is a piece of cake using only the mouse buttons.
- A last model is obtained by a least squares identification.

Validation: The models are validated by comparing the measured and simulated outputs (validation and identification data).

Display: The transfer functions of the obtained models are displayed on the same plot to compare the models in the frequency domain. The algorithm window is shown in Figure 2.9.

2.2.3 Continuous time and discrete time models

The bilinear transformation β maps the unit circle to the imaginary axis and the interior of the unit circle into the open left half plane.

$$\beta : s \to z = \frac{1+s}{1-s} \qquad \beta^{-1} : z \to s = \frac{z-1}{z+1}$$

On state space level we can represent this transformation as follows:

Theorem 1
Let $G(s)$ be the transfer matrix with realization $[A_c, B_c, C_c, D_c]$ where A_c has no eigenvalues in $s = 1$. Let $H(z)$ be the discrete time system defined by

$$H(z) = G(\frac{1+s}{1-s})$$

Then $H(z)$ is a discrete time LTI system with realization $[A_d, B_d, C_d, D_d]$ satisfying:

$$\begin{array}{ll} A_d = (I + A_c)(I - A_c)^{-1} & A_c = (A_d - I)(A_d + I)^{-1} \\ B_d = \sqrt{2}(I - A_c)^{-1}B_c & B_c = \sqrt{2}(A_d + I)^{-1}B_d \\ C_d = \sqrt{2}C_c(I - A_c)^{-1} & C_c = \sqrt{2}C_d(I + A_d)^{-1} \\ D_d = D_c + C_c(I - A_c)^{-1}B_c & D_c = D_d - C_d(A_d + I)^{-1}B_d \end{array}$$

It can be shown that the bilinear transformation preserves stability.

2.3 Grey Box identification, parameter estimation

Grey box system identification techniques are used if a priori knowledge about the system and its structure is avaliable. For instance, the differential equations of a mechanical system, or at least part of it, can easily be derived. The (characteristics) parameters of most components can be measured (length, mass) and they appear in the differential equations as coefficients. Sometimes these coefficients are nonlinear functions of the physical parameters. It may also happen that some characteristics, like friction coefficients, are difficult to measure. The task is then to estimate them based on input-output measurements. Since we want to estimate physical parameters, it is obvious to identify the system in continous time. Transferring a system from continous to discrete time generates equations wich sometimes are non linear in the parameters. This can be avoided by staying in the continous time domain. Time varying systems can also be easier to identify in the continous time domain.

2.3.1 Grey box identification of an induction motor

The technique will be explained here on an example, in which we try to identify the parameters of an induction motor. For full details we refer to [22].
Assume that no physical parameters (resistances, inductances) are known. We start by writing down the equations of the induction motor:

$$\begin{array}{rcl} a_2\frac{d^2V_s^q}{dt^2} + a_1\frac{dV_s^q}{dt} + a_0V_s^q & = & b\frac{dI_s^d}{dt} + c_3\frac{d^3I_s^q}{dt^3} + c_2\frac{d^2I_s^q}{dt^2} + c_1\frac{dI_s^q}{dt} + c_0I_s^q \\ a_2\frac{d^2V_s^d}{dt^2} + a_1\frac{dV_s^d}{dt} + a_0V_s^d & = & -b\frac{dI_s^q}{dt} + c_3\frac{d^3I_s^d}{dt^3} + c_2\frac{d^2I_s^d}{dt^2} + c_1\frac{dI_s^d}{dt} + c_0I_s^d \end{array} \qquad (2.4)$$

with:

V_s^q, V_s^d	q-phase and d-phase of stator voltage in the stationary reference frame
I_s^q, I_s^d	q-phase and d-phase of stator current in the stationary reference frame
I_r^q, I_r^d	q-phase and d-phase of rotor current in the stationary reference frame

If we first define R_s to be the stator resistance, R_r the rotor resistance (referred to the stator), L_m the stator-rotor mutual inductance, L_s the stator self inductance, L_r the rotor self inductance, ω the rotorspeed (rad/sec) and let $\sigma = L_s L_r - L_m^2$, then it turns out that the parameters in the model can be associated to the physical parameters as:

$$\begin{aligned} a_2 &= L_r^2 & c_3 &= L_r\sigma \\ a_1 &= 2L_rR_r & c_2 &= R_sL_r^2 + R_rL_rL_s + R_r\sigma \\ a_0 &= \omega^2L_r^2 + R_r^2 & c_1 &= 2R_sR_rL_r + R_r^2L_s + \omega^2\sigma L_r \\ b &= L_m^2\omega_r R_r & c_0 &= R_r^2R_s + \omega^2R_sL_r^2 . \end{aligned}$$

The inputs are I_s^q and I_s^d. The output is V_s^q. First, we describe the system equations in the Laplace domain.

$$a_2s^2V_s^q + a_1sV_s^q + a_0V_s^q = [bsI_s^d + c_3s^3I_s^q + c_2s^2I_s^q + c_1sI_s^q + c_0I_s^q$$

Suppose that we apply a linear operator (system) G(s) to both sides of this equation. If we neglect the initial conditions the following equation results:

$$G(s)[a_2s^2V_s^q + a_1sV_s^q + a_0V_s^q] = G(s)[bsI_s^d] + G(s)[c_3s^3I_s^q + c_2s^2I_s^q + c_1sI_s^q + c_0I_s^q]$$

After measuring V_s^q,I_s^q and I_s^d, generate signals of the form $G(s)[V_s^q]$, $sG(s)[V_s^q]$, $s^2G(s)[V_s^q]$ and $G(s)[I_s^q]$, $sG(s)[I_s^q]$, ... from the inputs and outputs. We get a linear relation between the generated signals which has the same coefficients as the differential equation:

$$\begin{aligned} &a_2G(s)[s^2V_s^q] + a_1G(s)[sV_s^q] + a_0G(s)[V_s^q] \quad = \\ &bG(s)[sI_s^d] + c_3G(s)[s^3I_s^q] + c_2G(s)[s^2I_s^q] + c_1G(s)[sI_s^q] + c_0G(s)[I_S^q] \end{aligned}$$

One requirement for chosing $G(s)$ is to take its denominator degree at least as large as the highest order of differentiation that appears in the differential equation. This prevents the filters to amplify the high frequency noise.

2.3.2 Derivation of a first identification scheme

The filtered signals are put in a large data matrix H_m. Every column of H_m consists of a filtered signal as described. The identification problem reduces now to estimate the vector Θ wich is related to H_m by the following equation:

$$H_m\Theta = 0$$

where

$$\Theta = \begin{bmatrix} a_2 & a_1 & a_0 & b & c_3 & c_2 & c_1 & c_0 \end{bmatrix}^t$$

The vector Θ can be estimated with a method that's called the General Total Least Squares Method. However, in this particular application, the coefficients of the differential equations are

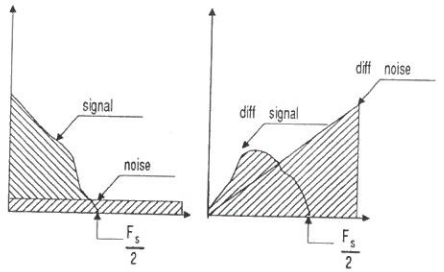


Figure 2.10: The magnitude of some typical signal and noise spectra are shown in the left figure. The right figure shows the spectra of the differentiated signals (multiply by $s = j\omega$). The ideal Nyquist frequency is indicated as well. Differentiating causes the signals and noises to be amplified in the same amount. The signal to noise ratio is decreasing if the Nyquist frequency is larger than the band width of the signal, because some frequency bands contain only noise and no signal. Only the noise will be amplified in these frequency bands. So it is important not to choose the sampling frequency too high.

time varying. This causes a problem when applying the filter $G(s)$. The reason is that in the time domain:

$$G[a(t)y(t)] \neq a(t-\tau)G[y(t)]$$

where $\tau > 0$ is a delay (see below).
One cannot apply the law of convolution, when $G(s)$ is applied to a time varying differential equation. Van hamme [27] has proven that this can only be tolerated if G(s) has a flat amplitude characteristic and a linear phase characteristic. So we require:

$$\begin{aligned} |\, G(j\omega)\,| &= 1 \\ \angle(G(j\omega)) &= k\omega \text{ (for some positive constant k)} \end{aligned}$$

in a large frequency region. This means that the signals that will be generated: $(G(s)[V_s^q]$, $G(s)[sV_s^q]$, $G(s)[s^2V_s^q])$ are derivatives of V_s^q. So we need pure differentiators. Differentiators are not very popular in signal processing because they amplify the noise in the signal (see Fig. (2.10)). But high frequency components in the signal are also amplified a lot. So the frequency band from DC to the Nyquist frequency has to be filled with the signal, which means that we need a suitable excitation of the motor. Oversampling has to be avoided because it degrades the signal to noise ratio as explained in Fig. (2.10). The area under the curve of the differentiated noise spectrum increases with frequency although the area under the curve of the differentiated signal, which was originally band limited, is bounded. The resulting identification scheme is shown in Figure 2.11. All the filters, except for the anti-aliasing filters, are digital. The digital approximation of a differentiator poses an extra problem. A phase characteristic of 90 degrees (property of a differentiator) is impossible to achieve. This is obvious for an FIR filter, which has a delay (derivative of the phase characteristic) proportional to the filter order. If we want higher order filters, which give a better fitting of the desired frequency response, extra delays will be introduced in the flow diagram of the signal. This causes no real problem, provided the delays are

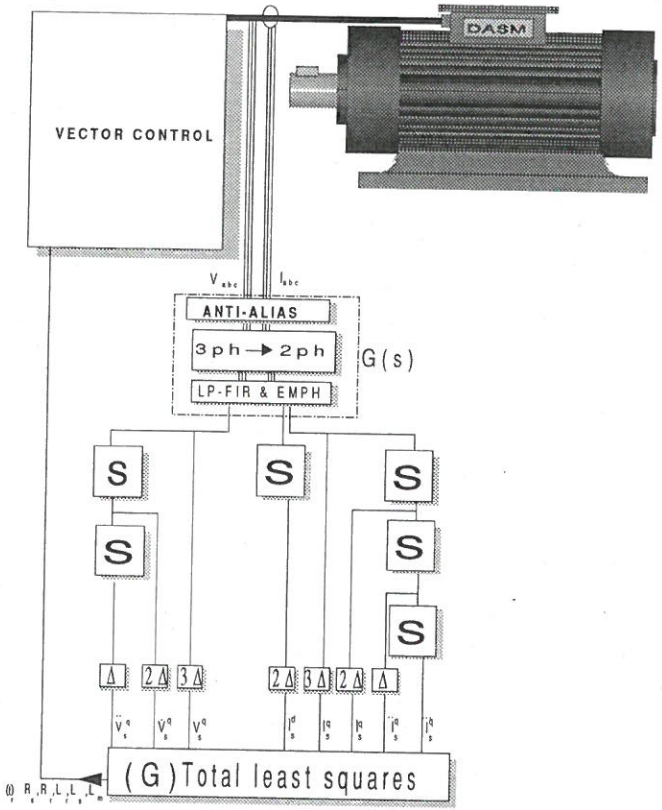


Figure 2.11: Identification scheme where S is a Differentiator, Δ:1 sample delay, EMPH: filter to emphasize certain frequencies in the signal to have a better fit in the emphasized frequency band, $G(s)$ is the transfer function we have defined before.

parameter	exact	estimate
R_s	.94 Ω	.948 Ω
R_r	.61 Ω	.62 Ω
L_r	.0668 H	.067 H
L_s	.0668 H	.0668 H
L_m	.065 H	.0652 H

Table 2.1: *Exact and estimated physical parameters*

integer valued, because the signals can be reshifted in time after the filter action. For the design of IIR filters, we also need extra delays to have stable filters. We can allow those extra delays, provided they are also integer valued. The previous inequality now becomes an equality:

$$G[a(t)y(t)] = a(t-\tau)G[y(t)]$$

with τ the integer delay of G.

2.3.3 It's better when also the rotor speed is measured !

The problem with induction motors is that no measurements are done at the rotor side. This means that the beheavior of the rotor is difficult to *see* from the stator side. This results in a very bad estimation quality for some coefficients of the system equations. The solution is to measure the rotor speed and rearrange the system equations in the following way:

$$d_2\left(\frac{d^2V_s^q}{dt^2} + \omega_r^2 V_s^q\right) + d_1\frac{dV_s^q}{dt} + d_0 V_s^q =$$
$$e\omega_r\frac{dI_s^d}{dt} + f_4\left(\frac{d^3I_s^q}{dt^3} + \omega_r^2\frac{dI_s^q}{dt}\right) + f_3\frac{d^2I_s^q}{dt^2} + f_2\frac{dI_s^q}{dt} + f_1\left(\frac{d^2I_s^q}{dt^2} + \omega_r^2 I_s^q\right) + f_0 I_s^q$$

with $\sigma = L_sL_r - L_m^2$, $d_2 = L_r^2$, $d_1 = 2L_rR_r$, $d_0 = R_r^2$, $e = L_m^2R_r$, $f_4 = L_r\sigma$, $f_3 = 2R_rL_rL_s$, $f_2 = 2R_sR_rL_r + R_r^2L_r$, $f_1 = L_r^2R_s$ and $f_0 = R_r^2R_s$. The result of this regrouping is an almost time invariant system (temperature effects have a large time constant compared to the sampling interval). This is easier to identify and more robust. Some components are put together to get more accurate results, e.g. $\frac{d^3I_s^q}{dt^3}$ and $\omega_r^2\frac{dI_s^q}{dt}$ share the same coefficient $L_r\sigma$. Although the system we want to identify is a time invariant system, the characteristics of G(s) must still satisfy the requirements we have been deriving before. The coefficients we want to identify are time invariant, but the coefficients of the differential equations still vary in time due to the speed changes. The differential equations of the machine have not changed, they are just written in another way. The resulting identification scheme's is given in figure 2.12.

2.3.4 Some simulations

The test consists of reducing the rotor speed continuously from maximum speed (300 rad/sec) to stall in 4 seconds. The simulated motor was a 3 KW induction motor with the parameters $R_r = .61\Omega$, $R_s = .94\Omega$, $L_m = .065H$, $L_s = .0668H$ $L_r = .0668H$. The evolution of the physical parameter estimates in time are given in figure 2.13. After a few seconds the estimated parameters converge to the values in Table 2.1. The method can still be improved upon. The coefficients of

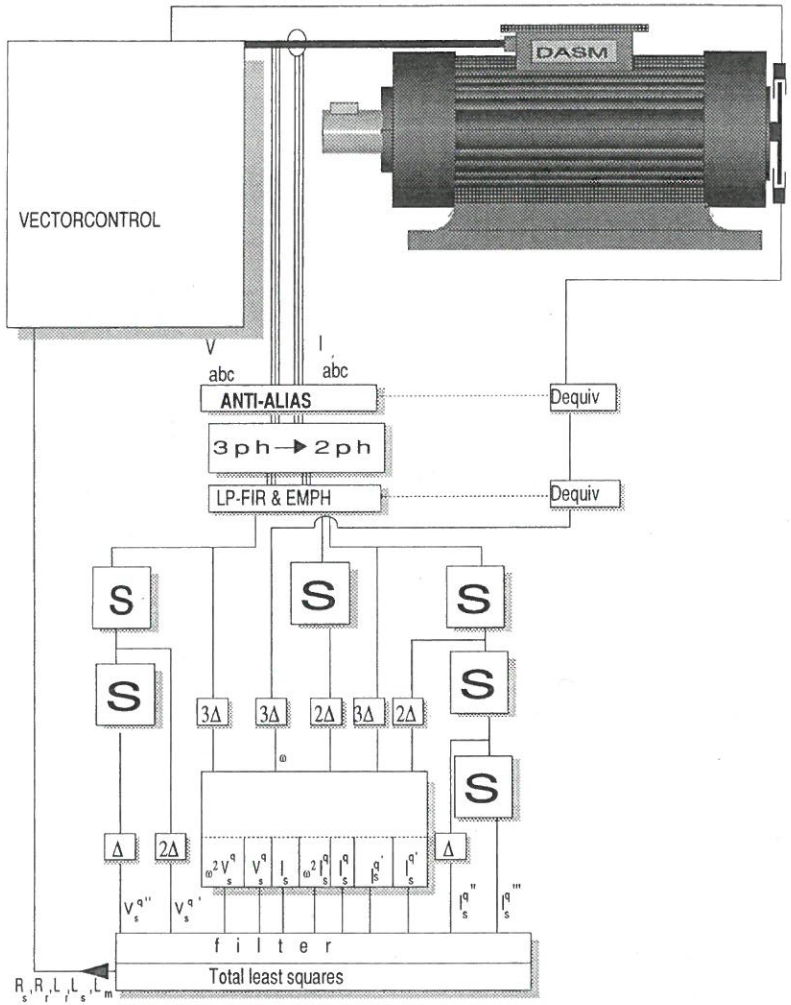


Figure 2.12: Global identification scheme of the second method. The large block in the middle at the bottom of the scheme contains all the non linear combinations of the measured signals. The blocks named *dequiv* are delays which shift the rotor speed signal in time by the same amount as the anti-alias filter and the correcting filters.

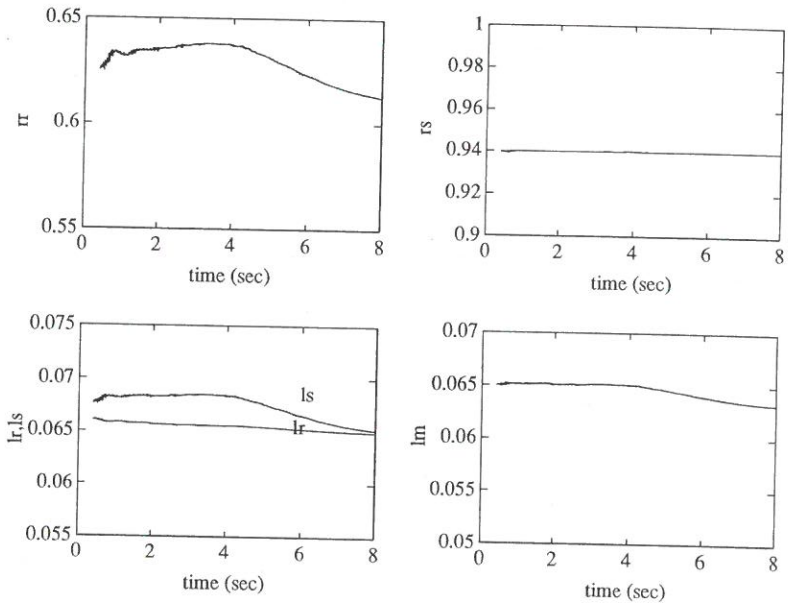


Figure 2.13: Estimated induction motor parameters

the differential equations are related by non-linear equations. This results in a General Total Least Squares problem with non linear constraints. This can be solved with very efficient optimization software. The results are much better than the unconstrained version. This method works very well even at very high signal to noise ratios. The error covariances of the solution can be up to 10000 times smaller with the constraint version.

2.4 Physical modelling with Bond Graphs

So far, we have described techniques to generate models from input-output measurements. The technique of bondgraphs provides another angle of attack to the modelling problem, in which now the insights are more based on physical relations. Especially for engineering systems bond graphs provide a structured methodology to obtain relationships between variabels and for instance, to set up a set of equations, in which only some of the parameters need to be estimated. Basic works on bondgraphs are [11] [18] [19]. Below we show an example of a bond graph model for the power train of a continuous variable transmission (figures 2.14 and 2.15).

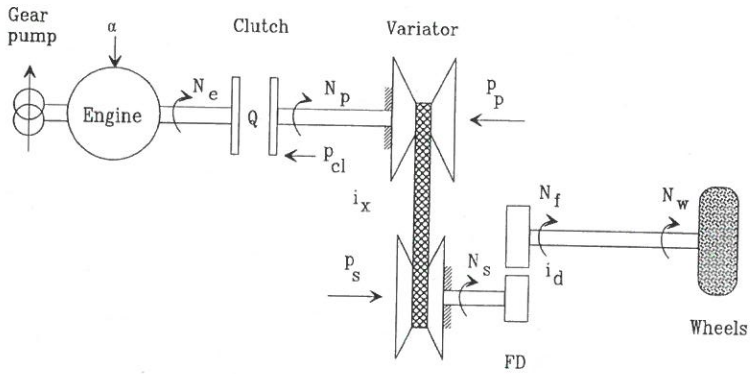


Figure 2.14: The powertrain contains a mechanical continuous variable transmission with a wet multi-plate clutch and a metal pushbelt variator. It's a universal CVT which can be used for all types of vehicles. The variator ratio i_x can be varied by changing the pressures p_p and/or p_s. The clutch control signal Q is determined by the pressure p_{cl}. The operation of the different components is not explained here.

2.5 Nonlinear system identification using neural state space models

The use of artificial neural networks in nonlinear system identification is motivated by the fact that any continuous nonlinear function can be approximated arbitrarily well on a compact interval by a feedforward neural network with one or more hidden layers. Up till now input/output models like NARX or NARMA(X) models that are parametrized by neural nets are most well known and

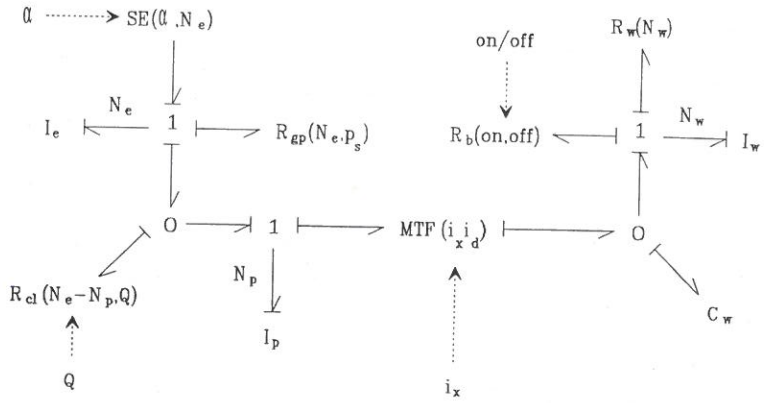


Figure 2.15: *The causal bond graph of the powertrain, to model and compute the interactions between the various inertances, compliances and resistances. The partial arrows represent power signals, while the dotted arrows show where external signals enter the system. The 1-junctions represent classical force equilibria (Kirchoff's Voltage Law: KVL), while the 0-junctions correspond to speed equilibria (Kirchoff's Current Law: KCL). The causal assignments are indicated by a small causal line on each power signal.*

many learning algorithms exist in order to find locally optimal interconnection weights, such as e.g. backpropagation, prediction error algorithms, extended Kalman filtering, quasi Newton and conjugate gradient methods.

On the other hand also nonlinear state space models, parametrized by neural nets can be considered and are called *neural state space models*. For neural state space models learning algorithms exist both for the deterministic identification case and the stochastic case with process noise and measurement noise. Another attractive feature of these models is that an LFT (Linear Fractional Transformation) representation can be obtained where the neural net model is represented as an uncertain linear model with parametric uncertainties caused by nonlinear perturbations. This leads to a new method to bridge the well known 'gap' between system identification and control on deriving meaningful bounds of uncertainty on the parameters, derived from I/O data on a plant. The procedure consists of first doing nonlinear system identification by neural nets, secondly represent the model by an LFT and finally use this representation in a standard robust performance control scheme. This scheme makes it possible to model nonlinear phenomena such as saturation effects, hysteresis, chaos and systems operating around many working points and to use the identification results in a modern robust control framework. More details can be found in [24] [25].

2.6 Conclusions

In this section, we have been discussing mathematical models for systems and how to obtain them. The list of possibilities was certainly not complete. Important identification methods for linear

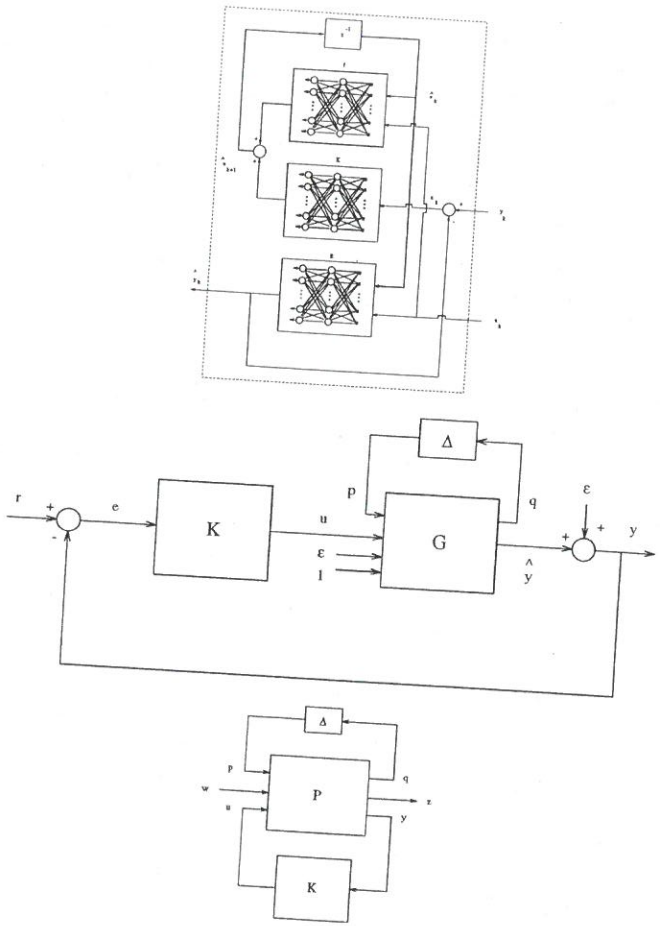


Figure 2.16: a/ Nonlinear state space model for nonlinear system identification on systems corrupted by process and measurement noise, b/ 1-DOF control scheme with the neural state space model represented as an LFT, c/ standard robust performance control scheme.

systems include subspace identification algorithms, for which now Graphical User Interface driven software tools are available. We have also shown some examples of other modelling techniques such as grey box modelling, bond graphs and neural networks. The idea that will be pursued in the chapters to come is that all of these modelling techniques provide models which can be used in deriving model-based control strategies.

Chapter 3

Control System Design Specifications

One could view a closed-loop configuration as a way of embedding a given plant in a 'larger' higher-dimensional system so that the dynamic behavior of the global system is better than that of the original system. In this chapter, we first discuss possible control configurations, in which the controller can be considered as a 'free' system, which needs to be tuned so that certain *specifications* are satisfied (if possible). We'll discuss one-degree and two-degree-of-freedom control configurations, and diverse control specifications such as stability, disturbance rejection, tracking, decoupling, several time domain specifications, sensitivity requirements, robustness issues and minimum variance control.

3.1 Control configurations

Depending on the number of 'free' blocks that enter the closed-loop control strategy and that are to be designed by the control engineer, one speaks about one-degree or two-degree-of-freedom controllers.
In what follows, the reader should keep in mind that all arrows represent *vector signals* in the most general case (and not just one single signal as one would maybe inclined to think).

3.1.1 One-degree-of-freedom control (1DOF)

Feedforward control

Intuitively, the best controller would be a precompensator which is the inverse of the system to be controlled (see figure 3.1). In such a configuration, ideally, the real output y_r and the desired output y_{sp} are exactly the same, provided that the process model describes the process accurately. However, if the model is not accurate enough, or worse, if the inverse of the system is unstable (as is the case when the original system has so-called non-minimum phase zeros in the linear case), this is obviously a dangerous practice. Moreover, even if the precompenstaor is stable, any unexpected disturbance of whatever origin will deteriorate the performance.
For continuous time systems, in order to compute the feedforward signal, it might be necessary to differentiate the reference signal once or even more times. This is only possible if the reference

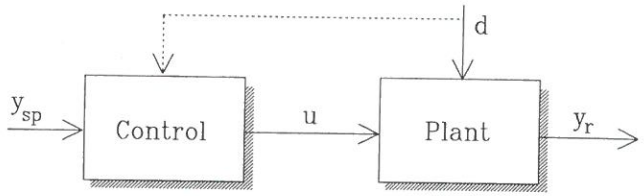


Figure 3.1: *Feedforward control: There is a precompensator which preceedes the plant. This control strategy - while extremely naive - rarely delivers satisfactory results (and when used without caution might even be dangerous).*

trajectory is sufficiently smooth, i.e. the reference trajectory has finite derivatives up to the required order.

Feedback control

In a closed-loop or feedback control configuration the control signal depends on the error e, the difference between the desired output y_{sp} and the real output y_r (Figure 3.2). This type of control

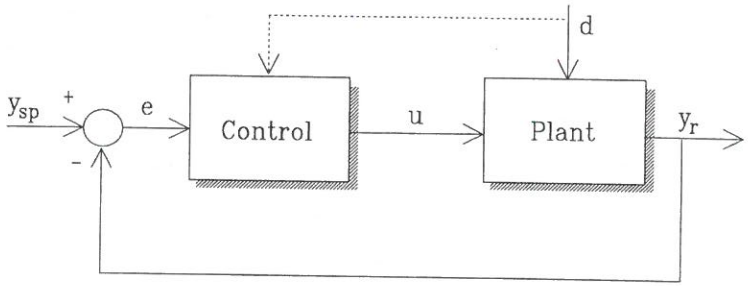


Figure 3.2: *Unit Feedback control: The feedforward control configuration of 3.1 is now extended with a closed-loop, which compares the actual output y_r back to the desired specified output y_{sp}.*

has the advantage of relative insensitivity for disturbances, uncertainties in the process and process parameter changes.

3.1.2 Two-degree-of-freedom control (2DOF)

In a two-degree-of-freedom control strategy, feedforward control and feedback control are combined. The feedback is necessary for stabilization and to change the system's dynamics, but alone it might be insufficient to get rid of tracking errors, so that a feedforward control must be added. There are two possible configurations. For both configurations, it should be noticed that the controller has two input signals, namely the desired output y_{sp} (feedforward) and the error signal e (feedback).

2DOF control using explicit feedforward

In this configuration, the feedforward control signal u_{ff} and the feedback control signal u_{fb} are added explicitly (figure 3.4).

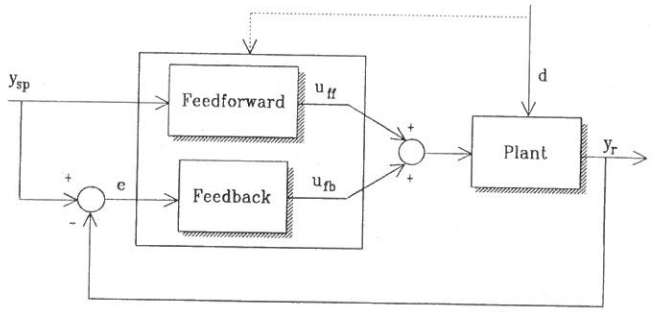


Figure 3.3: *Explicit feedforward/feedback control configuration*

2DOF control using prefiltering as feedforward

In this configuration, the feedback controller is designed first. The feedforward controller (realized by the prefilter) is added next (figure 3.3). As we have said before, it is not always possible to design such a prefilter, particularly in the case when the closed loop transfer function contains 'unstable' zeros (zeros in the RHP), since these zeros become unstable poles in the prefilter.

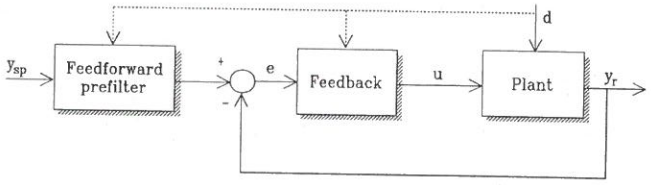


Figure 3.4: *Prefiltering technique: The feedback control is designed first, and then a precompensator is added to enhance performance.*

3.1.3 Transfer functions of interest in the 1 DOF control strategy

In a 1DOF controller, the cascade connection of controller followed by plant is closed by negative[1] unity feedback Consider the feedback configuration of figure 3.5. Using simple formula manipulations with transfer matrices (starting from $y_1 = Ce_1$ and $y_2 = Ge_2$), we easily find the following transfer matrices from the inputs u_1, u_2 and u_3 to the outputs y_1, y_2 and y_3.

[1] It is an historical convention among engineers to use *negative* feedback.

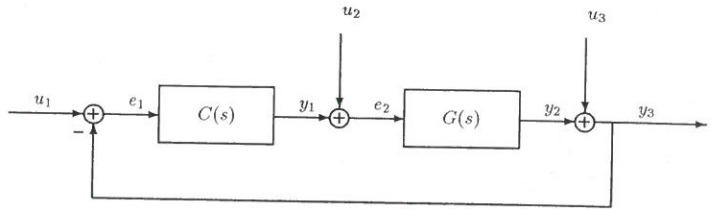


Figure 3.5: Classical one-degree-of-freedom control configuration. Recall that all arrows represent vector signals. Typically, u_1 contains the so-called reference inputs, of which there are as many as there are outputs in y_3 (hence the closed-loop system is 'square', it has as many inputs as there are outputs); u_2 is called the process-noise and u_3 are called the disturbances. The signal e_1 is the controller input and e_2 is the plant input. All the transfer matrices from any vector signal to any other vector signal have a certain name and most performance and robustness specifications can be quantified in terms of these transfer matrices.

	y_1	y_2	y_3	e_1	e_2
u_1	$(I+CG)^{-1}C$	$(I+GC)^{-1}GC$	$(I+GC)^{-1}GC$	$(I+Ge)^{-1}$	$(I+CG)^{-1}C$
u_2	$-(I+CG)^{-1}CG$	$(I+GC)^{-1}G$	$(I+GC)^{-1}G$	$-(I+GC)^{-1}G$	$(I+CG)^{-1}$
u_3	$-(I+CG)^{-1}C$	$-(I+GC)^{-1}GC$	$(I+GC)^{-1}$	$-(I+GC)^{-1}$	$-(I+CG)^{-1}C$

When manipulating expressions with these transfer matrices (recall that in the general case, these are matrices with rational entries, so multiplication is not commutative), the following relations are useful:

$$CG(I+CG)^{-1} = (I+CG)^{-1}CG \qquad (I+GC)^{-1}GC = GC(I+GC)^{-1}$$
$$G(I+CG)^{-1} = (I+GC)^{-1}G \qquad C(I+GC)^{-1} = (I+CG)^{-1}C$$
$$(I+GC)^{-1}GC + (I+GC)^{-1} = I$$

Some of these transfer functions have a standard name:

Loop gain: $L = GC$. If the loop gain is large in a certain sense, we see that the transfer matrix from $u_1 \to y_3 \approx I$, and that the transfer matrix from $u_3 \to y_3 \approx 0$. The first approximate equality means that we approximately achieve *decoupling*, while the second one means that the disturbances in u_3 are 'rejected' (they do not appear in the outputs y_3).

Loop return difference: $I+GC$: The name originates in the following argument: Cut the unit feedback link and call the 'linking' signal z. Then the transfer function from z to the error $z - y_3$ satisfies $(z - y_3) = (I + GC)z$.

Sensitivity: $S = (I + GC)^{-1}$: As we will see below this transfer matrix plays an important role in the disturbance rejection $u_3 \to y_3$ and in the first-order sensitivity analysis of the closed-loop system.

Complementary sensitivity: $T = (I + GC)^{-1}GC$: The name comes from the fact that for all frequencies, we have $S + T = I$.

Control sensitivity: $C(I + GC)^{-1}$: The *control sensitivity matrix* is a measure for the control effort necessary to yield a certain performance.

Most often, control specification for performance and robustness can be expressed as certain requirements on one or more of these transfer matrices. For instance, a *large loop gain* might be advantageous but for certain systems, it can not be too large (because an increasing loop gain drives the closed-loop poles towards to open-loop plant zeros so that for non-minimum phase plants, the closed-loop system will eventually grow unstable). The *sensitivity matrix* is used to measure the tracking and disturbance rejection capabilities of the control system. It is also related to the sensitivity of the closed-loop system to a perturbation δG on the plant G. The *complementary sensitivity matrix* is used for measuring the stability margins of the controlled system for multiplicative output uncertainty.

In terms of the sensitivity, complementary sensitivity and control sensitivity transfer functions, one also finds $T = LS = SL$. As an example, one may use these properties to find that the transfer matrix from

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \rightarrow \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \text{ is given by } \begin{pmatrix} (I+CG)^{-1}C & -(I+CG)^{-1}CG \\ G(I+CG)^{-1}C & -G(I+CG)^{-1} \end{pmatrix} = \begin{pmatrix} CS & -CSG \\ T & SG \end{pmatrix}$$

3.2 Control specifications

Whether we have a 1DOF or 2DOF control configuration, the control blocks in the closed-loop are to be considered as 'free parameters' which are to be designed, taking into account certain specifications. This is what this section is about.
There might be several ways of implementing a certain control strategy: manual (trial-and-error) or based on expertise (expert systems), mathematics, numerical optimization, neural networks, fuzzy systems,.... Some of these methods will be treated in the chapters to come, but first, in this section, we will be dealing with some typical control specifications.
Although most of the statements in this section carry over to non-linear controllers, we will confine ourselves here to linear control strategies.

3.2.1 Closed-loop stability

One of the most important specifications is the requirement that the closed-loop system should be stable. This means that *all* the transfer matrices from the inputs u_1, u_2 and u_3 to the outputs y_1, y_2 and y_3 should represent stable systems. In terms of state space models for these transfer matrices, this implies that all the system matrices should have their eigenvalues with the unit disk (for discrete time systems) or the left half plane (for continuous time systems).
A controller that achieves this goal, is called a *stabilizing* controller.

3.2.2 Disturbance rejection and tracking

Disturbance rejection refers to the suppression of the effect of disturbances entering the closed-loop via the inputs u_3 onto the outputs y_3. Since the transfer matrix from $u_3 \rightarrow y_3$ is the sensitivity

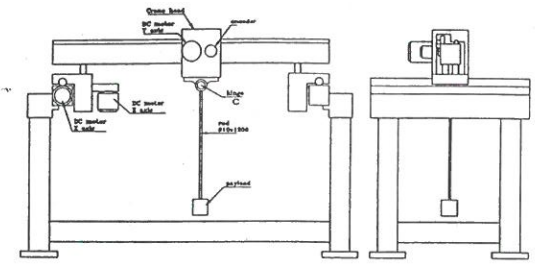


Figure 3.6: Schematic representation of a granty crane

matrix $S(s)$, this implies that $S(s)$ should be small in a certain sense. As we have already seen, this could be achieved by a 'large' loop gain $G(s)C(s)$.
In many cases, the loop gain GC is strictly proper, i.e. $\lim_{s\rightarrow\infty} GC = 0$. This is the case when the loop gain contains at least one integrator. We then have

$$\lim_{\omega\rightarrow\infty} S(j\omega) = I\ .$$

Hence, $S(j\omega)$ can be made small only over a finite frequency range. This implies that perfect control in the sense of disturbance rejection is impossible over all frequencies!
We also find

$$\lim_{\omega\rightarrow\infty} T(j\omega) = 0$$

Note that $T(s)$ describes the tracking capabilities of our system ($u_1 \rightarrow y_3$). We would like this transfer matrix to be close to unity, which is impossible for all frequencies.

Example: A granty crane (which is depicted in Figure 3.6) can be modelled with a transfer function like

$$G(s) = \frac{100(s^2+1)}{s(s^2+100)}$$

This model has 3 poles and 2 zeros on the imaginary axis. It can be stabilized by a unit controller($C = I$). The sensitivity $S(s)$ and complementary sensitivity $T(s)$ are given by:

$$\begin{aligned} S(s) &= (I+GI)^{-1} = \frac{s(s^2+100)}{s^3+100s^2+100s+100} \\ T(s) &= GI(I+GI)^{-1} = \frac{100(s^2+1)}{s^3+100s^2+100s+100} \end{aligned}$$

These functions are plotted in figure 3.7. When $S = 0$ (at the zeros of S: 0 and $10j$ where $j = \sqrt{-1}$), then $T = I$, and the other way around. The loop gain GC has a pole on 0 (an integrator). This implies that $S(s) = 0$ at $\omega = 0$ and thus $T(0) = 1$. The loop gain is also strictly proper. Hence, one can also observe that $\lim_{\omega\rightarrow\infty} S(j\omega) = I$ and $\lim_{\omega\rightarrow\infty} = T(j\omega) = 0$.

3.2.3 Decoupling

In multivariable systems with several inputs and outputs, every input generically influences every output variable. Engineers don't like this. What they would really like is that the first input

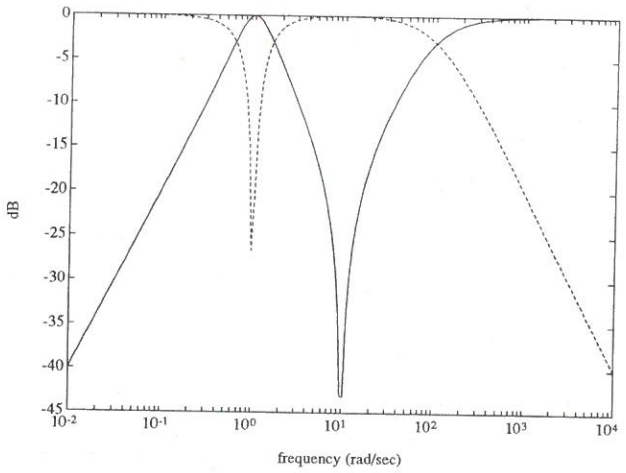


Figure 3.7: The sensitivity (full line) and complementary sensitivity (dashed line) function for a unit feedback of $G = \frac{100(s^2+1)}{s(s^2+100)}$. Recall that $S(s) + T(s) = I$, which is clearly visible on this picture.

controls only the first output, the second input the second output etc.... This is called *decoupling*. We have already seen that with a large loop gain we can approximately decouple MIMO systems. However, some systems can be completely decoupled as is shown in the following:

Theorem 2
Let $P(s)$ be a square, stable, strictly proper transfer function with $detP(s) \neq 0$ (not identically zero). Then there exists a proper stabilizing controller $C(s)$ such that $H_{y_3u_1}$ is diagonal.

Proof: $P(s)^{-1}$ is a matrix of rational (probably unstable) not proper transfer functions. Let $\{p_1, \ldots, p_k\}$ be the unstable poles of the entries of $P(s)^{-1}$ so that $\Pi_{i=1}^{k}(s-p_i)P(s)^{-1}$ has no poles in the **RHP**. Suppose q is the greatest degree of any polynomial term in $\Pi_{i=1}^{k}(s-p_i)P(s)^{-1}$. Then put

$$R(s) = \frac{1}{(s+1)^m}\Pi_{i=1}^{k}(s-p_i)P(s)^{-1}$$

where $m \geq q$ and the controller

$$C(s) = R(s)(I - PR)^{-1} \tag{3.1}$$

Then we find that

$$H_{y_3u_1} = PR = \frac{\Pi_{i=1}^{k}(s-p_i)}{(s+1)^m}I$$

which is diagonal. Observe that every I/O map is the same. □

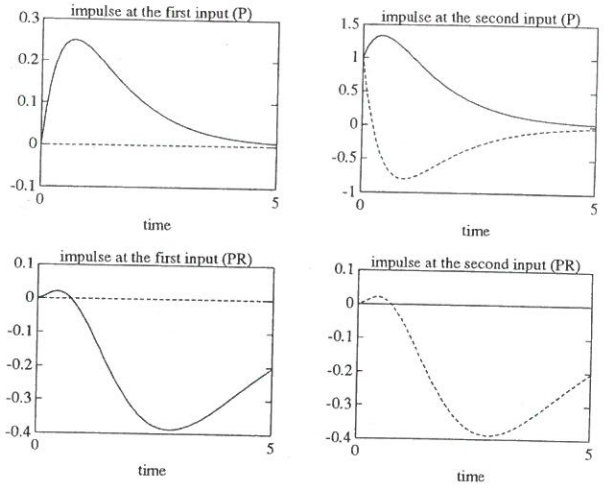


Figure 3.8: The impulse responses of a 2×2 stable system. The full line is the signal at the first output and the dashed line is the second output. The 2 upper plots are without controller and the 2 lower plots are with a decoupling controller. With the controller, the first input only influences the first output, and the second input only the second output

Example: Take the following 2 by 2 system:

$$P(s) = \frac{1}{(s+1)(s+3)}\begin{pmatrix} 1 & s+5 \\ 0 & s-3 \end{pmatrix}$$

It is coupled as can be seen on figure 3.8. $P(s)^{-1}$ is given by

$$P(s)^{-1} = \frac{(s+1)(s+3)}{s-3}\begin{pmatrix} s-3 & -(s+5) \\ 0 & 1 \end{pmatrix}$$

so that

$$R(s) = \frac{(s+1)(s+3)}{(s+1)^3}\begin{pmatrix} s-3 & -(s+5) \\ 0 & 1 \end{pmatrix}$$

Hence

$$H_{y_3u_1} = \frac{s-3}{(s+1)^3}I$$

The fact that now the closed-loop transfer from u_1 to y_3 is decoupled can also be seen in figure 3.8. An impulse at the first input influences only the first output, and similar for the second input and output. Remark also that the impulse responses are equal.

The idea now is to first design a decoupling controller $C(s)$ as given by (3.1) and then design separate single-loop controllers for the decoupled input-output pairs.

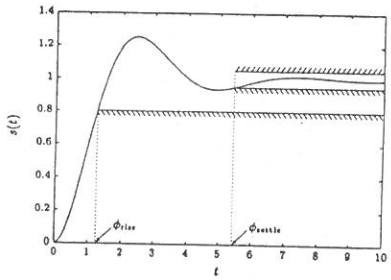


Figure 3.9: Illustration of some common definitions of the rise time and the settling time and specifications on them.

3.2.4 Time-domain spec's

The steady state error e_{ss} is equal to $(I + L(0))^{-1}$, which is the difference between one and the closed loop transfer function at $DC(s = 0)$. It is a measure for the steady state accuracy of a system. e_{ss} is also equal to the DC sensitivity, i.e., for small changes in the open loop DC gain, it gives the percentage change in closed loop DC gain divided by the percentage change in the open loop DC gain. For example, if e_{ss} equals 0.1, a 10% change in open loop DC gain will cause a change in the closed loop DC gain of about 1%. Notice that the e_{ss} is zero when $L(s)$ contains an integral factor.

The rise time T_r is the smallest time at which the step response is at least 90% of its asymptotic value (which is the closed loop DC gain). It can be verified experimentally that this time is proportional to the inverse of the cross-over frequency ω_{co}: a system with a high cross-over frequency will also have a high *response speed*.

The settling Time T_s is the smallest time after which the step response stays within 5% of its asymptotic value.

The overshoot os is the maximum amount by which the step response ever exceeds its asymptotic value, expressed as a percentage of the asymptotic value.

The undershoot us is the maximum amount the step response dips below zero, expressed as a percentage of the asymptotic value of the step response.

Some of these specifications are illustrated in Figure 3.9

3.2.5 Sensitivity

For reasons of simplicity, we only concentrate on SISO (single-input single-output systems) here, although most of the results we present here have their MIMO (multiple-input multiple output) counterpart.

First order fractional sensitivity

Consider the complementarity sensitivity transfer function for the 1 DOF-system of Figure 3.5 for a SISO system:

$$T = \frac{GC}{1 + GC}$$

which is also the transfer function from $u_1 \rightarrow y_3$. Bode observed that:

$$[\frac{\delta T(s)}{\delta G(s)}] / [\frac{T(s)}{G(s)}] = (1 + G(s)C(s))^{-1} = S(s)$$

which is the sensitivity transfer function. We thus have a *basic rule of thumb*:

$$\frac{\delta T(s)}{T(s)} \simeq S(s)\frac{\delta G(s)}{G(s)}$$

where $\simeq$ means equal to first order. In words: A relative perturbation of the plant results in a relative perturbation of the complementary sensitivity transfer function proportional to the sensitivity transfer function. Hence, when $S(s)$ is small, $T(s)$ (and therefore the tracking performance) will be relatively insensitive to plant perturbations (e.g. modelling errors).
Roughly speaking, at a frequency ω where $|S(j\omega)| = 0.1$, a 10 % change in the complex number $|G(j\omega)|$ yields a change in $T(j\omega)$ of only approximately 1 %.

Logarithmic sensitivity

We can express the previous expression as

$$S(s) = \frac{\delta \log T(s)}{\delta \log G(s)}$$

For this reason $S(s)$ is called the *logarithmic sensitivity* of T with respect to G. Recall that

$$\log T(s) = \log |T(s)| + j\angle T(s)$$

where $\angle T(s)$ is a phase angle in radians. $\log |T(s)|$ is unambiguously well-defined for all s (as long as $T(s) \neq 0$) but the phase angle $\angle T(S)$ is ambiguous: It is only defined up to within an integer multiple of 2π [2]. On any simply-connected region in the complex plane on which $T(s) \neq 0$, it is possible to make a selection of particular phase angles such that $\angle T(s)$ is continuous on the region, which makes $\log T(s)$ analytic. (When this process of phase selection is applied on the imaginary axis, it is called *phase unwrapping*). The real part $|\log T(s)|$ is expressed in *nepers* in engineering. One neper is the gain that corresponds to a phase angle of one radian and is approximately 8.7 dB. One dB corresponds to about 6.6 degrees of phase. We can rewrite the expression for the logarithmic sensitivity as

$$\begin{aligned} \delta \log |T(s)| &\simeq \Re[S(s)]\, \delta \log |G(s)| - \Im[S(s)]\, \delta\angle G(s) \\ \delta\angle T(S) &\simeq \Im[S(s)]\, \delta \log |G(S)| + \Re[S(s)]\, \delta\angle G(s) \end{aligned}$$

[2] Recall De Moivre's lemma:

$$\begin{aligned} T(s) &= |T(s)|e^{j\angle T(s)} \\ &= |T(s)|(\cos(\angle T(s)) + j\sin(\angle T(s)) \\ &= |T(s)|(\cos(\angle T(s) + 2\pi k) + j\sin(\angle T(s) + 2\pi k) \end{aligned}$$

which clarifies the problem.

Here $\Re[.]$ and $\Im[.]$ are the real and imaginary parts of the quantities between brackets.
These formulas reveal the first order changes in magnitude and phase of the closed-loop input-output transfer function caused by magnitude and phase variations in G.

Example:
Consider a change in gain, i.e.

$$\delta G(s) = \alpha G(s)$$

so that

$$\delta \log G(j\omega) \simeq \alpha$$

Hence

$$\begin{aligned} \delta \log |T(j\omega)| &\simeq \alpha \Re[S(j\omega)] \\ \delta \angle T(j\omega) &\simeq \alpha \Im[S(j\omega)] \end{aligned}$$

It follows that, if we specify $\Re[S(j\omega_0)] = 0$ at a certain frequency ω_0, the magnitude of the transfer function $H_{y_3u_3}$ is first order insensitive to gain variation α at the frequency ω_0.

To give a specific example, take [3]

$$G(s) = \frac{1}{s^2}\frac{10-s}{10+s}$$

and controller

$$C(s) = \frac{44.14s^2 + 107.3s + 39}{s^3 + 10s^2 + 55.25s + 78.14}$$

Consider the effect on $|T(j\omega)|$ of a gain perturbation of $\alpha = 25\%$, which is about 2 dB. The nominal magnitude of the I/O transfer function is

$$|T(j\omega)| = \left| \frac{G(j\omega)C(j\omega)}{1 + G(j\omega)C(j\omega)} \right|$$

The actual magnitude with the 25% gain increase in G is

$$|T^{pert}(j\omega)| = \left| \frac{1.25G(j\omega)C(j\omega)}{1 + 1.25G(j\omega)C(j\omega)} \right|$$

The first order approximation of the perturbed I/O transfer function is

$$|T^{approx}(j\omega)| = |T(j\omega)| \exp\left(0.25\Re\left(\frac{1}{1 + G(j\omega)C(j\omega)}\right)\right)$$

The magnitudes of these three tranfer functions are shown in figure 3.10. For this example, the first order prediction gives a good approximation of the perturbed magnitude of the I/O transfer function, even for this 2dB gain change in G (which is not directly 'small').

Next we study the effects on T of a phase variation in G :

$$G(j\omega) + \delta G(j\omega) = e^{j\theta(\omega)} G(j\omega)$$

[3] This is a typical plant consisting of two integrators and an all-pass transfer function which could be considered as a first order approximation of a delay.

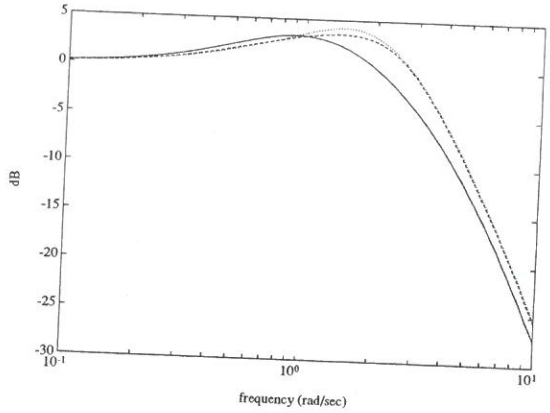


Figure 3.10: When G is replaced by $1.25G$, the I/O transfer function's magnitude changes from $|T|$ (full line) to $|T^{pert}|$ (dashed line). $|T^{approx}|$ (dotted line) is a first order approximation of $|T^{pert}|$. In this example the effect of a plant gain change as large as 25% is well approximated using the differential sensitivity.

Then

$$\begin{aligned} \delta \log |T(j\omega)| &\simeq -\theta(\omega)\, \Im[S(j\omega)] \\ \delta \angle T(j\omega) &\simeq \theta(\omega)\, \Re[S(j\omega)] \end{aligned}$$

To guarantee that $|T(j\omega_0)|$ is first order insensitive to phase variations in $G(j\omega)$, we need the specification

$$\Im[S(j\omega_0)] = 0$$

Consider now a specific example. Take G and the controller C as in the example of the gain variation. Let the perturbed plant be

$$G(s) + \delta G(s) = \frac{1}{s^2}\frac{5-s}{5+s}$$

so the phase perturbation is

$$\theta(\omega) = 2\left(\tan^{-1}\frac{\omega}{10} - \tan^{-1}\frac{\omega}{5}\right)$$

which is plotted in figure 3.11. The maximum phase shift of $-38.9°$ corresponds to about 6dB of gain variation. Figure 3.12 shows the nominal magnitude of T, the actual perturbed magnitude caused by the phase shift in G, and the perturbed magnitude predicted by the first order analysis:

$$|T^{approx}(j\omega)| = |T(j\omega)| \exp\left(-\theta(\omega)\Im\frac{1}{1 + G(j\omega)C(j\omega)}\right)$$

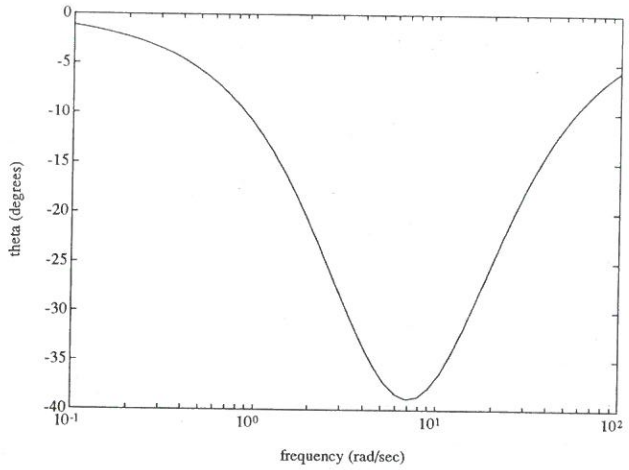


Figure 3.11: A specific phase shift $\theta(j\omega)$ in G.

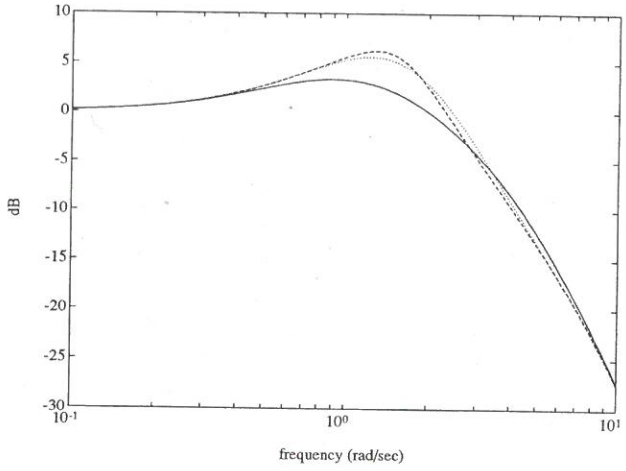


Figure 3.12: The magnitudes of T (full line), T^{pert} (dashed line) for a phase variation. The dotted line is the first order approximation $|T^{pert}|$.

Other SISO logarithmic sensitivities

The logarithmic sensitivity of the I/0 transfer function $T(s) = H_{y_3u_3}(s)$ was given by the sensitivity transfer function $S(s)$. In the general case however, the logarithmic sensitivity of a closed-loop transfer function need not be another closed-loop transfer function. There are however several other important closed-loop transfer functions that have logarithmic sensitivities that are closed-loop transfer functions. This is of course an important thing to remember when designing controllers.

Closed-loop H	$\delta \log H \;/\; \delta \log G$
$(1+GC)^{-1}$	$-(1+GC)^{-1}GC$
$C(1+GC)^{-1}$	$-(1+GC)^{-1}GC$
$G(1+GC)^{-1}$	$(1+GC)^{-1}$
$(I+GC)^{-1}GC$	$(1+GC)^{-1}$

3.2.6 Robustness specifications

Gain and phase margins for SISO systems

The stability degree σ [4] of a closed-loop system is required to be positive. On the other hand, if the loop transfer $L(s)$ itself is stable, the *relative stability* of the closed loop system can be expressed by three different measures:

- The *positive gain margin* (*pgm*) is the smallest increase in loop gain that results in an unstable closed loop system. A positive gain margin smaller than 0 dB means that the design yields an unstable closed loop system. In practice, a *pgm* of 2 dB is required in order to guarantee sufficient stability.
- The *negative gain margin* (*ngm*) is the smallest decrease in loop gain that results in an unstable closed loop system. A negative gain margin smaller than 0 dB means that the design yields an unstable closed loop system. In practice, a *ngm* of 2 dB is required in order to guarantee sufficient stability.
- The *phase margin* (*pm*) is the smallest decrease or increase in loop phase that results in an unstable closed loop system. A phase margin lower then zero degrees means that the design yields an unstable closed loop system. In practice, a *pm* of 45 degrees is required in order to guarantee sufficient stability.

MIMO systems

Recent years have been characterized by a wealth of results in robust control system design for systems with several inputs and outputs (MIMO), which has now become a major research topic,

[4] The stability degree of a transfer matrix is defined by $\max\{\Re[p] \mid p \text{ is a pole of } H\}$.

with important mathematical tools such as *the small gain theorem, structured singular value* etc.... It would however lead us to far to describe these evolutions in detail here and we have to refer to more advanced courses on control theory.

3.2.7 Minimum variance control

In Chapter 2, we have seen how linear models could be obtained in the form

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Ee_k \\ y_k &= Cx_k + Du_k + Fe_k \end{aligned}$$

where u_k and y_k are the input resp. output vectors of the system and e_k is a disturbance input. We have also shown in Chapter 2 how such a model could be obtained from input-output measurements via so-called system identification techniques and we have given as an example the identification of a glass production plant. The two outputs to be controlled in this plant are the tube diameter and the wall thickness, with as a control specification that these should be as constant as possible over time. Because of the disturbance inputs however, it is impossible to have an exact control precisely on the set points. Instead, the diameter and thickness will fluctuate randomly around the desired values. One could however try to obtain a controller that minimizes the variances of the deviations from the set point. This is exactly what a minimum variance control strategy will achieve. We will not give the full details here on how such a controller can be calculated. It suffices to say that it could consist of a special observer (namely a Kalman filter)(see also Chapter 5) and a static feedback on the estimated state of the system (see also Chapter 5). In Chapter 4, we describe another control strategy for this plant which combines a static decoupling feedforward with two PIID-controllers. Some results are shown in Figure 3.13.

3.3 Conclusion

In this Chapter, we have given a non-exhaustive treatment of control configurations and specifications. It was not our goal to really enumerate all possibilities, but just to give a general idea about possible specifications. Many more details can be found in any book on control theory.

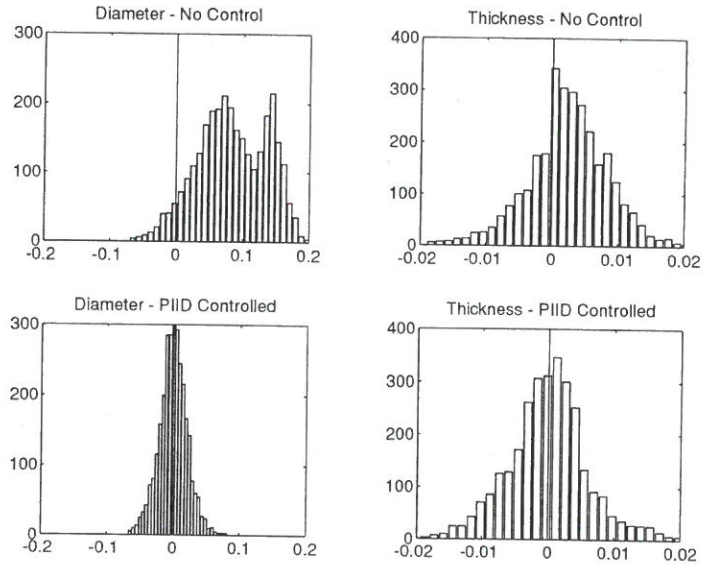


Figure 3.13: *Quality improvement. The top 2 figures show a histogram of the measured diameter and thickness. The reference setpoint for production is at zero. The bottom two figures show the histograms of the controlled system. The enhancement in variance of both diameter and thickness and the fact that they are now centered symmetrically around the desired setpoint is clearly visible.*

Chapter 4

PID controllers

In this Chapter, we consider the 1 DOF control strategy of Figure 3.5 of Chapter 3, for SISO systems, i.e. systems with one input and one output. More specifically, we will concentrate on a particular, widely used controller, which has only 3 to 4 parameters that can be tuned to achieve a certain specified performance, namely a PID-controller (Proportional, Integrative, Derivative controller). In Section 4.1., we discuss the basic PID control configuration. Several tuning rules are treated in Section 4.2. Model-based PID-tuning is the subject of Section 4.3. In Section 4.4. we briefly discuss implementational aspects while we conclude this Chapter with our glass tube production process (again).

4.1 PID-control laws

In industrial environments, the PID-controller is still the most favourite feedback controller: a vast amount of practical experience learns that the PID-controller has a structure which is sufficiently versatile to cope in an excellent way with a multitude of process control problems. Other advantages of the PID-law are the low number of design parameters and the fact that the controller parameters can easily be related to performance measures.
The PID-controller is a linear controller. It transforms the error signal $e(t)$ into the control signal $u(t)$. Since it is a linear controller, it can be studied in the time domain and in the frequency domain.

In the time domain, the PID control signal looks like:

$$u(t) = K_p(e(t) + \frac{1}{T_i}\int e(t) + T_d\frac{d}{dt}e(t))$$

After Laplace transformation, this becomes:

$$U(s) = K_p(1 + \frac{1}{sT_i} + sT_d)E(s)$$

Basically, the controller consists of three terms, each with a typical function. The first term is proportional to the error signal, the second term is proportional to its integral and the third term is proportional to its derivative.
However, to be practical, this algorithm needs to be modified, especially for the implementation of the derivative action, since pure derivation is a practical nonsense, as it has unbounded amplification at high frequencies, which is physically impossible and even unwanted. Indeed, the slightest

high frequency noise, superposed on the error signal, would be enormously amplified. Furthermore, the controller output at a step in the desired output would change annoyingly abrupt (in theory, an impulse is required).
'Smoothing' can be introduced by processing the error signal with a low-pass filter, before taking the derivative. In practice, both operations are implemented in one filter. The transfer function of the PID-controller is then given by:

$$C(s) = K_p(1 + \frac{1}{sT_i} + \frac{sT_d}{st_c + 1})$$

where t_c must be small, compared to T_d, to have a derivative action. When t_c is chosen too big, the third term becomes a proportional one, even for low frequencies. A possible PID-scheme is given in figure 4.1.

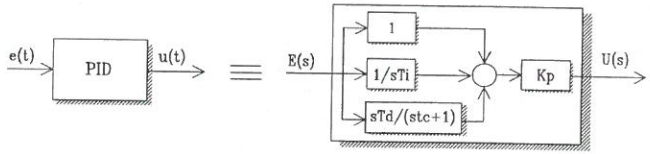


Figure 4.1: *The PID-control law.*

4.2 PID Tuning

Four parameters can be used to tune a PID.

1. *The gain* K_p
2. *The integration time or reset time* T_i
3. *The derivation time or lead time* T_d
4. *The time constant* t_c *of the low-pass filter*

Up to now, the experimental rules-of-thumb derived by Ziegler and Nichols more than 50 years ago, are still widely used for tuning industrial PID-controllers, because it is not necessary to have a complete, dynamic model of the plant. Only a few, fairly simple experiments are required to find suitable controller parameters. Since these rules are empirical rules, it should be recognized that problems can occur. However, in many cases, these rules give rise to a control system with very acceptable performance trade-offs. There are basically two cases:
Transient response method
To apply this method, it is required that the plant's step response has a similar shape as the one given in figure 4.2. From this step response, the time delay or lag L and the steepest slope or reaction rate $R = \frac{K}{\tau}$ can easily be derived (after a delay of L seconds, the output increases K units in τ seconds). Once these parameters are computed, the controller parameters are selected from the following table.

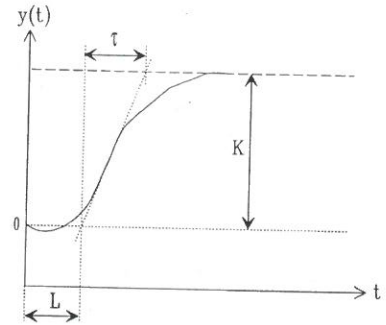


Figure 4.2: *The time delay L and the reaction rate $R = \frac{K}{\tau}$ are easily calculated from a step response.*

	K_p	T_i	T_d
P-control	$\frac{1}{RL}$		
PI-control	$\frac{0.9}{RL}$	$3.3L$	
PID-control	$\frac{1.2}{RL}$	$2L$	$\frac{L}{2}$

However, if the plant's step response differs drastically from that in figure 4.2, this method is not applicable anymore, since L and R can't be measured/computed anymore.

Stability limit method

First, the plant is controlled using proportional control only ($T_i = \infty, T_d = 0$). Once a stable design is found (start with a very small value for the gain K_p), K_p must be increased slowly until a permanent oscillation occurs. At this point, the gain K_p and the oscillation period P_o are recorded (figure 4.3). If they are denoted by K_o and P_o, the controller parameters are determined from the following table.

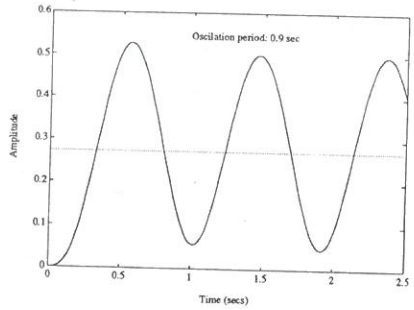


Figure 4.3: *The oscillation period P_o for the stability limit method. The ultimate controller gain K_p, responsible for this permanent oscillation, is denoted as K_o.*

	K_p	T_i	T_d
P-control	$0.5K_o$		
PI-control	$0.45K_o$	$\frac{P_o}{1.2}$	
PID-control	$0.6K_o$	$\frac{P_o}{2}$	$\frac{P_o}{8}$

Design rule:

1. *Do some simple experiments and determine the characteristics that are required for the method you want to apply.*
2. *Use the apropriate table to calculate the controller parameters.*

4.3 Model-based PID-tuning

The performance of modelling and identification techniques increased enormously in the last decades. Therefore, it is very reasonable to assume that a suitable linear model of the plant is available. In the next subsections, model-based tuning techniques and design rules are presented.

4.3.1 Tuning for stabilization

One important reason why a feedback loop is used in many control problems, is the relative ease with which stabilization can be obtained. To design a stabilizing PID-controller, several techniques

are available. Two different approaches can be followed: one can design a controller by asking for absolute stability (all closed loop poles are explicitly required to be in the Left Half Plane (LHP)) or one can design a controller by asking for relative stability (sufficiently high phase and gain margins are required). In both cases, the resulting controller will stabilize the system.

Absolute stability

If only one parameter needs to be chosen (K_p or K_i or K_d), the root-locus method can be used to analyze the stability of the closed loop system in terms of the free parameter. The free parameter of the controller $C(s) = K\frac{N_C(s)}{D_C(s)}$ is denoted by K (K_p for P-control, $\frac{K_p}{T_i}$ for I-control and K_pT_d for D-control). If the plant's model is given by $P(s) = \frac{N_P(s)}{D_P(s)}$, the loop transfer function $L(s)$ can be rewritten as:

$$L(s) = KL_f(s) = K\frac{N_P(s)N_C(s)}{D_P(s)D_C(s)}$$

The poles of the closed loop system can be calculated in terms of the free parameter K, by solving:

$$1 + KL_f(s) = 0$$

for various values of K. The values of K for which all roots are in the LHP correspond to a stabilizing controller.
Notice that for K equal to zero, the roots correspond to the poles of the loop transfer function, while for K going to ∞, they correspond to the zeros of the loop transfer function. For non-minimum phase plants (some zeros in the RHP), the last observation shows explicitly that the free parameter K is limited by the requirement of closed loop stability.

Design rule:

1. *Choose an initial value of K and calculate the closed loop poles.*
2. *If they are all in the RHP, decrease K till they are in the LHP. Once they are all in the LHP, the corresponding K-value stabilizes the plant. When they are all 'deep' in the LHP, K may be increased without destabilizing the system.*

Although it is possible to repeat this method in situations where more then one parameter must be tuned, other techniques are preferred, since root-loci are not practical anymore.
In general, explicit stabilization conditions in terms of the controller parameters are derived. Once these conditions are given, an (in)equality solver can be used to determine a stabilizing design (notice that multiple solutions are possible).
If the loop transfer function is given by $L(s) = \frac{N_L(s)}{D_L(s)}$, the closed loop denominator (the critical polynomial) becomes:

$$l(s) = D_L(s) + N_L(s)$$

Necessary and sufficient conditions for $l(s)$ to have only roots in the LHP may be obtained by applying the Routh-Hurwitz criterion to the polynomial $l(s)$.

Design rule:

1. *Calculate the loop transfer function $L(s) = \frac{N_L(s)}{D_L(s)}$ and construct the polynomial $l(s) = D_L(s) + N_L(s)$.*
2. *Apply the Routh Hurwitz criterium or the coefficient rule.*
3. *Solve the resulting set of inequalities (in the design parameters).*

Relative stability

As indicated before, a sufficiently high gain margin and/or phase margin guarantee closed loop stability in the case of a stable loop transfer. In that case, (frequency domain) loop-shaping techniques can be used to calculate a stabilizing controller.

Design rule (for PI-control):

1. *Find the frequency where the original system has a phase shift of $-(180-pm-pl)$. Choose this frequency as the new cross-over frquency ω_{co}.*
2. *Select T_i such that the contribution of the PI-controller to the phase shift at ω_{co} is equal to pl. For PI-control, this results in:*

   $$T_i\omega_{co} = \tan(90 - pl)$$

3. *Calculate K_p such that the gain at $\omega = \omega_{co}$ becomes 1:*

   $$L(j\omega_{co}) = 1$$

This method guarantees a phase margin pm. An additional phase lag of pl at the new cross-over frequency ω_{co} is allowed. For the other PID-like controllers, similar procedures can easily be derived.

4.3.2 Performance-based tuning

Once a stabilizing design is found, it is human's nature to ask for other stabilizing designs, of higher quality then the previous design. To express a system's quality, different performance measures may be considered. Two main categories of specifications can be distinguished: specifications in the frequency domain and specifications in the time domain.
The influence of the PID-controller parameters on the closed loop performance measures will be discussed now.

P-control

In case of proportional control, the controller transfer function is just a gain:

$$C(s) = K_p$$

and the control signal is proportional to the control error. From the Bode plot of a proportional controller (figure 4.4), it follows that the loop gain will be multiplied by K_p, while the phase shift is not influenced at all. The purpose of proportional control is to increase the loop gain, in order to obtain a higher accuracy (DC-gain) and a large cross-over frequency. However, due to stability requirements, K_p is limited. For plants with a tendency to oscillate, one can easily verify that the rise time T_r decreases, but that the settling time T_s and the overshoot os often increase.

I-control

In case of integral control, the controller transfer function is just an integrator:

$$C(s) = \frac{K_p}{sT_i}$$

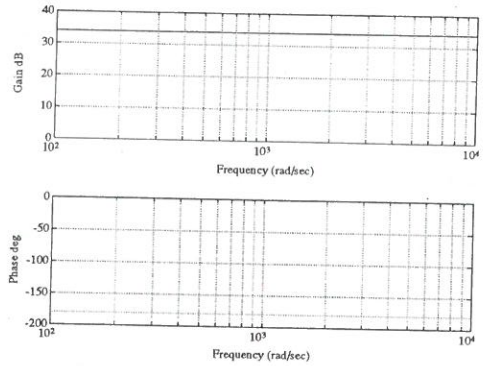


Figure 4.4: *The Bode plot of a P-controller* ($K_p = 50$).

and the control signal is proportional to the integral of the control error. From the Bode plot of an integral controller (figure 4.5), it follows that the loop gain will be multiplied by $\frac{K_p}{T_i}$ at $\omega = 1$ and that an additional phase shift of -90 degrees will be introduced. The purpose of integral control is

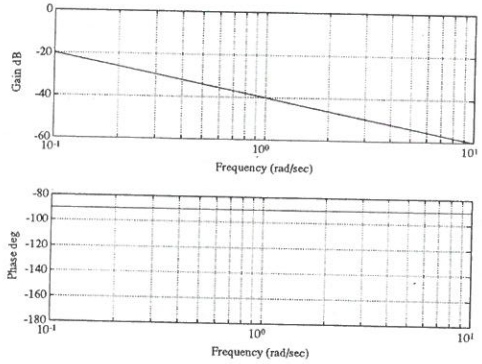


Figure 4.5: *The Bode plot of a I-controller* ($K_p = 1, T_i = 100$).

to make the loop gain at *DC* equal to ∞, in order to obtain a zero steady state error. However, the additional phase shift forces the designer to decrease the bandwidth, to ensure relative stability (so $\frac{K_p}{T_i}$ is limited).

D-control

In case of proportional control, the controller transfer function is just a derivative term:

$$C(s) = sK_pT_d$$

and the control signal is proportional to the derivative of the control error. From the Bode plot of a derivative controller (figure 4.6), it follows that the loop gain will be multiplied by K_pT_d at $\omega = 1$ and that an additional phase shift of $+90$ degrees will be introduced. The purpose of derivative

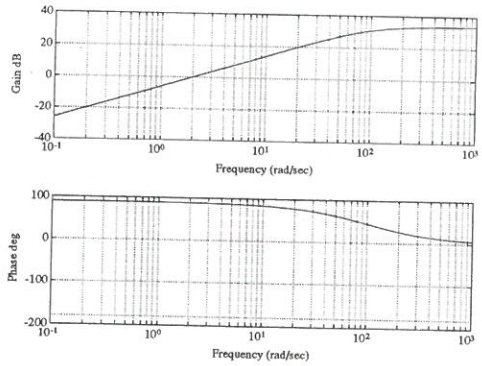


Figure 4.6: *The Bode plot of a D-controller* ($K_p = 1, T_d = 0.5, t_c = 0.01$). *The derivative action is bounded for frequencies* $\omega \geq \frac{1}{t_c} = 100$.

control is to increase the response speed (decrease T_r). However, due to the amplification at high frequencies (noise sensitivity), the derivative action must be limited. In theory, it would be sufficient to place an extra low-pass filter $\frac{1}{st_c+1}$ before the differentiator, to 'remove' the high frequency components in the error signal. In practice, the low-pass filter and the derivative action are implemented by a single filter. In that case, the controller transfer function is:

$$C(s) = \frac{sK_pT_d}{st_c + 1}$$

PI-control

In PI-control, the advantages of P- and I-control are combined. P-action is added for bandwidth and speed, the I-action is added to guarantee a zero steady state error. From the Bode plot of a PI-controller (figure 4.7), it follows that a high loop gain is introduced for lower frequencies, while the additional phase shift vanishes for higher frequencies. The positive and the negative effects of PI-control become more outspoken by making the integration time T_i smaller: the high gain for low frequencies increases, but the phase margin reduces.

PD-control

In PD-control, the advantages of P- and D-control are combined. P-action is added for bandwidth and speed, the D-action to further decrease the rise time (while P is limited by stability requirements). From the Bode plot of a PD-controller (figure 4.8), it follows that a high phase lead and a high gain are introduced for higher frequencies: the speed of response increases, together with the sensitivity to noise. The positive and the negative effects of PD-control become more outspoken

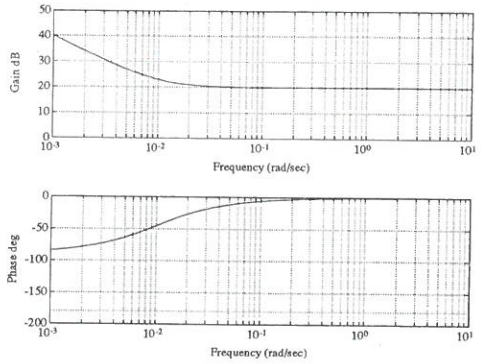


Figure 4.7: *The Bode plot of a PI-controller* ($K_p = 10, T_i = 100$). *The integral action is higher for frequencies* $\omega \leq \frac{1}{T_i} = 0.01$.

by making the integration time T_d larger: the phase lead for high frequencies increases, but the noise sensitivity increases also!

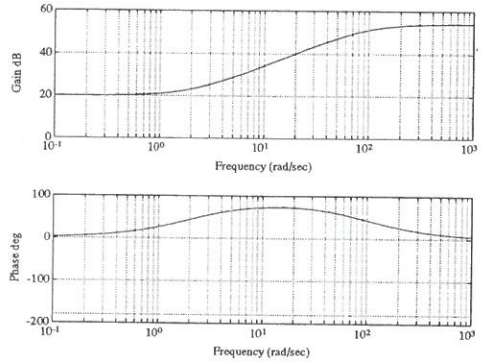


Figure 4.8: *The Bode plot of a PD-controller* ($K_p = 10, T_d = 0.5, t_c = 0.01$). *The derivative action is bounded for frequencies* $\omega \leq \frac{1}{T_d} = 2$ *or* $\omega \geq \frac{1}{t_c} = 100$.

PID-control

An attempt to combine all the previous advantages, while accepting the corresponding disadvantages, is the PID-controller, which is described in a previous section. It is easy to understand that the relative sizes of the different controller parameters determine which effects become more/less important. By analyzing the Bode plot of the PID-controller (figure 4.9), it is possible to do some

predictions of the closed loop behavior.

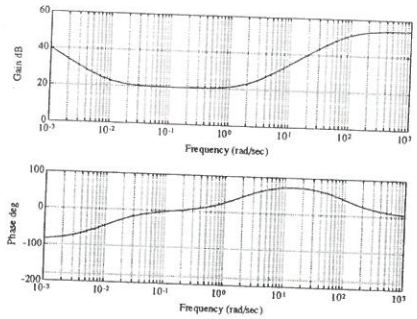


Figure 4.9: *The Bode plot of a PID-controller* ($K_p = 10, T_i = 100, T_d = 0.5, t_c = 0.01$).

4.3.3 Classical PID tunning with Xmath

PID-tuner is a graphical tool for the interactive design of PID-like controllers, using the functions of *Xmath*[1] and the facilities of a graphical user interface *GUI*. The tool itself consists of three different windows, each with its own functionalities. Detailed help messages on these windows and features are implemented.

- *The main window* is used by the designer who 'turns knobs' in the parameter control panel and 'reads the meters' in the performance measures display area (figure 4.10). It appears automaticaly when the tool is started.
- *The performance plot window* visualizes the characteristics of the controlled system. It can be called by pressing the *Plot* button on any Windows menu (figure 4.11).
- *The ranges window* is used to change the ranges of sliders, bargraphs and plots. It can be called by pushing the *Ranges* button on any *Windows menu* (figure 4.12).

In the main window of the tool, a control panel is used to change the type of controller and its design parameters. Pushbuttons can be used to (de)activate the different controller terms (choose the control law), while sliders and edit-boxes can be used to set the values of the control parameters.
Two possibilities are available to analyze the performance of the actual design:

- Bargraphs and read-boxes (in the main window) indicate the value of the most important quality measures, if the closed loop system is stable. A warning message occurs if the closed loop system is unstable.
- In the performance plot window, the Bode plot and the Nyquist plot of the loop transfer $L(s)$ are shown, together with the closed loop step response.

[1] Xmath is a trademark of Integrated Systems Inc., Santa Clara, California, USA.

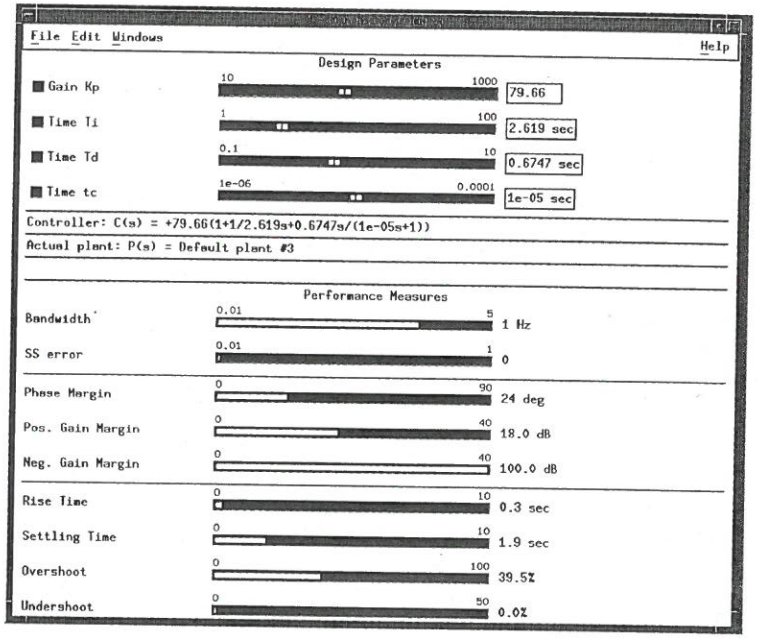


Figure 4.10: *The main window consists of a control panel that is used to change the design parameters and tune the controller, a display that gives the current controller transfer function (the "current design"), a display with the name of the plant, a display area that is used for warning messages and a display area that is used to report various performance measures for the current controller.*

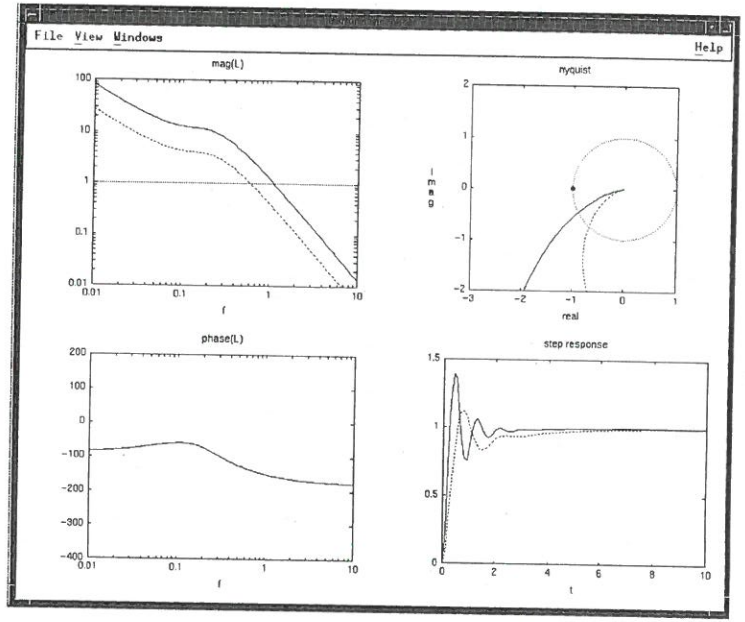


Figure 4.11: *In the performance plot window, the Bode plot and the Nyquist plot of the loop transfer $L(s)$ are shown, together with the closed loop stepresponse, for the actual design (full line) and the previous design (dotted line).*

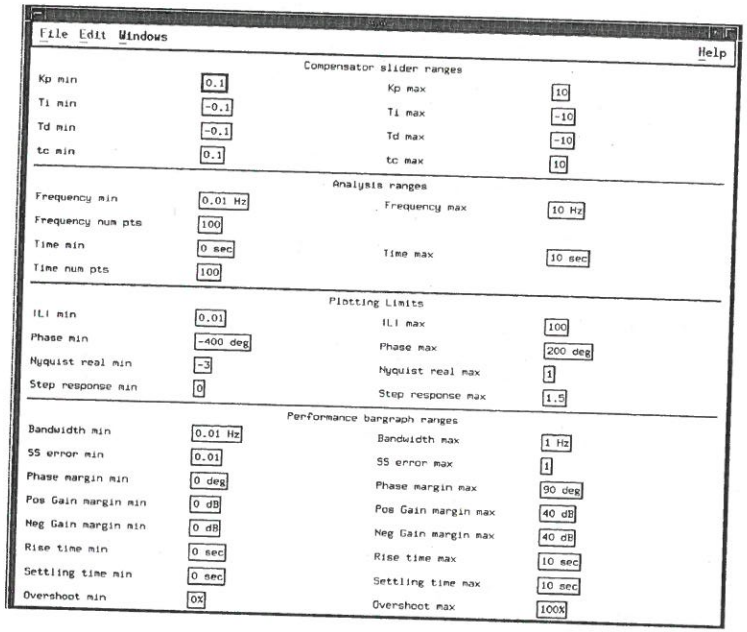


Figure 4.12: *The range window is used to modify the ranges of the sliders, bargraphs and plots of the Main Window and the Performance Plot Window.*

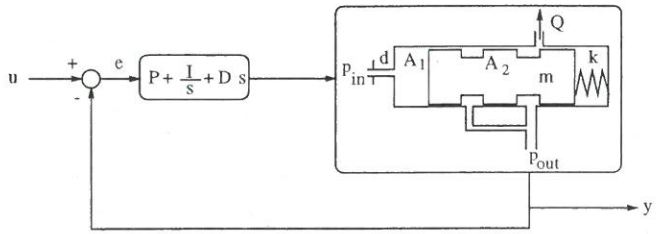


Figure 4.13: *Control scheme for an actuator in a continuous variable transmission: The actuator itself is however not completely specified yet. The mass of the damper, the spring constants, the surfaces and the diameter can still be specified as to optimize the design. But also the three parameters of the controller can be used to even further optimize the performance. All of this can be done within the framework of multi-objective optimization.*

4.3.4 PID-Tuning with Optimization

With the recent availibility of sophisticated software packages, such as Matlab and Xmath and also the increasing degree of sophistication of numerical optimization techniques, it becomes feasible to design and optimize *virtual* engineering systems before they are ever built. Without going into detail, we just show here (without much explanation for which we refer to [26]) the combined design of an actuator system of the continuous variable transmission (see also Chapter 2) and that of a PID controller. For the actuator system, still several parameters need to be optimized so that some specifications are satisfied. For the PID controller, the same remark holds true. In this sense, one can design both systems *simultaneously* (while in the more classical system design methodologies, first the actuator would be designed and actually implemented, while the controller would only be invoked afterwards, typically if all other remedies are failing).
The problem we have discussed here can be formalized using so-called *multi-objective optimization* techniques and certainly has great potentials to design *mechatronic* systems.

	Valve design	PID design
Unknowns	Mass m	Gain K_p
	Spring constant k	Reset time T_i
	Surfaces A_1 and A_2	Lead time T_d
	Diameter d	Time constant τ_c
Specifications	Stable	Stabilizing
	Linearity range	Tracking
	Speed of response	Bandwidth
	Robustness	Sensitivity
	Size limits	Limited control signal

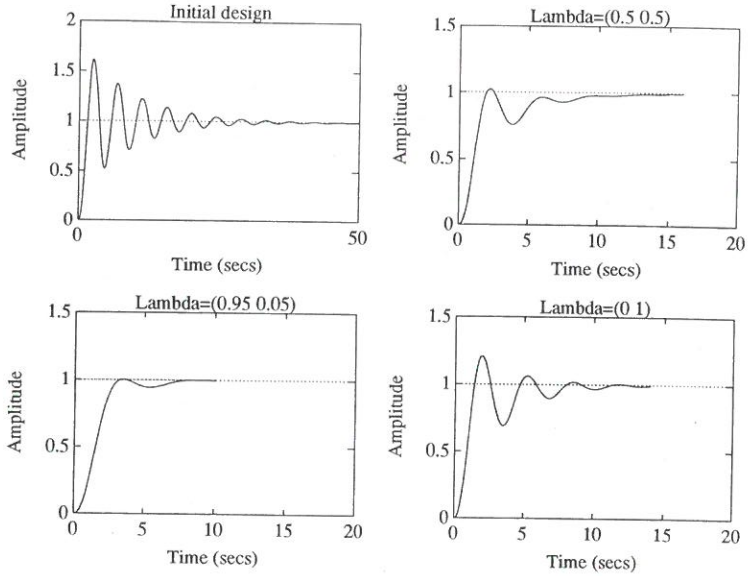


Figure 4.14: *Results of optimal design of the actuator and its controller. Shown is the step response for several values of the parameters λ_1 and λ_2, which express the relative importance of tuning the actuator parameters versus the control parameters in the optimization of the overall system.*

4.4 Implementation

Having discussed in some detail possible design methodologies, in this section, we briefly treat some omplementational issues.

4.4.1 Analog or digital?

Once a suitable PID-controller is found, it has to be implemented in some way. Classical PID-controllers were implemented in hardware, using all kinds of analog technology (electronical, mechanical, pneumatical, hydraulical). Since the controller parameters were determined by geometrical and physical properties, they were mostly fixed for a given controller. However, due to the enormous success of the digital computer because of its flexibility, the actual controllers are mostly implemented in software, where parameters can be changed easily.
The ideal continuous time PID-controller $C(s)$ is given by:

$$C(s) = K_p(1 + \frac{1}{sT_i} + sT_d)$$

$C(s)$ relates the controller output $U(s)$ to the error signal $E(s)$:

$$U(s) = C(s)E(s)$$

If the sampling interval T is small enough, the continuous time behavior of the controlled process can be approximated rather well by stepwise constant signal approximations. In that case, the discrete PID-controller will be implemented by proper numerical approximations of the integration and differentiation operations.
The proportional part is readily implemented as:

$$u_p(k) = K_p e(k)$$

where k refers to the sampling instant kT. In the z-domain, this gives:

$$U_p(z) = K_p E(z)$$

In the discrete time domain, the integral action can be approximated by the sum of all previous errors:

$$u_i(k) = u_i(k-1) + \frac{K_p T}{T_i} e(k)$$

In the z-domain, this becomes:

$$U_i(z) = \frac{K_p T z}{T_i(z-1)} E(z)$$

Finally, differentiation in the discrete domain can be approximated as the first difference:

$$u_d(k) = K_p T_d \frac{(e(k) - e(k-1))}{T}$$

and after taking the z-transform:

$$U_d(z) = K_p T_d \frac{(z-1)}{Tz}$$

Combining all the above results yields the discrete time PID-controller:

$$U_p(z) = K_p(1 + \frac{Tz}{T_i(z-1)} + \frac{T_d(z-1)}{Tz}) E(z)$$

4.4.2 Anti-windup

The actuator is the device through which the control input is applied to the process. Since these components have a limited range (e.g. motor driving amplifiers have a minimal and maximal voltage or current output), they introduce a saturating non-linearity into the process (figure 4.15).

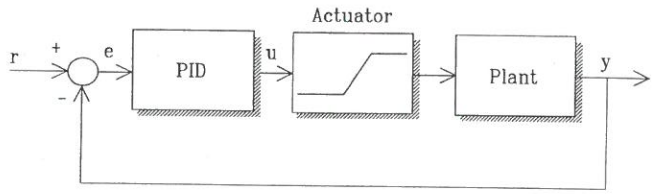


Figure 4.15: *Integrator wind up, due to actuator saturation.*

As an example, consider the PI-control of a certain process. Suppose a setpoint change brings the actuator into saturation. In that case, the error signal will decrease slower then without

saturation, so the integrating output term will be higher: as a consequence, the actuator will be saturated during a longer time, since it will not leave the saturation, unless the error has changed sign for a sufficiently long time in order to bring the integral output to a low value. Further, during saturation, the feedback is actually not working: it can be replaced by a fixed-level input, which gives rise to undesirable overshoots for processes with a low stability degree. The previous effect is called *integrator wind up*.

This phenomenon can be avoided by stopping the integration when the actuator saturates. A clever way to do so is to force the output of the controller to stay at the saturation limit u_l as long as the actuator is saturated. This can be done by implementing an extra feedback loop which forces the difference between actuator input and actuator output to be zero at all times. The integral part of the controller can be used for this purpose (figure 4.16).

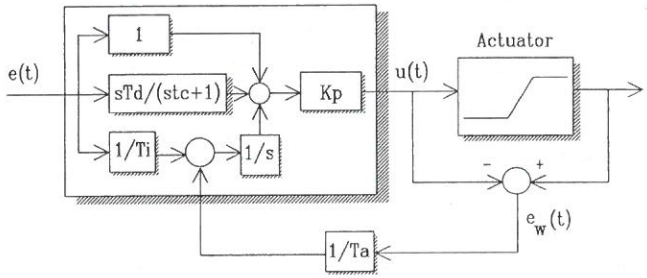


Figure 4.16: *Anti-windup by introducing an extra feedback loop: as long as there is no saturation, $e_w = 0$ and the PID-controller works in the normal way. As soon as $e_w \neq 0$, the extra loop tries to bring e_w back to zero, to a rate which is inversely proportional to T_a. This may drastically reduce the saturation time and improve the step response! Be careful to take T_a not too small, if there is an active D-action ($T_d \neq 0$), as spurious peaks originating from it may drive the actuator into saturation for a short time: too quick resetting during this short peak results in an unnecessary small value for the integrator output after the peak has vanished.*

4.4.3 Incremental PID

For analog PID-control, the output of the regulator is always generated in a direct way. However, digital control allows to bring the algorithms in an incremental form, where at each sampling instance, the regulator output increment is calculated. This is particularly useful when the actuator is of the integrating type (e.g. a stepping motor). The number of pulses to be fed to the stepping motor is then proportional to the regulator increments (the actuator itself reconstructs the required control signal by integrating the increments!). The incremental algoritm is also called the *velocity form* of the PID-algorithm, as opposed to the *position form* of the non-incremental case.

Advantages of this incremental form are:

- Shorter wordlength calculations are needed, only the final summation stage needs higher precision.

- Except for integrating type actuators, the terminal stage is an integrator, which is useful for anti-windup compensation.

4.4.4 Leadlag, a special class of PID-controllers

An alternative for the parallel PID-implementation (summing the three terms) is the series implementation of a PI- and a PD-action. In the Laplace domain, this implementation looks like:

$$C(s) = K_p'(1 + \frac{1}{sT_i'})(1 + sT_d')$$

Its parallel equivalent can easily be calculated, using the following relations:

$$K_p = K_p' \frac{T_i' + T_d'}{T_i'} \quad T_i = T_i' + T_d' \quad T_d = \frac{T_i' T_d'}{T_i' + T_d'}$$

However, it can be proven that the series equivalent of a given parallel form does not always exist. Consider the controller's transfer function:

$$C(s) = K_p(1 + \frac{1}{sT_i} + sT_d) = \frac{K_p}{sT_i}(s^2 T_d T_i + sT_i + 1)$$

To convert this to the series form, it is required that the polynomial $s^2 T_d T_i + sT_i + 1$ has two real roots, so its discriminant must be non-negative:

$$T_i^2 - 4T_d T_i \geq 0$$

From this, it follows that the series implementation of a parallel form only exists if:

$$T_i \geq 4T_d$$

Although the parallel form is more general, the series form is sometimes preferred, while the implementation and the avoidance of windup are very easy.

4.5 Example: The glas tube manufacturing plant

In order to conclude this section, we show how even a quite complicated industrial process such as the glass tube production process we have been discussing before, which is certainly multivariable, can be controlled using PID-control strategies. Having obtained the state space model with the techniques of Chapter 3, we now design a controller. The specifications are such that the variance on the outputs has to be as small (high product quality). This implies that the the disturbances have to be rejected as much as possible. Typical disturbances consist of a mixture of noise, steps and trends. The noise is generated by the short term variations and the non-linearities of the process and also by the non-homogeneous composition of the quartz sand that is melted. The step and especially the ramp disturbances are generated by the change over from one barrel of sand to another : the step change in the composition of the sand is "integrated" in the oven and is transmitted to the output as a linear trend. As a control configuration, we choose a double PIID controller as in Figure 4.17, combined with a static feedforward, which decouples the system for low frequencies. Efficient suppression of steps and ramps is guaranteed by the double integrating action in the PIID controller. Another specification is the "change over time" between different setpoints. This time interval should be as small as possible, to avoid loss of production. This

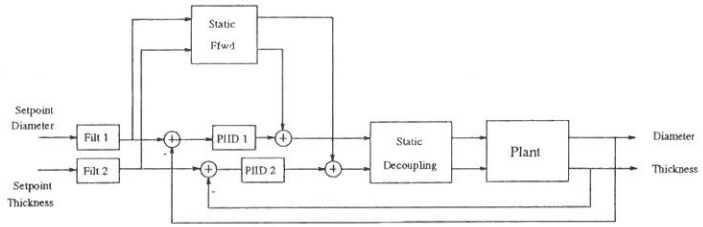


Figure 4.17: *The control scheme used for the PIID control of the glass tube manufacturing plant. The controller consists of two feedforward filters, a static feedforward, a static decoupling and two PIID controllers to control the decoupled loops. The PIID parameters are tuned using a multi objective optimization algorithm.*

implies that the settling time of the "reference to output behavior" has to be as small as possible. Furthermore, the overshoot and oscillations should be restricted.
Finally, the controller has to be robust towards changes in the process. The design of the PIID should also take into account that the input signals to the actuators have to be smooth (to avoid damage of the actuators) and within the saturation limits.
A PIID controller was chosen over a more complicated state feedback controller, because of its simple structure and implementation. The parameters of the controller were tuned by translation of the above specifications in a multi-objective problem, which was solved using optimization software.
The results were shown in Chapter 3 in Figure 3.13.

4.6 Conclusions

PID-controllers are still widely used in industry. In this Chapter, we have discussed besides the classical Ziegler-Nichols based tuning rules, some more recent insights, all of which are *model-based*. In combination with powerful and user-friendly Graphical User Interfaces and multi-objective optimization software, and decoupling strategies which turn a MIMO system into a set of SISO systems, PID-control strategies still have great potentials in an industrial environment.

Chapter 5

Model-based control

In this section, we present some *model-based* control strategies. For linear systems, the most important ones are always based on the concept of pole-placement. Typically, a pole-placement model-based controller will consist of two major parts: First, there is an observer, which itself is a dynamical system that uses the model of the system and its inputs and outputs to estimate the *state* of that system. Next, there is a *static feedback* which translates the observed estimated state into a control input to be applied to the real system. It turns out that many optimal control schemes (such as the Linear Quadratic Gaussian Regulator (minimum variance), and more recent robust control strategies such as H_∞) can be interpreted as a combination of an observer and a static feedback from the estimated state to the control input.
The motivation to study sophisticated multivariable control strategies is not necessarily to implement these solutions, but to compare existing control implementations to what could be achievable if the advanced theories are used. Moreover, using these advanced techniques, it is also possible to explore the so-called *limits-of-performance* of the control system (actually of *any* control system).

In Section 5.1, we have a closer look at pole-placement strategies, which basically consist of changing the dynamics of a given system. In Section 5.2., we discuss how observes can be designed. The combination of an observer followed by a static feedback is discussed in Section 5.3. In Section 5.4., we show that a controller which is optimal with respect to a certain quadratic object function, reduces to a static state feedback. In Chapter 5.5. we give an example of a neural network controller as an illustration of nonlinear control systems. Finally, conclusions are given in Section 5.6.

5.1 Pole placement - state feedback

In this section we discuss the poleplacement problem in some detail. Poleplacement works for both discrete and continuous time systems. While we derive the solution for discrete time systems, we will also give a small example for a continuous time system, which also illustrates some important points.

5.1.1 The pole placement problem

The pole placement problem is the following:
Given a linear state space model

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

How can we design a static state feedback matrix F in

$$u_k = -Fx_k + v_k$$

such that $A - BF = \tilde{A}$, where $\tilde{A}$ is a given matrix and v_k is a new additional (free) input.

Why would we want to do something like that? First observe that the state equations become

$$\begin{aligned} x_{k+1} &= (A - BF)x_k + Bv_k = \tilde{A}x_k + Bv_k \\ y_k &= (C - DF)x_k + Dv_k \end{aligned}$$

We see that the system $[A, B, C, D]$ has been modified into $[A - BF, B, C - DF, D]$ and that we have replaced the input u_k by an input v_k. We might try to modify the dynamic behavior of the system by changing its A matrix if for one reason or another we are not pleased with the actual A matrix. For instance, we could require that our system should have a certain *marginal stability regio*, a specified *damping ratio* or a certain stability degree [1]. Such requirements can be translated into certain specifications on the roots of the characteristic polynomial of the system which are the eigenvalues of A and $\tilde{A}$.

5.1.2 Solution of the pole placement problem

Assume that the closed-loop system poles are the eigenvalues of a (not necessarily but most often diagonal) matrix $\tilde{\Lambda}$ (For instance, if we want $\tilde{A} = A - BF$, we can take the eigenvalue decomposition of $\tilde{A}$ as $\tilde{A} = X\tilde{\Lambda}X^{-1}$). Hence, there must exist a similarity transformation X such that

$$X^{-1}(A - BF)X = \tilde{\Lambda}$$

which can be rewritten as

$$AX - X\tilde{\Lambda} = BFX$$

We can split up this equation by introducing an auxiliary matrix G as

$$\begin{aligned} AX - X\tilde{\Lambda} &= BG \\ FX &= G \end{aligned}$$

If G would be known, the first equation would be a linear equation in the unknown matrix X. Equations with this structure are called *Sylvester equations*. If G would be known, then we find immediately

$$F = GX^{-1}$$

Hence, we have found an algorithm to assign the controllable modes of (A, B) to any position:

[1] defined by $max\{\Re[p] \mid p \text{ is a pole of } H\}$.

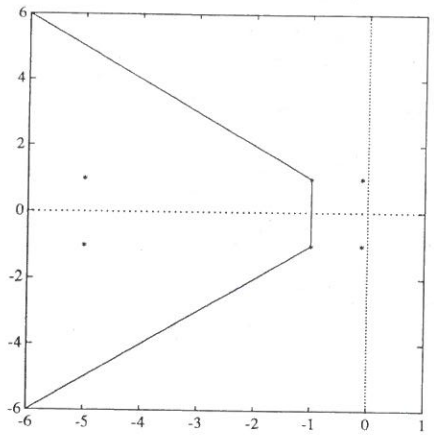


Figure 5.1: The stability degree of this system is 1

1. Pick an arbitrary matrix G.

2. Solve the Sylvester equation for X.

3. Obtain the static feedback gain $F = GX^{-1}$.

The following properties can be proved:

- The Sylvester equation has precisely one solution if A and $\tilde{\Lambda}$ have no common eigenvalues.
- It has been shown in [5] that this is the case for almost all matrices G.
- Observe that X are the eigenvectors of the closed-loop system matrix. Hence this method could also be studied in order to *shape* the closed-loop modes.

5.1.3 A numerical example

Let us consider the following system

$$A = \begin{pmatrix} -1 & 2 & 0 & 3 \\ 4 & -2 & -1 & 2 \\ 3 & 5 & -6 & 1 \\ 2 & -1 & 1 & 2 \end{pmatrix} \qquad B = \begin{pmatrix} -1 & 2 \\ -1 & 3 \\ 2 & -1 \\ 4 & 2 \end{pmatrix}$$

The poles of A are $3.8937, -4.8451 \pm 1.1310j$ and -1.2035, hence the system is clearly unstable. Assume we want to *stabilize* the system by reallocating the poles to those of the matrix

$$\tilde{\Lambda} = \begin{pmatrix} -1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -2 \end{pmatrix}$$

As a random choice for G we take

$$G = \begin{pmatrix} 0 & 2 & -1 & 2 \\ 1 & 2 & 4 & 1 \end{pmatrix}$$

The solution of the Sylvester equation is:

$$X = \begin{pmatrix} 4.8665 & 3.7535 & -38.0000 & -6.9091 \\ 4.6179 & 3.1895 & -43.1250 & -7.7273 \\ 7.7115 & 3.5318 & -58.3750 & -13.7273 \\ -3.6631 & 0.1625 & 31.7500 & 7.4545 \end{pmatrix}$$

This results in the feedback matrix

$$F = \begin{pmatrix} 1.0093 & 0.0794 & -0.5875 & 0.2042 \\ 1.9521 & -1.6137 & -0.0585 & 0.1629 \end{pmatrix}$$

It can now be verified that

$$A - BF = X\tilde{\Lambda}X^{-1}$$

5.1.4 Continuous time example

Consider the simple system

$$\dot{x} = x - u$$

with static state feedback $u = -fx$, $f \in \mathbb{R}$ so that

$$\dot{x} = (1 + f)x$$

We can make the eigenvalue $1 + f$ as negative as we want. As a consequence, also the input will become large. Let $x(0) = 1$ so that $u(t) = -fe^{(1+f)t}$. As $f \to -\infty$, the input u approaches a Dirac impulse. Observe that

$$\int_0^\infty u(t)dt = \frac{f}{1+f}$$

so that for $f \to -\infty$, $\int_0^\infty u dt \to 1$. Clearly, u is 'bunching up' near $t = 0$ for $-f$ large.
A conclusion is that in general large changes will require large gain matrices F. In some way this corresponds to our intuition that drastic changes to a physical system need a lot of effort! However, such large loop gains might give problems in a practical implementation. As a matter of fact, there is an engineering trade-off between *speed* on the one hand and the *size* of the input on the other hand.

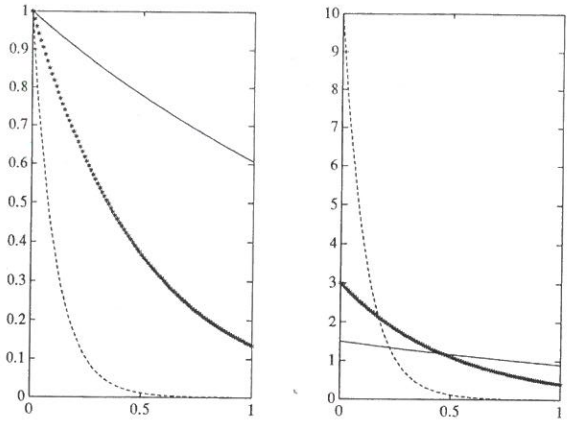


Figure 5.2: a/ Faster control of the states as $f \to \infty$. b/ The input approaches a Dirac impulse as the gain $f \to \infty$. (full line $f = -1.5$, stars $f = -3$, dashed $f = -10$.

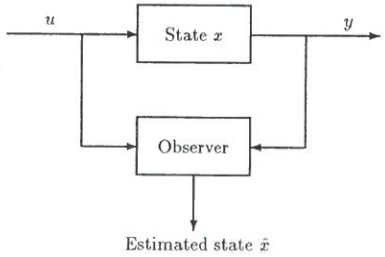


Figure 5.3: An observer is a dynamical system that uses the model of the system and inputs and outputs to estimate the state of the system.

5.2 How do we find the state: Observers

In our derivations so far, we have always assumed that the state x of our system was available. Usually this is not the case. Therefore we will design a dynamical system (which is called a state observer) that computes an estimate of the state. An additional feedback term in this so-called *observer* will make sure that the estimates do not diverge from the actual values due to system modeling errors or erroneous initial conditions.

An observer is a dynamical system which has as inputs u, the inputs to the system and y, the corresponding outputs of the system. It has as outputs (an estimate of) the state of the system.

Let the continuous time system equations be given as

$$\begin{aligned} \dot{x} &= Ax + Bu + w \\ y &= Cx + Du + v \end{aligned}$$

Where w and v are noise signals. The Luenberger observer is the following dynamical system with state z:

$$\begin{aligned} \dot{z} &= Az + Bu + L(y - Cz - Du) \\ &= (A - LC)z + (B - LD)u + Ly \end{aligned}$$

If we define the state estimation error

$$e = x - z$$

then it is easily found that

$$\dot{e} = (A - LC)e - Lv + w$$

This means that the dynamics of the estimation process is governed by the eigenvalues of the matrix $A - LC$. These eigenvalues can be selected by designing an appropriate matrix L with pole placement. In particular, all eigenvalues of $A - LC$ should be stable.

Note that for the design of a Luenberger observer, there is a fundamental trade-off between sensitivity of the estimation error and speed of convergence. The smaller the eigenvalues of $A - LC$, the faster the estimation error e will go to zero. However, for smaller eigenvalues, rapid changes in the signals will also induce rapid changes in the estimation error. The estimated state will change *nervously* and the state feedback based controller will behave quite nervous. Hence there is a trade-off between the ability to track the state and the ability to reject high frequency changes. An optimal solution, which is a trade-off between these choices, is the Kalman filter.

Example:

Consider the following first order system

$$\begin{aligned} \dot{x} &= -x + u \\ y &= x + w \end{aligned}$$

where w is sensor noise. The Luenberger observer becomes

$$\dot{z} = (A - LC)z + Bu + Ly$$

On figure 5.4 we show what happens for different choices of L, such that $A - LC = -10$ and -2. There is no input u, $u = 0$, but the initial state of x is 1. The initial state of the observer, $z(0)$, is 0. For $A - LC = -10$, the observer goes faster to the true value of the state x, but it is more subject to noise than the observer with $A - LC = -2$. The Sylvester equation approach can also be used here. Let the eigenvalues of $\tilde{\Lambda}_{obs}$ coincide with the desired observer dynamics. Then, there must exist a similarity transformation Y such that

$$Y(A - LC)Y^{-1} = \tilde{\Lambda}_{obs}$$

which can be rewritten as

$$YA - \tilde{\Lambda}_{obs}Y = YLC$$

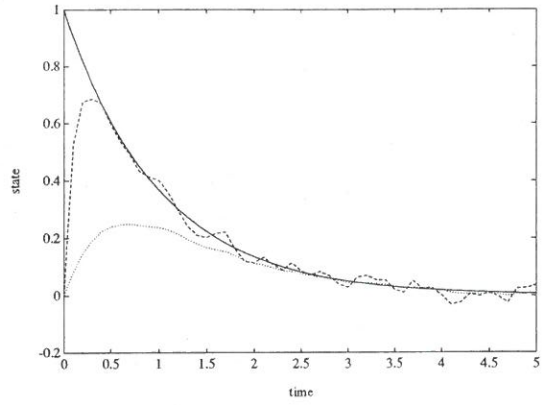


Figure 5.4: The trade off between speed of convergence and sensitivity to noise. The full line is the actual state x; the dashed line is the state of fast observer; the dotted line is the state of the slow observer. The fast observer converges faster to the real state, but is more sensitive to sensor noise.

Next define

$$H = YL$$

If we now pick a random matrix H, then the corresponding matrix Y can be found by solving the Sylvester equation

$$YA - \tilde{\Lambda}_{obs}Y = HC$$

and then obtain the observer pole placement matrix

$$L = Y^{-1}H$$

5.3 Dynamics of estimated state feedback controllers

We are now ready to combine the insights obtained so far. The idea is to first design an observer using poleplacement and then design a static feedback matrix, which feeds back the estimated state to the control input (instead of the real system's state, which is in most cases is unavaible). Combining the state x of the system and the state estimation error e in one state vector, together with the selected static estimated state feedback $u = -Fz$, results in the closed-loop system equation (which is autonomous):

$$\begin{pmatrix} \dot{x} \\ \dot{e} \end{pmatrix} = \begin{pmatrix} A - BF & BF \\ 0 & A - LC \end{pmatrix} \begin{pmatrix} x \\ e \end{pmatrix} + \begin{pmatrix} I & 0 \\ I & -L \end{pmatrix} \begin{pmatrix} w \\ v \end{pmatrix}$$

It is easy to verify that the set of closed-loop eigenvalues is the union of the sets of the eigenvalues of $A - BK$ and $A - LC$. Usually, the estimator poles are selected somewhat faster

than the control poles (4 a 10 times) such that their effect on the desired closed-loop dynamics is neglectable. Observe that, using poleplacement, the dynamics of the observer and controller part can be designed independently from one another.

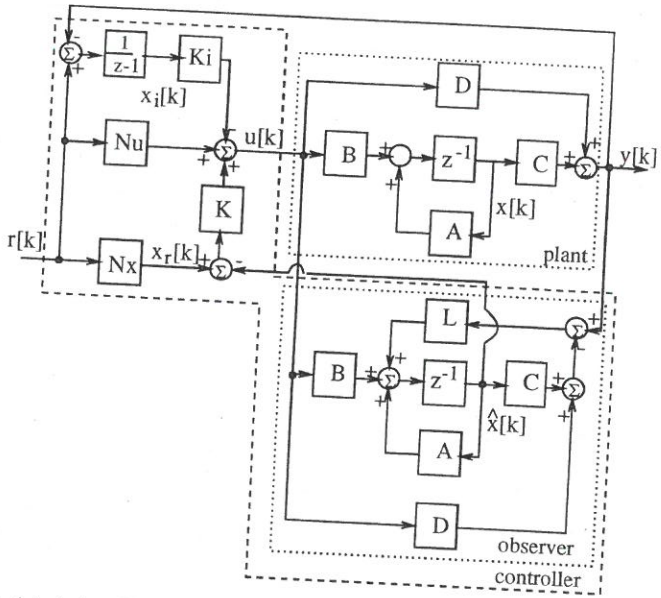


Figure 5.5: *A typical model-based control configuration, which consists of a model-based observer, followed by a static feedback on the estimated state (illustrated here for a discrete time system). One clearly sees how the observer contains a duplicate of the system's model and the observer gain L. The estimated state $\hat{x}_k$ is fed back to the control input. In this example here, there is also some preprocessing for the reference input r_k (matrix N_x) and there is also some direct output feedback (matrices K_i, N_u).*

5.4 The linear quadratic regulator

In this Section, we show how static state feedback is the optimal control strategy when one wants to minimize a certain performance criterion which is quadratic in the inputs and the states.

5.4.1 The problem

Consider the *quadratic cost function* of the form:

$$J = \sum_{k=0}^{N} (x_k^t Q x_k + u_k^t P u_k)$$

where both Q and P are given nonnegative definite matrices [2]. Next, we choose as a control law *static gain feedback*, i.e:

$$u_k = -Fx_k$$

The central question is now whether we can find a *stabilizing* feedback matrix F so that the quadratic cost function is minimized.

5.4.2 An example

Consider the system

$$x_{k+1} = x_k + u_k$$

and apply static state feedback $u_k = -fx_k$ where $f \in \mathbb{R}$. Suppose that our cost function is given as:

$$J = \sum_{k=0}^{\infty}(x_k^2 + \rho u_k^2)$$

The idea is that when ρ is small, we don't care about the magnitude of the control inputs (cheap control) and we are mainly interested in a fast response, requiring that the terms x_k^2 are small. If on the other hand ρ is large, we weight our control inputs heavily in the quadratic cost function (high cost). We find easily that $x_k = (1-f)^k x_0$ so that

$$\begin{aligned} J = \sum_{k=0}^{\infty}(x_k^2 + \rho u_k^2) &= x_0^2 \frac{(1+\rho f^2)}{1-(1-f)^2} \quad \text{if} \quad 0 < f < 2 \\ &= \infty \quad \text{if} \quad f \geq 2 \text{ or } f \leq 0 \end{aligned}$$

Observe that the cost is quadratic in x_0. If $0 < f < 2$, we find the optimal value of the feedback parameter f by taking the derivative of the cost with respect to f and setting the result equal to 0. The result is

$$f_{\text{opt}} = \frac{\sqrt{1+4\rho}-1}{2\rho}$$

For this choice of f the optimal cost is

$$\frac{\rho(\sqrt{1+4\rho}-1}{2\rho-\sqrt{1+4\rho}+1}$$

For low cost, cheap control, we find:

$$\lim_{\rho \to 0} f_{\text{opt}} = 1$$

which is deadbeat control. It has a cost of $x_0^2(1+\rho)$ which does increase as ρ increases.
For high cost control, we find that $f_{opt} \approx \frac{1}{\sqrt{\rho}}$ if ρ is large. Hence f_{opt} is small. It barely stabilizes the system: the closed-loop eigenvalue is $1 - \frac{1}{\sqrt{\rho}}$ but the plant input is small.

[2] It may sometimes happen that these matrices are also time-varying as P_k and Q_k. However, this represents no essential complications in the derivation that follows, so we assume that they are time-invariant.

5.4.3 The general case

Given the linear time invariant discrete time system

$$x_{k+1} = Ax_k + Bu_k$$

with the initial state x_0 given and the control objective:
Minimize over all control sequences u_k, the cost function:

$$J = \frac{1}{2}\sum_{i=0}^{N-1}[x_k^t Q x_k + u_k^t P u_k] + \frac{1}{2}x_N^t R x_N$$

We can interprete the state equations for $k = 0, 1, \ldots, N$ as $k+1$ constraints and use the technique of *Lagrange multipliers* to find the minimum of J. There will be one Lagrange multiplier vector, l_{k+1} for each value of k. First, we write the so-called Lagrangean as

$$J = \sum_{k=0}^{N}[\frac{1}{2}x_k^t Q x_k + \frac{1}{2}u_k^t P u_k + \frac{1}{2}x_N^t R x_N + l_{k+1}^t(-x_{k+1} + Ax_k + Bu_k)]$$

Setting the derivatives with respect to u_k, l_{k+1} and the x_k to zero results in

control equations	$u_k^t P + l_{k+1} B = 0$
state equations	$-x_{k+1} + Ax_k + Bu_k = 0$
adjoint equations	$x_k^t Q - l_k^t + l_{k+1}^t A = 0$
final state	$Rx_N - l_N = 0$

The *adjoint equations* may also be written as

$$l_k = A^t l_{k+1} + Qx_k$$

Obviously, these three sets of equations constitute a set of coupled difference equations defining the optimal solutions l_k and x_k and hence u_k. However, we need to know the initial conditions on x_k and the final conditions on l_k. The initial condition x_0 must be given. The solution to the two-point boundary value problem remains difficult however. One method, called the *sweep method*, is to assume that

$$l_k = S_k x_k$$

for a certain matrix S_k. Then the control equations become

$$\begin{aligned} Pu_k &= -B^t S_{k+1} x_{k+1} \\ &= -B^t S_{k+1}(Ax_k + Bu_k) \end{aligned}$$

Solving for u_k, we obtain

$$\begin{aligned} u_k &= -(P + B^t S_{k+1} B)^{-1} B^t S_{k+1} A x_k \\ &= -R_{k+1}^{-1} B^t S_{k+1} A x_k \end{aligned}$$

with an obvious definition for R_{k+1}. From the adjoint equations we find

$$S_k x_k = A^t S_{k+1} x_{k+1} + Qx_k$$

and substituting the state equation we find

$$S_k x_k = A^t S_{k+1}(Ax_k + Bu_k) + Qx_k$$

Now we use the expression for u_k:

$$S_k x_k = A^t S_{k+1}(Ax_k - BR_{k+1}^{-1}B^t S_{k+1}Ax_k) + Qx_k$$

and collect all terms on one side:

$$[S_k - A^t S_{k+1}A + A^t S_{k+1}BR_{k+1}^{-1}B^t S_{k+1}A - Q]x_k = 0$$

This equation must hold for any x_k whatever, hence the coefficient matrix must be identically zero, from which follows a backward equation in S_k, which is called the *Riccati difference equation*:

$$S_k = A^t[S_{k+1} - S_{k+1}BR_{k+1}^{-1}B^t S_{k+1}]A + Q \tag{5.1}$$

The boundary condition on the recursion is obtained from the final state

$$l_N = Rx_N = S_N x_N$$

so that

$$S_N = R$$

Solving for u_k we obtain

$$u_k = -K_k x_k$$

where

$$K_k = (P + B^t S_{k+1}B)^{-1}B^t S_{k+1}A$$

This matrix K_k is the desired optimal time-varying feedback gain matrix. Note that the optimal gain sequence is independent of the initial condition x_0. It can be precomputed from the matrices P, Q, A, B as long as the length N of the problem is known.

One can also evaluate the cost function J using the optimal values:

$$J_{\min} = \frac{1}{2}x_0^t S_0 x_0$$

Hence if we have computed S_0, we can immediately evaluate the minimal cost.

5.4.4 An example

Consider the discrete-time control system defined by

$$x_{k+1} = 0.3679x_k + 0.6321u_k, \qquad x_0 = 1$$

Determine the control sequence that minimizes the following cost function:

$$J = \frac{1}{2}\sum_{k=0}^{9}[x_k^2 + u_k^2] + \frac{1}{2}x_{10}^2$$

Note that in this example $Q = 1$, $P = 1$ and $R = 1$. Also, determine the minimum value of the cost function J.
The *Riccati difference equation* we have to solve is:

$$S_k = (0.3679)[S_{k+1} - S_{k+1}(0.6321)(1 + 0.6321S_{k+1}0.6321)^{-1}(0.6321)S_{k+1}](0.3679) + 1$$

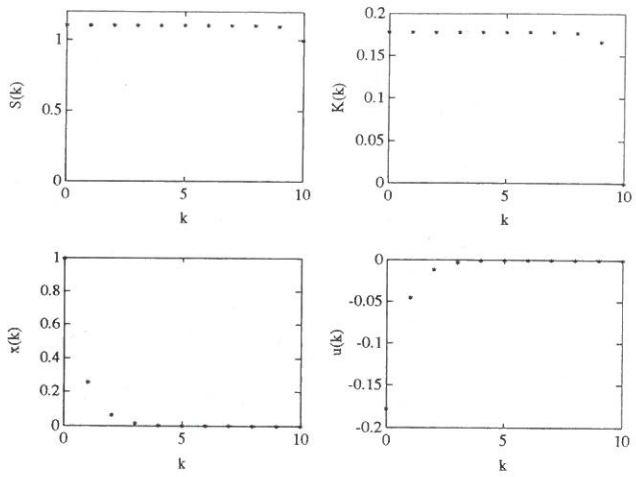


Figure 5.6: The solution of the Riccati equation S_s (upper left plot), the feedback gain K_k (upper right), the state x_k (lower left) and the optimal control sequence u_k (lower right) for the system $x_{k+1} = 0.3679x_k + 0.6321u_k$

which can be simplified to

$$S_k = 1 + 0.1354S_{k+1}[1 + 0.3996S_{k+1}]^{-1}$$

The boundary condition for S_k is

$$S_N = S_{10} = R = 1$$

Compute now S_k backward from $k = 9$ to $k = 0$. This is shown in Figure 5.6. Notice that the values of S_k rapidly approach the steady-state value S_{ss}. The steady-state value S_{ss} can be obtained from

$$S_{ss} = 1 + 0.1354S_{ss}(1 + 0.3996S_{ss})^{-1}$$

Solving this equation for S_{ss}, we have

$$S_{ss} = 1.1037 \quad \text{or} \quad -2.2674$$

Since S_k must be positive, we find the steady-state value for S_k to be 1.1037.
The feedback gain K_k can be computed from:

$$K_k = [1 + (0.6321)S_{k+1}(0.6321)]^{-1}(0.6321)S_{k+1}(0.3679)$$

This is also shown in Figure 5.6. Also the state x_k and the optimal control $u_k = -K_k x_k$ are shown. One can evaluate the cost function J using the optimal values:

$$\begin{aligned} J_{min} &= \frac{1}{2}x_0^t S_0 x_0 \\ &= \frac{1}{2}1(1.1037)1 \\ &= 0.5518 \end{aligned}$$

The important conclusion from the example is that the control gains K_k keep their constant value until at the very end of the time horizin, in which they start varying. This suggest that, when the time horizon N is large, a 'steady state' controller with its gain matrix fixed and equal to the initial gains will not differ very much from the optimal one.
We will explore this idea in the next section .

5.4.5 The steady state solution

One can show that as $N \to \infty$, the solutions S_k converge to a steady-state value S_∞. This steady state solution can be found from the so-called *algebraic Riccati equation*, which is the following quadratic matrix equation:

$$S_\infty = Q + A^t S_\infty A - A^t S_\infty B(P + B^t S_\infty B)^{-1} B^t S_\infty A$$

$$K_\infty = (P + B^t S_\infty B)^{-1} B^t S_\infty A$$

The steady-state optimal control law is then time-invariant. Kalman has shown that the closed-loop system matrix $A - BK_\infty$ is asymptotically stable if the system (A, B) is controllable and $(A, P^{1/2})$ is observable.

5.4.6 Solving the algebraic Riccati equation

Recall the state equations and the adjoint equations:

$$\begin{aligned} Ax_k &= x_{k+1} - Bu_k \\ &= x_{k+1} + BP^{-1}B^t l_{k+1} \\ A^t l_{k+1} &= l_k - Qx_k \end{aligned}$$

We can combine these equations as

$$\begin{pmatrix} A & 0 \\ -Q & I_n \end{pmatrix} \begin{pmatrix} x_k \\ l_k \end{pmatrix} = \begin{pmatrix} I_n & BP^{-1}B^t \\ 0 & A^t \end{pmatrix} \begin{pmatrix} x_{k+1} \\ l_{k+1} \end{pmatrix}$$

If we take the z-transform, we find

$$\begin{pmatrix} A & 0 \\ -Q & I_n \end{pmatrix} \begin{pmatrix} X(z) \\ L(z) \end{pmatrix} = \begin{pmatrix} I_n & BP^{-1}B^t \\ 0 & A^t \end{pmatrix} \begin{pmatrix} X(z) \\ L(z) \end{pmatrix} z$$

With

$$y = \begin{pmatrix} X(z) \\ Y(z) \end{pmatrix} = \begin{pmatrix} v \\ w \end{pmatrix}$$

and $z = \lambda$ we get

$$\begin{pmatrix} A & 0 \\ -Q & I_n \end{pmatrix} y = \begin{pmatrix} I_n & BP^{-1}B^t \\ 0 & A^t \end{pmatrix} y\lambda$$

which has the format of a *generalized eigenvalue problem*. Note that the two matrices involved are non-singular if A is non-singular. In that case, the generalized eigenvalue problem can be converted to an ordinary eigenvalue problem.

Let us now show that if λ_i is an eigenvalue, then also $1/\lambda_i$ will be one. In words, the eigenvalue spectrum is symmetric with respect to the unit circle. Indeed, if y_i is the eigenvector corresponding to λ_i, then we find

$$\begin{aligned} (A - \lambda_i I)v_i - \lambda_i BP^{-1}B^t w_i &= 0 \\ -Qv_i + (I_n - \lambda_i A^t)w_i &= 0 \end{aligned}$$

Now define

$$\begin{aligned} g_i &= -w_i \\ h_i &= v_i/\lambda_i \end{aligned}$$

then

$$\begin{aligned} (\lambda_i I - A)h_i - BP^{-1}B^t g_i &= 0 \\ -\lambda_i Q h_i - (I_n - \lambda_i A^t)g_i &= 0 \end{aligned}$$

which can be written as

$$\begin{pmatrix} A & 0 \\ -Q & I_n \end{pmatrix}^t \begin{pmatrix} -w_i \\ v_i/\lambda_i \end{pmatrix} = \begin{pmatrix} I_n & BP^{-1}B^t \\ 0 & A^t \end{pmatrix}^t y\frac{1}{\lambda}$$

The last pair of matrices has the same set of eigenvalues as the pair without transposes. The conclusion is that if λ_i is an eigenvalue, then $1/\lambda_i$ is an eigenvalue too.
In what follows, we assume that the eigenvalues λ_i are distinct and $|\lambda_i| < 1$. Then $1/\lambda_i$ are all lying outside the unit circle.
We will also assume for simplicity that the vectors $v_1, \ldots, v_n$ are linear independent.
Let us assume that there exists an $n \times n$ matrix S such that

$$w_i = Sv_i \quad i = 1, \ldots n$$

Then we find from the generalized eigenvalue problem that

$$\begin{pmatrix} A & 0 \\ -Q & I_n \end{pmatrix} \begin{pmatrix} v_i \\ Sv_i \end{pmatrix} = \begin{pmatrix} I_n & BP^{-1}B^t \\ 0 & A^t \end{pmatrix} \begin{pmatrix} v_i \\ Sv_i \end{pmatrix} \lambda_i$$

which can be rewritten as

$$\begin{aligned} Av_i &= \lambda_i(I_n + BP^{-1}B^t S)v_i \\ \lambda_i A^t S v_i &= -Qv_i + Sv_i \end{aligned}$$

If we premultiply the first equation with $A^t S$, we find

$$\lambda_i A^t S v_i = A^t S(I_n + BP^{-1}B^t S)^{-1} A v_i$$

so that we can eliminate λ_i to find

$$(S - Q - A^t S(I_n + BP^{-1}B^t S)^{-1}A)v_i = 0$$

Since $v_1, \ldots, v_n$ are independent, we must have

$$S - Q - A^t S(I_n + BP^{-1}B^t S)^{-1}A = 0$$

It is easily found [3] that this is precisely the algebraic Riccati equation!

Hence, we have found the following algorithm to solve the algebraic Riccati equation:

- Solve the generalized eigenvalue problem

$$\begin{pmatrix} A & 0 \\ -Q & I_n \end{pmatrix} \begin{pmatrix} v_i \\ w_i \end{pmatrix} = \begin{pmatrix} I_n & BP^{-1}B^t \\ 0 & A^t \end{pmatrix} \begin{pmatrix} v_i \\ w_i \end{pmatrix} \lambda_i$$

- Let v_i, w_i be the vectors corresponding to the eigenvalues λ_i within the unit circle. Define

$$V = (\, v_1 \;\; v_2 \;\; \ldots \;\; v_n \,) \qquad W = (\, w_1 \;\; w_2 \;\; \ldots \;\; w_n \,)$$

- Then the solution S_∞ to the algebraic Riccati equation is given by

$$S_\infty = WV^{-1}$$

5.4.7 Another example

Consider the system

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)$$

where

$$\begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

and the costfunction is given by

$$J = \frac{1}{2}\sum_{k=0}^{\infty}[x^*(k)Qx(k) + u^*(k)Pu(k)]$$

where

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad P = 1$$

Determine the optimal control law to minimize the cost function. Also, determine the minimum value of J.
The corresponding *generalized eigenvalue problem* is

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_i \\ w_i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} v_i \\ w_i \end{bmatrix} \lambda$$

[3] Use the general matrix inversion formula $(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}$.

The generalized eigenvalues are

$$\begin{matrix} 0.45685 \\ -0.45685 \\ 2.18890 \\ -2.18890 \end{matrix}$$

Notice that $0.45685 = 1/2.18890$. The matrix with the corresponding eigenvectors is

$$\begin{bmatrix} 0.1233 & 0.2433 & 0.8463 & -0.3548 \\ 0.2699 & -0.5326 & 0.3866 & 0.1621 \\ 0.7374 & 0.3899 & -0.2830 & 0.4429 \\ 0.6068 & -0.7107 & -0.2329 & -0.8073 \end{bmatrix}$$

The steady-state solution of the Riccati equation is then

$$\begin{aligned} S &= \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} v_1 & v_2 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} 0.7374 & 0.3899 \\ 0.6068 & -0.7107 \end{bmatrix} \begin{bmatrix} 0.1233 & 0.2433 \\ 0.2699 & -0.5326 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} 3.7913 & 1.0000 \\ 1.0000 & 1.7913 \end{bmatrix} \end{aligned}$$

The optimal control law is

$$\begin{aligned} u(k) &= -(P + B^t SB)^{-1}B^t SA \\ &= -\begin{bmatrix} 1 & 0.7913 \end{bmatrix} x(k) \end{aligned}$$

The minimal cost is

$$\begin{aligned} J_{min} &= \frac{1}{2}x^t(0)Sx(0) \\ &= \frac{1}{2}\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 3.7913 & 1.0000 \\ 1.0000 & 1.7913 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= 1.8956 \end{aligned}$$

5.4.8 Elimination of cross terms

A common choice for Q comes from the output equation:

$$y_k = Cx_k + Du_k$$

Then

$$y_k^t y_k = x_k^t C^t C x_k + u_k^t D^t D u_k + 2x_k^t C^t D u_k$$

In general, one may use the quadratic cost function

$$J = \sum_{k=0}^{\infty}(x_k^t P_{xx} x_k + u_k^t P_{uu} u_k + 2x_k^t P_{xu} u_k)$$

In what follows, we assume that P_{uu} is invertible. We can eliminate the cross-terms in x_k and u_k by defining a new input vector v_k as

$$v_k = u_k + M^t x_k$$

where

$$M = P_{xu} P_{uu}^{-1}$$

Then, we find for the state space equations

$$\begin{aligned} x_{k+1} &= (A - BM^t)x_k + Bv_k \\ y_k &= (C - DM^t)x_k + Dv_k \end{aligned}$$

and the cost function is now transformed into

$$J = \sum_{i=1}^{\infty} (x_k^t Q_{xx} x_k + v_k^t P v_k)$$

where

$$Q_{xx} = P_{xx} - P_{xu} P_{uu}^{-1} P_{xu}^t$$

Hence, we have eliminated the cross-term.

5.5 Neural optimal control design

Recently, methods have been developed for designing either static or dynamic nonlinear optimal controllers, parametrized respectively by feedforward or recurrent neural networks. A new feature of these method is the possibility for realizing transitions between working points with output feedback (no observer needed). This is extremely important for industrial applications, in which it frequently happens that around a working point (equilibrium point) *linear* control strategies are succesful, but when one goes from one point to another, the transition between these points typically causes lots of troubles. (At present, this problem is mostly solved with heuristic solutions such as for instance gain scheduling).
Local stability at the target point is guaranteed by imposing a linear controller design at that point. This allows to design first a suitable linear controller around the target point and incorporate these results into the neural controller, that realizes the transition between two points in state space. This procedure is illustrated by the examples of swinging up an inverted or a double inverted pendulum, with local stability at the upper equilibrium point.
We will not dive into the mathematical details of how these neural networks have to be designed. Or only purpose is to show how they can also be used in setting up powerful multivariable nonlinear control strategies. For details, the reader is referred to [24] [25].

5.6 Conclusions

In this chapter, we have been discussing more recent advanced control system design methodologies (we have not tried in any way to be exhaustive).
First, we have been describing how poleplacement strategies are very useful to design model-based observer-based static feedback controllers for linear systems. We have also shown how some optimal controllers can be cast in this framework. Finally we have given a simple example of the power of neural net controllers, a topic which receives currently much research interest.
The topics we have been dealing with here are only the top of an iceberg which consists of a vast

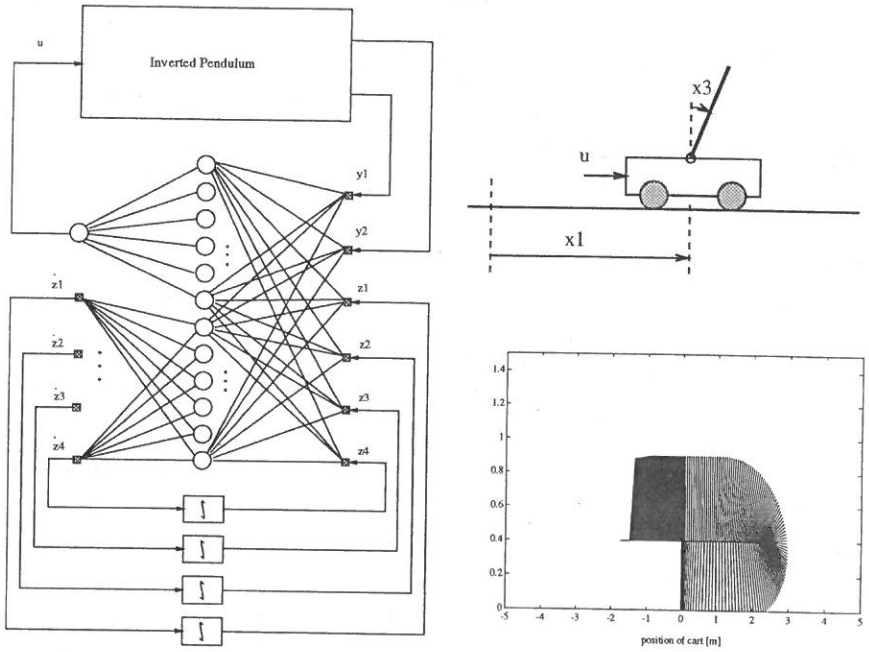


Figure 5.7: Inverted pendulum controlled by a recurrent neural network (dynamic output feedback) for performing the swinging up problem. The outputs y_1 and y_2 are the position of the cart (x_1) and the angle of the pole (x_3). The state variables are $x_1, \dot{x}_1, x_3, \dot{x}_3$ and u is the force applied to the cart. In this case the neural controller behaves as an LQG (Linear Quadratic Gaussian) controller around the upper equilibrium point, ensuring the pole will stay balanced.

collection of theories and methodologies, including robust control, loop transfer recovery, internal model control, nonlinear control, adaptive control, fuzzy control....
We would certainly need two days instead of two hours to give an introduction to these important recent developments.

Bibliography

[1] Aling H., Milletti U., Kosut R.L., Mesaros M.P., Van Overschee P., De Moor B. *An interactive system identification module for Xmath.* Proc. of the American Control Conference, June 2-4, 1993, San Francisco, USA, pp. 3071-3075

[2] Aling H., Kosut R., Van Overschee P., De Moor B., Boyd S. *Xmath Interactive System Identification Module.* Manual **ISID I** , Integrated Systems Inc., Santa Clara, CA, USA, 1993.

[3] Anderson B.D.O., Moore J.B. *Optimal control: Linear quadratic methods.* Prentice-Hall, 1990.

[4] Aström K., Wittenmark B. *Computer Controlled Systems: Theory and Design.* Prentice Hall, 1984.

[5] Battacharyya S.P., Desouza E. *Pole placement via Sylvester's equation.* Systems and Control Letters 1, 198?, pp.261-263.

[6] Black H.S., Obituary, see article: *Inventing the negative feedback amplifier.* IEEE Spectrum, Dec. 1977, pp.54-60.

[7] Bode H.*Network Analysis and Feedback Amplifier Design.* D. Van Nostrand, New York City, N.Y., 1945.

[8] Bode H.W. *Obituary statement: Harry Nyquist.* IEEE Trans. Aut. Control, Vol. AC-22, no.6, dec. 1977.

[9] Box G.E., Jenkins G.M. *Time series analysis, forecasting and control.* Revised edition, Holden-Day series in time series analysis and digital processing, Holden-Day, Oackland, 1976.

[10] Boyd S., Barrat C. *Linear controller design: Limits of performance.* Prentice Hall Information and System Sciences Series, Englewood Cliffs, NJ, 1991.

[11] Breedveld P.C. *Physical systems in terms of bondgraphs.* PhD thesis, TUTwente, The Netherlands, 1984.

[12] De Moor B. *Numerical algorithms for state space subspace system identification.* Academia Analecta, Klasse der Wetenschappen, Koninklijke Akademie voor Wetenschappen, België. To be published, 43 pp., 1994.

[13] De Moor B., Van Overschee P. *Graphical User Interface Software for System Identification.* ESAT-SISTA Technical Report 1994/06, Department of Electrical Engineering, Katholieke Universiteit Leuven, Belgium, January 1994.

[14] Van Overschee P., De Moor B., Aling H., Kosut R., Boyd S. *Xmath Interactive System Identification Module II,* Manual **ISID II**, 100 pp., Integrated Systems Inc., Santa Clara, CA, USA, 1993.

[15] Di Stefano III, Stubberud, Williams. *Feedback and control systems.* Schaum's outline series, Mc-Graw-Hill Book Company, 1976.

[16] Franklin G.F., Powell J.D. *Digital Control of Dynamic Systems.* Addison-Wesley Publishing Company, World Student Series, 1980.

[17] Kailath T. *Linear Systems.* Prentice Hall, Englewood Cliffs, New Jersey, 1980.

[18] Karnopp D.C., Margolis D.L., Rosenberg R.C. *System dynamics : a unified approach.* Second edition, John Wiley & Sons, New York, USA, 1990.

[19] Karnopp D.C., Rosenberg R.C. *Analysis and simulation of multiport systems.* MIT press, Cambridge, USA, 1968.

[20] Ljung L. *System identification - Theory for the User.* Prentice Hall, Englewood Cliffs, NJ, 1987.

[21] Ljung L. *Matlab, System Identification Toolbox, User's Guide,* The Mathworks Inc., Natick, Ma, USA, 1991

[22] Moons C. , De Moor B. *Parameter identification of induction motor drives* , Internal Report 93-42I, 1993

[23] Morari M., Zafirou E. *Robust process control.* Prentice Hall, 1989, Englewood Cliffs, New Jersey.

[24] Suykens J., B. De Moor, J. Vandewalle, *Static and dynamic stabilizing neural controllers, applicable to transition between equilibrium points,* to appear in *Neural Networks.*

[25] Suykens J., B. De Moor, J. Vandewalle, *Nonlinear system identification using neural state space models, applicable to robust control design,* K.U. Leuven, Belgium, Department of Electrical Engineering, ESAT-SISTA report TR 1994-07, submitted for publication.

[26] Vanvuchelen P., De Moor B. *A Multi-Objective Optimization Approach for Parameter Setting in System and Control Design.* Proc. of the 16th IFIP Conference on system modelling and optimization, Compiegne, July 5-9, 1993, France pp. 63-66.

[27] Van hamme *Identification of Linear Systems from time or Frequency-Domain Measurements,* PhD. Dissertation V.U. Brussels, 1992.

[28] Wiberg D.M. *State space and linear systems.* Schaum's Outline Series, McGraw-Hill Book Company, 1971.

[29] Ziegler J.G., Nichols N.B. *Optimum settings for automatic controllers.* Trans. ASME 64, pp.759-768, 1942.