Algorithms and Tools for System Identification Using Prior Knowledge

LINKÖPING STUDIES IN SCIENCE AND TECHNOLOGY

Thesis no: 456

Peter Lindskog

LiU-TEK-LIC-1994:42
ISBN 91-7871-422-2
ISSN 0280-7971

Linköping, September 1994
Algorithms and Tools for System Identification Using Prior Knowledge

Peter Lindskog

Department of Electrical Engineering
Linköping University, S-581 83 Linköping, Sweden
E-mail: lindskog@isy.liu.se

Linköping 1994
Abstract

One of the hardest problem in system identification is that of model structure selection. In this thesis two different kinds of *a priori* process knowledge are used to address this fundamental problem.

Concentrating on linear model structures, the first prior taken advantage of is knowledge about the systems' dominating time constants and resonance frequencies. The idea is to generalize FIR modeling by replacing the usual delay operator with discrete so-called Laguerre or Kautz filters. The generalization is such that stability, the linear regression structure and the approximation ability of the FIR model structure is retained, whereas the prior is used to reduce the number of parameters needed to arrive at a reasonable model. Tailorized and efficient system identification algorithms for these model structures are detailed in this work. The usefulness of the proposed methods is demonstrated through concrete simulation and application studies.

The other approach is referred to as *semi-physical modeling*. The main idea is to use simple physical insight into the application, often in terms of a set of unstructured equations, in order to come up with suitable nonlinear transformation of the raw measurements, so as to allow for a good model structure. Semi-physical modeling is less "ambitious" than physical modeling in that no complete physical structure is sought, just combinations of inputs and outputs that can be subjected to more or less standard model structures, such as linear regressions. The suggested modeling procedure shows a first step where symbolic computations are employed to determine a suitable model structure – a set of regressors. We show how constructive methods from commutative and differential algebra can be applied for this. Subsequently, different numerical schemes for finding a subset of "good" regressors and for estimating the corresponding linear-in-the-parameters model are discussed. We also deal with more informal tools such as a programming environment.

And perhaps more importantly, software tools supporting the suggested approach have been designed and implemented.
Preface

The common theme of this thesis is the use of a priori knowledge in system identification. After a brief introduction to modeling in Part I, two different approaches in this direction are investigated in Parts II and III, namely

II modeling using Laguerre and Kautz filters, and

III semi-physical modeling.

Some of the material in Chapters 3 and 4 is rewritten and reorganized results from


Various parts of the material in Chapter 5 can be found in


The basic idea behind semi-physical modeling is discussed in

I guess it's hard to complete a thesis without having an inspiring atmosphere to work in. Yes, guess must be the word to use as the people of the automatic control group really provide the stimulating and friendly environment that is so important for all research.

I would first like to thank my supervisors, Professors Lennart Ljung and Torkel Glad, for creating and maintaining this excellent atmosphere. No matter their work load they have always taking interest in my (weird?) questions, and without their flooding well of ideas, inspiration and support this thesis would never have been completed.

I am also deeply grateful to Professor Bo Wahlberg who co-authored and later proof read some of the material in Part II. Our discussions over the net have always been fruitful.

Special thanks goes to Torbjörn Andersson, Håkan Fortell, Dr. Krister Forsman and Dr. Inger Klein for proof reading various part of the manuscript. Their numerous comments and suggested improvements have been invaluable. Errors, misconceptions and “confuse a cat ltd.” sections no doubt remain, and they are my own!

Finally, I would like to thank my parents, Leif and Ing- Britt, my brother Stefan and my sister Linda for their never-ending love and support. Thank you all!
Contents

Notational Conventions vii

I Modeling in General, Identification in Particular 1

1 Introduction 3
  1.1 Systems, Experiments and Models .................................................. 3
  1.2 The A Priori Knowledge .............................................................. 4
  1.3 Organization of the Thesis ...................................................... 6

2 System Identification 7
  2.1 The Data, the Model Structure and the Selection Criterion ............ 7
  2.2 Black Box Structures .............................................................. 9
  2.3 Physically Parameterized Structures ............................ 10
  2.4 Parameter Estimation .............................................................. 11
  2.5 Model Validation ................................................................. 12
  2.6 Tools for the Identification Procedure .......................... 13

II Modeling Using Laguerre and Kautz Filters 15

3 Orthonormal Functions 17
  3.1 Orthonormal Functions – an Introduction .................................... 18
  3.2 FIR Modeling ................................................................. 19
  3.3 Time Series ................................................................. 22
  3.4 Earlier Work and Trends .................................................. 23

4 Discrete Laguerre and Kautz Functions 25
  4.1 Choice of Basis ............................................................... 25
  4.2 Linear Regression Formulation ............................................ 33
  4.3 Links to Other Model Structures ........................................ 35
Contents

4.4 State Space Formulations .......................................................... 37
4.5 Effects of Initial Conditions ......................................................... 43
4.6 Searching for the Filter Coefficients .............................................. 45
4.7 Examples ..................................................................................... 46

5 Applications ................................................................................... 49
  5.1 Aircraft Flight Flutter ................................................................. 49
  5.2 Neutron Flux Noise ................................................................. 53

III Semi-Physical Modeling .............................................................. 57

6 The Semi-Physical Modeling Problem ............................................. 59
  6.1 Introduction .............................................................................. 59
  6.2 An Illustrative Example ............................................................. 60
  6.3 Required Tools ......................................................................... 65
  6.4 Some Other Approaches .......................................................... 68

7 Algebraic Tools in Semi-Physical Modeling ..................................... 71
  7.1 Commutative Algebra and Gröbner Bases ...................................... 71
  7.2 Differential Algebra and Characteristic Sets ................................. 78
  7.3 Obtaining Algebraic Differential Equations .................................... 85
  7.4 Aspects on the Computational Complexity ..................................... 90
  7.5 Representing Derivatives in Discrete Time .................................... 92

8 Selecting the Regression Model ...................................................... 95
  8.1 Model Selection Procedures ...................................................... 96
  8.2 Selecting Regressors Interactively – An Example ......................... 100

9 A Tool for Semi-Physical Modeling ................................................. 105
  9.1 Software Requirements and Design Considerations ....................... 105
  9.2 SEMI - A Tool for Doing Semi-Physical Modeling ....................... 109
  9.3 A Modeling Scenario with SEMI .............................................. 112

10 Conclusions and Future Extensions .............................................. 119

Bibliography ...................................................................................... 121

Subject Index .................................................................................... 129
Notational Conventions

Unless otherwise locally said, the symbols below have the following global meaning.

**Symbols**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{N}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$</td>
<td>Natural, rational, real and complex numbers, respectively.</td>
</tr>
<tr>
<td>$\mathbb{N}^+, \mathbb{Q}^+, \mathbb{R}^+$</td>
<td>Positive natural, rational and real numbers, respectively.</td>
</tr>
<tr>
<td>$\mathbb{R}^d$</td>
<td>Euclidian $d$-dimensional space.</td>
</tr>
<tr>
<td>$\mathbf{x}, \mathbf{X}$</td>
<td>Boldface letters are used for vectors, matrices and sets.</td>
</tr>
<tr>
<td>$I$</td>
<td>Identity matrix of appropriate dimension; in Chapter 7 it denotes an ideal.</td>
</tr>
<tr>
<td>$T$</td>
<td>The sampling interval.</td>
</tr>
<tr>
<td>$y(t), y(t)$</td>
<td>Output signal at index (usually time) $t$.</td>
</tr>
<tr>
<td>$u(t), u(t)$</td>
<td>Input signal at index (usually time) $t$.</td>
</tr>
<tr>
<td>$e(t), v(t)$</td>
<td>Disturbance signal at index (usually time) $t$.</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of samples used for parameter estimation.</td>
</tr>
<tr>
<td>$N^*$</td>
<td>Number of samples in the data set.</td>
</tr>
<tr>
<td>$\mathbf{y}^N, \mathbf{u}^N$</td>
<td>Output data vector containing $N$ measurements.</td>
</tr>
<tr>
<td>$\mathbf{Z}^N$</td>
<td>Input data vector containing $N$ measurements.</td>
</tr>
<tr>
<td>$\mathbf{Z}^N$</td>
<td>Output-input data matrix, $\mathbf{Z}^N = [\mathbf{y}^N \mathbf{u}^N]$.</td>
</tr>
<tr>
<td>$G(q), G(q, \theta)$</td>
<td>Transfer function from $u(t)$ to $y(t)$.</td>
</tr>
<tr>
<td>$G^0(q), G^0(q, \theta)$</td>
<td>True transfer function from $u(t)$ to $y(t)$.</td>
</tr>
<tr>
<td>$H(q), H(q, \theta)$</td>
<td>Transfer function from $e(t)$ to $y(t)$.</td>
</tr>
<tr>
<td>$H^0(q), H^0(q, \theta)$</td>
<td>True transfer function from $e(t)$ to $y(t)$.</td>
</tr>
<tr>
<td>$\theta, \Theta, \eta$</td>
<td>Parameter vectors.</td>
</tr>
<tr>
<td>$\hat{\theta}_N, \hat{\Theta}_N, \hat{\eta}_N$</td>
<td>Parameter vectors estimated from $N$ samples.</td>
</tr>
<tr>
<td>$\varphi(t), \Phi(t)$</td>
<td>Regression matrices in a regression model.</td>
</tr>
<tr>
<td>$\hat{y}(t; \theta)$</td>
<td>Predictor.</td>
</tr>
<tr>
<td>$\epsilon(\cdot)$</td>
<td>Prediction error $y(t) - \hat{y}(\cdot)$.</td>
</tr>
<tr>
<td>$V_N(\cdot)$</td>
<td>Loess function calculated from $N$ samples.</td>
</tr>
<tr>
<td>$R_N$</td>
<td>Regression matrix calculated from $N$ samples.</td>
</tr>
</tbody>
</table>
### Notational Conventions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{M}$, $g(t, \theta, \varphi(t))$</td>
<td>Model structure.</td>
</tr>
<tr>
<td>$\mathcal{M}^*$</td>
<td>Model set.</td>
</tr>
<tr>
<td>$\mathcal{M}(\theta)$</td>
<td>A particular model corresponding to the parameters $\theta$.</td>
</tr>
<tr>
<td>$D_M$</td>
<td>Set of values over which $\theta$ ranges in a model structure.</td>
</tr>
<tr>
<td>$L^2(-\pi, \pi)$</td>
<td>The space of real-valued square integrable functions on $(-\pi, \pi)$.</td>
</tr>
<tr>
<td>$k[x_1, \ldots, x_n]$</td>
<td>Differential polynomial ring in $x_1, \ldots, x_n$ with coefficients from a field $k$, usually $\mathbb{Q}$, $\mathbb{R}$ or $\mathbb{C}$.</td>
</tr>
<tr>
<td>$\langle f_1, \ldots, f_m \rangle$</td>
<td>Ideal generated by the polynomials $f_1, \ldots, f_m$.</td>
</tr>
<tr>
<td>$[f_1, \ldots, f_m]$</td>
<td>Differential ideal generated by the differential polynomials $f_1, \ldots, f_m$.</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>Set of differential polynomials; sometimes a differential ideal.</td>
</tr>
<tr>
<td>$S_1 \setminus S_2$</td>
<td>The set ${x : x \in S_1 \text{ and } x \not\in S_2}$.</td>
</tr>
</tbody>
</table>

### Operators and Functions

- $\subset$ Subset.
- $\cup$ Union.
- $\cap$ Intersection.
- $x^T$ Transpose of $x$.
- $X^{-1}$ Inverse of the matrix $X$.
- $\dot{x}(t), \ddot{x}(t), x^{(i)}(t)$ First, second and $i$:th time derivative of $x(t)$.
- $q$ Forward shift operator, $qf(t) = f(t + T) = f(t + 1)$.
- $q^{-1}$ Backward shift operator, $q^{-1}f(t) = f(t - T) = f(t - 1)$.
- $p$ Shorthand for $\frac{d}{dt}$.
- $\sigma$ Either differentiation operator $p$ or forward shift operator $q$.
- $L_k(q, a)$ Discrete Laguerre functions. $a$ is the Laguerre filter coefficient.
- $\Psi_k(q, b, c)$ Discrete Kautz functions. $b$ and $c$ are the Kautz filter coefficients.
- $\{\mathcal{F}_k(q)\}$ Sequence of $k$ orthonormal functions $\mathcal{F}(\cdot)$.
- $\arg \min_x f(x)$ Minimizing argument of $f(x)$.
- $E\{\cdot\}$ Mathematical expectation.
- $\dim(\cdot)$ Dimension of vectors and matrices.
- $\det(X)$ Determinant of matrix $X$.
- $O(n)$ Function with the property that $O(n)/n$ is bounded as $n \to \infty$.
- $||\cdot||$ (Euclidian) norm of a vector. Absolute value if the argument is a scalar.

- $>_{\text{lex}}$ Lexicographic monomial ordering. See Definition 7.6, page 73.
- $\text{Mdeg}(f)$ Multidegree of $f$. See Definition 7.7, page 73.
- $\text{Lc}(f)$ Leading coefficient of $f$. See Definition 7.7, page 73.
- $\text{Lm}(f)$ Leading monomial of $f$. See Definition 7.7, page 73.
- $\text{Lt}(f)$ Leading term of $f$. See Definition 7.7, page 73.
- Rem$(f, F)$ Remainder of $f$ on division by $F$. See Theorem 7.1, page 74.
- $S(f_1, f_2)$ $S$-polynomial of $f_1$ and $f_2$. See Definition 7.9, page 74.
- $\text{GB}(I)$ Minimal and unique Gröbner basis of the ideal $I$.
- $l_f$ Lie derivative operator computed with respect to $f$.
- $\lambda_f$ Ring-homomorphism operator computed with respect to $f$. 
Notational Conventions

< Ranking in Section 7.2.
Ld(f) Leader of the differential polynomial f. See page 80.
Lv(f) Leading variable of the differential polynomial f. See page 80.
ChS(\Sigma) Characteristic set of \Sigma. See page 81.
S(f) Separant of f. S(f) = \frac{\partial f}{\partial Ld(f)}.
I(f) Initial of the differential polynomial f. See page 81.
Lu(f) The highest unreduced derivative of f. See Definition 7.18, page 81.
Lp(f) The highest degree term of Lu(f). See Definition 7.18, page 82.
Lj(f, v) The coefficient of the differential monomial v in f. See Definition 7.18, page 82.
rem_p(f, A) Partial remainder of f with respect to A. See page 82.
rem(f, A) Remainder of f with respect to A. See page 83.
\sqrt{\Sigma} Radical ideal of \Sigma.

Abbreviations

ADE Algebraic Differential Equation.
AR AutoRegressive.
ARMA AutoRegressive Moving Average.
ARMAX AutoRegressive Moving Average with eXternal input.
ARX AutoRegressive with eXternal input.
BJ Box-Jenkins.
ETFE Empirical Transfer Function Estimate.
FIR Finite Impulse Response.
IO Input-Output.
MIMO Multiple Input Multiple Output.
MISO Multiple Input Single Output.
ODE Ordinary Differential Equation.
OE Output Error.
PCR Principal Component Regression.
PLS Partial Least-Squares.
SNR Signal to Noise Ratio.
SISO Single Input Single Output.

The operations performed in Chapter 7 soon lead to complicated structures. To simplify the understanding of the examples, we therefore underline the expressions currently being considered.
Part I

Modeling in General, Identification in Particular

Almost in the beginning was curiosity.

Isaac Asimov

God may be subtle, but He is not malicious.

Albert Einstein
Introduction

1.1 Systems, Experiments and Models

In everyday life models play and have always played an important role. They are used to explain various phenomena in our surroundings and to foresee coming events.

Even our early ancestors made frequent use of models. The hunter's survival, e.g., was tightly coupled to his hunting skills and his ability to avoid predators. Without pondering on the model concept, he no doubt learned to distinguish between a prey and a man-eater, how to hunt the prey and how to defend himself against the predator. When confronted with an animal species never seen before he used earlier gained experience to classify it and predict its behavior. Sometimes the prediction turned out to be incorrect, i.e., the animal reacted different than expected, and the internal model had to be refined. But as the number of such experiences increased the hunter's predictability improved although it was never perfected - the model still had its limited range of validity.

Thus the hunter's model describes the behavior of animals - the true system, and it was developed through education, practice and experience - the performed experiments. In fact, every model as we think of it is coupled to the tuple system and experiment. In our context the system can be almost anything, from the entire universe to the behavior of one single atom, or more formally any object or collection of objects whose characteristics we want to study, cf. Ljung and Glad [65] and Cellier [12]. In practice this formulation means that a system, in some sense, can be observed and possibly controlled, i.e., we can collect data from it by performing experiments. Here, the phrase "in some sense" stress that only some of the system's entities can be investigated, not all of them.

Ljung and Glad [65] distinguish between mental, verbal, physical and mathematical models. The hunter's animal model belongs to the first category, and show intuitive, hard-to-formalize features. A verbal model is qualitative in nature; words are used to describe the system's reaction given certain stimuli. The rule "if it's raining cats and dogs then the subsoil water level will increase significantly" can be a part of such a model. Knowledge-Base Systems (KBS, sometimes also referred to as expert systems) are examples of more complete and fully formalized verbal models.

The third group is physical models. What is usually meant is that physical, often small scale, models of ships, aircraft etc. are developed with the purpose to investigate
the behavior of the real system under realistic conditions.

In this work we solely focus on the last category, mathematical models. In such a model the relationships between the variables observable in the system are described by mathematical relations (often equations). To achieve a suitable generality we will deal with equations that are dynamic, i.e., the quantities may depend on earlier values on one or more variables. Notice, however, that this does not exclude the presence of purely algebraic, or static, relations.

Dynamic real-world systems are most often modeled by differential equations, which reflect time-continuous properties. To be able to analyze the derived models we use computers. Hence, the original model must at some point be converted to, or approximated by, a time-discrete counterpart – a number of difference equations. In this framework we only consider equations that have a single independent variable, usually the time \( t \). As a consequence, Partial Differential Equations (PDE) are beyond the scope of this thesis. The remaining modeling domain is, however, still of great generality and applicability. As a matter of fact, it is much too general for some of the chapters in this study and further restrictions will be imposed later on.

There are many reasons for developing mathematical models of already existing systems. Besides the sheer curiosity to know and understand mother Nature, some of the most common are given by the following list (which without doubt is incomplete):

- Because of safety and/or economical reasons it might be impractical or impossible to perform experiments on the real system. Thus experiment with the model instead.

- We are interested in physical states which must be monitored but that are not directly available through measurements, and therefore we try to deduce their values using a model.

- More advanced control structures require a model of the system.

- The model can be used in an operator decision support system. Before making a decision the operator can ask the model questions such as “What will happen if I choose this setting?”.

Yet another situation is when the system itself has not come to existence. Before settling the design it is important to choose the “correct” design parameter configuration. To figure out what “correct” means it is customary to construct a number of models of the intended system and pick the one that best meet the design criteria.

### 1.2 The A Priori Knowledge

Let us now turn to the central issue of this thesis, the use of a priori knowledge in system identification. Study for a moment the sketch of the chemical distillation plant in Figure 1.1 (from Stephanopoulos [90]). Suppose that the aim is to model the amount of C produced given some sort of information about A, B, C, the steam, and so forth.

Physically parameterized modeling means that all physical insight about the behavior of the chemical plant is condensed into the model structure, which typically contains both known and unknown parameters. Almost surely, such a modeling approach requires chemical expertise and is likely to be quite time-consuming. But no matter how many
system aspects that are accounted for, we still do not have complete process knowledge, and indeed believing so would be childish. There are always uncertain factors affecting the system. Besides, what is the point of developing a complicated model if in using it some of its features are ignored? All in all, a sound rule of thumb is to let the modeling effort reflect the intended use of the model.

Perhaps it was these observations that originally spurred and motivated the development of the black box modeling approach, where the models' internal dynamics are assumed to be unknown. But to follow Graebe [38] real systems are never that "black". What instead usually is meant by a black box is that it is true that there are some process information at hand but in discarding it we will not loose that much. Of course, this is generally not true - there ought to be many situations where partial a priori process information can be of great help in determining a suitable model. This "middle zone", somewhere in between pure physical (where all parameters are assumed to be known) and black box modeling, is usually termed grey box modeling, and is from an identification viewpoint the domain of interest in this thesis.

However, it seems somewhat naive to try to develop identification methods that cover all possible types of a priori assumptions. Suppose that we are allowed to perform a number of simpler experiments on the chemical plant above. One possibility is to add a batch of the reactants and measure at what rate the product C is obtained. The measurements could also reveal some other phenomena such as product fluctuations. Sometimes we do not even have to perform these experiments, as the information might be available from other, similar plants. At hand we thus have approximate knowledge about the systems dominating time constants and resonance frequencies. This particular a priori knowledge will be exploited when we in Part II try to estimate linear time-invariant models.

Now, a regular linear model may very well be insufficient for our purposes. The identification procedure can then only be successful if the "major" nonlinearities are built into the model structure. Unless other things are explicitly stated, "major" will throughout this thesis mean relations that arise from elementary physics. To be more
specific, this involves balance equations (conservation of energy, mass, electrons etc.) and simple constitutive relationships. Thus, these approximations express our a priori knowledge. Returning a last time to the chemical plant, it should be possible to apply the principle of conservation of energy both for the mixing vessel and the distillation column. Perhaps these balance expressions can be complemented with geometric aspects (area of the column etc.) as well as of known chemical reaction formulae.

At the heart of the matter is here nonlinear model structure selection. From these seemingly unstructured relationships (not always in state space form) we are searching for transformations of the raw measurements from the underlying process, so as to allow for a “good” (and typically nonlinear) input-output description. This search and tools for supporting it are the topics of Part III, and we will refer to it as semi-physical modeling. For natural reasons, a tool for doing this involves both symbolic and numerical calculations. Symbolic calculations are first used for solving systems of nonlinear equations, whereupon numerical calculations are invoked in the model estimation step, and possibly for model reduction. We should already here stress that semi-physical modeling is less “ambitious” than physically parameterized modeling, in that no complete physical structure is sought, just suitable inputs and outputs that can be subject to more or less standard model structures (preferably to linear regressions).

This thesis describes how the above detailed a priori knowledge can be incorporated in the identification procedure. In particular, it aims at illustrating how this can be done by means of computer.

1.3 Organization of the Thesis

The thesis has been divided into three separate parts, which are organized as follows. In Part I, comprising the first two chapters, we take a closer look at the art of mathematical modeling. Chapter 2 more or less serves two purposes. Firstly, it provides the theoretical framework necessary for the coming chapters, and secondly, it highlights some existing software tools that support the identification procedure.

Part II is devoted to modeling using so-called orthonormal basis functions. Its use in an identification context is introduced and motivated in Chapter 3. In Chapter 4 we start off by discussing two specific choices of such bases, namely Laguerre and Kautz functions. Tailorized and efficient estimation algorithms for both these cases are also developed in the same chapter. Subsequently, in Chapter 5, two industrial applications, aircraft flight flutter and neutron flux noise, are modeled using the suggested approach.

Semi-physical modeling is treated in Part III. After more formally having stated the problem and outlined tools required in Chapter 6, the parts that have a symbolic nature are addressed in Chapter 7. Among other things, we show how constructive analytical tools such as commutative and differential algebra can be used in arriving at a suitable model structure. The numerical requirements, which include various model reduction schemes, are thereafter detailed in Chapter 8. Eventually, in Chapter 9, all these algorithms are tied together thereby giving a software tool that supports semi-physical modeling in practice.

The thesis is finally summarized in a stand alone chapter containing some concluding remarks and suggestions for future research.
System Identification

In this chapter we will briefly describe the ideas behind the field of parametric system identification, and we start by stating the problem and discussing commonly encountered model structures. Then we proceed to address parameter estimation in general and the least-squares algorithm in particular. Some methods and means for model validation are also presented. To balance these more technical parts we lastly investigate a few software tools that support the identification procedure.

The presentation neither states new results nor is complete. The purpose with the chapter is to introduce main concepts, algorithms and ideas from which subsequent chapters can depart. As indicated above, we focus on parametric models and restrict the work to the ones “living” in the time-discrete domain, where we additionally assume that the parameters are time-invariant. In some sections of Part II we, however, break these assumptions by estimating nonparametric models for comparison as well as for model structure selection guidance. Descriptions of how to do this and more comprehensive treatments of system identification are given in, for example, Söderström and Stoica [88] and in Ljung [61], upon which this survey is based.

2.1 The Data, the Model Structure and the Selection Criterion

The system identification procedure is highly iterative in nature and show three main ingredients, all of which in some degree are subject to personal judgments:

- The data $Z^N$: To be able to estimate models we first need data, which will depend both on the true system and on the experimental conditions. We will by $Z^N$ denote a data set containing $N$ measurements that are organized as

$$Z^N = \begin{bmatrix} y^N & u^N \end{bmatrix},$$

$$y^N = \begin{bmatrix} y(1) & y(2) & \cdots & y(N) \end{bmatrix}^T,$$

$$u^N = \begin{bmatrix} u(1) & u(2) & \cdots & u(N) \end{bmatrix}^T.$$
Chapter 2. System Identification

with $y^N$ being a sequence of ordered outputs and $u^N$ a sequence of observable inputs. Sometimes there is no observable input at all, which means that the models must be based only on measured outputs ($z^N = y^N$). In the literature this particular situation is labeled time series modeling.

- **The model structure $\mathcal{M}$**: It is generally agreed upon that the single most difficult step in identification is that of model structure selection (and this is the very reason for why we later spend so much time on it). Roughly speaking the problem can be divided into three subproblems. The first one is to specify the type of model set to use. To be more specific, this involves selection between linear and nonlinear models, between black boxes, semi-physical and physically parameterized models, and so forth. Next, the size of the model set must be decided. This includes the choice of possible variables and combination of variables to use in the model description. It also involves fixing orders and degrees of the model types, usually to some intervals. Once these two issues are settled we in principle have determined a model set $\mathcal{M}^\ast$ over which the search for a model can be conducted. Notice though that $\mathcal{M}^\ast$ can be a very large subset, but by using a priori knowledge it can often be reduced significantly. The last item to consider is how to parameterize the model set $\mathcal{M}^\ast$ so that the estimation algorithms stand as good chances as possible to find reasonable parameter values. To classify the thesis, the latter issue is the main topic of Part II, while Part III largely is concerned with the former two matters.

Assuming from now on that the members of $\mathcal{M}^\ast$ can be parameterized by a finitely dimensional parameter vector $\theta \in D_\mathcal{M} \subset \mathbb{R}^d$, a particular model corresponding to $\theta$ is denoted $M(\theta)$. The model structure to which the model belongs is defined by the mapping

$$\mathcal{M} : D_\mathcal{M} \ni \theta \rightarrow M(\theta) \in \mathcal{M}^\ast.$$  \hfill (2.4)

We will usually not use this system theoretic notation, but instead denote the family of candidate models by a function

$$\hat{y}(t|\theta) = g(t, \theta, \varphi(t)), \hfill (2.5)$$

where $\hat{y}(t|\theta)$ accentuates that the function $g(\cdot)$ is predicted. The search for a model is thus conducted over the parameters $\theta$, and is based on $\varphi(t)$ which is composed of input signals up to index $t$ and output signals up to index $t - 1$.

- **The selection criterion $V_N(\theta, Z^N)$**: The purpose of the selection criterion is to rank different models according to some pre-determined cost function (hence each model is assigned a quality mark). The criterion can come in several shapes, although we shall here adopt a scalar measure of the fit between the predicted and the measured value, i.e., a measure based on

$$V_N(\theta, Z^N) = \frac{1}{N} \sum_{t=0}^{N} l(y(t) - \hat{y}(t|\theta)), \hfill (2.6)$$

where $l(\cdot)$ is a positive scalar valued function, typically chosen to be quadratic, i.e., $l(\cdot) = |\cdot|^2$. Sometimes $l(\cdot)$ is modified to increase "slower" than quadratic for large errors, since the criterion then becomes more robust to outliers (i.e. to large seldom occurring measurement errors).
2.2 Black Box Structures

Once these items are settled we have implicitly defined the searched for model. It then "only" remains to estimate the parameters $\theta$ and to decide upon whether the model is good enough or not. If the model cannot be accepted some or even all of the entities above have to be reconsidered; in the worst case we must start from the very beginning and collect new data. Thus system identification is iterative, and the model acceptance criterion show personal taste characteristics.

### 2.2 Black Box Structures

Quoting Ljung [63] "A black box structure is one where the parameterization in terms of $\theta$ is chosen so that the family of models $\{g(t,\theta,\varphi(t)), \theta \in D_M\}$ covers "many common and interesting" ones as possible". In practice, this means that very few facts about the true system is taken into account.

The most common class of model structures used for black box modeling is the following linear one:

$$y(t) = G(q, \theta)u(t) + H(q, \theta)e(t), \quad (2.7)$$

where $q$ is the shift operator, and $G(q, \theta)$ and $H(q, \theta)$ are rational transfer functions, both assumed to be proper and $H(q, \theta)$ additionally assumed to be monic. Here, $e(t)$ is the disturbance contribution, which for analysis purposes normally is assumed to be white noise. The one step ahead predictor of the system is (substitute the noise term with $y(t) - \hat{y}(t|\theta)$)

$$\hat{y}(t|\theta) = H^{-1}(q, \theta)G(q, \theta)u(t) + \left[1 - H^{-1}(q, \theta)\right]y(t), \quad (2.8)$$

and hence $g(\cdot)$ becomes

$$g(t, \theta, \varphi(t)) = H^{-1}(q, \theta)G(q, \theta)u(t) + \left[1 - H^{-1}(q, \theta)\right]y(t). \quad (2.9)$$

A straightforward and general parameterization of Equation (2.7) is given by (dropping for simplicity the $\theta$ argument)

$$y(t) = q^{-nk} \frac{B(q)}{A(q)F(q)}u(t) + \frac{C(q)}{A(q)D(q)}e(t), \quad (2.10)$$

with $nk$ representing the delay between the input signal $u(t)$ and the output signal $y(t)$, and with polynomials defined as

$$A(q) = 1 + a_1q^{-1} + \ldots + a_{na}q^{-na}, \quad (2.11)$$
$$B(q) = b_1 + b_2q^{-1} + \ldots + b_{nb}q^{-nb+1}, \quad (2.12)$$
$$C(q) = 1 + c_1q^{-1} + \ldots + c_{nc}q^{-nc}, \quad (2.13)$$
$$D(q) = 1 + d_1q^{-1} + \ldots + d_{nd}q^{-nd}, \quad (2.14)$$
$$F(q) = 1 + f_1q^{-1} + \ldots + f_{nf}q^{-nf}. \quad (2.15)$$

Applying Equation (2.8) to this model structure gives the predictor

$$\hat{y}(t|\theta) = q^{-nk} \frac{B(q)D(q)}{C(q)F(q)}u(t) + \left[1 - \frac{A(q)D(q)}{C(q)}\right]y(t). \quad (2.16)$$
In the following chapters we will only consider special cases of Structure (2.10), namely the ones with the following acronyms:

- **FIR** : \( y(t) = q^{-nk}B(q)u(t) + e(t) \), \( q \neq 0 \) (2.17)
- **ARX** : \( y(t) = q^{-nk} \frac{B(q)}{A(q)} u(t) + \frac{1}{A(q)} e(t) \) (2.18)
- **OE** : \( y(t) = q^{-nk} \frac{B(q)}{F(q)} u(t) + e(t) \) (2.19)
- **ARMAX** : \( y(t) = q^{-nk} \frac{B(q)}{A(q)} u(t) + \frac{C(q)}{A(q)} e(t) \) (2.20)
- **BJ** : \( y(t) = q^{-nk} \frac{B(q)}{F(q)} u(t) + \frac{C(q)}{D(q)} e(t) \) (2.21)
- **AR** : \( y(t) = \frac{1}{A(q)} e(t) \) (2.22)
- **ARMA** : \( y(t) = \frac{C(q)}{A(q)} e(t) \) (2.23)

where the last two model structures are reserved for time series modeling.

Of course, nonlinear black box structures have also been studied in the literature. Let us though defer a discussion about such approaches to Part III as it is first then nonlinear models are really considered.

### 2.3 Physically Parameterized Structures

Physically parameterized model structures are designed from physical insights and are customized for one specific application or for a class of applications. The model description is fixed, often through a quite laborious procedure, and contains known as well as unknown constants (representing resistances, densities, friction coefficients, and so on). This also means that all introduced variables \( x(t) \) have physical significance (speed, angle, temperature etc.).

Since most laws of physics are time-continuous it is here most natural to adopt the framework of differential equations. Although we often start our work in this domain, the resulting model is sooner or later converted into the time-discrete world. Before this, however, the gained relationships are often summarized in a time-continuous state space form like

\[
\begin{align*}
\dot{x}(t) &= f(t, x(t), \theta, u(t), e(t)) \\
y(t) &= h(t, x(t), \theta, u(t), e(t))
\end{align*}
\] (2.24)

where \( \theta \) denotes the unknown parameters. An even more general formulation is

\[
\begin{align*}
f(t, \dot{x}(t), x(t), \theta, u(t), e(t)) &= 0 \\
g(t, x(t), \theta, u(t), e(t)) &= 0 \quad \text{(2.25)} \\
y(t) &= h(t, x(t), \theta, u(t), e(t))
\end{align*}
\]

where the derivatives are allowed to occur implicitly. Here, the functions \( g(\cdot) \) emphasize that not only differential equations are used to describe the system, but also algebraic
ones. This class of structures (described by differential-algebraic equations, DAE’s for short) is the most general starting point allowed in the semi-physical modeling procedure (see Part III).

2.4 Parameter Estimation

We have earlier suggested to employ the prediction error, $\varepsilon(t, \theta)$,

$$\varepsilon(t, \theta) = y(t) - \hat{y}(t|\theta), \quad (2.26)$$

as a measure of the models' quality. By using a quadratic norm the loss function $V_N(\theta, Z^N)$ is defined as the scalar

$$V_N(\theta, Z^N) = \frac{1}{N} \sum_{t=t_0}^{N-1} \frac{1}{2} \varepsilon^2(t, \theta). \quad (2.27)$$

The parameter estimate $\hat{\theta}_N$ is then given by the minimizing argument of the loss function, or in mathematical terms by

$$\hat{\theta}_N = \arg \min_{\theta} V_N(\theta, Z^N), \quad (2.28)$$

where $\hat{\theta}_N$ is searched for over the subset $D_M$. This particular family of estimation algorithms is for obvious reasons labeled Prediction Error Methods (PEM).

The innocent-looking “arg min”-expression typically conceals substantial amount of computations, and recursive search algorithms, such as

$$\hat{\theta}_N^{(k+1)} = \hat{\theta}_N^{(k)} - J_N(V_N(\theta_N^{(k)}, Z^N)), \quad (2.29)$$

must often be used. Here, $J_N(\cdot)$ is a function expressing in what direction to update the parameters. See Ljung [61] for an introduction and Dennis and Schnabel [18] for a comprehensive treatment of this type of algorithms. Besides the computational complexity, one additional complication is here that the global minimum is not always found.

There is however one important subclass of model structures for which $\hat{\theta}_N$ can be calculated efficiently and explicitly, namely when the parameters enter the predictor linearly. This also implies that the predictor can be written as a linear regression

$$\hat{y}(t|\theta) = \theta^T \varphi(t), \quad (2.30)$$

which, e.g., is the case for the FIR, ARX and AR structures, but not for OE, ARMAX, BJ and ARMA structures. Notice that $\varphi(t)$ is allowed to hold any past signals, linear as well as nonlinear. Thus, the Structure (2.30) can be viewed as a general nonlinear predictor, a fact that will be utilized in Part III.

The least-squares estimate of a linear regression structure can readily be determined by solving the so-called normal equations ($t^0$ is the index of the first sample used)

$$\hat{\theta}_N = R_N^{-1} f_N = \left[ \sum_{t=t_0}^{N-1} \varphi(t) \varphi^T(t) \right]^{-1} \left[ \sum_{t=t_0}^{N-1} \varphi(t)y(t) \right], \quad (2.31)$$

provided that $R_N^{-1}$ exists. The many nice features of this algorithm have been a major motivating source for the methods to be presented, especially for those outlined in Part II.
2.5 Model Validation

After estimation, the obvious query is whether the derived model is adequate for its intended use or not. This is the subjective and overall hard problem of model validation. According to Bohlin [6] the sound way to attack the matter is to confront the model with all available type of information, including a priori knowledge, experimental data and experience of using the model. To gain confidence in a model one general advice is to employ as many validation tools, preferably of different kinds, as possible.

The first conceivable mean is to use simple common sense reasoning. If, e.g., the estimated parameter represents the length of a rod it must at least be positive. Hence, this type of tests belongs to the a priori category, which is especially important when the parameters have physical interpretations.

The overwhelming majority of methods developed are, however, based on experimental data. A very basic test is to investigate the variances of the estimated parameters. High variances compared to the parameter values indicate that something is wrong. Another quite useful possibility (see Part II) is to estimate several models in parallel. The frequency response of a parametric model can, e.g., be viewed in the same diagram as the nonparametric so-called spectral estimate (consult Ljung [61] for the algorithm). With some restrictions these curves should roughly be of the same shape.

Now, the most versatile validation procedure, all categories, is simulation. The true system and the derived model are then both fed with the same input signals, whereupon the measured true outputs are compared to the ones calculated from the model. For a just comparison it is here desirable that the simulation are based on fresh data, i.e., on data not earlier used for estimation (an approach which among statisticians is known as cross validation). In certain situations, e.g., when the estimated model is unstable, simulations can be quite hazardous. It is then usually much better to compare the measured output with the output from the k-step ahead predictor, although it can also be risky (see Chapter 8).

To end the section it is, especially on fresh data, often worthwhile to investigate the residuals $\varepsilon(t, \hat{\theta}_N) = y(t) - \hat{y}(t|\hat{\theta}_N)$ (or the prediction errors) of the model as well. A very simple test is to plot these residuals and investigate if the resulting sequence shows the expected disturbance pattern. In addition, the often assumed whiteness of the residual sequence can be revealed through the following sample covariances

$$\hat{R}_\varepsilon^N(\tau) = \frac{1}{N} \sum_{t=1}^{N-\tau} \varepsilon(t, \hat{\theta}_N)\varepsilon(t+\tau, \hat{\theta}_N),$$

(2.32)

which for $|\tau| \neq 0$ should be small, at least if $N$ is large. Furthermore, the assumed independency between the input and the residual signals can be inspected by computing

$$\hat{R}_{\varepsilon u}^N(\tau) = \frac{1}{N} \sum_{t=\tau}^{N} \varepsilon(t, \hat{\theta}_N)u(t-\tau)$$

(2.33)

for various time lags $\tau$. These should again be small, since otherwise there is more information in the output originating from the input, which means that there is still unmodeled dynamics present. Several other residual tests exist, see for example Draper and Smith [20].

Lastly, we should emphasize that the more tests a model is confronted with, the better is the chance of finding what it cannot be used for.
To reach a major group of potential users it is absolutely necessary that theory is condensed into an easy-to-use form. Today, this is almost synonymous with program packages, and indeed, system identification is no exception from this rule. On the contrary, it would be almost impossible to manage without computer assistance. But even then the identification task is generally not a simple one. This is essentially due to the immense possibilities and the subjective nature of the problem. Practically, this means that the identification procedure cannot be fully automated; it must be performed interactively \( \text{cf.} \) Bohlin [8] and Ljung [61]). For good and bad, this naturally calls for better trained and skilled users. However, user interaction also opens up new interesting opportunities. First of all, the user's chances to guide the model search is improved. It also becomes simpler to identify and avoid pitfalls, and as is pointed out by Larsson and Persson [57] the software can more easily be used to teach identification to the user. But to accomplish this better and more powerful computer tools of various kinds are needed.

According to Nagy [76] the second generation identification packages\(^1\) (belonging to "the interactive identification package period", 1970-) typically contain support for

- acquisition and handling of data,
- nonparametric identification methods,
- parametric identification methods for different types of model structures, and
- various means for evaluation of the models.

The commands are entered directly by the user in some sort of interactive program environment. Along with the numerical algorithms these programs provide rudimentary help facilities and plotting capabilities, but that is about all.

One of the first tools in this spirit was IDPAC [104], which was developed by Wieslander in the early seventies. It and most of its modern successors are built on top of existing numerical program platforms. This is, e.g., the case for the wide-spread System Identification Toolbox (SITB) [62], which is implemented in MATLAB [72], and for the add-on identification package available for MATRIX\(^x\) [49]. To various degrees these tools support the items listed above, although they are restricted to linear model structures. There also exist a number of programs allowing estimation of parameters in nonlinear systems. However, most of them concentrate on parameter optimization without paying much attention to model validation. As one exception from this rule we may mention IDKIT [7, 39, 40, 41], which among other things support estimation and validation of parameters in nonlinear state space structures. Needless to say, there exist many more second generation identification tools on the market, especially those that are customized for specific applications.

As earlier stressed, the rapid development of new hardware and software techniques have opened up new prospects in the field of identification. As of today, we are not that far away from realizing the third generation identification tools, which besides the

\(^1\)The first generation was born during the 1960's and are nowadays often labeled as "the batch routine era". The programs were usually written in a traditional program language like FORTRAN and consisted of a number of subroutines. Their functionality were pretty much the same as the second generation packages, with the major difference being that they were executed in a batch mode.
numerical and the graphical capabilities also provides decision support and advanced help. As the success of an identification session often depends on hard-to-formalize expert tricks one such means is Knowledge-Based Systems (KBS) of different kinds. Experimentations and results with such programs have lately been reported. Among many others we may here mention IFS [57] used together with IDPAC, and ID TOOL EXPERT [76] which complement the SITB. Third generation support can of course come in many other forms. One possibility is to integrate traditional modeling tools, such as graphical oriented bond graphs, with parameter identification. An approach in this direction is BondTool\textsubscript{M} developed by Nagy [76].

Most of the software schemes detailed in the following chapters have been implemented. The algorithms in Part II have been coded into a MATLAB package, fully designed to utilize the features provided by the SITB. This package belongs to the second generation software packages, whereas the tool for semi-physical modeling described in Part III can be categorized to belong to the third generation, since it involves symbolic support for model structure selection.
Part II

Modeling Using Laguerre and Kautz Filters

The following sentence is false.
The preceding sentence is true.

From the thought-provoking masterpiece

Gödel, Escher, Bach:
An Eternal Golden Braid
by Douglas R. Hofstadter.
Assume that a number of input-output data has been collected from a plant of interest. To get a quick feeling of the process' behavior and complexity one often starts to view the input-output relationship via nonparametric frequency domain models, such as the spectral analysis estimate [61, 88]. This approach is indeed appealing, since very little about the system in question must be known beforehand. Apart from the assumption that the system is linear the only practical consideration is the choice of weighting function to smooth out the frequency response (see, for instance, Ljung [61] for computational details). However, nonparametric models have their drawbacks. They do not explicitly employ a finitely-dimensional parameter vector, which in particular means that they cannot directly be used for simulations. Parametric models, on the other hand, overcome this shortage, but unfortunately new decisions concerning model structures and model orders must be taken.

The result of, e.g., the spectral estimate can readily be visualized in an ordinary Bode diagram. Information about resonance frequencies, number of peaks, dominating time constants, amplitude roll-off and so on are then visually revealed. In building standard parametric black-box models this knowledge should be exploited to decide upon which model order to use. Still, direct information about resonance frequencies and the magnitude of the time-constants are seldom, or never, explicitly utilized. Furthermore, step response experiments often precede the model estimation step itself. Such experiments are especially useful for determining the sampling interval, which again means that the dominating time constants of the system roughly are identified.

Said another way, all accessible a priori knowledge is not immediately considered when estimating a black box model of the system. The idea behind the orthonormal model structures to be discussed in Chapter 4 is precisely to tackle and take advantage of the approximately known dominating time constants and resonance frequencies. As we shall see this is done by generalizing the Finite Impulse Response (FIR) model structure, or in the time series case, by complementing the AutoRegressive (AR) ditto. The advantages of these structures turn out to be preserved, whereas the disadvantage in terms of high model orders is mastered.

In this introductory chapter we first define the concept of orthonormal basis functions. We then proceed to discuss benefits and drawbacks of the FIR and the AR model.
structures in Sections 3.2 and 3.3, respectively. Finally, Section 3.4 gives some historical
milestones, standard references and other orthonormal modeling approaches.

3.1 Orthonormal Functions — an Introduction

Suppose that the true, linear, stable, time-invariant and time-discrete system is governed
by the following general equation:

\[ y(t) = f\delta u(t-t) + v(t), \]  

where \( \{y(t)\} \) and \( \{u(t)\} \) are the output and input signal sequences, respectively. Notice
that by starting the sum at \( k = 1 \) we have assumed a unit time-delay between the
input and the corresponding output, meaning that the system is strictly proper. The
sequence of so-called Markov parameters \( \{g_k\} \), is commonly referred to as the system's
impulse response, cf. Kailath [51]. The last term, \( v(t) \), is the part of \( y(t) \) which cannot
be explained by the input, and is the disturbance contribution. We shall in the sequel
assume that \( \{v(t)\} \) is a stationary process with zero mean value, \( E\{v(t)\} = 0 \), and
covariance function

\[ R_v(\tau) = E\{v(t)v(t-\tau)\}. \]  

In other words, the noise properties are assumed to be time-invariant, i.e., it does not
depend on absolute but on relative time.

Using the delay operator, \( q^{-1} \), Equation (3.1) can be rewritten as

\[ y(t) = G^0(q)u(t) + v(t). \]  

\[ G^0(q) = \sum_{k=1}^{\infty} g_k q^{-k}. \]  

The system's transfer function is now given by the complex-valued function \( G^0(z) \), with
\( z \in \mathbb{C} \). For stable systems we have, see [61],

\[ \sum_{k=1}^{\infty} |g_k^0| < \infty, \]  

which in particular implies that \( G^0(z) \) is analytic on and outside the unit circle. One
immediate consequence of this statement is that the true system has all its poles inside
the unit disc.

In constructing the Laurent series expansion

\[ G^0(z) = \sum_{k=1}^{\infty} g_k^0 z^{-k} \]  

we have actually used a set of orthonormal functions, namely \( \{z^{-k}\} \) for \( k \in \mathbb{N} \). To verify
this let us first formally define what we mean by a set of orthonormal functions.
Definition 3.1 The sequence of functions \( \{ \mathcal{F}_k(e^{i\omega}) \} \), \( k \in \mathbb{N} \), forms an orthonormal set, i.e., an orthonormal basis, in \( L^2(-\pi, \pi) \) (see [55] and [91] for details) if and only if

\[
\frac{1}{2\pi} \int_{-\pi}^{\pi} \mathcal{F}_n(e^{i\omega})\mathcal{F}_m(e^{-i\omega})d\omega = \delta_{nm},
\]

where \( \delta_{nm} \) is the Kronecker delta defined by

\[
\delta_{nm} = \begin{cases} 
1 & \text{if } n = m \\
0 & \text{if } n \neq m.
\end{cases}
\]

Using this definition the following calculations show that \( \{z^{-k}\} \) forms an orthonormal basis.

Example 3.1 The ordinary set \( \{ \mathcal{F}_k(e^{i\omega}) = e^{-i\omega k} \}, k \in \mathbb{N} \), forms an orthonormal set, since for arbitrary \( n, m \in \mathbb{N} \) we have

\[
\frac{1}{2\pi} \int_{-\pi}^{\pi} \mathcal{F}_n(e^{-i\omega})\mathcal{F}_m(e^{i\omega})d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-i\omega n}e^{i\omega m}d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i(m-n)\omega}d\omega = \begin{cases} 
1 & \text{if } n = m \\
0 & \text{if } n \neq m.
\end{cases}
\]

By generalizing Equation (3.6), see [46] for further details, we get

\[
G^0(z) = \sum_{k=1}^{\infty} g_k^0 \mathcal{F}_k(z),
\]

with \( \mathcal{F}_k(z) \) being a set of orthonormal functions. The obvious question that arises here is what orthonormal functions to use in different situations?

3.2 FIR Modeling

Before addressing this question we note that the model structure of ours is described by an infinitely-dimensional parameter vector, which cannot be estimated in practice. To overcome this problem one apparent idea is to try to approximate the infinite sums above with finite ones. Thus we are confronted with functional approximation theory. See, e.g., Kreyszig [55] for a mathematical introduction and overview.

In an identification context, a simple approach is as follows. For stable systems the Markov parameters in Equation (3.1) necessarily tend to zero as \( k \) tends to infinity. Therefore, just truncate the impulse response at \( k = nb \), which yields the well-known Finite Impulse Response model

\[
y(t) = \sum_{k=1}^{nb} g_k^0 u(t-k) + \sum_{k=nb+1}^{\infty} g_k^0 u(t-k) + v(t),
\]

\( \epsilon(t) \)
where \( e(t) \) is the model error, composed of the truncation term and the original stochastic disturbances.

The FIR model structure really has some tractable features:

- It can be designed to approximate any stable transfer function with arbitrary accuracy just by choosing the order large enough, see Rabiner [79].
- It is guaranteed to be stable, since the output is compounded of finitely many elements that are all finite. See Figure 3.1.
- It can be realized efficiently, and numerical problems, such as quantization and roundoff errors, can often be made negligible, consult [79].
- It can be represented as a linear regression

\[
    y(t) = \theta^T \varphi(t) + e(t),
\]

\[
    \theta = [g_1^0 g_2^0 \ldots g_{nb}^0]^T,
\]

\[
    \varphi(t) = [u(t-1) u(t-2) \ldots u(t-nb)]^T,
\]

where the unknown coefficients, \( \theta \), can be analytically estimated through the computational appealing least-squares algorithm.

- \( G(z) \) can according to Ljung [61] be estimated consistently with the least-squares method whenever the unknown additive noise is uncorrelated with the input signal. Notice that this is the case even though the noise is colored.

- Overparametricizations is not too cumbersome. The model order \( nb \) can thus be overestimated without severe problems provided that the input signal is sufficiently rich. Effects of such overfits are certainly more troublesome for other model structures, such as the ARX one, where the difficulties increase rapidly with decreasing Signal to Noise Ratios (SNRs). This well-known fact stems from the inclusion of old outputs in the regression vector. Consult, e.g., Ljung and Wahlberg [68] for a more thorough treatment of factors affecting the accuracy of the estimated parameters.
3.2 FIR Modeling

- For physical systems, regardless of the amount of data collected, we are always faced with an error due to unmodelled dynamics (undermodeling). The normally hard problem of assessing this error is considered to be less difficult for FIR models, see [37, 48, 102, 103], than for more general model structures.

The first listed item inevitably leads to the question of how large \( nb \) to choose in order to obtain an adequate model. In answering this, recall that a FIR model directly should reflect the system's impulse response. For systems with dominating poles near the unit circle this response decays more slowly than for systems with poles near the origin. Hence, the truncation term of Equation (3.8) will be large in the former case, and the only way to decrease it is to increase the model order.

Poles, but also zeros, near the unit circle is more common than one first might expect. For example, in sampling a continuous system it is well-known [4] that the poles are mapped to

\[
q_i = e^{p_i T},
\]

where \( q_i \) denotes the time-discrete counterpart of the continuous pole \( p_i \). It is easy to verify that this transformation maps the left half plane onto the unit disc, and thus stability is preserved. For moderately fast sampled systems the argument \( p_i T \) will be small, leading to poles close to the unit circle and consequently a high FIR model order is required. Because of computation, information and sensitivity aspects this cannot be tolerated in practice. The problem is actually that the delay operator, \( q^{-1} \), has a too short memory (only one sampling step), and thus the standard basis \( \{q^{-k}\} \) is inappropriate in these situations. Let us justify this statement in an example.

**Example 3.2** Regard the simple noise-free and continuous system (where \( p \) denotes the differential operator)

\[
y(t) = \frac{1}{p + 1} u(t),
\]

which after sampling becomes (\( T \) is the sampling interval)

\[
y(t) = \frac{1 - e^{-T}}{q - e^{-T}} u(t) = G(q) u(t)
\]

for \( t = kT, k \in \mathbb{N} \). To model this we can employ a FIR model structure

\[
y(t) = g_1 u(t - T) + \ldots + g_{nb} u(t - nbT).
\]

The transfer function can now be expressed as a power series

\[
G(q) = \frac{1 - e^{-T}}{q - e^{-T}} = (1 - e^{-T}) \frac{q^{-1}}{1 - e^{-T} q^{-1}} = (1 - e^{-T}) \sum_{k=1}^{\infty} (e^{-T})^{k-1} q^{-k},
\]

where we see that a truncation of the impulse response at \( k = nb \) gives an error of magnitude \( O([e^{-T}]^n) \). For a short sample interval we must thus choose \( nb \) large to get a feasible FIR approximation.

The order problem is by no means unique for the FIR model structure. It can, as will be discussed after this, also be a problem in time series analysis.
3.3 Time Series

Sometimes it is difficult or virtually impossible to identify and measure the external stimuli to a system. Try for example to model the fluctuations on the stock market. As an output we can collect the overall share price at different time instances. But how on earth can we determine and measure the input signals relevant to such a system?

One way is to lump all external stimuli into an unknown disturbance term. Hence we build models only from the observed outputs. If we order the observations in time we have what commonly is referred to as a time series. Time series modeling is frequently used for phenomena which possess a "wave-like" structure, such as radio, radar and sound waves. They are also, as in the example above, often employed to explain fluctuations in various economic conditions. The literature on time series is by now quite extensive, see, e.g., the book by Anderson [1] for an early reference, and the work by Priestley [78], Hannan and Deistler [44], Ljung [61] and Söderström and Stoica [88] for more recent results.

Let us limit the time series modeling to the linear world and assume that the measured output \{y(t)\} is given as a random process

\[ y(t) = H^0(q)e(t), \]  

(3.13)

with \( H^0(q) \) being monic and \( \{e(t)\} \) a sequence of random variables with zero mean, variance \( \lambda \) and bounded fourth moments. The output spectral density then equals

\[ \Phi_y(e^{j\omega}) = \left| H^0(e^{j\omega}) \right|^2 \frac{\lambda}{2\pi}. \]  

(3.14)

Suppose further that the complex function \([H^0(z)]^{-1}, z \in \mathbb{C}\) is analytic in \(|z| > 1\) and continuous in \(|z| \geq 1\), which in particular means that the noise transfer function is stable. Then

\[ [H^0(z)]^{-1} = \sum_{k=0}^{\infty} \tilde{h}_k z^{-k}, \quad |z| \geq 1, \]  

(3.15)

where \( \tilde{h}_0 = 1 \). As for the input-output system we can truncate this expression at \( k = na \), yielding the \( na \)th order autoregressive (AR) approximation of (3.13):

\[ \tilde{H}^0(q)y(t) = e(t), \quad \tilde{H}^0(q) = 1 + \sum_{k=1}^{na} \tilde{h}_k q^{-k}. \]  

(3.16)

The crucial question is here how large \( na \) to choose in order to come up with a useful AR approximation. Again this is a matter of the rate of convergence of the infinite sum of Equation (3.15), and effectively this strongly resembles the problem that was discussed for the FIR model structure in the previous section. Compare Figures 3.1 and 3.2. The important difference is that the noise transfer function is expressed in an inverted manner. The factor causing slow convergence is therefore associated with zeros close to the unit circle, and not with poles as in the FIR case.

In contrast to the simple mapping between continuous and discrete poles, there is unfortunately no simple transformation showing how the zeros of a time continuous system are transformed by sampling. Still, it is possible to do some analysis under the assumption that the sampling period is sufficiently small. In the ordinary input-output
case this is carried out in Åström et al. [3], whereas the sampled noise model counterpart is detailed in Wahlberg [94]. The main result in the latter work is that the corresponding time discrete zeros approximately equal

\[ 1 + \tilde{z_j}T, \]  

(3.17)

when the sampling interval is small enough (\(\tilde{z_j}\) denotes a time continuous zero). To describe rapidly sampled time series data we must thus rely on AR models of high order. The order problem also appears when approximating narrow band processes. This is demonstrated by Wahlberg et al. in [93, 100].

All this motivates the search for alternative approximations which are less sensitive to the location of the dominating poles/zeros. Before approaching this matter in the next chapter let us for a moment dwell on earlier results and use of orthonormal functions.

### 3.4 Earlier Work and Trends

The idea of using orthonormal functions is by no means new. From a mathematical point of view it can, at least, be traced back to the early 19:th century, where such great mathematicians as Gauss, Laplace and Legendre made major theoretical contributions. Over the years the results and proofs have matured, and by now a rather uniform and elegant theory is available. Among the textbooks we may as a standard reference point to the one by Szegö [91], in which many fundamental concepts of orthogonal functions are given.

The many good properties of orthonormal functions were soon noticed by researchers dealing with various systems and signals. Perhaps the most important source of inspiration was the nowadays classical work on prediction and filter design by Lee and Wiener in the 1930's. They used time continuous so-called Laguerre functions for this purpose; consult Lee [59] for a survey. In 1954 Kautz showed how to orthogonalize a set of arbitrary time continuous exponential functions [52]. The computer development in the sixties then motivated the search for the analogous time discrete counterpart, which was solved by Young and Huggins [107] and by Broome [10]. It turns out that their framework is quite general; the discrete-time orthonormal Laguerre functions is, for example, readily obtained as a special case.
Besides the traditional use of orthonormal functions in filter synthesis the same framework was adopted in the field of system identification. For time continuous systems Mäkilä has recently applied orthonormal functions in order to approximately describe stable infinite-dimensional systems (e.g., systems with time delays) [70, 71]. Generalized orthonormal basis functions for discrete multivariate systems are discussed in Heuberger et al. [46, 47]. Therein they show how to construct orthonormal functions so that a series expansion of the system becomes simple, i.e., of low order. Returning to the SISO case, identification of discrete coefficients in finite length series expansions is from a statistical point of view considered in Wahlberg [97, 98], and for time series data in Wahlberg and Hannan [99, 100]. These results constitute the basis for the material in Chapter 4.

The rapidly growing interest in robust control ($H_\infty$ design and the like) has stressed the need for models that are delivered with error bounds in the frequency domain [103]. To accurately quantify this error Goberdensingh et al. [34], Goodwin et al. [36] and many others have suggested the use of orthonormal basis functions as one possible modeling approach. Further comments and ideas for the important problem of how to develop identification techniques better suited for robust control design can be found in the 1992 July issue of "IEEE-Transactions on Automatic Control". The robustness issue has also motivated further research in the area of adaptive control and controller tuning. As examples, discrete Laguerre functions are from this perspective studied in Dumont et al. [21] and in Gunnarsson and Wahlberg [43]. The use of other orthonormal filters in adaptive systems is investigated by Den Brinker [9] and by Finn et al. [24].
Discrete Laguerre and Kautz Functions

We are now prepared to deal with the earlier posed question concerning the choice of orthonormal functions. This issue occupies the opening of this chapter. Having determined suitable basis functions, we proceed to formulate the suggested structure as a linear regression (Section 4.2). Then, after pointing out some relationships to other linear model structures, we continue the chapter by discussing algorithm design in Sections 4.4–4.6. The usefulness of these algorithms is finally illustrated in Section 4.7. The last five sections constitute the main results of the chapter, while part of the material in Section 4.2 and most of the ideas in Section 4.1 can be found elsewhere (see the references below).

The overall aim in the algorithmic sections is to come up with orthonormal identification schemes based on hard statistical facts. However, we will in general not pursue such statistical analysis here, but instead point at adequate references. Nonetheless, we will occasionally convey and comment statistical considerations affecting the estimation computations.

4.1 Choice of Basis

The order problem is doubtless the most severe drawback of the FIR model structure. Earlier we claimed that the reason for this is that the delay operator has a too short memory, and thus a search for operators with longer memory is motivated. It turns out that well-damped and oscillatory systems ought to be treated separately. The former case is somewhat easier to analyze and hence treated first.

Well-damped Systems

In [95] and [98] Wahlberg points out the usefulness of generalizing the FIR network of Figure 3.1, by replacing the first block by a first order low-pass filter, and by replacing the unit shifts in the other blocks by first order all-pass filters (all having the same pole as the low-pass filter). With the particular choices of filters shown in Figure 4.1, a so-called Laguerre network is obtained. The name originates from that
Chapter 4. Discrete Laguerre and Kautz Functions

Figure 4.1: Laguerre network. The first block is a low-pass filter, while the grey blocks are first order all-pass filters.

\[ L_k(z, a) = \frac{\sqrt{1 - a^2}}{z - a} \left( \frac{1 - az}{z - a} \right)^{k-1}, \quad -1 < a < 1, \quad k \geq 1 \]  \hspace{1cm} (4.1)

is the $z$-transform of the so-called discrete Laguerre functions. Notice that by restricting the value of the filter parameter ($a$) stability is preserved. Furthermore, straightforward calculations show that \( \{L_k(z, a)\} \) forms an orthonormal set in \( L_2^2(-\pi, \pi) \). Also, observe that by letting $a = 0$ we arrive at a FIR model structure, which implies that it is just a special case of the more general Laguerre network.

Another quite important point is the following lemma due to Wahlberg [98].

**Lemma 4.1 (The Discrete Laguerre Expansion)**
If the true system $G^0(z)$ is strictly proper ($G^0(\infty) = 0$), analytic in $|z| > 1$, and continuous in $|z| \geq 1$, then there exists a sequence $\{c^0_k\}$ such that

\[ G^0(z) = \sum_{k=1}^{\infty} c^0_k L_k(z, a) \]  \hspace{1cm} (4.2)

provided that the filter parameter $a$ is chosen within $-1 < a < 1$.

**Proof:** See Wahlberg [98].

The lemma states that no matter the value of $a$ (corresponding to stable filters), the true system can be approximated with arbitrary accuracy using Laguerre functions. But how should $a$ be chosen so that the rate of convergence becomes acceptable in practice? The answer to this question is given by the following example.

**Example 4.1** Reconsider the first order sampled system of Example 3.2, i.e.,

\[ y(t) = \frac{1 - e^{-T}}{q - e^{-T}} u(t) = G(q)u(t). \]

1 After Edmond Laguerre (1834 – 1886), French mathematician, who did research work in geometry and the theory of infinite series, see, e.g., Kreyszig [56].
4.1 Choice of Basis

As a preliminary introduce the bilinear transformation

\[ w = \frac{q - a}{1 - aq}, \quad q = \frac{w + a}{1 + aw}, \quad -1 < a < 1, \]

which maps the unit disc onto the unit disc. Based on this transformation the transfer function can be rewritten as a power series:

\[
G(q) = G \left( \frac{w + a}{1 + aw} \right) = \frac{(1 - e^{-T})(1 + aw)}{(w + a) - e^{-T}(1 + aw)}
\]

\[
= \frac{(1 - e^{-T})}{(1 + aw)} \left[ w - \frac{(e^{-T} - a)}{1 - a e^{-T}} \right]
\]

\[
= \frac{(1 - e^{-T})(w^{-1} + a)}{(1 - a e^{-T})} \frac{1}{1 - \left( \frac{e^{-T} - a}{1 - a e^{-T}} \right) w^{-1}}
\]

\[
= \frac{(1 - e^{-T})(w^{-1} + a)}{(1 - a e^{-T})} \sum_{k=0}^{\infty} \left[ \frac{e^{-T} - a}{1 - a e^{-T}} \right] w^{-k}
\]

\[
= \frac{(1 - e^{-T}) \sqrt{1 - a^2}}{(1 - a e^{-T})} \sum_{k=1}^{\infty} \left[ \frac{e^{-T} - a}{1 - a e^{-T}} \right]^{k-1} \left[ \frac{1 - a q}{q - a} \right]^{k-1}
\]

By truncating this series at \( k = nb \) we obtain an error of magnitude

\[
O \left( \left[ \frac{e^{-T} - a}{1 - a e^{-T}} \right]^{nb} \right).
\] (4.3)

This example deserves some comments. The original system has a pole at \( e^{-T} \), and by choosing \( a = e^{-T} \) no real approximation is needed, which means that convergence is obtained in only one step. For a real-world system the location of the pole is, of course, unknown, but pilot step or impulse response tests (see the methods outlined by Fu and Dumont [27]) as well as nonparametric identification results can be used to gain insight into its approximate location. Thus, by taking \( a \) in the \( \text{a priori} \) given neighborhood of the true pole, the rate of convergence is improved. In other words, an appropriate choice of filter parameter reduces the number of parameters needed to obtain useful models. Notice also that any choice of \( a \), \( -1 < a < 1 \), implies that the error term tends to zero as \( nb \) tends to infinity, which confirms Lemma 4.1.

Let us at this point quantitatively and briefly investigate what orders to acquire to get a reasonable system approximation. From Ljung [61] we know that the limiting parameter estimate \( \theta^* \) as the number of data tends to infinity is given by

\[
\theta^* = \arg \min_{\theta} \int_{-\pi}^{\pi} |G(\omega) - G(e^{i\omega})|^2 \Phi_u(\omega) d\omega,
\] (4.4)

provided that the assumed additive unitary white noise and the input signals are independent of each other. Here \( \Phi_u(\omega) \) denotes the input spectral density. Theoretically this formulation means that the estimated transfer function is as close to the real transfer function as possible (\( L^2(-\pi, \pi) \) optimal). It turns out that for Laguerre structures this calculation can be done analytically by solving an ordinary Lyapunov equation, as is
described by Wahlberg in [96]. Leaving out further details about this algorithm we can investigate what happens when the model order is varied.

**Example 4.2** Consider the simple continuous-time system

\[ y(t) = \frac{1}{p + 0.2} u(t). \]

After sampling, with sampling interval \( T = 0.5 \), the system has a pole at \( z \approx 0.9 \). To comply with the requirements for Equation 4.4 to hold, the input and noise signals are both chosen to be independent unitary white noises. Some “optimal” Laguerre approximations of various order when \( a = 0.7 \) are given in Figure 4.2. As can be seen an order between 3 and 5, say 4, seems to be sufficient to describe the true system. Selecting this order and varying \( a \), Figure 4.3 clearly illustrates the importance of choosing a close to the true pole. As also can be seen in this figure a FIR model of order four cannot appropriately reflect the real system. However, the Laguerre model of the same order (\( a = 0.7 \)) shows an almost perfect fit. Its truncation error is of magnitude \( O(0.10) \), while the corresponding value of the FIR model is \( O(0.67) \). To retain the “quality” of the Laguerre model by using a FIR model its order must be increased substantially (an order of 23 gives \( O(0.10) \)).

For simple first order systems the Laguerre approach is thus far superior to FIR modeling, provided that \( a \) can be chosen near the true pole.

**Oscillatory Systems**

Since the Laguerre filters are of first order it is not that far-fetched to believe that slow convergence still is a problem for resonant systems.

**Example 4.3** Consider the system

\[ y(t) = \frac{1}{p^2 + 0.2p + 1} u(t), \]

i.e., a system with resonant frequency \( \omega_0 = 1 \) and damping \( \xi = 0.1 \). The corresponding sampled system \( (T = 0.5) \) has two complex conjugated poles located at \( q \approx 0.84 \pm i0.45 \). Clearly, it is impossible to choose \( a \) in the immediate vicinity of the true poles. In doing a somewhat crude visual optimization the best value of \( a \) seems to be around 0.6, which therefore is chosen. Varying \( nb \) and letting \( \Phi_a(e^{i\omega}) = 1 \), it is from the Bode diagram of Figure 4.4 obvious that low-order Laguerre models have great difficulties in approximating resonant systems. But as also is shown, it is even more troublesome to describe the system using low-order FIR models.

To remedy this deficiency Kautz [52] in the fifties showed how to orthogonalize an arbitrary set of continuous time exponential functions, with the ambition to improve analog filter design. Some ten years later, the discrete counterpart was investigated by Young together with Huggins [107] and by Broome [10]. The suggested representation, which is thoroughly summarized in Wahlberg [97], holds as a special case the discrete Laguerre functions. Based on Broome’s framework another interesting special case is when the Laguerre filter parameter is generalized to the complex situation. The result
4.1 Choice of Basis

Figure 4.2: Bode plots of some Laguerre models ($\alpha = 0.7$, $n\beta$ varies) for approximating a well-damped system with a pole at $z \approx 0.9$ (Example 4.2).

Figure 4.3: Bode plots of some Laguerre models ($\alpha$ varies, $n\beta = 4$) describing a well-damped system with a pole at $z \approx 0.9$ (Example 4.2).
is what we in the sequel shall call the discrete Kautz functions. Skipping the derivation (see the mentioned paper by Wahlberg) the Kautz functions are defined as

\[ \Psi_k(z, b, c) = \begin{cases} \frac{\sqrt{1 - z^2}(z - b)}{z^2 + b(c - 1)z - c} \left[ \frac{-cz^2 + b(c - 1)z + 1}{z^2 + b(c - 1)z - c} \right]^{(k-1)/2} & \text{k odd,} \\ \frac{\sqrt{(1 - z^2)(1 - b^2)}}{z^2 + b(c - 1)z - c} \left[ \frac{-cz^2 + b(c - 1)z + 1}{z^2 + b(c - 1)z - c} \right]^{(k-2)/2} & \text{k even,} \end{cases} \]

\[ -1 < b < 1, \quad -1 < c < 1, \quad k \geq 1. \] (4.5)

This time the delay operators of the FIR network are replaced by second order filters, but now the first block has band-pass characteristics, while the rest of the blocks are of all-pass type. From Figures 4.1 and 4.5 the kinship between the Laguerre and the Kautz structures is evident. Observe that the restrictions on the filter parameters also here guarantee stable models.

Broome [10] has shown that the Kautz basis functions \( \{\Psi_k(z, b, c)\} \) are orthonormal in \( L^2(-\pi, \pi) \). They are also complete, i.e., Lemma 4.1 can in a straightforward manner be generalized to hold for these functions as well [97]. Another analogy with the Laguerre case is the choice of \( b \) and \( c \), which, for fast convergence, should be chosen so that the roots of \( z^2 + b(c - 1)z - c \) as well as possible correspond to the resonant poles of the true system. In fact, if \( b \) and \( c \) could be determined to exactly correspond to the true poles, only one second order filter is required to model the system.
4.1 Choice of Basis

Example 4.4 Returning to the system of Example 4.3, Figure 4.6 shows two optimal (in $L^2(-\pi, \pi)$ sense) Kautz models with $b = 0.87$ and $c = -0.61$. These values leave us with models where the poles are believed to be located at $z = 0.70 \pm 0.35$, i.e., the Euclidian distance to the true poles is around 0.17. Despite this imperfection a twelfth order Kautz model is able to approximate the true system.

From this example and the discussion above we conclude that Kautz models are appropriate for describing systems that have a single resonant mode.

Systems with Several Dispersed Poles

Far from all processes can be modeled by first or second order structures, even though surprisingly many industrial systems can be reasonably described by a first, or perhaps second, order model with a time-delayed input. The extensive area of modeling mechanical vibrations is an extreme example where high model orders often must be used. It is not unusual that data from such an experiment show some twenty widely spread resonant frequencies, which means that models of about order forty should be expected. Since the aim is to deploy a general identification framework this case must also be covered.

The obvious way to support systems with several scattered poles is, of course, to use several Laguerre and/or Kautz structures in parallel, i.e.,

$$G^0(z) \approx \sum_{l=1}^{n_C} \sum_{k=1}^{nb_l} \tilde{b}_l^C(z, a_l) + \sum_{l=1}^{n^\Psi} \sum_{k=1}^{nb_l^\Psi} \tilde{b}_l^\Psi(z, b_l, c_l),$$

(4.6)

where superscript $L$ ($\Psi$) refers to the Laguerre (Kautz) part of the model, $n = n_C + n^\Psi$ is the total number of structures in parallel, and $nb_l$ denotes the number of filters for each structure. Notice that this extension results in models that no longer are orthonormal.
Chapter 4. Discrete Laguerre and Kautz Functions

Figure 4.6: Bode plots of two Kautz models (with filter parameters corresponding to assumed poles located at $z \approx 0.70 \pm i0.35$ and with $nb$ chosen to be 4 and 12, respectively) describing a resonant system with true poles at $z \approx 0.84 \pm i0.45$ (Example 4.4).

and that one could expect problems with systems that have poles close to each other. However, by using the earlier mentioned Kautz procedure it is possible to orthogonalize these sets of submodels too. Since the models then will become more complicated and not equally intuitive we will not adopt this approach here. In staying with Equation (4.6) one should for each of the parallel structures select filter parameters in the same manner as before, i.e., in the vicinity of the true poles. The following example illustrates the benefits with the suggested extension.

Example 4.5 The third order system

$$y(t) = \frac{0.1}{(p + 0.1)(p^2 + 0.2p + 1)} u(t)$$

has, after sampling ($T = 0.5$), poles at $z \approx 0.95$ and $z \approx 0.84 \pm i0.45$, i.e., three poles at the same distance from the origin. Inheriting input signals from the previous example, the system is modeled using single FIR, Laguerre and Kautz models of order eight. The parameter settings for the latter two models were chosen so that the distance to the true pole/poles where about 0.05. For comparison, and with parameter values as in the single cases, a total eighth order (4 Laguerre and 2 Kautz filters) combined Laguerre/Kautz model is also calculated. According to Figure 4.7 the ranking between the non-combined models is not apparent. However, it is clear that they do not compete in the same league as the combined model.

By using several Laguerre/Kautz filters in parallel it is thus possible to render reasonable low-order models for linear SISO systems of arbitrary order.
4.2 Linear Regression Formulation

Up to this point very little has been said about modeling time series data. Fortunately, as is demonstrated by Wahlberg and Hannan [99, 100], the Laguerre/Kautz filters can be successively used also in this situation. The natural approach is to extend the AR structure of Equation (3.16) to the broader Laguerre/Kautz class, i.e., to

$$H^0(z) \approx 1 + \sum_{i=1}^{n_c} \sum_{k=1}^{na_f} \bar{a}_{k,i}^f L_k(z, a_i) + \sum_{i=1}^{n_p} \sum_{k=1}^{na_p} \bar{a}_{k,i}^p \Psi_k(z, b_i, c_i),$$

(4.7)

with notation as in Equation (4.6). With the only visible difference being that $na_i$ here denotes the number of filters for each structure. In case there is an input signal present it is quite natural to combine Equations (4.6) and (4.7), and by allowing an optional direct term $b_0 u(t)$ we get

$$y(t) = -\left[ \sum_{i=1}^{n_c} \sum_{k=1}^{na_f} \bar{a}_{k,i}^f L_k(q, a_t) + \sum_{i=1}^{n_p} \sum_{k=1}^{na_p} \bar{a}_{k,i}^p \Psi_k(q, b_t, c_t) \right] y(t) + b_0 u(t) +$$

$$+ \left[ \sum_{i=1}^{n_c} \sum_{k=1}^{na_f} \bar{b}_{k,i}^f L_k(q, a_t) q^{-(nk^f-1)} + \sum_{i=1}^{n_p} \sum_{k=1}^{na_p} \bar{b}_{k,i}^p \Psi_k(q, b_t, c_t) q^{-(nk^p-1)} \right] u(t)$$

$$+ e(t),$$

(4.8)

which can be viewed as a generalization of the ARX model structure. Observe that this structure also is an extension in the sense that larger input time delays than one can be
taken care of through the \( nk_l \) terms (which all are assumed to be strictly positive).

For ease of notation let

\[
\theta = \begin{bmatrix}
\tilde{a}_{L,1} & \cdots & \tilde{a}_{L,n} \circ \cdots & \tilde{a}_{L,1} & \cdots & \tilde{b}_{L,1} \circ \cdots & \tilde{b}_{L,n} \circ \cdots & \tilde{b}_{L,1} \circ \cdots & \tilde{b}_{L,n} \circ \cdots \end{bmatrix}^T,
\]

and introduce the filtered signals

\[
\begin{align*}
y_{k,1}(t) &= L_k(q, a_l)y(t), \\
y_{k,\Psi}(t) &= \Psi_k(q, b_l, c_l)y(t), \\
u_{k,1}(t) &= L_k(q, a_l)u(t + 1 - nk_l), \\
u_{k,\Psi}(t) &= \Psi_k(q, b_l, c_l)u(t + 1 - nk_l).
\end{align*}
\]

By collecting these signals in a vector

\[
\varphi(t) = 
\begin{bmatrix}
-\tilde{y}_{1,1}(t) & \cdots & -\tilde{y}_{n_{\circ}, n_{\circ}}(t) & -y_{1,1}(t) & \cdots & -y_{n_{\circ}, n_{\circ}}(t) \\
u(t) & u_{1,1}(t) & \cdots & u_{n_{\circ}, n_{\circ}}(t) & u_{1,1}(t) & \cdots & u_{n_{\circ}, n_{\circ}}(t)
\end{bmatrix}^T,
\]

Structure (4.8) can be expressed as an ordinary linear regression:

\[
y(t|\theta) = \theta^T\varphi(t) + e(t). \quad (4.9)
\]

Recall that the least-squares estimate of \( \theta \) is defined as the vector \( \hat{\theta} \) that minimizes the quadratic loss function (from now on \( N = N + 1 - t_0 \))

\[
V_N(\theta, Z^N) = \frac{1}{N} \sum_{t=t_0}^N \frac{1}{2} [y(t) - \hat{y}(t|\theta)]^2. \quad (4.10)
\]

The minimizing solution to this optimization problem is given by the well-known normal equations

\[
R_N \hat{\theta}_N = f_N, \quad \Rightarrow \quad \hat{\theta}_N = R_N^{-1} f_N
\]

\[
R_N = \frac{1}{N} \sum_{t=t_0}^N \varphi(t)\varphi^T(t), \quad f_N = \frac{1}{N} \sum_{t=t_0}^N \varphi(t)y(t). \quad (4.11)
\]

provided that the inverse of \( R_N \) exists. Regardless of the quality of the collected data it must, at least, be required that every filter setting is unique, since otherwise the regression matrix would be singular. Furthermore, and unlike the ARX case where filter transients disappear in finitely many steps, it is here much harder to determine the first adequate sample when forming the regression matrix. For the Laguerre/Kautz structure filter transients will generically not vanish in finitely many samples, and hence \( t_0 \) must be carefully chosen so that these effects are negligible. This matter will be further dealt with in Section 4.5.

In the light of this framework it is clear that calculating a Laguerre/Kautz model consists of two separate and simple steps of low computational complexity, namely the recursive creation of regressors by prefiltering data, and as a second step solving the resulting least-squares problem. The prefiltering operation is a well-established method in identification to avoid differentiation of data, to, and this is important, reduce the bias from model mismatch [61], but also for shaping the bias distribution due to undermodeling [101]. Another advantage is that the particular choices of basis functions discussed
above correspond to all-pass filters, which are known to be robust to implement and use in numerical computations [97].

By only allowing single Laguerre models (without an autoregressive part) it can be shown, as a consequence of Lemma 3.1 in [98], that the estimated covariance matrix will have a $1 \times 1$ block Toeplitz structure, provided that the input signal is quasi-stationary\(^2\). More formally this means that the entries of the $nb \times nb$ regression matrix $R_N$ only depend on the difference between the indexes, i.e.,

$$R_{i,j} = R_\tau = \frac{1}{N} \sum_{t=\tau}^{N} \varphi(t-\tau)\varphi^T(t), \text{ for } \tau = i - j. \quad (4.12)$$

In Theorem 3.1 of [97] this result is generalized to the Kautz case, with the major difference being that the blocks then are of size $2 \times 2$.

Computationally, the Toeplitz structure gives some distinct advantages when solving the normal equations. Firstly, there is a large number of algorithms, such as the one outlined by Levinson (see Ljung [61] for further details), which utilizes this structure to reduce the number of necessary operations. Secondly, in the earlier mentioned papers by Wahlberg, the importance of having a well-conditioned least-squares problem is emphasized. When using higher-order ARX models for approximating complex systems this is typically not the case. By taking appropriate Laguerre or Kautz filter parameters the condition number of the resulting Toeplitz regression matrix is improved. Thus the estimates will be more reliable, since numerical problems caused by round-off errors etc. will be less severe now. Furthermore, if the input signal is sufficiently rich (persistently exciting) then we have a well-posed Laguerre or Kautz estimation problem. As a matter of fact, the Toeplitz form is the key concept Wahlberg uses in deriving error bounds, consistency results and simple frequency domain variance expressions for these model structures.

### 4.3 Links to Other Model Structures

Having outlined an algorithm for estimating Laguerre/Kautz models we now turn to discussing connections to other and more common model structures. As a starting point consider the ARMAX description

$$A(q)y(t) = B(q)u(t) + C(q)e(t),$$
$$C(q) = 1 + a_1q^{-1} + \ldots + a_{nc}q^{-nc}. \quad (4.13)$$

We have earlier stressed that direct estimation of the ARMAX coefficients requires iterative optimization methods (for which convergence to the global minimum is not guaranteed). However, an ARMAX model can be approximated by a high-order ARX model if $1/C(q)$ is approximated by a FIR filter $D(q)$. The resulting model structure is

$$D(q)A(q)y(t) = D(q)B(q)u(t) + e(t),$$
$$D(q) = 1 + d_1q^{-1} + \ldots + d_{nd}q^{-nd}. \quad (4.14)$$

\(^2\)The limits $\lim_{N \to \infty} \frac{1}{N} \sum_{t=\tau+1}^{N} u(t)u(t-\tau)$ exist for all $\tau \in \mathbb{N}$.
which after reparameterization (let $A^M(q) = D(q)A(q)$ be a polynomial of order $nd + na$
and $B^M(q) = D(q)B(q)$ a polynomial of order $nd + nb$) can be estimated using the
least-squares algorithm. One problem with this approach is that if the noise model $C(q)$
has zeros near the unit circle, an ARX model of high order is needed. To reduce this
problem in practice, one often applies prefilters before processing the data, i.e., the data
is transformed to
\[
y^F(t) = \frac{1}{C^*(q)} y(t), \quad u^F(t) = \frac{1}{C^*(q)} u(t).
\] (4.15)

This has the same effect as using an ARMAX model with a fixed $C$-polynomial, $C^*(q)$.
However, for this method to work well a priori knowledge about the time constants of
the system is needed (i.e. a “good” $C^*(q)$ is needed).

The general Laguerre/Kautz model structure of Equation (4.8) can somewhat im-
properly be compacted to
\[
\frac{A(q,a,b,c)}{C^*(q,a,b,c)} y(t) = \frac{B(q,a,b,c)}{C^*(q,a,b,c)} u(t) + e(t),
\] (4.16)

which means that it largely corresponds to an ARMAX description with a fixed $C$-
polynomial. A single Laguerre model has, for example, the fixed $C$-polynomial
$C^*(q) = (q - a)^n$. Despite this close kinship, Wahlberg [98] has shown that the Laguerre/Kautz
structure is much more numerically robust than the corresponding ARMAX ditto, which
typically is not well scaled for high model orders. Hence, one should avoid transforming
Laguerre/Kautz models into ARMAX descriptions, but instead employ special state
space structures, where the states are directly related to the network in question. We
return to this matter in the next section.

Although an as broad class as possible has been studied hitherto, two special model
subclasses are of particular interest. The first one is the output error model structure
\[
y(t) = \frac{B(q)}{F(q)} u(t) + e(t).
\] (4.17)

By simply removing the autoregressive part of Structure (4.16) the result coincides with
the following output error description
\[
y(t) = \frac{B(q,a,b,c)}{C^*(q,a,b,c)} u(t) + e(t).
\] (4.18)

One obvious advantage with this formulation is that the unknown parameters can be
consistently estimated using the least-squares algorithm, while this is not possible for
(4.17). Notice that this is the case even though the noise is colored, provided that the
input and noise signals are independent of each other.

Finally, if there is no measurable excitation signal the time series data can be modeled
by
\[
A(q)y(t) = C(q)e(t).
\] (4.19)

Estimating these ARMA parameters again requires complex optimization schemes, where
the estimation problem itself is not well-posed for large model orders [100]. To reduce
these difficulties the following Laguerre/Kautz time series structure can be applied
\[
\frac{A(q,a,b,c)}{C^*(q,a,b,c)} y(t) = e(t).
\] (4.20)
Conceptually, this approach is equivalent to AR modeling with a fixed pre-whitening filter $C^*(q, a, b, c)$.

### 4.4 State Space Formulations

To be able to use standard tools for analysis and design of control systems on Laguerre and Kautz models, this section is devoted to the development of the corresponding state space theory. To begin with, only SISO systems of low order (one or two poles) are treated. These results are then generalized to more complicated cases, namely systems with several dispersed poles, and finally, systems with several inputs.

#### The Laguerre Case

Consider the single Laguerre model structure (with $nk = 1$)

$$y(t) = -\sum_{k=1}^{na} \bar{a}_k \mathcal{L}_k(q, a)y(t) + \sum_{k=1}^{nb} \bar{b}_k \mathcal{L}_k(q, a)u(t) + e(t), \quad |a| < 1 \quad (4.21)$$

where $\bar{a}_k$ and $\bar{b}_k$ are weights delivered from the least-squares algorithm. This equation can be decomposed as

$$y^u(t) = \sum_{k=1}^{nb} \bar{b}_k \mathcal{L}_k(q, a)u(t), \quad (4.22)$$

$$y(t) = -\sum_{k=1}^{na} \bar{a}_k \mathcal{L}_k(q, a)y(t) + y^u(t) + e(t), \quad (4.23)$$

with Equation (4.22) chiefly corresponding to the Laguerre network of Figure 4.1. Since the filters are of first order it is natural to introduce new state variables after each such filter block. Numbering the states from the left to the right and highlighting that their excitation source is $u(t)$, the result is a number of simple and recursive difference equations:

$$\bar{x}_1^u(t + 1) = a \bar{x}_1^u(t) + \sqrt{1 - a^2}u(t)$$

$$\bar{x}_2^u(t + 1) = \bar{x}_1^u(t) - a \bar{x}_1^u(t + 1) + a \bar{x}_2^u(t)$$

$$\vdots$$

$$\bar{x}_{nb}^u(t + 1) = \bar{x}_{nb-1}^u(t) - a \bar{x}_{nb-1}^u(t + 1) + a \bar{x}_{nb}^u(t). \quad (4.24)$$

By taking $\bar{x}^u(t) = [\bar{x}_1^u(t) \bar{x}_2^u(t) \ldots \bar{x}_{nb}^u(t)]^T$, Equations (4.24) can be expressed almost as an ordinary state space model

$$\dot{\bar{x}}^u(t + 1) = \bar{F} \bar{x}^u(t) + \bar{G} u(t) \quad (4.25)$$

$$y^u(t) = \bar{H}^u \bar{x}^u(t), \quad (4.26)$$
where \( \dim(\tilde{E}) = \dim(\tilde{F}) = nb \times nb \) and
\[
\tilde{E} = \begin{bmatrix}
1 & 0 & \ldots & 0 & 0 \\
0 & 1 & \ldots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \ldots & 1 & 0 \\
0 & 0 & \ldots & 0 & 1
\end{bmatrix}, \quad \tilde{F} = \begin{bmatrix}
a & 0 & \ldots & 0 & 0 \\
1 & a & \ldots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \ldots & a & 0 \\
0 & 0 & \ldots & 1 & a
\end{bmatrix},
\tag{4.27}
\]
\[
\tilde{G} = \begin{bmatrix}
\sqrt{1-a^2} & 0 & \ldots & 0 & 0
\end{bmatrix}^T, \quad \tilde{H}^u = \begin{bmatrix}
b_1 & b_2 & \ldots & b_{nb-1} & b_{nb}
\end{bmatrix}.
\]

Multiplying (4.25) with \( \tilde{E}^{-1} \) gives
\[
\tilde{a}^u(t+1) = \tilde{F} \tilde{a}^u(t) + \tilde{G} u(t)
\]
\[
y^u(t) = \tilde{H}^u \tilde{a}^u(t).
\tag{4.28}
\]

Notice that this operation always is valid since \( \det(\tilde{E}^{-1}) \) is independent of \( a \) and equal to 1.

Applying the same method to Expression (4.23) but now with input signal \(-y(t)\) yields
\[
\tilde{x}^y(t+1) = \tilde{F} \tilde{x}^y(t) - \tilde{G} y(t)
\]
\[
y(t) = \tilde{H}^y \tilde{x}^y(t) + y^u(t) + e(t),
\tag{4.29}
\]

with
\[
\tilde{H}^y = \begin{bmatrix}
\tilde{a}_1 & \tilde{a}_2 & \ldots & \tilde{a}_{na-1} & \tilde{a}_{na}
\end{bmatrix}.
\tag{4.30}
\]

Together Equations (4.28) and (4.29) form a state space description of dimension \( na + nb \).

However, this realization is not minimal. A minimal realization, with \( \max(na, nb) \) states, can readily be constructed by using the so-called dual forms of Equations (4.28)-(4.29) (see, e.g., Kailath [51]), i.e.,
\[
x^u(t+1) = \tilde{F}^T x^u(t) + \tilde{H}^u^T u(t)
\]
\[
y^u(t) = \tilde{G}^T x^u(t),
\tag{4.31}
\]
\[
x^y(t+1) = \tilde{F}^T x^y(t) - \tilde{H}^y^T y(t)
\]
\[
y^u(t) = \tilde{G}^T [x^y(t) + x^u(t)] + e(t).
\tag{4.32}
\]

Observe that this classical transformation only works when the input and output signals are scalars. After some straightforward substitutions Equations (4.32) can be rewritten as
\[
x^y(t+1) = \tilde{F}^T x^y(t) - \tilde{H}^y^T \tilde{G}^T [x^y(t) + x^u(t)] - \tilde{H}^y^T e(t)
\]
\[
y^u(t) = \tilde{G}^T [x^y(t) + x^u(t)] + e(t).
\tag{4.33}
\]

By introducing new state variables \( x(t) = x^y(t) + x^u(t) \) we arrive at the desired state space structure
\[
x(t+1) = \tilde{F}^T x(t) + \tilde{H}^u^T u(t) - \tilde{H}^y^T e(t)
\]
\[
y(t) = \tilde{G}^T x(t) + e(t).
\tag{4.34}
\]
The Kautz Case

The course of action when deriving a single Kautz state space structure is fairly similar to the one used in the Laguerre case. Yet the procedure is more involved and temporary state variables will be introduced. Recall the definition of the Kautz model structure (with \( nk = 1 \))

\[
y(t) = - \sum_{k=1}^{na} \bar{a}_k \bar{\Psi}_k(q, b, c)y(t) + \sum_{k=1}^{nb} \bar{b}_k \bar{\Psi}_k(q, b, c)u(t) + e(t), \tag{4.35}
\]

where for stability reasons \(-1 < b < 1\) and \(-1 < c < 1\). In analogy to the Laguerre case, we can separate this equation into two parts:

\[
y^u(t) = \sum_{k=1}^{nb} \bar{b}_k \bar{\Psi}_k(q, b, c)u(t), \tag{4.36}
\]

\[
y(t) = - \sum_{k=1}^{na} \bar{a}_k \bar{\Psi}_k(q, b, c)y(t) + y^u(t) + e(t). \tag{4.37}
\]

We would also like to introduce states so that the recursions of Figure 4.5 can be utilized in the computations. Unfortunately, it is somewhat tricky to directly express the desired states, but as a starting point we can introduce temporary state variables in pairs:

\[
\begin{bmatrix}
  \bar{x}^{u}_{2k-1}(t) \\
  \bar{x}^{u}_{2k}(t)
\end{bmatrix}
= \begin{bmatrix}
  q \\
  q^2 + b(c-1)q - c
\end{bmatrix}
\begin{bmatrix}
  u(t) \\
  y^u(t)
\end{bmatrix}
\]

\[
\begin{bmatrix}
  -c q^2 + b(c-1)q + 1 \\
  q^2 + b(c-1)q - c
\end{bmatrix}
\begin{bmatrix}
  \bar{x}^{u}_{2k-3}(t) \\
  \bar{x}^{u}_{2k-1}(t)
\end{bmatrix}
\]

\[
k > 1.
\]

The corresponding difference equations are

\[
\begin{align*}
\bar{x}^u_1(t+1) &= -b(c-1)\bar{x}^u_1(t) + c\bar{x}^u_2(t) + u(t) \\
\bar{x}^u_2(t+1) &= \bar{x}^u_1(t) \\
\bar{x}^u_3(t+1) &= b(c-1)\bar{x}^u_1(t) - c\bar{x}^u_2(t) + \bar{x}^u_3(t) - b(c-1)\bar{x}^u_3(t) + c\bar{x}^u_4(t) \\
\bar{x}^u_4(t+1) &= \bar{x}^u_3(t) \\
&\vdots \\
\bar{x}^u_{nb-1}(t+1) &= b(c-1)\bar{x}^u_{nb-3}(t) - c\bar{x}^u_{nb-3}(t+1) + \bar{x}^u_{nb-2}(t) - b(c-1)\bar{x}^u_{nb-2}(t) + c\bar{x}^u_{nb}(t) \\
\bar{x}^u_{nb}(t+1) &= \bar{x}^u_{nb-1}(t)
\end{align*}
\]

Vectorizing, \( \bar{x}^u(t) = [\bar{x}^u_1(t) \bar{x}^u_2(t) \ldots \bar{x}^u_{nb}(t)]^T \), and rearranging the terms, Equations (4.39) can be written as

\[
\begin{align*}
\bar{E}\bar{x}^u(t+1) &= \bar{F}\bar{x}^u(t) + \bar{G}u(t),
\end{align*}
\]

(4.40)
Chapter 4. Discrete Laguerre and Kautz Functions

where \( \dim(\tilde{E}) = \dim(\tilde{F}) = nb \times nb \) (assumed to be even) and

\[
\tilde{E} = \begin{bmatrix}
I & 0 & \cdots & 0 \\
\tilde{E}_a & I & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & I \\
0 & 0 & \cdots & \tilde{E}_a & I
\end{bmatrix}, \quad \tilde{F} = \begin{bmatrix}
\tilde{F}_\beta & 0 & \cdots & 0 \\
\tilde{F}_a & \tilde{F}_\beta & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & \tilde{F}_a & \tilde{F}_\beta
\end{bmatrix}, \quad \tilde{G} = \begin{bmatrix}
1 \\
0 \\
\vdots \\
0
\end{bmatrix}
\] (4.41)

with \( I, \tilde{E}_a, \tilde{F}_a \) and \( \tilde{F}_\beta \) being the \( 2 \times 2 \) matrices

\[
I = \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}, \quad \tilde{E}_a = \begin{bmatrix}
c & 0 \\
0 & 0
\end{bmatrix},
\]

\[
\tilde{F}_a = \begin{bmatrix}
b(c-1) & 1 \\
0 & 0
\end{bmatrix}, \quad \tilde{F}_\beta = \begin{bmatrix}
-b(c-1) & c \\
1 & 0
\end{bmatrix}.
\] (4.42)

Since \( \det(\tilde{E}) = 1 \) we can multiply Equation (4.40) from the left with the inverse of \( \tilde{E} \). By also changing state variables it is possible to elicit the desired structure. The operation is completed by letting \( \tilde{x}^u(t) = T \tilde{x}^u(t) \), which gives

\[
\tilde{x}^u(t+1) = T \tilde{E}^{-1} \tilde{F} \tilde{F}^{-1} \tilde{x}^u(t) + T \tilde{E}^{-1} \tilde{G} u(t)
\]

\[
y^u(t) = \tilde{H} u \tilde{x}^u(t),
\] (4.43)

with \( T \) being the block diagonal transformation matrix

\[
T = \begin{bmatrix}
T_\alpha & 0 & \cdots & 0 \\
0 & T_\alpha & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & T_\alpha \\
0 & 0 & \cdots & 0 & T_\alpha
\end{bmatrix}, \quad T_\alpha = \begin{bmatrix}
\sqrt{1-c^2} & -b \sqrt{1-c^2} \\
0 & \sqrt{(1-b^2)(1-c^2)}
\end{bmatrix},
\] (4.44)

and

\[
\tilde{H} u = \begin{bmatrix}
\tilde{b}_1 & \tilde{b}_2 & \cdots & \tilde{b}_{nb} & \tilde{b}_{nb}
\end{bmatrix}.
\] (4.45)

Notice that \( \det(T) = \left[(1-c^2)\sqrt{1-b^2}\right]^{\frac{\alpha}{\beta}} \), which implies that \( T^{-1} \) exists if and only if \( |b| \neq 1 \) and \( |c| \neq 1 \). Thus we have a state space description for the moving average part, which easily can be extended in exactly the same way as in the Laguerre situation. In fact, the discussion starting just before Equation (4.29) can, with the same notation, be equally applied for the Kautz case. The interpretation of the states and the associated system matrices must, of course, be changed so that they agree with the expressions above.

The derived Laguerre and Kautz state space representations are both limited in that the time delay is fixed to one. If \( nk = 0 \), then the output should be complemented with a factor \( b_0 u(t) \), but some additional modifications (see Result 4.1 below) must then also
be conducted. The standard trick for handling larger delays, \( nk > 1 \), is to extend the state vector with the old inputs as

\[
x^d(t) = \begin{bmatrix} x_1^d(t) \\ \vdots \\ x_{nk-1}^d(t) \\ u(t-1) \\ \vdots \\ u(t-nk+1) \end{bmatrix}
\]  

(4.46)

Taking this into account we get the following state space result.

**Result 4.1** **The Single Laguerre/Kautz State Space Structure.**

\( nk = 0 \):  
\[
x(t + 1) = Fx(t) + [G + KD] u(t) + Ke(t)
\]  
\[
y(t) = Hx(t) + Du(t) + e(t),
\]

\( nk = 1 \):  
\[
x(t + 1) = Fx(t) + Gu(t) + Ke(t)
\]  
\[
y(t) = Hx(t) + e(t),
\]

\( nk > 1 \):  
\[
\begin{bmatrix} x(t+1) \\ x^d(t+1) \end{bmatrix} = \begin{bmatrix} F & G & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & 1 \\ 0 & 0 & \ldots & 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x^d(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u(t) + \begin{bmatrix} K \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} e(t)
\]

\[
y(t) = \begin{bmatrix} H \\ 0 \\ 0 \\ \ldots \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x^d(t) \end{bmatrix} + e(t),
\]

where \( F = \tilde{F}^T - \tilde{H}^u \tilde{G}^T, \ G = \tilde{H}^u, \ K = -\tilde{H}^v, \ H = \tilde{G}^T \) and \( D = \tilde{b}_0 \).

These matrices are given by

- Expressions (4.27), (4.28) and (4.30) in the Laguerre case, and by
- Expressions (4.30) and (4.41)-(4.45) in the Kautz case.

**The Combined Case**

Recall the definition of the combined Laguerre/Kautz model structure given by Equation (4.8). The output \( y(t) \) can be viewed as a sum of outputs from single Laguerre and Kautz models. The extended combined state vector \( x^c(t) \), which also contains states for handling arbitrary \( nk \), is then simply one where all the states from each single model have been added. With notational abuse, let the state space structure for each such system be described by the 5-tuple \( < F^c_i, G^c_i, K^c_i, H^c_i, D > \), where \( c \) either refers to a simple Laguerre or a simple Kautz structure. The individual vectors and matrices are here calculated according to the previous two subsections, but generalized to cover all different time delays \( (nk \geq 0) \). Notice that only one of the simple models are allowed to


have a direct term \((nk = 0)\). The structure of a combined Laguerre/Kautz state space description is summarized in Result 4.2.

**Result 4.2** THE COMBINED LAGUERRE/KAUTZ STATE SPACE STRUCTURE.

\[
x^c(t + 1) = \begin{bmatrix} F_{11}^C & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & F_{n_1}^C & 0 \\ 0 & \cdots & \cdots & F_{n_2}^C \\ \end{bmatrix} x^c(t) + \begin{bmatrix} G_{11}^C \\ \vdots \\ G_{1n}^C \\ G_{n1}^C \\ \end{bmatrix} u(t) + \begin{bmatrix} H_{11}^C \\ \vdots \\ H_{1n}^C \\ H_{n1}^C \\ \end{bmatrix} e(t)
\]

\[
y(t) = \begin{bmatrix} H_{11}^C & \cdots & H_{1,2}^C & \cdots & H_{1,n}^C \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ H_{n,1}^C & \cdots & H_{n,2}^C & \cdots & H_{n,n}^C \\ \end{bmatrix} x^c(t) + D u(t) + e(t),
\]

where the single Laguerre and Kautz matrices are given by Result 4.1.

**MISO Systems**

From a control engineering viewpoint, the most relevant of the previously discussed structures is the one having an output error form. Let us therefore constrain the modeling to such a structure, but additionally allow several \((m)\) input signals. Thus we get the following model description

\[
y(t) = \sum_{i=1}^{m} b_{0,i} u_i(t) + \sum_{i=1}^{m} \left[ \sum_{l=1}^{n_C} \sum_{k=1}^{n_C} \tilde{b}_{l,i}^C \mathcal{C}_k(q, a_{l,i}) q^{(1-nk^C)} \right] u_i(t)
\]

\[
+ \sum_{i=1}^{m} \left[ \sum_{l=1}^{n_\psi} \sum_{k=1}^{n_\psi} \tilde{b}_{l,i}^\psi \mathcal{Q}_k(q, b_{l,i}, c_{l,i}) q^{(1-nk^\psi)} \right] u_i(t) + e(t).
\]

Notice that this still is a linear regression, and hence the parameters can be estimated using the least-squares algorithm. A state space model for Equation (4.47) can be derived as follows. First, let every simple Laguerre structure be denoted

\[
\tilde{x}_{l,i}^C(t + 1) = \tilde{F}_{l,i}^C \tilde{x}_{l,i}^C(t) + \tilde{G}_{l,i}^C u_i(t)
\]

\[
y_{l,i}^C(t) = \tilde{H}_{l,i}^C \tilde{x}_{l,i}^C(t) + D_i u_i(t),
\]

where each \(\tilde{F}_{l,i}^C, \tilde{G}_{l,i}^C\) and \(\tilde{H}_{l,i}^C\) is given by Equation (4.28), but again extended so that any time delay is handled. Notice that these matrices differ from the ones used before, since no transformation to cope with the autoregressive part is necessary here. Also observe that the first two matrices are calculated given \(a_{l,i}\), whereas the estimated parameters are located in \(\tilde{H}_{l,i}^C\), and possibly in \(D_i = \tilde{b}_{0,i}\). Next, do the same for each Kautz structure, but then with matrices in accordance to Equation (4.43).

Secondly, let \(\tilde{x}_{l}^C(t) = [\tilde{x}_{l,1}^C(t)^T \ldots \tilde{x}_{n_1}^C(t)^T \tilde{x}_{1,1}^\psi(t)^T \ldots \tilde{x}_{n_\psi}^\psi(t)^T]^T\), which gives a
4.5 Effects of Initial Conditions

Recall the least-squares parameter estimate

\[ \hat{\theta}_N = R_N^{-1} f_N = \left[ \sum_{t=0}^{N} \varphi(t) \varphi^T(t) \right]^{-1} \left[ \sum_{t=0}^{N} \varphi(t) y(t) \right]. \]  

To avoid that transients affect this estimate negatively it is necessary to correctly initialize the model structure. From a filtering point of view this relates to removing the effects of unknown initial conditions \( \mathbf{x}(0) \) in the predictor model. In case the regression matrix \( \varphi(t) \) is formed using the standard basis \( \{q^{-k}\} \) it is well-known that transients wear off in finitely many samples. This is due to the fact that the delay operator has a memory of only one sampling step. For example, for the FIR model structure these
effects completely vanish in \( nb+1 \) steps, and thus it is reasonable to start the summation at \( t^0 = nb + 1 \).

The filters in the Laguerre/Kautz predictor model, however, generically show infinite impulse responses, and consequently filter transients affect the whole data set. To decrease this problem two different approaches can be followed.

1. Since the system is assumed to be asymptotically stable the impulse response decays monotonically to zero as \( t \to \infty \). With this in mind it is natural to skip a number of initial samples and choose \( t^0 \) at a point where the transient contribution is considered to be "small". In determining \( t^0 \) such factors as filter coefficients and the length of the estimation data set must be taken into account. Clearly, this method does not lead to more calculations, but having small data sets one cannot afford wasting data. It can then be worthwhile to also estimate the initial state.

2. Estimating \( x(0) \) normally requires non-quadratic optimization algorithms, since the linear regression structure is lost as products of unknown parameters appear in the output \([97]\). An important exception is the SISO output error structure of Equation (4.49), i.e., (with \( i = 1 \) and \( C \) suppressed)

\[
\begin{align*}
\bar{x}(t+1) &= \tilde{F}\bar{x}(t) + \tilde{G}u(t) \\
y(t) &= \tilde{H}\bar{x}(t) + Du(t) + e(t),
\end{align*}
\]  

(4.51)

where all the unknown parameters are located in \( \tilde{H} \), and possibly in \( D \). The dual form of this system is

\[
\begin{align*}
x(t+1) &= \tilde{F}^T\bar{x}(t) + \tilde{H}^T u(t) \\
y(t) &= \tilde{G}^T\bar{x}(t) + Du(t) + e(t),
\end{align*}
\]  

(4.52)

and solving for \( y(t) \) gives

\[
y(t) = \tilde{G}^T\tilde{F}^T\bar{x}(0) + \varphi^T(t)\theta + e(t),
\]  

(4.53)

with \( \varphi(t) \) and \( \theta \) defined in Section 4.2 but with the autoregressive part left out. As \( \tilde{G}^T\tilde{F}^T \) is known beforehand the unknown parameters \( (x(0) \text{ and } \theta) \) enter this equation linearly. Therefore, the regression and parameter vectors can be extended as

\[
\Phi(t) = \begin{bmatrix} (\tilde{G}^T\tilde{F}^T)^T \\ \varphi(t) \end{bmatrix}, \quad \Theta = \begin{bmatrix} x(0) \\ \theta \end{bmatrix},
\]  

(4.54)

and a new linear regression model is obtained:

\[
y(t|\Theta) = \Theta^T\Phi(t) + e(t),
\]  

(4.55)

where \( \Theta \) readily can be estimated using the least-squares algorithm. This method should only be used for small data sets, since the computations will be considerably slower and, which is perhaps more severe, the regression matrix will be ill-conditioned for a large \( N \), as \( \tilde{G}^T\tilde{F}^T \to 0 \) as \( t \to \infty \).

Finally, we should point out that transient effects are negligible for large data sets. It is then reasonable to assume a zero initial state and start the summation at \( t^0 = 1 \).
4.6 Searching for the Filter Coefficients

The number of estimated parameters needed to acquire a good enough model crucially depends on the choice of filter coefficients. To arrive at a low-order model one appealing idea is to also estimate the filter settings. It is then, however, impossible to analytically find the coefficients that minimize the loss function, which in practice means that iterative search strategies must be employed. Consult the excellent book [18] by Dennis and Schnabel as a standard reference for such algorithms.

By introducing \( \eta \) as a vector of length \( d \) containing the filter coefficients

\[
\eta = \begin{bmatrix} a_1 & \ldots & a_n \epsilon & b_1 & \ldots & b_n \epsilon & c_1 & \ldots & c_n \epsilon \end{bmatrix}^T,
\]

\( \hat{\Theta}_N(\eta) \) can easily be calculated for a fixed \( \eta \) using prefiltering and the least-squares method (for ease of notation we have here assumed a SISO system as the MISO case is treated in almost the same way). If we stick to a quadratic criterion the problem of finding the filter coefficients can be expressed as an optimization problem:

\[
\hat{\eta}_N = \arg \min_\eta V_N(\eta, Z^N) = \arg \min_\eta \frac{1}{N} \sum_{t=0}^{N-1} \frac{1}{2} [y(t) - \hat{\Theta}_N^T(\eta) \Phi(t, \eta)]^2,
\]

which in numerical analysis sometimes is referred to as a "nonlinear least-squares problem". Denoting the prediction error as

\[
\varepsilon(t, \eta, Z^N) = y(t) - \hat{\Theta}_N^T(\eta) \Phi(t, \eta),
\]

the criterion has the gradient (a vector of length \( d \))

\[
\frac{dV_N(\eta, Z^N)}{d\eta} = V'_N(\eta, Z^N) = -\frac{1}{N} \sum_{t=0}^{N-1} \nabla(t, \eta, Z^N) \varepsilon(t, \eta, Z^N).
\]

The components of \( \nabla(\cdot) \) can be numerically approximated by the Euler equations

\[
\nabla_i(t, \eta, Z^N) = \frac{\partial \varepsilon(t, \eta, Z^N)}{\partial \eta_i} \approx \frac{\varepsilon(t, \eta + \delta \varepsilon_i, Z^N) - \varepsilon(t, \eta, Z^N)}{\delta},
\]

with \( \varepsilon_i \) being a column vector with a one in the \( i \)th position and zeros elsewhere and \( \delta \) being a small scalar perturbation. Notice that in calculating the gradient \( d + 1 \) least-squares problems of the original type are solved.

Following the discussion in Ljung [61] a general family of iterative search routines is given by

\[
\eta_N^{(k+1)} = \eta_N^{(k)} - \mu_N^{(k)} [H_N^{(k)}]^{-1} V'_N(\eta_N^{(k)}, Z^N),
\]

where \( \eta_N^{(k)} \) denotes the \( k \)th iterate. Here \( H_N^{(k)} \) is a \( d \times d \) matrix used to modify the search direction and \( \mu_N^{(k)} \) is a step size whose purpose is discussed later on. In words, we search for the new iterate along a modified gradient of the criterion function. Taking \( H_N^{(k)} = V''_N(\eta_N^{(k)}, Z^N) \) and \( \mu_N^{(k)} = 1 \) results in a pure Newton-Raphson algorithm. The calculation of the second order derivative, the Hessian, is however rather costly. The Hessian is therefore often approximated by

\[
V''_N(\eta_N^{(k)}, Z^N) \approx \sum_{t=0}^{N-1} \nabla(t, \eta_N^{(k)}, Z^N) \nabla^T(t, \eta_N^{(k)}, Z^N),
\]
which gives an ordinary \textit{Gauss-Newton algorithm}. This expression is far more tangible than the one from the pure Newton-Raphson algorithm, since the so-called Gauss-Newton direction \( g_N^{(k)}(\eta_N^{(k)}, Z^N) \) readily can be calculated as (skipping the arguments)

\[
g_N^{(k)}(\cdot) = \left[ \sum_{t=t_0}^{N} \nabla(\cdot)\nabla^{T}(\cdot) \right]^{-1} \sum_{t=t_0}^{N} \nabla(\cdot)z(\cdot). \tag{4.63}
\]

i.e., it is obtained through a simple least-squares computation.

The \((k+1)\)th iterate is thus given by

\[
\eta_N^{(k+1)} = \eta_N^{(k)} - \mu_N^{(k)} g_N^{(k)}(\eta_N^{(k)}, Z^N).
\]

The step size \(\mu_N^{(k)}\) is here adjusted\(^3\) so that all filters remain stable and so that the loss function is strictly less than it was in the previous iteration. In other words, a constrained minimization problem is solved, which implies that the algorithm becomes more cautious. One simple adjustment mechanism is to start with a step size of 1, and halve it until the mentioned two requirements are fulfilled. If, say, after \(n_\mu = 11\) steps \((\mu_N^{(k)} = 2^{-10})\) these requirements are still not met the algorithm is terminated with \(\eta_N^{(k)}\) as the solution. The algorithm also terminates and returns \(\eta_N^{(k+1)}\) in case \(|g_N^{(k)}(\eta_N^{(k)}, Z^N)|\) is sufficiently small (should be zero if a minimum is found), or when a certain maximum number of iterations \((n_{\text{max}})\) has been carried out.

Lastly, one should be aware of some properties of this algorithm. The number of least-squares problems solved can be quite large: in the best case \((d+1)\) and in the worst \(n_{\text{max}}(d+n_\mu)\). Furthermore, the scheme does not always find the global minimum, but instead a local one. Yet we are in a better than normal position since with “good” initial parameter values (the \textit{a priori} knowledge) the algorithm stands a better chance to find the searched-for minimum.

\subsection*{4.7 Examples}

In this last section we will perform a more thorough simulation study. In particular properties such as sensitivity to the SNR and effects of a non-zero initial condition are investigated for first and second order systems. As before the calculations are carried out in MATLAB \cite{72}, by taking advantage of the System Identification ToolBox (SITB) \cite{62}.

\begin{example}
Regard the simple first-order system

\[
y(t) = \frac{0.02}{q - 0.98} u(t), \tag{4.65}
\]

corrupted by additive white noise with zero mean and variance \(\lambda\). The input signal data set is chosen to be of size 500, each sample shifting randomly (normal distribution) between \pm 1. Knowing the model class as well as the model order we can estimate first-order output error and Laguerre models. Since OE models are estimated iteratively it seems fair to invoke the nonlinear search scheme also in the Laguerre case. To further achieve a just comparison both algorithms were started from the same spot, whereupon they performed at most 15 iterations. The estimation results given 5000 different input and noise realizations are summarized in Table 4.1. The

\cite{Dennis and Schnabel [18] suggest the term \textit{damped} for algorithms when an adjusted step size is used.\end{example}
4.7 Examples

<table>
<thead>
<tr>
<th>Model</th>
<th>λ</th>
<th>Pole_{init}</th>
<th>Failure (%)</th>
<th>Found (%)</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>OE</td>
<td>0.01</td>
<td>0.20</td>
<td>3.4</td>
<td>90.3</td>
<td>0.276-0.992</td>
</tr>
<tr>
<td>OE</td>
<td>0.60</td>
<td>0.00</td>
<td>6.6</td>
<td>86.6</td>
<td>0.606-0.992</td>
</tr>
<tr>
<td>OE</td>
<td>0.90</td>
<td>0.00</td>
<td>0.0</td>
<td>93.9</td>
<td>0.629-0.992</td>
</tr>
<tr>
<td>Laguerre</td>
<td>0.20</td>
<td>0.00</td>
<td>0.0</td>
<td>99.9</td>
<td>0.967-0.990</td>
</tr>
<tr>
<td>Laguerre</td>
<td>0.60</td>
<td>0.00</td>
<td>0.0</td>
<td>99.9</td>
<td>0.967-0.990</td>
</tr>
<tr>
<td>Laguerre</td>
<td>0.90</td>
<td>0.00</td>
<td>0.0</td>
<td>99.9</td>
<td>0.967-0.990</td>
</tr>
<tr>
<td>OE</td>
<td>0.03</td>
<td>0.20</td>
<td>7.2</td>
<td>70.0</td>
<td>-0.969-0.991</td>
</tr>
<tr>
<td>OE</td>
<td>0.60</td>
<td>0.06</td>
<td>5.3</td>
<td>71.4</td>
<td>-0.985-0.991</td>
</tr>
<tr>
<td>OE</td>
<td>0.90</td>
<td>0.07</td>
<td>0.7</td>
<td>75.5</td>
<td>-0.282-0.991</td>
</tr>
<tr>
<td>Laguerre</td>
<td>0.20</td>
<td>0.06</td>
<td>0.0</td>
<td>97.2</td>
<td>-0.959-0.995</td>
</tr>
<tr>
<td>Laguerre</td>
<td>0.60</td>
<td>0.00</td>
<td>0.0</td>
<td>97.8</td>
<td>0.950-0.995</td>
</tr>
<tr>
<td>Laguerre</td>
<td>0.90</td>
<td>0.00</td>
<td>0.0</td>
<td>97.8</td>
<td>0.950-0.995</td>
</tr>
<tr>
<td>OE</td>
<td>0.06</td>
<td>0.20</td>
<td>17.2</td>
<td>51.6</td>
<td>-0.983-0.995</td>
</tr>
<tr>
<td>OE</td>
<td>0.60</td>
<td>0.12</td>
<td>17.2</td>
<td>54.0</td>
<td>-0.988-0.995</td>
</tr>
<tr>
<td>OE</td>
<td>0.90</td>
<td>0.52</td>
<td>17.2</td>
<td>58.9</td>
<td>-0.952-0.995</td>
</tr>
<tr>
<td>Laguerre</td>
<td>0.20</td>
<td>0.43</td>
<td>17.2</td>
<td>88.3</td>
<td>-0.986-0.998</td>
</tr>
<tr>
<td>Laguerre</td>
<td>0.60</td>
<td>0.03</td>
<td>17.2</td>
<td>92.0</td>
<td>-0.433-0.998</td>
</tr>
<tr>
<td>Laguerre</td>
<td>0.90</td>
<td>0.00</td>
<td>17.2</td>
<td>92.3</td>
<td>0.800-0.998</td>
</tr>
</tbody>
</table>

Table 4.1: OE and Laguerre identification experiments on a first-order system. Here, "λ" is the variance of the additive noise. "Pole_{init}" expresses the search starting point. "Failure" shows the percentage of models that have an estimated pole below 0.9, "Found" the percentage of models with an estimated pole in between 0.97 and 0.99 and "Interval" denotes an interval in which all 5000 estimates lie (see Example 4.6).

column named "Failure" shows the percentage of models having an estimated pole below 0.9, i.e., the true pole is clearly not recognized. "Found" states the percentage of models with an estimated pole in between 0.97 and 0.99, i.e., an interval in which we regard the true pole as identified. "Interval" lastly, denotes an interval in which all 5000 estimates lie.

Table 4.1 indicates some interesting points. First of all, the success of the OE estimation procedure strongly depends on the signal to noise ratio. A low SNR substantially increases the risk for getting stuck in a local minimum, a situation which is not uncommon for output error structures (cf. Ljung [61J]). This risk is obviously reduced using the Laguerre concept, and the better value on a is less the risk. Furthermore, even though a is poorly chosen the Laguerre approach appears to find the true pole more often than the corresponding OE procedure. Thus, the former estimator seems to be less sensitive to the iteration starting point than the latter one.

Let us finally take a closer look at the impact of a non-zero initial condition on the estimate.

Example 4.7 Consider the input-output description

$$y(t) = \frac{q - 0.8}{q^2 - 1.7q + 0.5} u(t),$$  \hspace{1cm} (4.66)

which has poles located at $q \approx 0.85 \pm i0.42$. By converting the system to, e.g., observable canonical state space form, we can easily extend it to reflect a non-zero initial condition $x(0)$. 

Table 4.2: Effects of a non-zero initial condition. \( N \) is the size of the data set, then follows the true initial state, while the last two columns denote the estimated zero when the initial state is assumed to be zero and when it is estimated (in the Kautz case) (see Example 4.7).

Assuming again that the model class and the model order are known we can fit second-order OE and Kautz structures to simulated noise-free data when the number of samples \( N \) and \( x(0) \) are varied. For the Kautz case we have here turned off the iterative search algorithm, and instead fixed \( b \) and \( c \) to exactly correspond to the true poles of the system. A typical simulation result is detailed in Table 4.2. The fourth column shows the location of the estimated zero, while the fifth and last column shows the result when also the initial condition is estimated (in the Kautz case).

The values in Table 4.2 are expected. Filter transients affect the Kautz estimate, especially when the data record is small. Notice though that such problems do not occur for the OE case, and that it is fully handled in the Kautz case if only the initial state is estimated.
So far we have only considered simulated data. It is now time to investigate two industrial data sets both showing a resonant behavior. In Section 5.1 we first examine a simplified input-output modal analysis problem taken from the aircraft industry. Then, in Section 5.2 time series analysis is performed on neutron flux noise from a nuclear power plant. It is shown that data from these two applications accurately can be described by low-order Laguerre/Kautz models, whereas more traditional methods (ARX, OE, ARMAX and AR, ARMA) in general require higher model orders.

5.1 Aircraft Flight Flutter

Before a new airplane type is put into service one must evaluate its reliability through a quite extensive test flight program. Among many other things it is interesting to examine the mechanical limitations of the different aircraft components. A frequently used measure of this is the so-called flight flutter condition in which an aircraft component at a specific airspeed starts to oscillate. Above the “flutter” speed disturbances will grow uncontrolled until finally they are limited by nonlinearities, whereas at lower speeds dynamic structural vibrations will be damped out. The first situation may also lead to structural failure, therefore making the flutter behavior a natural measure of the maximum performance of the aircraft.

The flutter experiments are usually carried out by attaching special transducers to various points on the airframe, in this case to the wings, thereby introducing mechanical vibrations artificially. Flying at a predetermined and constant speed data is collected and analyzed off-line, giving information about whether to allow the aircraft to fly faster or not. See Schoukens and Pintelon [84] for further experimental details.

Data gained during flight flutter tests is typically noisy with a rather low SNR, and for safety and economical reasons only short data records are permitted. Thus Laguerre or Kautz models are likely to perform well. The data that will be investigated here originates from LMS International (Belgium) and was first used by Schoukens and Pintelon (see [84]). There the identification was carried out in the frequency domain using the ELiS (Estimation of Linear Systems) algorithm. Here we will stay within the time domain trying to fit the data to Laguerre/Kautz model structures.
Flutter data is obtained using burst swept sine (4 – 40 Hz) excitations that generate a force input $u(t)$ leading to an acceleration response, which is taken as the measured output $y(t)$. The data is sampled at 100 Hz and consists of three different data sets each containing 2048 samples. The first data set is exclusively used for modeling, while the other two sets are reserved for validation tests. As in [84] the goal of the identification is to model the frequencies which fall into the frequency band 4 to 11 Hz. The raw input-output data is therefore filtered through a fifth order Butterworth band-pass filter with this pass-band. The energy of the filtered time domain signals is then concentrated below 8 seconds, and hence only the first 800 samples are used for the identification procedure. The resulting excitation and response signals are depicted in Figure 5.1.

Since the smoothed amplitude plots of the spectral and the empirical transfer function estimates (ETFE) (which are assumed to reflect the true system well) in Figure 5.2 clearly show four peaks, a model of at least order 8 should be searched for. A natural first choice is to combine 4 different second order Kautz models, each with the potential of describing one of the peaks. The next step is to utilize the amplitude curves to determine the initial Kautz filter parameters, $b_i$ and $c_i$ for $i \in \{1, 2, 3, 4\}$. Thus the
5.1 Aircraft Flight Flutter

<table>
<thead>
<tr>
<th>Peak</th>
<th>Resonance frequency (Hz)</th>
<th>Filter parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.8</td>
<td>$b_1$ = 0.952, $c_1$ = -0.953</td>
</tr>
<tr>
<td>2</td>
<td>6.0</td>
<td>$b_1$ = 0.928, $c_1$ = -0.953</td>
</tr>
<tr>
<td>3</td>
<td>6.9</td>
<td>$b_1$ = 0.907, $c_1$ = -0.953</td>
</tr>
<tr>
<td>4</td>
<td>8.6</td>
<td>$b_1$ = 0.858, $c_1$ = -0.968</td>
</tr>
</tbody>
</table>

Table 5.1: Suggested Kautz filter parameters.

![Figure 5.3: Smoothed spectral analysis estimate, chosen initial a priori band-pass filters (left), along with the frequency response of the eighth order Kautz model (4 second order Kautz structures in parallel) that was obtained with these filter settings (right).](image)

In practice this means that only the individual, and not the actual compounded, effects of the filters are taken into account. Selecting filter parameters as in Table 5.1 gives the level-adjusted frequency responses shown in Figure 5.3. In the same figure we have also plotted the result from the eighth order Kautz model that was estimated with these filter settings. As can be seen, the peaks numbered 1 and 2 are not captured with requisite accuracy, which partly is due to the approximate choice of filter parameters.

To compensate for this imperfection it is reasonable to stick with 4 parallel Kautz structures and try higher model orders. Yet it turns out that the low frequency dynamics are not that easily captured, which indicates that the chosen model structure is not flexible enough.

To enhance the flexibility one idea is to increase the number of models in parallel.
Chapter 5. Applications

Figure 5.4: Bode plot of the twelfth order Laguerre/Kautz model with $na = 4$, $a = 0.73$ and Kautz filter parameters according to Table 5.1, together with the smoothed spectral analysis estimate.

Figure 5.5: Comparison between measured and simulated twelfth order Laguerre/Kautz model output.

Having already incorporated the information about the presumed peaks in the model structure it is natural to investigate what happens if the model description is extended by a Laguerre part. In order to cope with the low frequency misfit we here choose the Laguerre filter parameter to be 0.73, which corresponds to a frequency breakpoint at 5.0 Hz. Although this choice is somewhat ad hoc we can compensate for it by increasing the Laguerre order until an acceptable model is found. Taking the Laguerre order to 4, i.e., $4 \cdot 2 + 4 = 12$ estimated parameters altogether, we obtain a model that is able to reflect all the peaks. This is illustrated in Figure 5.4. A further increase in order only marginally improves the fit; the additional parameters mostly adjust for the high frequency discrepancy and frequencies outside the domain of interest. Moreover, the model is rather insensitive to the choice of Laguerre filter parameter. Any breakpoint in between 4 and 8 Hz gives a frequency response comparable to the one in Figure 5.4.

Figure 5.5 shows a portion of a time domain simulation using fresh data. The agreement between measured and simulated output is quite striking and we conclude that the twelfth order Laguerre/Kautz model has the ability to describe most of the flutter
5.2 Neutron Flux Noise

In a nuclear reactor of boiling water type (see [50] for a functional survey) it is extremely important to be able to estimate and predict the core dynamics. Off-line identification results are chiefly used as a guideline in, for example, plant start-up procedures, whereas on-line methods constitute the basis for continuous monitoring of the core stability. Here we will concentrate on off-line experiments.

Given the measured neutron flux noise, time series modeling is normally carried out in order to determine the reactor core stability. Traditionally, it is the so-called Decay Ratio (DR) defined as

\[
DR = e^{-\frac{2\pi \ln(r)}{\varphi}}
\]

that is used as the stability measure (consult Ingeström [50] for a motivation as well as a derivation). In Equation (5.2) \( r \) is the distance from the origin to the dominating resonant pole, while \( \varphi \) is the associated phase angle. See Figure 5.6 below.

![Figure 5.6: Definition of the Decay Ratio.](image)

The core is considered well-damped if the Decay Ratio is less than 0.25, which because of safety reasons is desirable under normal operations. However, during start-up procedures the reactor passes through different phases where the core stability, and with that the Decay Ratio, varies significantly. Practically, this put quite hard demands on dynamics. For comparison a number of ordinary ARX, OE and ARMAX models were also fitted to the data. Using the ARX approach no model of order 40 or lower has the ability to satisfactory describe the 4 resonant modes. Concerning OE and ARMAX modeling a large number of models (of maximum order 40) have been evaluated, but so far none has had the capacity of reflecting the 6 Hz peak. In fact, the best models derived are able to capture the other three peaks, although to accomplish this a much higher model order is needed (20 or more parameters).

Finally, it should be remarked that the frequency response of the accepted Laguerre/Kautz model is very similar to the one obtained in [84] using the ELiS algorithm (at least up to 10 Hz). Yet the number of estimated parameters is larger (21) in the latter case. Furthermore, the Kautz model was derived from one single data set, while all available data (3 sets) were used in obtaining the ELiS model.
Chapter 5. Applications

Figure 5.7: Neutron flux data used for model building.

The model structure. It should be of low and preferably constant order over a large operational range. It should also be robust in the sense that the Decay Ratio is delivered with reliable and easy to compute error bounds. Therefore, and because of the promising time series simulation results gained by Wahlberg and Hannan [100], we will try to employ Kautz modeling instead of the more common AR and ARMA approaches.

ABB Atom (Sweden) has kindly provided a number of experimental data sets taken at various reactor operational points. For simplicity we shall only contemplate one of them here, and refer to [50] for further experiments. The chosen record, sampled at 12.5 Hz, is first divided into two subsets each containing 4000 samples. The first set is used for the model fitting, while the other set is kept for validation tests. Secondly, the sample mean of the neutron flux data is removed, which gives the typical noise signal of Figure 5.7.

Starting to examine the smoothed spectral and periodogram estimates in Figure 5.8 two resonant peaks can be distinguished. Henceforth, we initially combine 2 second order Kautz models, each reflecting one of the pinnacles. Since we in this case are dealing with time series data the parameters $b_i$ and $c_i$ should be chosen near the zeros of the true system (see Section 3.3). But these zeros are unknown! One idea is to estimate an ARMA model of order, say, 12 and take $b_i$ and $c_i$ corresponding to the two zeros nearest the unit circle. The model order is here chosen as the highest value such that no obvious pole-zero cancellation occurs. This gives initial filter parameters according to Table 5.2.

With these filter settings we get a fair but not good enough model. To fine tune the model there are in principle two options; either we increase the model order or we invoke the iterative search algorithm discussed in the previous chapter. By taking 2 fourth order Kautz structures in the first case (8 parameters in all), and pursuing both these approaches in parallel the resulting frequency responses are given in Figure 5.8. The agreement between the two parametric models is excellent, and now the two peaks are captured accurately enough. Although the parametric models behave similarly, we select the latter one since it only involves 4 parameters.

To further validate this model we can look at the autocorrelation between the residuals $\varepsilon(t)$ at different time lags. This is shown in Figure 5.9. The dotted lines mark the 99% confidence level under the assumption that the residuals are white, which seems to be the case. Looking at the pole-zero configuration in Figure 5.9 the estimated Decay
5.2 Neutron Flux Noise

Smoothed spectral estimate.

Optimized Kautz model of order 4.

I0

\( f \)

Frequency (rad/sec)

\( 10^2 \)

\( 10^1 \)

\( 10^0 \)

\( 10^{-1} \)

\( 10^{-2} \)

Amplitude

Figure 5.8: Smoothed spectral analysis and periodogram estimates (left). Amplitude responses of the eight and the optimized fourth order Kautz models (right). The smoothing was done using Hamming windows of size 40 and 20, respectively.

<table>
<thead>
<tr>
<th>Peak</th>
<th>Initial parameters</th>
<th>Optimized parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>( b_i )</td>
<td>( c_i )</td>
</tr>
<tr>
<td>1</td>
<td>0.840</td>
<td>-0.789</td>
</tr>
<tr>
<td>2</td>
<td>-0.350</td>
<td>-0.377</td>
</tr>
</tbody>
</table>

Table 5.2: Initial and optimized fourth order Kautz filter parameters.

Ratio equals 0.17, which means that the reactor process is well-damped. This value is the same as the one obtained by ABB using nonparametric methods, which indicates that the Kautz model is sufficient for our purposes. Moreover, the quality of the estimate, i.e., the variance of the Decay Ratio, can be approximately calculated based on the variance of the real and imaginary parts of the dominating pole by taking the first-order Taylor series expansion of the Decay Ratio. The resulting expression is somewhat messy (see [50]), so let us just mention that the variance here is estimated to be around 0.02.

Having estimated a low-order Kautz model we now proceed to estimate AR and ARMA models for comparison. In Figure 5.10 the area of interest, i.e., the area around the first peak, is magnified and a number of AR and ARMA frequency responses are plotted. Evidently, the AR model of order 4 cannot describe the interesting peak, but by enhancing the order to at least 10 the desired dynamics are reasonable reflected. The situation is doubtless better in the ARMA case, but still 6 or more parameters must be estimated. Notice also that the AR and ARMA models of highest order show frequency responses very similar to the chosen Kautz model. Our confidence in the latter model is thus increased.

It should here be pointed out that at other operational points we will not only have poles closer to the unit circle but also poles near each other. This implies that the power spectral density is concentrated around two adjacent frequencies with rather high peaks. To obtain sufficient accuracy it is then necessary to increase the number of estimated parameters. Notice though that the idea of describing each peak with a separate Kautz
model can be maintained, but that the number of estimated parameters for each such model probably must be increased. By instead adopting the AR approach the situation is even more troublesome, and very high model orders are typically needed (at least 25 – 30 parameters). This is fully in line with what is reported in [100].

On-line plant stability monitoring, i.e., the continuous surveillance of the reactor core, is also of great importance. The techniques used today are all based on AR schemes that unfortunately are rather sensitive to the choice of model order. The problem is that by using overparameterized AR structures asymptotic accuracy is deteriorated. This typical phenomenon is illustrated and explained in Ljung and Söderström [67] (pages 348 – 350). Recursive Kautz estimation algorithms may, without too much extra computational effort, be a solution to this problem.
Part III

Semi-Physical Modeling

Waterfall, by M. C. Escher
(lithograph, 1961).
The Semi-Physical Modeling Problem

6.1 Introduction

On the scale between pure black box and fully physically parameterized modeling\(^1\), the approach suggested in the previous part is certainly close to the former concept. The a priori knowledge used was gained without any real physical insight, and the idea was simply to combine knowledge from several black box experiments. By incorporating physical insight (usually in terms of mathematical relationships), but not to the extent that a formal physically parameterized model is constructed, the work on finding a suitable model structure is more oriented towards the other extreme. We will in the sequel refer to this approach as semi-physical modeling, although it really can be viewed as a special branch of grey box modeling.

More precisely, we will by semi-physical modeling mean the process to take physical insight about the behavior of the system into account, to use that insight to find adequate nonlinear transformations on the raw measurements so that the new variables - the new inputs and outputs - stand a better chance to describe the true system when they are subjected to standard model structures (typically linear in the new variables). Let us illustrate the point in a toy example (cf. with Ljung [63]).

### Example 6.1
Suppose that we want to build a model for how the voltage applied to an electrical heater affects the temperature in a room. Physical modeling entails writing down all equations relating to the power of the heater, heat transfer, heat convection, and so on. This involves several equations, expressions, unknown heat transfer coefficients etc. Normally, this is a quite time-consuming procedure and the result is not too seldom a complicated model that is hard to understand and analyze. As an engineer one would here probably wonder what model simplifications can be justified and carried out. On the other hand, a simple black box approach would be to use, say, an ARX model with the applied voltage as input and the room temperature as output. But that is too simple! A moment's reflection reveals that it is the heater power rather than the voltage that causes the temperature to change. Thus try to use an ARX model with the squared voltage as input and the room temperature as output.

\(^1\)Sometimes also referred to as transparent box modeling.
Clearly, semi-physical modeling is in frequent use in practice. Yet the fact remains that many failures of the identification process are to be blamed on not applying this principle [63]. These failures are no doubt mainly due to the lack of relevant and general software tools that support a systematic and interactive modeling procedure. The objective with this part of the thesis is to address this need by discussing and highlighting formal algorithms as well as more informal software tools that together form a computerized system for doing semi-physical modeling in reality. Before discussing the intended building blocks, we will first illustrate the main ideas behind semi-physical modeling by showing typical steps and problems encountered in a simple but non-trivial example.

6.2 An Illustrative Example

The oil crises in the early seventies highly motivated the search for alternative and more environmental friendly energy sources. It became, e.g., popular to utilize solar energy for the heating of houses. Data from such a system, earlier investigated by Ljung [61] and by Ljung and Glad [65], will in this section be reused to emphasize the intended semi-physical modeling procedure and its usefulness.

Consider the solar-heated house depicted in Figure 6.1. The sun heats the air in the solar cells, whereupon the pump transports the hot air to the heat storage – a box filled with pebbles. Later during the night the energy flow is reversed and the house is heated. The modeling aim is to describe the storage inlet temperature and investigate how it is affected by the pump speed and the sun intensity. At our disposal there are three measurable signals, namely

\( I(t) \) the sun radiation at time instance \( t \) (a non-controllable input),

\( u(t) \) the pump speed (a controllable input), or actually a binary variable indicating if the pump is on or off, and

\( y(t) \) the storage inlet temperature (the output).

**Figure 6.1**: Sketch over the solar-heated house.
To begin with, measures of these signals were taken every tenth minute over a period of forty-eight hours, i.e., 296 samples. The first 120 samples (20 hours) were exclusively used for the estimation procedure, whereas the remaining data were reserved for validation tests. Next, the outdoor mean temperature (21° C) was removed from the storage inlet temperature, and in order to avoid numerical problems (division by zero) a constant level of 0.1 was added to the pump speed. The resulting signals are given in Figure 6.2.

As a preliminary (guided by the “try simple things first” principle) it is of interest to see how an ordinary linear model structure, such as

\[ y(t) + a_1 y(t-1) + a_2 y(t-2) = b_1 u(t-1) + b_2 u(t-2) + c_1 I(t-1) + c_2 I(t-2), \]  

(6.1)

would perform on this data. Focusing on the second day-time period, it is from the simulation detailed in Figure 6.3 clear that the least-squares fitted model has great difficulties in explaining the solar-house dynamics. As a matter of fact, staying with this linear model class and varying the model order does not help - the same kind of discrepancy between measured and simulated output is still there.

If the “try simple things first” path is unacceptable a good advice is to proceed to the slightly more complicated “try simple physical things next”. In our case this means that physical insight into the heating process should be taken into account. After a moment’s reflection one realizes that a linear model is not very realistic since the sun intensity and the pump speed hardly are additive. The solar panel, and indirectly the sun intensity, can scarcely affect the storage temperature when the pump is off. We should instead...
expect a multiplicative relationship between the available input signals.

To investigate this suspicion we can see what happens if the system is modeled according to the law of conservation of energy. As a first step we can introduce the non-measurable mean solar panel temperature \( x_1(t) \). The heating of the air in the solar panel \( \dot{x}_1(t) \) must then be equal to the energy injected into the system minus the energy lost to the surroundings. The way the energy is supplied or lost might in reality be governed by very complicated relationships, but in semi-physical modeling one should strive to simplify them. Therefore, assume that the sun is directly responsible for the entire supply of energy \( \eta_1 I(t) \), and that the losses to the surroundings is divided into two separate parts: losses to the environment \( \eta_2 x_1(t) \) and losses to the storage when the pump is operating \( \eta_3 x_1(t) u(t) \), i.e.,

\[
\dot{x}_1(t) = \eta_1 I(t) - \eta_2 x_1(t) - \eta_3 x_1(t) u(t). \tag{6.2}
\]

In the same way, the increase in the storage temperature \( \dot{x}_2(t) \) equals the surplus energy from the solar panel \( \eta_3 x_1(t) u(t) \) minus the losses through the storage walls \( \eta_4 x_2(t) \). In other words

\[
\begin{align*}
\dot{x}_2(t) &= \eta_3 x_1(t) u(t) - \eta_4 x_2(t) \tag{6.3} \\
y(t) &= x_2(t). \tag{6.4}
\end{align*}
\]

As data is time-discrete it is, at one time or another (we return to this issue later on), natural to convert the model structure into a discrete time counterpart. With the rule
of simplicity in mind the derivatives can be approximated by employing Euler's method (consult Dahlquist and Björk [16] for further details)

\[ \dot{x}_i(t) \approx \frac{x_i(t + T) - x_i(t)}{T}, \quad i = \{1, 2\}, \quad T > 0, \quad (6.5) \]

which immediately yields

\[
\begin{align*}
    x_1(t + 1) &= (1 - T\eta_2)x_1(t) - T\eta_3u(t)x_1(t) + T\eta_1I(t) \\
    x_2(t + 1) &= (1 - T\eta_4)x_2(t) + T\eta_3u(t)x_1(t) \\
    y(t) &= x_2(t). 
\end{align*}
\]

(6.6)  \hspace{2cm} (6.7)  \hspace{2cm} (6.8)

If only the temperature in the panel was measured Equation (6.6) would have been redundant and \( \eta_3 \) and \( \eta_4 \) could have been estimated directly from the latter two equations by taking \( u(t) \) and \( x_1(t) \) as input signals. Since this is not the case \( x_1(t) \) must be eliminated, which here can be done in the following way. First, determine \( x_1(t) \) and its shifted relative \( x_1(t + 1) \) from Equation (6.7). Next, substitute the resulting two expressions into Equation (6.6). Utilizing Equation (6.8), performing a time shift and solving for \( y(t) \) finally gives the input-output description

\[
y(t) = (1 - T\eta_4)y(t - 1) + T^2\eta_1\eta_3u(t - 1)I(t - 2) - T\eta_3u(t - 1)y(t - 1) \\
+ T\eta_3(1 - T\eta_4)u(t - 1)g(t - 2) + (1 - T\eta_2)\frac{u(t - 1)y(t - 1)}{u(t - 2)} \\
- (1 - T\eta_2)(1 - T\eta_4)\frac{u(t - 1)y(t - 2)}{u(t - 2)}. 
\]

(6.9)

As it stands, estimating these parameters would require some kind of iterative search algorithm. From a computational point of view this is unfortunate, especially for complex systems with many parameters. To overcome this difficulty it is appealing to carry out the following reparameterization:

\[
\begin{align*}
    \theta_1 &= f_1(\eta) = (1 - T\eta_4), & \varphi_1(t) &= y(t - 1), \\
    \theta_2 &= f_2(\eta) = T^2\eta_1\eta_3, & \varphi_2(t) &= u(t - 1)I(t - 2), \\
    \theta_3 &= f_3(\eta) = -T\eta_3, & \varphi_3(t) &= u(t - 1)y(t - 1), \\
    \theta_4 &= f_4(\eta) = T\eta_3(1 - T\eta_4), & \varphi_4(t) &= u(t - 1)y(t - 2), \\
    \theta_5 &= f_5(\eta) = (1 - T\eta_2), & \varphi_5(t) &= \frac{u(t - 1)y(t - 1)}{u(t - 2)}, \\
    \theta_6 &= f_6(\eta) = -(1 - T\eta_2)(1 - T\eta_4), & \varphi_6(t) &= \frac{u(t - 1)y(t - 2)}{u(t - 2)}. 
\end{align*}
\]

(6.10)

Thus Equation (6.9) is rewritten as a true linear regression,

\[
y(t) = \sum_{i=1}^{6} f_i(\eta) \varphi_i(t) = \sum_{i=1}^{6} \theta_i \varphi_i(t) = \Theta^T \varphi(t),
\]

(6.11)

where we have a linear relationship between the new parameters \( \theta \) and the constructed regression vector \( \varphi(t) \). Notice that although the linearized parameters are known it might here be impossible to substitute back and uniquely determine the original parameters \( \eta_i \).
This is as far as symbolic computations can be of any help in assigning a suitable model structure. It is now the task of an estimator to come up with reasonable parameter values. Since Structure (6.11) is a linear regression its parameters can be estimated efficiently using the least-squares algorithm. As can be seen from the simulation in Figure 6.4 the sixth order least-squares fitted nonlinear model performs somewhat better than the previously discussed linear one. The mean temperature deviation (4.0 compared to 4.2 for the linear model) is, however, still too high.

At this point it is quite natural to wonder if all of the regressors are equally important in explaining the output. Or posed another way, can some of the regressors be removed without a decrease in the model's predictability? By only picking the first two regressors from the proposed model structure, i.e.,

$$y(t) = \theta_1 y(t-1) + \theta_2 u(t-1) f(t-2).$$

the answer is according to Figure 6.5 obvious – some of the regressors seem to be redundant. In fact, these regressors are not only redundant, their inclusion really results in a worse model. This is mainly due to a phenomenon known as overfit to the modeling data. The problem is actually that some of the parameters are adjusted to disturbances unique for one particular data record. For other data records which do not show the same disturbance pattern the model will react erroneously.

To finally summarize the example, the mean temperature deviation of the second order nonlinear model is 3.1, and as we expected the significant input signal is a mixture between the sun intensity and the pump speed. The example also indicates that linear
black box structures are infeasible in some situations. To then succeed in the modeling work it is very important that knowledge about the fundamental nonlinearities are built into the models.

Let us by way of conclusion discern some features that make semi-physical modeling different from physical modeling. As a first pragmatic rule use only fundamental physical principles and approximations, such as the law of conservation of energy. Approximations and idealizations can in many situations be justified in that the available controller anyway is limited in structure. The original parameters (which might have physical significance) are generically not estimated, and in fact they can sometimes not even be uniquely determined within the original model class even though the data is sufficiently rich. Instead, new parameters that enter the model linearly are introduced, thereby giving distinct advantages in terms of estimation algorithms. The model structure is also subject to some sort of model selection scheme, whose mission is to pick out $m$ out of $n$ regressors that in some way best can predict the output. Together these properties motivate the term semi-physical modeling.

### 6.3 Required Tools

The objective with this quite lengthy example was not to model a specific system, but to give a flavor of the steps involved in a typical semi-physical modeling session. It should at this point be clear that such an approach requires both *symbolic* and *numerical* calculations. These demands will from a user's perspective be discussed in more details.
next, while the matters of how to fulfill them are discussed in chapters to come.

The first and without hesitation most challenging step is to obtain a useful model structure, i.e., dynamic relationships between the given control signals \( u(t) \) and the measured output signals \( y(t) \). Based on physical considerations the model builder is often able to write down some fundamental equations, which are believed to approximately describe the system in question. The system description might in reality also contain inequalities, inequations as well as relationships not easily captured in a mathematical framework. We should already here stress that the latter type of knowledge is beyond the scope of this thesis. Apart from the measured signals, the given equations typically include non-measurable quantities \( x(t) \), known physical constants and a set of unknown parameters \( \eta \). With this as a starting-point the problem of finding a suitable model representation can be stated as follows.

**Requirement 6.1** *Given a set of user defined equations generate, without changing the solution space, a new set of equations which do not depend on \( x(t) \).*

In terms of algorithms this requirement suggests a general symbolic equation solver which eliminates the unknown signals from the original set of equations. In a broad context these equations may contain arbitrary nonlinearities, combinations of discrete- and continuous-time variables, time delays, and so forth. The given problem is for such a general framework considered to be extremely hard.

However, if all the given expressions are differential-algebraic, i.e., all relationships between \( u(t) \), \( y(t) \), \( x(t) \), \( \eta \) and their time derivatives (of arbitrary order) are purely algebraic, the situation is more manageable. In practice this means that there exist algorithms that are able to solve the stated problem. One such constructive procedure, similar to Gaussian elimination for linear equation systems, was outlined by Ritt [80] in the 1930's. A perhaps more intricate algorithm, which also allows inequations, was later developed by Tarski and Seidenberg, see [85]. A third tool for solving polynomial differential equations originates from commutative algebra (see Cox et al. [15] for an introduction). Algorithmically, the unknown signals are eliminated by computing one particular so-called \textit{Gröbner basis} for the system of equations according to a scheme invented by Buchberger [11] in the 1970's.

Although these algorithms are different in nature, one cannot say that one approach is always superior to the others. In an ideal situation, the user should be given the opportunity to choose any of them. Despite this need we will here restrict the discussion to \textit{characteristic sets} (the output when running Ritt's algorithm) and the more researched Gröbner basis approach. Both these concepts can deal with continuous time systems, but when confronted with difference equations (which was the case in the solar house example) we must resort to Gröbner bases. Notice that this does not disqualify Ritt's algorithm even though data is collected using computers, since nothing prevent us from staying within the differential-algebraic framework when performing the elimination. Afterwards, when an input-output description has been found, it can very well be transformed into the time-discrete domain by replacing the derivatives with suitable time-discrete approximations.

Notice also that the point of discretization affect the resulting model structure. For the solar house, discretization before elimination resulted in six suggested regressors, whereas elimination before discretization results in seven regressors. Nevertheless, both structures contain the important combination of input and output signals, with the
6.3 Required Tools

minor difference being that \( \theta_2 u(t - 1) I(t - 2) \) becomes \( \theta_2 u(t - 2) I(t - 2) \). In spite of this difference in time-shift the simulation results of these two models are comparable.

At this point the practiced model builder would most likely object and claim that many model descriptions do not fit into the above given polynomial framework. Exponentials are, e.g., commonly encountered in chemical and biological modeling, while other structures often include trigonometric functions. Thus, the model designer must, at least, be given the freedom to use simple nonlinearities such as sinusoids and square-roots in his or her equations. Then, as we intend to stay within the polynomial domain, the following need arises naturally.

**Requirement 6.2** Given a set of equations, transform this set into a form acceptable for the available polynomial algorithms, without throwing away possible solutions.

The mentioned algorithms can be very demanding in terms of computational time and space complexity. It is not unusual that systems compounded of some four or five equations and variables require several hours of CPU time and some 10 MB of memory. It would therefore be worth a lot, even though we have to resort to heuristic arguments, to beforehand be able to determine some sort of upper complexity bound for the set of equations. If the complexity is just too high the advice is either to reduce the number of equations, or alternatively, to simplify some of the nonlinear relationships.

**Requirement 6.3** Before sending the set of equations to the solver decide if the problem is computationally feasible.

The result from the symbolic equation solver is finally rearranged so as to fit into the linear regression framework. Thus we require a post-processing algorithm of the following kind.

**Requirement 6.4** Given an input-output description transform, if necessary, it to a discrete time counterpart. Also, rewrite the result to a linear-in-the-parameters predictor, or when this is impossible, return a linear-in-the-parameters IO-structure.

The reason for this last transformation is, with risk of becoming tedious, that the new, and linear, parameters can be efficiently (with respect to computational burden, numerical accuracy etc.) estimated using the least-squares algorithm. It should again be emphasized that any physical significance of the original parameters normally are lost in this operation.

The number of regressors delivered from the symbolic package might in reality be very high. For a number of reasons, including model understanding and computational accuracy, a good low order model is to be preferred to a higher order one. For large systems an "all possible subset" search for the few best regressors is as good as impossible due to the combinatorial explosion. Fortunately, there exist quite a few standard statistical procedures for sorting out the most relevant regressors. Two variants of such algorithms, the forward selection and the backward elimination procedures (see Draper and Smith [20]), were interactively used to select the regressors of the solar house model (to be discussed further in Section 8.2). The last need is thus a combined estimation and model selection scheme.
Requirement 6.5  From measured data and a number of suggested regressors estimate a “good” model of low order.

The requirements having a symbolic nature are further investigated in Chapter 7, while the last numerical requirement is dealt with in Chapter 8. Their integration and software issues are finally the main topics of Chapter 9.

6.4 Some Other Approaches

In a survey article from 1979 Mehra [74], and somewhat later Billings [5], point out that one major crux of nonlinear system identification lies in model structure selection. The richness in terms of possible regressors increase very rapidly even for polynomial nonlinearities of bounded degrees. As data from nonlinear systems also can exhibit phenomena (limit cycles, hysteresis, chaos etc.) that are impossible for linear systems, one realizes that the selection problem is a tough but challenging one. Over the years some general more or less versatile classes of nonlinear model structures have been studied.

One simple idea is to only consider memoryless (or static) nonlinearities at the input side and assume that the dynamics itself is linear. The result is normally known as a Hammerstein model [28, 61, 77], which in its simplest form looks like

\[ A(q)y(t) = B_1(q)u(t) + B_2(q)u^2(t) + \ldots + B_m(q)u^m(t). \]  (6.13)

Combining this approach with a method for picking out relevant regressors could potentially have been used to arrive at the second order nonlinear solar house model. On the other hand the sixth order model could not have been found since it does not belong to the considered model class. Observe that the number of regressors delivered to the model reduction algorithm still can be quite high, especially for systems with several inputs with large time lags and nasty unknown nonlinearities.

Another and more general black box approach is GMDH (Group Method of Data Handling) [23, 74] invented by Ivakhnenko in the sixties. It differs from the previous scheme in that a coarse type of model reduction is built into the method. Leaving out some of the details, a GMDH model is compounded of several layers (each holding a number of computational units – the nodes) and indeed looks very much like a neural network. Given a training set \( (y^T(t)) \) each node takes two signals (variables) from the previous layer as input and estimates a second order polynomial comprising of all possible mixtures of these two signals (including a static level), i.e., a model with 6 parameters is estimated. Then, from validation data \( (y^V(t)) \), the statistical significance of the estimated output is tested against a threshold based on a heuristic criterion. If the model is unacceptable the node is just deleted (“survival of the fittest” principle), thereby reducing the number of nodes investigated in the next layer. The procedure is continued either until only one output (the accepted predictor) passes the current layer or a predetermined maximum degree of the polynomial is reached.

At first sight one might think that the complexity using GMDH is moderate since only 6 parameters are least-squares fitted in each node. This is only partially true as the number of nodes increases with the number of signals from the previous layer. Returning to the solar house example, ignoring that some variables might be inverted and assuming a maximum time lag of 2, one set of possible network inputs is \( \{y(t - 1), y(t - 2), u(t), u(t - 1), u(t - 2), I(t), I(t - 1), I(t - 2)\} \), which means that to construct the first layer
(8 \times 7)/2 = 28\) such models must be estimated and investigated. Furthermore, through the pruning there is still a risk of missing the important combinations of input and output signals. On the other hand having a criterion that rarely deletes intermediate nodes results in enhanced computational complexity.

Speaking of layered structures it would almost be a breach of duty not to mention \textit{neural networks} when discussing nonlinear black box system identification. There the calculating unit – the node – can have several inputs, which are weighted and thresholded. The resulting signals are then summed and filtered through a so-called activation function, typically chosen to be a sigmoid although other choices have been studied in the literature. Once the size of the network is determined the parameters, i.e., the weights, are estimated using conventional methods. Consult, e.g., [14, 66, 86] for further details.

GMDH and neural networks have one thing in common, namely that the number of estimated parameters often is high. Thus they violate the important property of \textit{parsimony} in terms of parameters. From a modeling viewpoint it also seems hard to incorporate \textit{a priori} knowledge, at least into the neural network framework.

Yet another method that can be applied to a wide class of processes is the functional-series approach, which includes the Volterra and Wiener series expansions. See the excellent overview by Billings [5] for details. Even though the least-squares algorithm is applicable for discrete Volterra series, Billings states that the amount of computations required often is excessive. The price paid for generality also show up in the difficulty of incorporating \textit{a priori} information, and of interpreting the estimates and relating them to the physical characteristics of the underlying process.

Turning lastly to traditional grey box modeling, Graebe and Bohlin [39, 40, 41, 42, 7] have designed and developed \textit{IDKIT}, which supports identification of nonlinear state space models that may be either discrete or continuous, multi input/output, and multi rate sampled. Unlike our approach the search routines are based on the \textit{Newton-Raphson} method, which is more computationally demanding than the least-squares method. It is true that IDKIT also encourages exploration of several models concurrently, but the kind of data supported model reduction scheme outlined in the previous section is not included.

IDKIT, and for that matters other similar tools, are often limited to special model structures, usually state space descriptions. This means that the model designer is bound to a certain way of thinking when developing the model description. What is more severe is that pure algebraic relationships (which are quite common especially if geometric aspects are accounted for) must be left out or manipulated to fit into the allowed framework. The symbolic tools of the kind discussed in the previous section do not show this kind of limitation. Any set of polynomial equations can be sent to the algorithms and they will, in a sense and if possible, produce a model structure that is better for our purposes.
This chapter is devoted to the algebraic (symbolic) issues of semi-physical modeling, i.e., to Requirements 6.1–6.4. The first requirement concerns the delicate problem of eliminating a number of undesirable variables from a system of equations (here dynamic ones). Assuming that all nonlinearities are polynomial, we first discuss how the previously mentioned Gröbner bases and characteristic sets can be used to solve this problem. Although there exist other elimination methods (resultant theory etc.) the main reasons for working with Gröbner bases and characteristic sets are their relative ease of understanding and their availability in terms of computer implementations.

In the next section we give a brief introduction to commutative algebra and in particular to Gröbner bases. We show how these bases can be computed and above all how they can be used in semi-physical modeling. In Section 7.2 we repeat this procedure for differential algebra and characteristic sets. Most of the results and methods detailed in these two sections are well known and can be found in the references.

The problem of extending the modeling domain to also comprise elementary functions (sinusoids, exponentials, and so on), i.e., Requirement 6.2, is studied in Section 7.3. The main result here is a scheme which, given almost any ordinary differential equation (ODE), produces an algebraic differential equation satisfying the original ODE.

The complexity issue of Requirement 6.3 is then briefly addressed in Section 7.4. Apart from some bounds on the size of the input-output expression, we give a few heuristic advice on what types of problems that are computationally feasible.

Finally, in Section 7.5 the problem of how to numerically represent or approximate the derivatives is discussed (Requirement 6.4). We also point out some problems that may occur when trying to express the model in predictor form.

7.1 Commutative Algebra and Gröbner Bases

In commutative algebra, properties of systems of polynomial equations are studied; in many aspects it provides the same kind of constructive machinery for nonlinear equations as linear algebra offers to linear equations. The fact that nonlinear systems in general are harder to deal with than linear ones clearly show up in the large number of abstract definitions and concepts found in commutative algebra. In this section we will restrict
the discussion to those parts absolutely necessary to explain Gröbner bases\footnote{The name was coined by Bruno Buchberger to honor his thesis supervisor the Tyrolean mathematician Wolfgang Gröbner (1899 – 1980).} – the first tool for tackling the elimination problem.

The presentation closely follows Cox \textit{et al.} [15], in which the proofs of the stated theorems can be found. In some parts the material is complemented with ideas from Davenport \textit{et al.} [17], Geddes \textit{et al.} [31] and Forsman [26]. The last reference is especially important for us as it addresses the use of Gröbner bases for answering questions about nonlinear state space models.

\section*{Basic Definitions and Key Concepts}

Before addressing Gröbner bases themselves we adopt some convenient definitions. The traditional starting point is to consider \( m \) polynomials in \( n \) variables \( x_1, \ldots, x_n \) (where for simplicity the independent variable has been left out).

\begin{definition}
A monomial in \( x_1, \ldots, x_n \) is a product of the form \( x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n} \), where all \( \alpha_i \in \mathbb{N} \). Let \( \alpha = (\alpha_1, \ldots, \alpha_n) \) and \( x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n} \).
\end{definition}

\begin{definition}
A polynomial \( f \) in \( x_1, \ldots, x_n \) is the sum of a finite number of terms each of the form \( a_\alpha x^\alpha \), where \( a_\alpha \) is the coefficient of the monomial \( x^\alpha \).
\end{definition}

The coefficients \( a_\alpha \) are assumed to lie in a field, denoted \( k \). Roughly speaking, a field is a set where addition, subtraction, multiplication and division can be defined in the usual way (see Cox \textit{et al.} [15] for the mathematical formalities). Commonly encountered fields are the rational numbers \( \mathbb{Q} \), the real numbers \( \mathbb{R} \) and the complex numbers \( \mathbb{C} \). For convenience, we now introduce a notion for expressing all polynomials in \( x_1, \ldots, x_n \).

\begin{definition}
The set of all possible polynomials in \( x_1, \ldots, x_n \) with coefficients in \( k \) is denoted \( k[x_1, \ldots, x_n] \).
\end{definition}

Taking the sum or product of two polynomials both result in a new polynomial, while division typically does not. In manipulating polynomials we therefore exclude division from the list of allowed operations, which mathematically means that \( k[x_1, \ldots, x_n] \) forms a commutative \textit{polynomial ring} in \( x_1, \ldots, x_n \) with coefficients from a field \( k \).

The basic algebraic objects associated with a ring are its \textit{ideals}.

\begin{definition}
A subset \( I \subset k[x_1, \ldots, x_n] \) is an ideal if it holds that
\[ 0 \in I, \quad \text{if } f, g \in I \text{ then } f + g \in I, \quad \text{if } f \in I \text{ and } h \in k[x_1, \ldots, x_n] \text{ then } hf \in I. \]
\end{definition}

\begin{definition}
Let \( f_1, \ldots, f_m \in k[x_1, \ldots, x_n] \). The ideal generated by \( f_1, \ldots, f_m \) is denoted \( \langle f_1, \ldots, f_m \rangle \), and is the set of all polynomials \( f \) that can be written \( f = h_1 f_1 + \ldots + h_m f_m \) for some polynomials \( h_j \in k[x_1, \ldots, x_n] \).
\end{definition}

For a system of equations, \( f_1 = \ldots = f_m = 0 \), the ideal \( \langle f_1, \ldots, f_m \rangle \) can be thought of as the set of all “polynomial consequences” of the given equations. One such consequence is that the common zeros of \( f_1 = \ldots = f_m = 0 \) are also the common zeros to the equations which can be derived from \( \langle f_1, \ldots, f_m \rangle \). An ideal \( I \) is now said to be \textit{finitely generated}
if there exist polynomials $f_1, \ldots, f_m \in k[x_1, \ldots, x_n]$ such that $I = \langle f_1, \ldots, f_m \rangle$, and somewhat improperly the polynomials $f_1, \ldots, f_m$ are called a basis of $I$. In analogy with linear algebra we are here looking for a particular basis so that the desirable properties of the set of equations can be recognized.

To do this we need to specify what desirable means. From a computational viewpoint it is crucial that all terms (or actually monomials) of the polynomials are unambiguously ordered. This can be achieved in many ways, though for our purposes it suffices to consider the so called lexicographic ordering.

**Definition 7.6** Let $x^\alpha$ and $x^\beta$ be two monomials, with $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n) \in \mathbb{N}^n$. We say that $\alpha >_\text{lex} \beta$ if, in the vector difference $\alpha - \beta$, the left-most nonzero entry is positive. If $\alpha >_\text{lex} \beta$ then we also say that $x^\alpha >_\text{lex} x^\beta$.

**Example 7.1** $x_1x_2^3x_4 >_\text{lex} x_2^3x_4^2$ since $\alpha - \beta = (1, 0, 3, 2) - (0, 5, 0, 2) = (1, -5, 3, 0)$.

Notice that this ordering is a total ordering on $\mathbb{N}^n$, a fact that actually must hold for any monomial ordering. One interpretation of the ordering is that polynomials containing variables in $x_r, \ldots, x_n$, $r \in \mathbb{N} > 1$, are more attractive than those having a variable $x_s$ such that $s < r$. To connect to the modeling problem of ours, the measured variables belong to the desirable domain, whereas the non-measured quantities do not.

Having imposed a monomial ordering we can introduce the following terminology.

**Definition 7.7** Let $f = \sum_{\alpha} a_\alpha x^\alpha$ be a polynomial in $k[x_1, \ldots, x_n]$ ordered by $>_\text{lex}$.

(a) The multidegree of $f$ is $\text{Mdeg}(f) = \max(\alpha \in \mathbb{N}^n : a_\alpha \neq 0)$, where maximum is taken with respect to $>_\text{lex}$.

(b) The leading coefficient of $f$ is $\text{Lc}(f) = a_{\text{Mdeg}(f)} \in k$.

(c) The leading monomial of $f$ is $\text{Lm}(f) = x^{\text{Mdeg}(f)}$ (with coefficient 1).

(d) The leading term of $f$ is $\text{Lt}(f) = \text{Lc}(f) \cdot \text{Lm}(f)$.

**Example 7.2** Consider $f = -2ix_1^2x_3^3 + 2x_1x_2x_3 + 5x_3^2x_4^3$ in $\mathbb{C}[x_1, x_2, x_3]$. Using $>_\text{lex}$ we have $\text{Mdeg}(f) = (2, 0, 3)$, $\text{Lc}(f) = -2i$, $\text{Lm}(f) = x_1^2x_3^3$, $\text{Lt}(f) = -2ix_1^2x_3^3$.

**Gröbner Bases and How to Compute Them**

As shown by Hilbert (Hilbert's Basis Theorem) every ideal $I \subset k[x_1, \ldots, x_n]$ has a finite generating set. A Gröbner basis then is merely a special generating set of an ideal, having some nice properties.

$^2$A total ordering implies a strict order among the monomials. For our problem it would often be enough to divide the monomials into two unordered sets – the sets of desirable and undesirable monomials. What then matters is that these sets are ordered. Following this approach could potentially lead to lower computational complexity. The major reason for still keeping the lexicographic ordering will be illustrated later on.
Definition 7.8 Fix a monomial ordering (here \(\succ_{\text{lex}}\)). A finite subset \(G = \{g_1, \ldots, g_s\}\) of an ideal \(I\) is said to be a Gröbner basis if \(\langle \text{Lt}(g_1), \ldots, \text{Lt}(g_s) \rangle = \langle \text{Lt}(I) \rangle\).

Once the monomials are ordered it can be shown, as a consequence of Hilbert’s basis theorem, that every ideal \(I \subset k[x_1, \ldots, x_n]\) has a Gröbner basis, and that any Gröbner basis for an ideal \(I\) is also a basis of \(I\).

Although Definition 7.8 is concise it gives few hints on how to algorithmically construct a Gröbner basis. As in the univariate case it is not surprising that one scheme needed is a generalization of polynomial division to the multivariate case.

Theorem 7.1 Fix a monomial ordering (here \(\succ_{\text{lex}}\)) on \(\mathbb{N}^n\), and let \(F = \{f_1, \ldots, f_m\}\) be an ordered subset of polynomials in \(k[x_1, \ldots, x_n]\). Then every \(f \in k[x_1, \ldots, x_n]\) can be written

\[
f = a_1f_1 + \ldots + a_mf_m + r,
\]

where \(a_j, r \in k[x_1, \ldots, x_n]\), and \(r\) is either zero or a linear combination of monomials, none of which is divisible by any \(\text{Lt}(f_1), \ldots, \text{Lt}(f_m)\). We call \(r\) a remainder of \(f\) on division by \(F\), and denote it \(r = \text{Rem}(f, F)\); see Cox et al. [15] for the algorithm.

Example 7.3 Let \(f = x_1^2x_2 + x_1x_2^2 + x_2^3\) and \(F = \{f_1, f_2\} = \{x_1x_2 - 1, x_2^2 - 1\}\). Taking remainder, first with respect to \(f_1\) and then with respect to \(f_2\), yields

\[
f = x_1x_2^3 + x_1x_2^2 + x_2^3 = (x_1 + x_2) \cdot (x_1x_2 - 1) + (1) \cdot (x_2^2 - 1) + (x_1 + x_2 + 1).
\]

The other way around (first \(f_2\) then \(f_1\)) gives

\[
f = x_1x_2^3 + x_1x_2^2 + x_2^3 = (x_1) \cdot (x_1x_2 - 1) + (x_1 + 1) \cdot (x_2^2 - 1) + (2x_1 + 1).
\]

The example reveals the unpleasant fact that the remainder cannot always be uniquely determined; it may depend on the order the polynomials are listed. However, one nice feature of Gröbner bases is that the remainder is unique, and hence this cannot happen.

The other operation needed to construct a Gröbner basis is a procedure to cancel out the leading terms of two polynomials.

Definition 7.9 Let \(f_1, f_2 \in k[x_1, \ldots, x_n]\) be two nonzero polynomials.

(a) If \(\text{Mdeg}(f_1) = \alpha\) and \(\text{Mdeg}(f_2) = \beta\), then let \(\gamma = (\gamma_1, \ldots, \gamma_n)\), where \(\gamma_i = \max(\alpha_i, \beta_i)\) for each \(i = 1, 2, \ldots, n\). The least common multiple of \(\text{Lm}(f_1)\) and \(\text{Lm}(f_2)\) is denoted \(x^{\gamma} = \text{Lcm}(\text{Lm}(f_1), \text{Lm}(f_2))\).

(b) The S-polynomial of \(f_1\) and \(f_2\) is

\[
S(f_1, f_2) = \frac{x^{\gamma}}{\text{Lt}(f_1)} \cdot f_1 - \frac{x^{\gamma}}{\text{Lt}(f_2)} \cdot f_2.
\]
Example 7.4 Consider $f_1 = 3x_1^2x_2 - 2x_1^2$ and $f_2 = x_1^2x_2^2 - x_1x_2 + x_2$ in $\mathbb{R}[x_1, x_2]$. Using the lexicographic monomial ordering, $M\deg(f_1) = (3, 1)$, $M\deg(f_2) = (2, 2)$ and thus $\gamma = (3, 2)$. Moreover, $Lt(f_1) = 3x_1^2x_2$ and $Lt(f_2) = x_1^2x_2^2$ so $S(f_1, f_2) = \frac{x_1^2x_2^2}{3x_1^2x_2} \cdot f_1 - \frac{x_1^2x_2^2}{x_1^2x_2^2} \cdot f_2 = \frac{1}{3} x_1^2x_2 - x_1x_2$.

We are now in a position to give an alternative and more constructive description of a Gröbner basis.

Theorem 7.2 Let $I$ be a polynomial ideal. A basis $G = \{g_1, \ldots, g_s\}$ for $I$ is a Gröbner basis for $I$ if and only if for all pairs $i \neq j$ $\text{Rem}(S(g_i, g_j), G) = 0$.

This theorem is the key to Buchberger’s algorithm [11]. Start with a set of polynomial equations $G = F$. For each pair of equation indices $i \neq j$ in this set, compute the corresponding $S$-polynomial and its remainder with respect to $G$. If the remainder is nonzero, then include it in $G$. Repeat this procedure until Theorem 7.2 holds. Although $G$ is extended in all but the last iteration it can be shown that the Buchberger procedure eventually stops, and hence it is an algorithm.

However, the Gröbner basis obtained using this algorithm often contains more polynomials than necessary. Fortunately, for a given monomial ordering, the ideal $I$ has a Gröbner basis that is both minimal and unique (see Cox et al. [15] for algorithmic details). In the sequel we will denote this particular Gröbner basis $GB(I)$.

Example 7.5 Consider $F = \{f_1, f_2, f_3\} = \{x_1 - 2x_2x_3, x_1 - 2x_2x_3 - 1, x_2 + x_2x_3 - 2\}$ and the corresponding ideal $I = (f_1, f_2, f_3)$. With respect to $\succ_{\text{lex}}$ we have

$$GB(I) = \{5x_1 + 10x_2^2 - 16x_3 - 1, 5x_2 + 20x_3^2 - 7x_3 + 8, 10x_3^3 - 6x_3^2 + 8x_3 - 1\}.$$  

The most desirable polynomial here is the last one, since it does not contain the variables $x_1$ or $x_2$, while the first polynomial is the one least attractive as it contains $x_3$. The important observation is that $GB(I)$ is expressed in a generalized triangular form. We can apply one-variable techniques to find the roots of $x_3$ from the third polynomial of $GB(I)$, then use “back-substitution” to solve for $x_2$, and so on (cf. with Gauss elimination for linear systems). Since the equations $F = 0$ have the same zeros as $GB(I) = 0$ we here have an excellent tool for solving polynomial equation systems. Suppose now that a fourth polynomial, $x_1 - 1$, is added to $F$. Then $GB(I) = \{1\}$, which in particular means that there is no solution whatsoever to $F = 0$.

Gröbner Bases in Semi-Physical Modeling

Because of the close relationship between equation solving and elimination it is not surprising that the Gröbner basis approach can be equally applied in the latter case. So far, though, we have only considered static equation systems and nothing has been said about the dynamic relationships we are actually interested in. To remedy this we start
to consider explicit SISO state space systems of the form (when \( f, h \) are polynomials)

\[
\begin{align*}
\sigma x_1(t) &= f_1(x(t), \eta, u(t), \sigma u(t), \ldots, \sigma^k u(t)), \\
\vdots \\
\sigma x_n(t) &= f_n(x(t), \eta, u(t), \sigma u(t), \ldots, \sigma^k u(t)) \\
y(t) &= h(x(t), \eta, u(t)),
\end{align*}
\] (7.1)

where \( \eta \) as usual is a parameter vector and \( \sigma \) either is the differentiation operator \( p \) or the forward shift operator \( q \), thus meaning that continuous- and discrete-time systems are treated in parallel. Notice that this state space formulation is generalized in the sense that a finite number \( k \) of derivatives or time shifts of the input are allowed to describe the states. We will here assume that only \( u(t) \) and \( y(t) \) are measured, so by elimination we simply mean a procedure that returns a relation between \( u(t) \) and \( y(t) \).

To see how this can be accomplished we first introduce some extra definitions.

**Definition 7.10** For a continuous-time system of the form (7.1), the extended Lie derivative operator \( l_f \) with respect to \( f \) is defined by

\[
l_f = \sum_{i=1}^{n} f_i \frac{\partial}{\partial x_i(t)} + \sum_{i=0}^{\infty} p^{i+1} u(t) \frac{\partial}{\partial p^i u(t)}.
\]

We also introduce \( L_f^j(h) \) as the ideal

\[
L_f^j(h) = (y(t) - h, py(t) - l_f h, \ldots, p^j y(t) - l_f^j h).
\]

**Definition 7.11** For a discrete-time system of the form (7.1), the ring-homomorphism \( \lambda_f \) is defined by \( \forall i : \lambda_f x_i(t) = f_i \), and \( \lambda_f q^i u(t) = q^{i+1} u(t) \). Let \( A_f^j \) be the ideal

\[
A_f^j(h) = (y(t) - h, q y(t) - \lambda_f h, \ldots, q^j y(t) - \lambda_f^j h).
\]

As a last concept we introduce a list \( R \), which just orders the variables from the least to the most wanted. Effectively, this is just a way of generalizing the concept of the lexicographic monomial ordering when arbitrary variable names are used (and not just \( x_i \)). By \( R = [x_1(t), x_2(t), \sigma^2 y(t), \sigma y(t), y(t), \sigma u(t), u(t)] \) we thus say that polynomials in the states are less desirable than polynomials in the output and the input variables only.

Due to Forsman [26] we have the following important theorem.

**Theorem 7.3** A Gröbner basis for \( L_f^N(h) \) or \( A_f^N(h) \) with respect to an ordering of the type

\[
R = [x(t), \sigma^n y(t), \ldots, \sigma y(t), y(t), \sigma^k u(t), \ldots, \sigma u(t), u(t)]
\]

contains a minimal order input-output relation for the system (7.1).

The key point of Theorem 7.3 is that it is sufficient to carry out \( n \) Lie or ring-homomorphism operations. Also, observe that each \( \sigma^i y(t) \) and \( \sigma^j u(t) \) are treated as separate variables, and that no differentiations (time shifts) are performed in the Gröbner basis calculations. The original problem is instead reduced to a purely algebraic elimination problem. Let us illustrate the procedure in two examples.
7.1 Commutative Algebra and Gröbner Bases

Exam.  •  6 A state space model structure of a DC-motor is (from Rugh [83])

\[
\begin{align*}
\frac{di(t)}{dt} &= -\frac{R}{L}i(t) - \frac{K}{L}u(t)\omega(t) + \frac{V_0}{L} \\
\frac{d\omega(t)}{dt} &= -\frac{B}{J}\omega(t) + \frac{K}{J}u(t)i(t) \\
y(t) &= \omega(t),
\end{align*}
\]

with \( R, L, K, B \) and \( J \) being unknown parameters, \( V_0 \) the constant voltage applied to the armature circuit of the motor, \( \omega(t) \) the measurable angular velocity of the motor shaft, \( i(t) \) the non-measurable armature current, and \( u(t) \) the so-called field current which is acting as a control signal. Using Lie derivatives we get the ideal

\[
L^2_f(h) = (y(t) - \omega(t), py(t) + \frac{B}{J}\omega(t) - \frac{K}{J}u(t)i(t), p^2y(t) + \left(\frac{JRK + BKL}{J^2L}\right)u(t) - \frac{K}{J}pu(t), i(t) + \left(\frac{K^2}{JL}u(t)^2 - \frac{B^2}{J^2}\right)\omega(t) - \frac{KV_0}{JL}u(t)).
\]

By letting \( R = [i(t), \omega(t), p^2y(t), py(t), y(t), pu(t), u(t)] \) the following Gröbner basis is obtained:

\[
\text{GB}(L^2_f) = \{-JLy(t) - (BL + J)\omega(t) - \omega(t) + Ku(t)i(t) + KV_0u(t), \\
-\omega(t) + u(t), JLy(t)u(t) - KV_0u(t), \\
\}
\]

The last polynomial is an IO-relation, and is a possible model structure for describing the angular velocity given the field current. If, however, \( i(t) \) could be measured, the second polynomial is a better structure in the sense that it comprises fewer terms than the last polynomial. Hence, a lexicographic Gröbner basis also shows the benefit of measuring more signals.

Example 7.7 A discrete-time variant of a predator-prey system is (see, e.g., Luenberger [69])

\[
\begin{align*}
N_1(t + 1) &= \eta_1N_1(t) - \eta_3N_1(t)N_2(t) - \eta_2N_1(t)^2 \\
N_2(t + 1) &= -\eta_4N_2(t) + \eta_5N_1(t)N_2(t) \\
y(t) &= N_1(t),
\end{align*}
\]

where the number of preys \( y(t) = N_1(t) \) can be measured, while the number of predators \( N_2(t) \) is unknown. Applying the \( A_f \) operator on the output map gives the ideal

\[
A_f^2(h) = (y(t) - N_1(t), qy(t) - \eta_1N_1(t) - \eta_3N_1(t)N_2(t) - \eta_2N_1(t)^2, q^2y(t) - \ldots).
\]

The last of the four Gröbner basis elements for \( A_f^2(h) \) with respect to the ordering \( R = [N_2(t), N_1(t), q^2y(t), py(t), y(t)] \) is a difference polynomial, namely

\[
y(t)y(t + 2) + (\eta_3 - \eta_5)y(t)y(t + 1)^2 + \eta_4y(t + 1)^2 - \eta_3\eta_5y(t)^3y(t + 1) + (\eta_1\eta_5 + \eta_3\eta_4)y(t)^2y(t + 1) - \eta_1(1 + \eta_4)y(t)y(t + 1).
\]

It now turns out that Theorem 7.3 can be extended to hold when \( f \) and \( h \) are rational functions. Furthermore, explicit MISO state space systems can be handled in almost the same way, and in fact so can MIMO systems. In the latter case, though, one must specify
what is meant by an IO-relation (is, e.g., a relation involving several output signals an acceptable IO-relation, or should it involve only one output signal?).

Yet another situation that can be handled is implicit systems of the form

\[ f_1(\sigma x_1(t), x(t), \eta, u(t), \sigma u(t), \ldots, \sigma^k u(t)) = 0 \]
\[ \vdots \]
\[ f_n(\sigma x_n(t), x(t), \eta, u(t), \sigma u(t), \ldots, \sigma^k u(t)) = 0 \]
\[ h(y(t), x(t), \eta, u(t)) = 0, \] (7.2)

where all \( f_i \) and \( h \) are rational functions. The idea is to proceed in the same manner as for explicit systems; given the output map form the slightly redefined ideal \( L_f^n(h) \) \((A_f^n(h))\), and apply the Gröbner basis scheme to eliminate the states. However, for the algorithm to work properly it must here be required that the systems correspond to prime ideals\(^3\), which unfortunately is a property that is rather hard to test for.

To fully meet Requirement 6.1 we must also be able to handle systems not given in state space form. For this purpose it is more convenient to use differential algebra, since then, any set of polynomial differential equations (including pure algebraic ones) directly can be taken care of. In particular, this means that no pre-processing (taking Lie derivatives etc.) of the equations is necessary before the solver is invoked. But as the name indicates only differential and not difference equations are allowed, which strongly motivates, e.g., Gröbner bases as a reasonable complement.

### 7.2 Differential Algebra and Characteristic Sets

Next to commutative algebra we find the more general differential algebra\(^4\), which besides the usual operations (addition, subtraction, multiplication and division) also allow differentiation. In the sense of Ritt \([80]\) and Kolchin \([54]\), differential algebra focus on constructive methods for manipulating nonlinear differential polynomials. One of the most important concepts within this ramification is so-called characteristic sets, which in many ways play the same role as Gröbner bases do in commutative algebra. These sets, along with a scheme (often referred to as Ritt's algorithm) for computing them, are the main topics of this section.

The more theoretical material discussed herein originates from the two references mentioned above. The use of differential algebra for solving various control problems has been emphasized by Fliess and Glad, see, e.g., the overview \([25]\). An implementation of Ritt's algorithm is more thoroughly described in Glad \([33]\).

#### Basic Definitions and Key Concepts

In differential algebra the polynomial concept is enlarged to comprise differential polynomials, i.e., polynomials in a number of variables and their derivatives (of arbitrary but finite order). Following the structure of the previous section, a reasonable starting point is to consider systems of \( m \) differential polynomials \( f_1, \ldots, f_m \) in \( n \) variables, enumerated \( x_1, \ldots, x_n \) (the independent variable \( t \) is again dropped).

\(^3\)An ideal \( I \subseteq k[x_1, \ldots, x_n] \) is prime if whenever \( f_1, f_2 \in k[x_1, \ldots, x_n] \) and \( f_1 f_2 \in I \), then either \( f_1 \in I \) or \( f_2 \in I \). It can be shown \([26]\) that the ideals created in the explicit case are prime.

\(^4\)Founded by the American mathematician Joseph F. Ritt in the 1930's.
Example 7.8 \( \{x_1^{(17)}x_3^2 + 2.5x_2x_1, (x_2^{(5)})^3 + 4ix_2x_3, x_3^2 - 1\} \) is a set of differential polynomials. Notice how superscripts (17 and 5) within brackets are used to denote higher order derivatives.

The coefficients are here assumed to lie in a differential field \( k \), which is nothing else but an ordinary field equipped with a differentiation \( \frac{d}{dt} \) defined so that \( \forall a, b \in k \)

\[
\frac{d}{dt}a = \dot{a} \in k, \quad \frac{d}{dt}(a + b) = \dot{a} + \dot{b}, \quad \frac{d}{dt}(ab) = \dot{a}b + ab. \tag{7.3}
\]

An immediate consequence of this formulation is that the “usual” fields \( \mathbb{Q}, \mathbb{R} \) and \( \mathbb{C} \) with constant elements are differential fields. In analogy with the commutative case, we let \( k\{x_1, \ldots, x_n\} \) denote the infinite set of all possible differential polynomials in the indeterminates \( x_1, \ldots, x_n \) with coefficients from \( k \). Mathematically speaking, \( k\{x_1, \ldots, x_n\} \) forms a differential polynomial ring, and any differential polynomial in \( x_1, \ldots, x_n \) is an element in this ring.

Not surprisingly, the important concept of ideal carries over to differential algebra, where the objects are referred to as differential ideals.

Definition 7.12 A set \( \Sigma \) of differential polynomials in \( k\{x_1, \ldots, x_n\} \) is a differential ideal if the following two conditions hold.

a) If \( f_1, \ldots, f_m \) belong to \( \Sigma \), then so does \( q_1f_1 + q_2f_2 + \ldots + q_mf_m \) for any choice of differential polynomials \( q_1, \ldots, q_m \in k\{x_1, \ldots, x_n\} \).

b) The derivative of every differential polynomial in \( \Sigma \) is contained in \( \Sigma \).

Definition 7.13 Let \( f_1, \ldots, f_m \in k\{x_1, \ldots, x_n\} \). The differential ideal generated by \( f_1, \ldots, f_m \) is denoted \( \langle f_1, \ldots, f_m \rangle \), and is the set of all differential polynomials of the form (s finite)

\[
h_{10}f_1 + h_{11}f_1 + \ldots + h_{1s}f_1^{(s)} + \ldots + h_{m0}f_m + h_{m1}f_m + \ldots + h_{ms}f_m^{(s)},
\]

with the \( h_{ij} \) being arbitrary differential polynomials.

From the last definition it follows trivially that the solutions to \( f_1 = \ldots = f_m = 0 \) are solutions to every equation that can be created from \( \langle f_1, \ldots, f_m \rangle \). Among all expressions in this set we are looking for a particular representative having as “simple” a structure as possible.

The concept of ranking (cf. with ordering in the non-differential case) is the common way of expressing simplicity. From an algorithmic point of view, it is important that all variables and their derivatives are unambiguously ordered. Because of this any ranking, denoted \( < \), must satisfy two conditions, namely

\[
x_i^{(\nu)} < x_j^{(\nu + \sigma)}, \quad \forall \nu, \sigma \in \mathbb{N},
\]

if \( x_i^{(\nu)} < x_j^{(\mu)} \) then \( x_i^{(\nu + \sigma)} < x_j^{(\mu + \sigma)} \), \( \forall \nu, \sigma, \mu \in \mathbb{N} \), \( \tag{7.4} \)

\[5\]The field of analytic functions in the independent variable \( t \) is also a differential field, which means that time varying systems can be dealt with too.
where $x_i$ and $x_j$ are two different variables, and $<$ is interpreted as "is ranked lower than", or equally, "is more desirable than". The ranking constraints are quite natural, and still leave us with a rich flora of opportunities.

**Example 7.9** In three variables $x_1$, $x_2$ and $x_3$ there are many possible rankings, e.g.,

\[
\begin{align*}
&x_1 < \dot{x}_1 < \ddot{x}_1 < \ldots < x_2 < \dot{x}_2 < \ddot{x}_2 < \ldots < x_3 < \dot{x}_3 < \ddot{x}_3 < \ldots \quad (7.5) \\
&x_1 < x_1 < \ddot{x}_1 < \ldots < x_2 < \dot{x}_2 < \ddot{x}_2 < x_3 < \dot{x}_3 < \ddot{x}_3 < \ldots \quad (7.6) \\
&x_1 < x_2 < x_3 < \dot{x}_1 < \ddot{x}_1 < \ddot{x}_2 < \dot{x}_3 < \ddot{x}_3 < \ldots \quad (7.7)
\end{align*}
\]

The ranking concept provides a natural way of ranking differential polynomials too. We first determine the highest ranking derivative of each polynomial. This is the leader of the polynomial, $Ld(f)$, and the corresponding variable is called the leading variable, $Lv(f)$. Next, we let the ranking of the leaders determine the ranking of the polynomials. If several polynomials have the same leader, then the one with the highest power of the leader is ranked highest. If also the powers of the leaders are the same, then the polynomials have the same ranking, so this ordering is only a partial ordering.

**Example 7.10** Using ranking (7.5) on the differential polynomials

\[
\begin{align*}
f_1 &= \dddot{x}_1^2 x_2 x_3 + 5\dddot{x}_1^2 x_3 + 2x_2, & \Rightarrow Ld(f_1) &= \dddot{x}_3, & Lv(f_1) &= x_3, \\
f_2 &= 4x_2^{(5)} + \dddot{x}_1^2 x_1^{(17)} + 1, & \Rightarrow Ld(f_2) &= x_2^{(5)}, & Lv(f_2) &= x_2, \\
f_3 &= \dddot{x}_3^2 + x_1 \dddot{x}_2 x_3 + x_1^4, & \Rightarrow Ld(f_3) &= \dddot{x}_3, & Lv(f_3) &= x_3,
\end{align*}
\]

gives the polynomial ranking $f_2 < f_1 < f_3$. With ranking (7.7) we have instead $f_1 < f_3 < f_2$.

Based on the polynomial ranking we can introduce the concept of reducedness.

**Definition 7.14** Let $f_a$ be a differential polynomial. Another differential polynomial $f_b$ is said to be partially reduced with respect to $f_a$ if it contains no higher derivative of $Lv(f_a)$ than $Ld(f_a)$. Furthermore, $f_b$ is said to be reduced with respect to $f_a$ if it is partially reduced, and the power of $Ld(f_a)$ in $f_b$ is smaller than the power of $Ld(f_a)$ in $f_a$.

Using the polynomials detailed in Example 7.10 and ranking (7.5), we see that $f_1$ is partially reduced with respect to $f_3$, and vice versa. Moreover, $f_1$ is reduced with respect to $f_3$, but the converse is false since the power of $Ld(f_1)$ in $f_3$ is larger than that of $Ld(f_1)$ in $f_1$. Sets of differential polynomials where all the polynomials are reduced with respect to each other turn out to be especially important.

**Definition 7.15** A set of differential polynomials $\mathbf{A}$, where all polynomials are reduced with respect to each other, is called an autoreduced set.

Suppose that $\mathbf{A}$ and $\mathbf{B}$ are two autoreduced sets consisting of $m_1$ and $m_2$ differential polynomials, respectively, both listed in increasing rank. Starting from the left with $A_1$ and $B_1$, we can pursue a pairwise comparison to the right until either the leaders of the polynomials $A_j$ and $B_j$ differ, or all polynomials in one of the sets have been considered. If $A_j$ is ranked lower than $B_j$ or $j = m_2 + 1 \leq m_1$, then $\mathbf{A}$ is considered to be ranked lower than $\mathbf{B}$. 

Example 7.11  Let the ranking be (7.5) and consider the autoreduced sets

\[ A : \quad 5\dot{x}_1^2\ddot{x}_1 + 1, \quad \dot{x}_2^3 + x_2^2 + 3. \]
\[ B : \quad \ddot{x}_1^2 - x_1^2 + 7, \quad \ddot{x}_1^2 x_1 + 2x_2, \quad x_2x_3. \]

Here \( Ld(A_1) = Ld(B_1) = \ddot{x}_1 \), while \( Ld(A_2) = \dot{x}_2 < Ld(B_2) = \dddot{x}_2 \), so \( A \) is ranked lower than \( B \).

This finally brings us to characteristic sets.

Definition 7.16  Let \( \Sigma \) be a set of differential polynomials. If \( A \) is an autoreduced set in \( \Sigma \), such that no lower autoreduced set can be constructed in \( \Sigma \), then \( A \) is called a characteristic set of \( \Sigma \).

It can readily be shown that every set of differential polynomials has a characteristic set. Another interesting property is that if \( A \) is a characteristic set of \( \Sigma \), then \( \Sigma \) contains no nonzero polynomial reduced with respect to \( A \).

Ritt's Algorithm

Given a ranking the obvious question now is how to algorithmically construct a characteristic set for \( \Sigma \). Luckily, the answer was given, or let us say outlined, in the book by Ritt [80], who suggested an algorithm that, in principle, used only two subroutines. From an arbitrary finite set of differential polynomials the first one is for determining a characteristic set \( A \) in \( \Sigma \). The second requirement is a scheme for doing division in \( k\{x_1, \ldots, x_n\} \), so that unwanted variables are eliminated.

Finding a characteristic set \( A \) in \( \Sigma \) can readily be done when \( \Sigma \) is finite. Commence by selecting a lowest ranked polynomial in \( \Sigma \). Then collect all polynomials in \( \Sigma \) that are reduced with respect to the chosen polynomial, and accept a lowest one among those as the second polynomial. Proceeding in this fashion will finally result in a characteristic set \( A \) of \( \Sigma \), and is obtained by \( A = ChS(\Sigma) \).

If this was fairly simple, the division algorithm is much more involved. In the subsequent discussion two special "coefficients", separants and initials, are going to be of utmost importance.

Definition 7.17  Let \( f \) be a differential polynomial.

(a) The separant of \( f \), denoted \( S(f) \), is defined as \( S(f) = \frac{\partial f}{\partial Ld(f)} \).

(b) The initial of \( f \), denoted \( I(f) \), is the coefficient of the highest power of \( Ld(f) \). ■

Example 7.12  If \( f = 3x_1\ddot{x}_2 - x_2 + 6x_1^2 + 1 \) and the ranking is (7.5), then \( S(f) = 6x_1\dddot{x}_2 - 1 \) and \( I(f) = 3x_1 \).

The following three concepts are also useful, especially from a computational point of view.

Definition 7.18  Fix a ranking. Let \( f \) be a differential polynomial and \( \Sigma \) a set of such polynomials.

(a) The highest unreduced derivative of \( f \) with respect to \( \Sigma \) is denoted \( Lu(f) \).
(b) The highest degree term of $L_u(f)$ is given by $L_p(f)$.

(c) The coefficient of the differential monomial $v$ in $f$ is denoted $L_j(f,v)$.

**Example 7.13** Suppose that $\Sigma$ consists of two polynomials with leaders $\ddot{x}_2$ and $\ddot{x}_3$ under the ranking (7.5). If $f = 2x_1x_2^3 + 3x_1^2x_3^2 - x_2x_3^2$, then

$$L_u(f) = \ddot{x}_2, \quad L_p(f) = 3x_1^2x_2, \quad L_j(f, x_3^2) = 2x_1.$$ 

When $A$ is an autoreduced set and $f$ a differential polynomial, then dividing $f$ by $A$ can be thought of as finding differential polynomials $q$ and $r$ such that

$$qf = r + [A], \quad (7.8)$$

where $r$ is reduced with respect to $A$ and $+$ is the modulo operator. Because differentiation is allowed in the division process, the algorithm becomes more complicated than the commutative counterpart. It is therefore customary to first find an $r$, let us call it $r_p$, so that $r_p$ is partially reduced with respect to $A$. In Glad [33] it is shown that

$$\prod_{A_j \in A} S(A_j)^{\sigma_j} f = r_p + [A], \quad (7.9)$$

always holds for some nonnegative integers $\sigma_j$, where $r_p$ is partially reduced with respect to $A$ and no higher than $f$. The key observation here is that the partial remainder, denoted $r_p = \text{rem}_p(f, A)$, can be computed in finitely many steps by considering separants and easily identified elements from $[A]$ only. The algorithm for achieving this is pretty straightforward, see Glad [33] for further details, so we just exemplify the procedure.

**Example 7.14** Let the ranking be (7.5) and consider the autoreduced set $A$ with elements

$$A_1: \quad 2x_1x_2^2 - 1, \quad \text{Ld}(A_1) = \dot{x}_2, \quad S(A_1) = 4x_1\dot{x}_2,$$

$$A_2: \quad \ddot{x}_3 + 2x_3 - 3x_2, \quad \text{Ld}(A_2) = \ddot{x}_3, \quad S(A_2) = 1.$$ 

Let $f = 2\ddot{x}_2x_3^{(4)}$, which obviously is not partially reduced with respect to $A$. Starting with $r_p = f$, the following calculations result in an $r_p$ which is partially reduced with respect to $A$.

1a. $r_p = S(A_2) \cdot r_p - \frac{L_p(r_p)}{L_u(r_p)} \cdot A_2 = 6\ddot{x}_2^2 - 4\ddot{x}_2\dddot{x}_3$.

2a. $r_p = S(A_1) \cdot r_p - \frac{L_p(r_p)}{L_u(r_p)} \cdot A_1 = -(16x_1\ddot{x}_2\dddot{x}_3 + 12\dot{x}_1\dddot{x}_2^2)\dddot{x}_2$.

2b. $r_p = S(A_1) \cdot r_p - \frac{L_p(r_p)}{L_u(r_p)} \cdot A_1 = 32x_1\ddot{x}_1\dddot{x}_2^2\dddot{x}_3 + 24\dddot{x}_2^4 = \text{rem}_p(f, A)$.

In each step the algorithm cancels out the highest ranked derivative which is not already partially reduced with respect to $A$. In every substep (marked with letters) the highest power of this derivative is successively taken care of using a suitable derivative of $A_j$. In step 1a above, $x_3^{(4)}$ is canceled out using $A_2$, which is chosen since its leading variable is $x_3$. Because the difference in the order between $r_p$ and $A_2$ in $x_3$ is two, $A_2$ is differentiated twice in this step. Then, in step 2a $\dddot{x}_2^3$ and in step 2b $\dddot{x}_2$ are canceled out in the same manner using $A_1$. 

In a similar way, though without taking derivatives, Glad [33] has detailed a scheme, taking an \( r_p \) and a differential polynomial \( A_j \) as inputs, that returns an \( r \) reduced with respect to \( A_j \). We call \( r \) the remainder of \( r_p \) with respect to \( A_j \), and refer to the computations as \( r = \text{rem}(r_p, A_j) \). Assuming that the elements of \( A \) are ordered in increasing rank, the remainder of a differential polynomial \( f \) with respect to \( A \) is denoted \( r = \text{rem}(f, A) \). and recursively computed as

\[
r = \text{rem}(f, A) = \text{rem}(\ldots \text{rem}(\text{rem}_p(f, A), A_m), \ldots, A_2), A_1). \tag{7.10}
\]

It can be shown that

\[
\prod_{A_j \in A} S(A_j)^{\sigma_j} I(A_j)^{\nu_j} f = r + [A] \tag{7.11}
\]

for nonnegative integers \( \sigma_j \) and \( \nu_j \), where \( r \) is reduced with respect to \( A \) and no higher than \( f \). As a matter of fact, calculating remainders when \( f = r_p \) involves initials only (and not separants), which the following example illustrates.

**Example 7.15** Reconsider the system of Example 7.14, where \( r_p = 32x_1 x_1^2 x_3 + 24x_1^2 x_2^2 \). We first note that the polynomials are listed in increasing rank and that \( I(A_1) = 2x_1 \) and \( I(A_2) = 1 \). Let \( \alpha \) be the highest power of \( \text{Ld}(A_j) \), \( \beta \) the highest power of \( \text{Ld}(r) \) and \( \gamma = \beta - \alpha \). Since \( A_2 \) is ranked higher than \( A_1 \), \( r = \text{rem}(r_p, A_2) \) is computed first (\( \alpha = 1 \)).

1. \( r = r_p = 32x_1 x_1^2 x_3 + 24x_1^2 x_2^2 \).
   1a. \( r = I(A_2) \cdot r - \text{Ld}(A_2)^\gamma \cdot \text{Lj}(r, \text{Ld}(A_2)^\delta) \cdot A_2^{\delta = 1} = 24x_1^2 x_2^2 - 32x_1 (2x_3 - 3x_2) x_2^2 \).

Next, \( r = \text{rem}(r, A_1) = \text{rem}(r, A) \) is calculated (now \( \alpha \) equals 2).

2a. \( r = I(A_1) \cdot r - \text{Ld}(A_1)^\gamma \cdot \text{Lj}(r, \text{Ld}(A_1)^\delta) \cdot A_1^{\delta = 4} = 64x_1^2 x_1 (3x_2 - 2x_3) x_2^2 + 24x_1^2 x_2^2 \).
2b. \( r = I(A_1) \cdot r - \text{Ld}(A_1)^\gamma \cdot \text{Lj}(r, \text{Ld}(A_1)^\delta) \cdot A_1^{\delta = 3} = 48x_1^2 x_1 x_2^2 + 64x_1^2 x_1 x_2 (3x_2 - 2x_3) \).
2c. \( r = I(A_1) \cdot r - \text{Ld}(A_1)^\gamma \cdot \text{Lj}(r, \text{Ld}(A_1)^\delta) \cdot A_1^{\delta = 2} = 128x_1^3 x_1 x_2 (3x_2 - 2x_3) + 48x_1^2 x_2^2 \).

Notice how \( r \) is modified and used in each step. This scheme is actually nothing else but a variant of the division algorithm that was discussed in the commutative case.

Using the above described procedures, Ritt’s algorithm for a set of differential polynomials \( \Sigma \) is as follows.

1. Input is a ranking and a set of differential polynomials \( \Sigma = \{f_1, \ldots, f_m\} \).
2. Calculate a characteristic set \( A \) of \( \Sigma \), i.e., \( A = \text{ChS}(\Sigma) \).
3. If \( \text{rem}(f_i, A) = 0 \ \forall f_i \in \Sigma \), then Return(\( A \)).
4. Let \( \Sigma = \Sigma \cup \{r\} \) for an \( r = \text{rem}(f_i, A) \neq 0 \), and go to step 1.

The fact that this is an algorithm means that the condition in step 2 eventually becomes true, though the choice of remainder in step 3 has a tremendous impact on when this happens. In the procedure a lower ranked ("simpler") characteristic set is created in each step, and finally \( A \) will be the "simplest" conceivable characterization of \( [\Sigma] \). Furthermore, a solution of \( f_1 = \ldots = f_m = 0 \) is also a solution of \( A_1 = \ldots = A_p = 0 \).
Chapter 7. Algebraic Tools in Semi-Physical Modeling

and conversely a nonsingular (no separant or initial is identically zero) solution of \( A_1 = \ldots = A_p = 0 \) is a solution of \( f_1 = \ldots = f_m = 0 \).

Recall now that a differential ideal \( \Sigma \) is called prime if for any differential polynomials \( f_a, f_b \) such that \( f_a f_b \in \Sigma \), either \( f_a \in \Sigma \) or \( f_b \in \Sigma \). The real crux with Ritt’s algorithm is that without primeness \( A \) may not be a characteristic set for \( [\Sigma] \). The following theorem due to Ritt [80] can then be used.

**Theorem 7.4** If \( A = \{A_1, \ldots, A_p\} \) is an autoreduced set such that the first polynomial \( A_1 \) cannot be factored over the coefficient field and \( A_2, \ldots, A_p \) all are of degree one in their leaders, then \( A \) is a characteristic set of a prime differential ideal.

Thus, the theorem provides a way of afterwards checking primeness of \( \Sigma \).

In case \( \Sigma \) is not prime, Ritt [80] has outlined an algorithm (known as Ritt’s factorization test) that checks for primeness and if not returns two differential polynomials \( f_a \) and \( f_b \) which are reduced with respect to \( A \) such that \( f_a f_b \in [A] \). The problem with this test is that the computational complexity can be very high, though in many practical situations it can still be carried out [25]. Suppose that the differential ideal \( \Sigma \) is such that for any polynomial \( f \) and any integer \( i \) it holds that \( f^i \in \Sigma \Rightarrow f \in \Sigma \). An ideal with this property is called radical (in some literature perfect), denoted by \( \sqrt{\Sigma} \), and is of interest since a solution to the elements of \( \Sigma \) also will be solutions to all elements in \( \sqrt{\Sigma} \).

Suppose that the factorization test returned \( f_a \) and \( f_b \). Since it holds (see [80]) that

\[
\sqrt{[\Sigma, f_a f_b]} = \sqrt{[\Sigma, f_a]} \cap \sqrt{[\Sigma, f_b]},
\]

the original problem can be split into two parts. Applying the factorization test and Ritt’s algorithm recursively on the new sets of polynomials eventually results in

\[
\sqrt{[\Sigma]} = \Pi_1 \cap \Pi_2 \cap \ldots \cap \Pi_r,
\]

where all \( \Pi_1, \ldots, \Pi_r \) are prime differential ideals, whose characteristic sets \( A_1, \ldots, A_r \) are given by Ritt’s algorithm.

**Characteristic Sets in Semi-Physical Modeling**

Because the starting point in semi-physical modeling often is a set of differential equations Ritt’s algorithm can be employed directly. The only practical consideration is the choice of ranking, which preferably is chosen to be

\[
u_1^{(i)}(t) < v_2^{(i)}(t) < \ldots < v_r^{(i)}(t).
\]

This means that all states are less desirable than the outputs and their derivatives, which in turn are less desirable than the inputs and their derivatives. Notice that the user still has the freedom to choose ranking among the states etc.

**Example 7.16** Consider an immersion heater affecting the temperature \( y(t) \) of a liquid in a vessel. Assume that the voltage over the heater element is the control signal \( u_1(t) \), causing the non-measured temperature \( T(t) \) of the element to change. The resistance of the heater element is temperature dependent and according to the manufacturer it can be approximated by
7.3 Obtaining Algebraic Differential Equations

\[ R(t) = \frac{R_0}{1 - \alpha T(t)} \], where \( R_0 \) is a nominal resistance and \( \alpha \) an unknown constant (though with a value such that \( R(t) \) is well defined in the interesting operating region). Assuming homogeneous temperatures (all in Kelvin), a bit of straightforward modeling gives the heat balance equations

\[
\begin{align*}
W_1 T(t) &= \frac{1}{R_0} u_1(t)^2 (1 - \alpha T(t)) - K_1 (T(t) - y(t)) \\
W_2 y(t) &= K_1 (T(t) - y(t)) - K_2 (y(t) - u_2(t)),
\end{align*}
\]

with \( W_1, W_2 \) being heat capacity coefficients, \( K_1, K_2 \) heat transfer coefficients, and \( u_2(t) \) being the measured temperature of the surroundings. The natural input signals are \( u_1(t), u_2(t) \) and the output signal is \( y(t) \). Using ranking \( u_1(t) < u_2(t) < y(t) < T(t) \), Ritt's algorithm gives the differential polynomials

\[
A = \{ R_0 W_1 W_2 y(t) + \alpha W_2 u_1(t)^2 y(t) + R_0 (K_1 W_1 + K_1 W_2 + K_2 W_1) y(t) + \alpha (K_1 + K_2) u_1(t)^2 y(t) \\
+ K_1 K_2 R_0 y(t) - \alpha K_2 u_2(t) u_1(t)^2 - K_1 u_1(t)^2 - R_0 K_2 W_1 u_2(t) - K_1 K_2 R_0 u_2(t), \\
W_2 y(t) + (K_1 + K_2) y(t) - K_1 T(t) - K_2 u_2(t) \},
\]

where the first element is a differential polynomial in known variables only.

Let us now take a look at an example not given in state space form.

**Example 7.17** Many biochemical reactions are controlled by an enzyme catalyst. Mohler [75] models one such system by

\[
\begin{align*}
\dot{x}_1(t) &= a_1 x_2(t) - b x_3(t) x_1(t) \\
\dot{x}_2(t) &= b x_3(t) x_1(t) - (a_1 + a_2) x_2(t) \\
\dot{x}_3(t) &= (a_1 + a_2) x_2(t) - b x_3(t) x_1(t) \\
\dot{y}(t) &= a_2 x_2(t) \\
K_1 &= x_1(t) + x_2(t) + y(t) \\
K_2 &= x_2(t) + x_3(t),
\end{align*}
\]

where \( x_1(t), x_2(t) \) are concentrations of a chemical substrate upon which enzymes act, \( x_3(t) \) is the enzyme concentration, and \( y(t) \) is the end-product concentration, which is the only measurable quantity. The parameters \( a_1, a_2 \) and \( b \) are unknown positive rate constants. The first two equations express mass balances, the following two equations describe the assumption that the complex formed by combination of enzyme and substrate decomposes into an enzyme and an end-product, while the algebraic equations specify that the total amount of substrate \( (= K_1) \) and enzyme \( (= K_2) \) are both constant. If the ranking is \( y(t) < x_1(t) < x_2(t) < x_3(t) \), then the first element of the characteristic set obtained running Ritt's algorithm is

\[
a_2 \dot{y}(t) - b \dot{y}(t)^2 - a_2 b y(t) \dot{y}(t) + a_2 (a_1 + a_2 + b K_1 + b K_2) \dot{y}(t) + a_2^2 b K_2 y(t) - a_2^2 b K_1 K_2,
\]

which is a reasonable model structure to start with.

7.3 Obtaining Algebraic Differential Equations

The assumption that the model structure consists of a number of polynomial differential equations is clearly quite restrictive. Most of the functions encountered in traditional modeling belong to the larger class of *elementary functions* (which we from now on
concentrate on). These additionally involve the trigonometric, inverse trigonometric, exponential and logarithmic functions as well as all other functions that can be constructed from these by adding, subtracting, multiplying, taking quotient, or forming compositions. One attractive feature of the elementary functions is that they satisfy a differential polynomial, i.e., they are differentially algebraic\(^6\); see Rubel and Singer [82]. In practice, this means that almost any ordinary differential equation is a solution to an algebraic one.

**Example 7.18** A junction diode is sometimes modeled by a nonlinear resistor according to
\[
i(t) = I_S e^{\frac{u(t)}{U_T} - 1},
\]
where \(I_S\) and \(U_T\) are unknown diode specific parameters. The solution to this constitutive equation satisfies the algebraic differential equation (ADE)
\[
\frac{dI(t)}{dt} - \frac{1}{U_T} \frac{dU(t)}{dt} i(t) = 0.
\]

**Example 7.19** The tangential motion of a pendulum consisting of a “particle” of mass \(m\) at the end of a weightless rod of length \(L\) is given by
\[
\ddot{\phi}(t) = \frac{k}{m} \dot{\phi}(t) - \frac{g}{L} \sin(\phi(t)),
\]
where \(\phi(t)\) denotes the angle of the pendulum from the vertical axis, \(k\) is a friction coefficient, and \(g\) is the acceleration due to gravity. The solution to this equation satisfies the ADE
\[
\phi^{(3)}(t) \left( \phi^{(3)}(t) + \frac{k}{m} \dot{\phi}(t) \right) + \dot{\phi}(t)^2 \left( \phi(t)^2 + \frac{k^2}{m^2} \right) + \frac{k}{m} \dot{\phi}(t) \dot{\phi}(t)^3 + \dot{\phi}(t)^2 \left( \frac{k^2}{m^2} \dot{\phi}(t)^2 - \frac{g^2}{L^2} \right) = 0.
\]

A first observation is that solving an ordinary differential equation \(f(y^{(n)}(t), \ldots, y(t)) = 0\) is the process of finding \(y(t)\) satisfying \(f\), while obtaining an algebraic differential equation can be seen as the opposite, i.e., finding an algebraic differential equation having \(y(t)\) as a solution.

To begin with, let \(\Gamma\) be a set of ordinary differential expressions, each written with a common denominator (which is assumed to be nonzero). This in particular means that negative powers can never appear in the expressions to be considered. Also, introduce \(\Omega\) as the set of all “factors” in \(\Gamma\) that are “non-polynomial”. The following example clarifies this non-standard terminology.

**Example 7.20** If \(\Gamma\) is the following set of differential expressions
\[
\{ \dot{x}_1(t)^2 + \theta_1 \cdot \sin(e^{x_2(t)}) \cdot \dot{x}_1(t) - \theta_2 \cdot \sqrt{x_1(t)^2 + \dot{x}_2(t)} \cdot \cos(x_1(t)), \ \dot{x}_2(t) + \theta_3 \cdot \cos(x_1(t)) \},
\]
then
\[
\Omega = \{ \sin(e^{x_2(t)}), \ \sqrt{x_1(t)^2 + \dot{x}_2(t)}, \ \cos(x_1(t)) \}.
\]

\(^6\)Beyond the algebraic functions we find the **transcendently transcendental** functions which do not satisfy an algebraic differential equation (at least not a finite one). Among such functions we may mention the gamma and the Riemann zeta functions. See Rubel [81] for a more detailed account of these special functions.
### Table 7.1: The most common elementary functions and differential expressions which they satisfy.

<table>
<thead>
<tr>
<th>Expression Ω</th>
<th>Variable(s) to add</th>
<th>Expression(s) to add, Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d^n) (\frac{n}{m} \in \mathbb{Q}^+ \setminus \mathbb{N})</td>
<td>(z_i = d^n)</td>
<td>(z_i^m - d^n)</td>
</tr>
<tr>
<td>(d^a) (a \in \mathbb{R}^+ \setminus \mathbb{N})</td>
<td>(z_i = d^a)</td>
<td>(z_i d - a z_i d)</td>
</tr>
<tr>
<td>(e^d) (a \in \mathbb{R}^+ \setminus {1})</td>
<td>(z_i = e^d)</td>
<td>(z_i - dz_i)</td>
</tr>
<tr>
<td>(a^d) (a \in \mathbb{R}^+ \setminus {1})</td>
<td>(z_i = a^d)</td>
<td>(z_i = \ln(a) z_i d)</td>
</tr>
<tr>
<td>(\ln(d))</td>
<td>(z_i = \ln(d))</td>
<td>(z_i d - d)</td>
</tr>
<tr>
<td>(a \log(d)) (a \in \mathbb{R}^+ \setminus {1})</td>
<td>(z_i = a \log(d))</td>
<td>(\ln(a) z_i d - d)</td>
</tr>
<tr>
<td>(\sin(d))</td>
<td>(z_i = \sin(d), z_{i+1} = \cos(d))</td>
<td>(z_i - \bar{d} z_{i+1}, z_i^2 + z_{i+1}^2 - 1)</td>
</tr>
<tr>
<td>(\cos(d))</td>
<td>(z_i = \cos(d), z_{i+1} = \sin(d))</td>
<td>(z_i + \bar{d} z_{i+1}, z_i^2 + z_{i+1}^2 - 1)</td>
</tr>
<tr>
<td>(\tan(d))</td>
<td>(z_i = \tan(d))</td>
<td>(z_i - d (1 + z_i^2))</td>
</tr>
<tr>
<td>(\cot(d))</td>
<td>(z_i = \cot(d))</td>
<td>(z_i + d (1 + z_i^2))</td>
</tr>
<tr>
<td>(\sec(d))</td>
<td>(z_i = \sec(d))</td>
<td>(z_i = z_i \tan(d) d)</td>
</tr>
<tr>
<td>(\csc(d))</td>
<td>(z_i = \csc(d))</td>
<td>(z_i + z_i \cot(d) d)</td>
</tr>
<tr>
<td>(\arcsin(d))</td>
<td>(z_i = \arcsin(d))</td>
<td>(z_i \sqrt{1 - d^2} - d)</td>
</tr>
<tr>
<td>(\arccos(d))</td>
<td>(z_i = \arccos(d))</td>
<td>(z_i \sqrt{1 - d^2} + d)</td>
</tr>
<tr>
<td>(\arctan(d))</td>
<td>(z_i = \arctan(d))</td>
<td>(z_i (1 + d^2) - d)</td>
</tr>
<tr>
<td>(\arccot(d))</td>
<td>(z_i = \arccot(d))</td>
<td>(z_i (1 + d^2) + d)</td>
</tr>
<tr>
<td>(\text{arcsec}(d))</td>
<td>(z_i = \text{arcsec}(d))</td>
<td>(z_i d \sqrt{d^2 - 1} - d)</td>
</tr>
<tr>
<td>(\text{arccsc}(d))</td>
<td>(z_i = \text{arccsc}(d))</td>
<td>(z_i d \sqrt{d^2 - 1} + d)</td>
</tr>
</tbody>
</table>

Suppose, e.g., that one of the elements in \(\Omega\) is the exponential function \(e^{\bar{z}(t)}\). By introducing a new variable \(z_1(t)\) to be \(e^{\bar{z}(t)}\), we can regard the new expression \(z_1(t) - e^{\bar{z}(t)}\). Differentiating this with respect to \(t\) and utilizing the fact that \(z_1(t) = e^{\bar{z}(t)}\) we get

\[
\dot{z}_1(t) = \dot{\bar{z}}(t) z_1(t),
\]

which obviously is a differential polynomial describing the exponential function. The idea of introducing new variables and generating new differential expressions can, of course, be generalized to other simple elementary functions; see Table 7.1. The only time differentiation is avoided is when the considered function is a rational power, in which case the function is described by a static expression. Notice also how the trigonometric identity is used for sin- and cos-expressions. The main reason for this is that static relationships are considered to be “simpler” than those involving derivatives. For some of the trigonometric functions (sec, arcsin, etc.) we see that the added expression contains a subexpression that is “non-polynomial”. By simply adding this subexpression to \(\Omega\), the
procedure above can be reapplied, thereby giving a differential polynomial description for this function as well.

In fact, the chain rule assures that this idea can be applied for composite functions too. Take for instance the function $e^{\sqrt{x(t)}}$. As above we let $z_1(t) = e^{\sqrt{x(t)}}$, which after differentiation, substitution and common denominator reformulation yields

$$z_1(t)\sqrt{x(t)} - \dot{x}(t)z_1(t). \quad (7.16)$$

By introducing the variable $z_2(t) = \sqrt{x(t)}$ we arrive at

$$\dot{z}_1(t)z_2(t) - \dot{z}(t)z_1(t), \quad z_2^2(t) = x(t), \quad (7.17)$$

i.e., two differential polynomials having zero at $z_1(t) = e^{\sqrt{x(t)}}$. In the general case the inner expression is an arbitrary elementary differential subexpression $d(t)$. By the chain rule, $\dot{d}(t)$ will show up in the differentiated expression. The “non-polynomial” functions of $\dot{d}(t)$ can then be assigned new names and described by new expressions according to the third column of Table 7.1. Proceeding stepwise in this recursive manner eventually results in a number of differential polynomials satisfying the original set of “non-polynomial” functions. The last claim follows as all “non-polynomial” functions are replaced by suitable differential expressions. Also, because $\dot{d}(t)$ is a finite expression containing finitely many “non-polynomial” functions it follows that the outlined procedure stops, and thus it is an algorithm.

More formally, using pseudo-code, the algorithm is:

1. Input is a set of ordinary differential expressions $\Gamma$.
2. Put each expression in $\Gamma$ on common denominator form.
3. Initialize $\Omega$ to be the set of all “non-polynomial” functions in $\Gamma$.
4. Mark all elements of $\Omega$ as not considered.
5. while not all elements in $\Omega$ are marked as considered do
   a. Let $\Omega$ be any element of $\Omega$ not yet considered.
   b. For $\Omega$ introduce new variable(s) according to Table 7.1 (second column).
   c. $\Gamma = \Gamma \cup \{\Delta_\Omega\}$, where the expressions $\Delta_\Omega$ are written on common denominator form and given by the third column of Table 7.1.
   d. Mark the function $\Omega \in \Omega$ as considered. In case $\Omega = \sin(d) \cos(d)$, also add $\cos(d) \sin(d)$ to $\Omega$ and mark it as considered.
   e. $\Omega = \Omega \cup \{\text{all “non-polynomial” functions found in } \Delta_\Omega\}$
5. Return(Substitute($\Omega = z, \Gamma$)). Here, $z$ represents the introduced variables associated with the functions in $\Omega$.

Let us demonstrate the procedure in an example.

**Example 7.21** Consider again Example 7.20, where the first 3 steps already have been performed. The $\Omega$-expressions considered are underlined.

1. 4a. $\Omega = \sin(e^{x(t)})$.
2. 4b. $z_1(t) = \sin(e^{x(t)}), \quad z_2(t) = \cos(e^{x(t)})$. 


4c. \( \Gamma = \Gamma \cup \{ z_1(t) - \dot{z}_2(t)e^{x_2(t)}z_2(t), \, z_1(t)^2 + z_2(t)^2 - 1 \} \).

4d. \( \Omega = \{ \sin(e^{x_2(t)}), \, \sqrt{x_1(t)^2 + \dot{x}_2(t)}(t), \, \cos(x_1(t)), \, \cos(e^{x_2(t)}) \} \).

4e. \( \Omega = \{ \sin(e^{x_2(t)}), \, \sqrt{x_1(t)^2 + \dot{x}_2(t)}(t), \cos(x_1(t)), \, \cos(e^{x_2(t)})e^{x_2(t)} \} \).

II 4a. \( \Omega = \sqrt{x_1(t)^2 + \dot{x}_2(t)} \).

4b. \( z_3(t) = \sqrt{x_1(t)^2 + \dot{x}_2(t)} \).

4c. \( \Gamma = \Gamma \cup \{ z_3(t)^2 - x_1(t)^2 - \dot{x}_2(t) \} \).

4d. \( \Omega = \{ \sin(e^{x_2(t)}), \, \sqrt{x_1(t)^2 + \dot{x}_2(t)}(t), \, \cos(x_1(t)), \, \cos(e^{x_2(t)})e^{x_2(t)} \} \).

III 4a. \( \Omega = \cos(x_1(t)) \).

4b. \( z_4(t) = \cos(x_1(t)), \, z_5(t) = \sin(x_1(t)) \).

4c. \( \Gamma = \Gamma \cup \{ z_4(t) + \dot{x}_1(t)z_5(t), \, z_4(t)^2 + z_5(t)^2 - 1 \} \).

4d. \( \Omega = \{ \sin(e^{x_2(t)}), \, \sqrt{x_1(t)^2 + \dot{x}_2(t)}(t), \, \cos(x_1(t)), \, \cos(e^{x_2(t)})e^{x_2(t)} \} \).

IV 4a. \( \Omega = e^{x_2(t)} \).

4b. \( z_6(t) = e^{x_2(t)} \).

4c. \( \Gamma = \Gamma \cup \{ \dot{z}_6(t) - \dot{x}_2(t)z_6(t) \} \).

4d. \( \Omega = \{ \sin(e^{x_2(t)}), \, \sqrt{x_1(t)^2 + \dot{x}_2(t)}(t), \, \cos(x_1(t)), \, \cos(e^{x_2(t)})e^{x_2(t)} \} \).

The substitution in step 5 finally results in a set of 8 differential polynomials:

\[
\begin{align*}
\{ & \dot{z}_1(t)^2 + \theta_1 z_1(t)\dot{x}_1(t) - \theta_2 z_3(t)z_4(t), \, \dot{x}_3(t) + \theta_3 z_4(t), \\
& \dot{z}_1(t) - \dot{x}_2(t)z_5(t)z_2(t), \, z_1(t)^2 + z_2(t)^2 - 1, \, z_3(t)^2 - x_1(t)^2 - \dot{x}_2(t), \\
& \dot{z}_4(t) + \dot{x}_1(t)z_5(t), \, z_4(t)^2 + z_5(t)^2 - 1, \, \dot{z}_6(t) - \dot{x}_2(t)z_6(t) \}. \end{align*}
\]

The example covers a quite complicated situation. In practice, however, composite functions are rare. The typical situation seems to involve only simple "non-polynomial" functions, and most often no more than 3 of them.

Since the introduced variables typically reflect functions of signals not measured the result from the proposed algorithm is of little use as it stands. However, in terms of differential algebra we can impose the ranking

\[ u_i^{(j)}(t) < y_j^{(j)}(t) < x_k^{(j)}(t) < z_l^{(j)}(t) \quad (7.18) \]

and try to eliminate the \( x \) and \( z \) variables using Ritt's algorithm. This was the procedure successively pursued for Examples 7.18 and 7.19. For Example 7.20 on the other hand, the algorithm did not return a result within 24 hours even though a fast computer with plenty of internal memory was used. This indicates the importance of being able to estimate the complexity of the problem (a matter to be discussed in the next section).

The price paid for introducing new expressions according to Table 7.1 is that extra solutions are added. Take, e.g., the differential polynomial \( \dot{z}_1(t) - \dot{z}_2(t)z_2(t) \) reflecting \( e^{x(t)} \). It is easy to verify that also \( e^{x(t)} + C \) (\( C \) being a constant) satisfy this polynomial, and hence the uniqueness of \( e^{x(t)} \) is lost in the derived formulation. This is fortunately not so serious as it first might look like, because in the estimation step data will reveal the value of \( C \). The important point here is just that the description is flexible enough to comprise the solutions of the original set of polynomials. Another interesting observation is that the introduced polynomials by no means are the only ones describing the considered function. As an example, \( \sin(t) \) satisfies \( \dot{z}(t)^2 + z(t)^2 - 1 \) but also \( \dot{z}(t) + z(t) \). In line
with the previous definition of “simplicity” the suggested algorithm strives to keep the polynomial orders as small as possible, and thus the first one is preferred. Notice that this is not an obvious choice as it could be argued that a linear differential polynomial is simpler than a nonlinear one.

7.4 Aspects on the Computational Complexity

An algorithm is not of much practical value if its computational complexity with respect to some problem dependent properties is unreasonably high. The most common measure of complexity is the time or the number of arithmetic operations needed to arrive at an answer. Another limiting factor of great practical importance here is the storage space needed for intermediate calculations, and yet a third measure is the size of the output (number of terms, memory consumption, etc.).

In general, an algorithm for computing a Gröbner basis for a system of \( m \) equations of degree at most \( d \) in \( n \) variables requires \( d^O(n^2) \) operations, depending on some technical details, see Lazard [58]. Because of the exponential nature of these bounds the computational time needed grows extremely fast when \( d \) and \( n \) increase. Using the best available software and algorithms Lazard [58] stress that problems in 6 to 8 variables can be managed, but that there is no hope to solve problems with 10 or more unknowns. Concerning differential algebra, the complexity picture of Ritt’s algorithm has not yet been studied in much detail. However, a conjecture (based on the test runs performed) is that its average time consumption is comparable to that of Buchberger’s algorithm.

Although the number of equations are few and of seemingly nice structure the computational complexity may still be horrible. In recent years, the increasing awareness of this fact has triggered a lot of research. Apart from better software and better suited data structures for representing polynomials, the algorithms themselves have become much more efficient.

One reason for this improvement is due to the observation that both algorithms spend most of their time computing remainders. Buchberger’s original algorithm is therefore often equipped with various criteria for avoiding to compute \( S \)-polynomials for which one can predict that they reduce to 0. Following this approach, quite a few benchmark tests indicates that a 90% decrease in \( S \)-polynomials computed can be obtained [58]. To further reduce the computational time various heuristic strategies for choosing polynomials have also been suggested. Perhaps the most promising one, also applicable in Ritt’s algorithm, is to take remainders between simple polynomials first and then proceed successively with less simple ones. For Buchberger’s algorithm many other heuristic strategies have been proposed as well, see Gebauer and Möller [30] for the current state of the art situation, though for Ritt’s algorithm very little has been done in this direction.

Another observation is that the ordering (ranking) used for elimination generally renders more computations than other orderings (rankings). By employing so-called degree orderings in combination with linear algebra a lexicographic Gröbner basis can be achieved in less time than using the lexicographic ordering directly [58].

The huge amount of storage space needed even for moderate set of equations has two main causes. First of all, the number of terms in a generated intermediate polynomial can be quite high. The second reason is that the size and structure of the coefficients can be very complicated. To a certain extent these difficulties can be reduced using special data structures, and in some cases by employing modular techniques (factorization) [58].
However, the fact still remains that the memory space needed for Gröbner bases calculations in general is exponential in the number of variables [17]. Several test runs also indicate that the situation is no better for Ritt's algorithm.

To get a feeling of what complexity figures we are talking about let us again consider Example 7.21 (containing 8 equations in 8 variables). Using the computer algebra system Maple [13] and Ritt's algorithm on a SUN SPARC 2 workstation the Maple kernel grows to more than 24 MB within 10 minutes. At this point the computational speed decreases drastically because information must now be interchanged between the internal and the disc storage media. If this happens an answer is seldom returned within reasonable time – the problem is just too complicated. For Example 7.21 the computation was actually interrupted after 35 hours with the error message "object too large". In Maple this means that the algorithm tried to generate a polynomial with more than $2^{16} - 1$ (65535) terms. As a matter of fact, this is normally the reason why the computations halt, and not because there is too little memory available.

When it comes to the size of the output (in KB), it provides a lower bound on the storage complexity. Philosophically speaking, too large an output implies that it is not the algorithm that is too complex, it is the stated problem. As the modeler often has a good understanding of what orders and degrees that are feasible in the IO-expressions, upper bounds on these would be quite welcome.

For state space models of the form discussed in Section 7.1, Forsman [26] has proved that the order of the IO-expression is at most $n$ (the number of states); it is equal to $n$ if and only if all states are algebraically observable. Since this is the case for Examples 7.6 and 7.7, both having 2 states, the IO-expressions are second order differential and difference equations, respectively. Using differential algebra there is a similar result. Suppose that the system $J = \{S\}$ is a prime differential ideal described by a characteristic set $A = \{A_1, \ldots, A_p\}$ under some ranking. If no leading variable of an $A_j \in A$ is an input, then Glad [32] has shown that the order of the system is

$$\nu = \sum_{i=1}^{p} \nu_i,$$

(7.19)

with $\nu_i$ being the order of the leader of the $i$:th differential polynomial in $A$. The main point here is that $\nu$ will be the same regardless of the ranking used as long as the inputs are the same. Thus impose a ranking known to be computationally efficient, such as the state space ranking

$$u_1(t) < x_1(t) < \ldots < x_n(t) < \dot{x}_1(t) < \ldots < \dot{x}_n(t) < \ldots < y_j(t).$$

(7.20)

The name comes from the fact that if $\Sigma$ is a state space model, then $A = \Sigma$, so with $n$ states the IO-expression is at most of order $n$ (and equal to $n$ when all states are algebraically observable [32]). With this ranking it immediately follows that the order of the IO-expression of Example 7.16 is at most 2. For Example 7.17 the characteristic set obtained using state space ranking has the leaders $y(t), x_3(t), \dot{x}_1(t)$ and $\dot{x}_2(t)$, and accordingly a second order input-output relation is expected.

Since the terms in this relation can be of any degree the knowledge about the order gives no direct clue to the size of the IO-expression. Unfortunately, it here seems hard to develop bounds that are not overly conservative, at least for bounds reflecting the number of terms in the IO-polynomial. Using the Gröbner basis setting one of the
few existing results is given by Forsman [26], who has derived an upper bound for the total degree\(^7\) of the resulting input-output polynomial. The idea is to first compute \(H = \{h, t_f h, \ldots, t_f^n h\}\) (in the time-discrete case \(H = \{h, \lambda h, \ldots, \lambda^n h\}\)). If the input and its derivatives are treated as parameters, then the total degree, \(d_t\), of the input-output relation is bounded by the product of the \(n\) largest \(\text{Tdeg}(\cdot)\) [26]. In Example 7.6 \(u(t)\) and \(pu(t)\) are treated as parameters, meaning that \(\text{Tdeg}(H_i) = 1\) for \(i = 1, 2, 3\), and hence the total degree of the input-output relation is 1, i.e., it is linear in the output and its derivatives. Notice that this way of treating the inputs typically results in complicated coefficients, see the expressions within large brackets on page 77. With the same procedure the total degree of the IO-expression of Example 7.7 is at most 4 (in fact it is 4). By simple combinatorics the maximum number of terms (still regarding the inputs as parameters) of the IO-polynomial is

\[
\left( \begin{array}{c} n + 1 + d_t \\ d_t \end{array} \right) = \frac{(n + 1 + d_t)!}{d_t!(n + 1)!}.
\]

(7.21)

For Examples 7.6 and 7.7 we get at most 4 and 35 terms, respectively. The latter quantity is interesting in the sense that the true IO-polynomial only contains 6 terms, which indicates that this bound is rather conservative. Also, and like the other bounds previously discussed, the maximum possible number of terms grows rapidly when \(n\) and \(d_t\) increase.

Despite the pessimistic complexity figures one should, however, remember that these measures reflect the worst possible case, and that the "average" modeling situation seems to be much more manageable [15]. One reason for this is that models derived from physics tend to have an inherently sparse structure; the models having worst case complexity are often pen and paper constructions. Although this observation, the complexity nuisance is still a severe or and as we have to live with it the obvious question arising is what model sizes (number of variables etc.) and model structures are tractable in general?

Here we have to be very subjective. A rule of thumb is that the number of non-measurable variables should be less than 5, but this very much depends on the structure of the problem. The "more" linear the system is the better the chance of success; bilinear systems, e.g., are very to be reasonable in this respect. Furthermore, models where the nonlinearities show up late (when taking Lie derivatives etc.) are nice; nonlinear terms in the output usually leads to excessive computations, especially if the degree of such a term is high.

### 7.5 Representing Derivatives in Discrete Time

When working with time-continuous models the elimination procedure hopefully results in one or more differential equations in the measured input and output signals. Then, to meet Requirement 6.4, we first need a way to calculate or approximate the derivatives of these signals. This can in principle be done in two different ways.

In view of the definition of derivatives the simplest and most natural approach is to replace the derivatives with the usual difference quotient, i.e., to use the Euler forward

\[f = 2x_1^2x_2 + x_1x_2^2x_2,\]

then the total degree of \(f\) is \(\text{Tdeg}(f) = \max(2+3+0, 2+0+1, 0+1+0) = 5\). More precisely, using the terminology of Definition 7.7, the total degree of \(f\) is the maximum \(\sum_{i=1}^{n} a_i\), such that the coefficient \(a_n\) is nonzero.
Table 7.2: Suggested regressors for Examples 7.6, 7.7, 7.16 and 7.17. The derivatives of the time-continuous models are approximated using the Euler forward method. A \( \dagger \) means that the coefficient of the corresponding regressor is a known constant. The first 1 in the last example implies that a static level should be estimated.

\[
x^{(n)}(t) = \left( \frac{1}{T} \right)^n \sum_{k=0}^{n} \frac{n!}{(n-k)!k!} (-1)^{n-k} x(t + k), \quad n \in \mathbb{N}^+.
\] (7.22)

The weakness of Euler's method lies in the fact that the approximation error is proportional to the sampling interval \( T \), and hence it must be chosen quite small in order to attain acceptable quality. To overcome this problem in simulation the usual advice is to employ higher order approximations (Runge-Kutta methods etc.; consult Dahlquist and Björk [16] for an overview of other methods.). In our situation such an approach is much less attractive, since it would lead to a set of difference equations of higher order than the original set of differential equations.

However, this very idea constitute the basis for the second possibility, namely to regard all derivatives as separate signals that are created from input-output data using appropriate filters. What approach to follow here depends heavily on the modeling situation, and is a typical choice of the model designer. We will not further judge this choice, though from now on we mainly restrict the discussion to the former approach.

Having a time-invariant difference equation it is a trivial symbolic operation to shift the signals so that the highest time index of any signal is \( t \). Assuming causality, the resulting expression will contain an output variable \( y(t) \). Knowing this, we can try to solve for \( y(t) \), and at the same time separate the parameters and the regressors into two parts: \( f(\eta) \) and \( \varphi(t) \). The main point here is that \( \varphi(t) \) holds combinations of variables whose values are known at time instance \( t \), and thus \( y(t) = f(\eta)^T \varphi(t) \) is a predictor form. Pursuing this procedure for some of the models earlier discussed gives regressors according to Table 7.2. Notice that it may very well happen that an \( \eta_i \) is completely known, which in an ideal situation means that this term should be moved to the known
left-hand side. For the sake of flexibility, we here allow the modeler to decide whether
to estimate this coefficient or not.

From Table 7.2 an apparent problem with the predictor formulation is that variables
may show up in the denominator of a regressor. As a consequence, there is a risk that
such a regressor becomes numerically undefined. Somewhat less obvious is that in solving
for $y(t)$ it might be impossible to separate the parameters from the regressors, which the
following example illustrates.

**Example 7.22** In Cartesian coordinates the motion of a planar pendulum is modeled by
(neglecting the air resistance)

$$
\begin{align*}
ml\ddot{x}(t) + f(t)x(t) \\
ml\ddot{y}(t) + f(t)y(t) + mgL \\
x^2(t) + y^2(t) - L^2,
\end{align*}
$$

where $m$ is the mass and $L$ the length of the pendulum, $g$ is the gravitational constant, $f(t)$ is
the tension in the rod, and $x(t), y(t)$ are the horizontal and the vertical positions, respectively.

Here the algebraic equation states that the motion is constraint to be along a circle of radius $L$. Applying Ritt's algorithm under the ranking $y^{(j)}(t) < x^{(j)}(t) < f^{(j)}(t)$ in conjunction with
Euler's method for approximating the derivatives yields the time-discrete output equation

$$
\begin{align*}
L^4 y(t) - L^2 y(t-2)^2 y(t) + L^2 y(t-2)y(t-1)^2 - 2L^4 y(t-1) + \\
gy(t-2)^4 - 2gL^2 y(t-2)^2 + L^4 y(t-2) + L^4 g &= 0.
\end{align*}
$$

Solving for $y(t)$ gives regressors with $L^2 (L^2 - y(t-2))$ in the denominator, which means that the
parameters and the regressors cannot be completely separated. The immediate consequence of
this observation is that the predictor cannot be reparameterized to fit into the linear regression
framework.

In addition, as the output can occur implicitly in the IO-relation, it can also be
impossible to solve for $y(t)$, at least uniquely.

An alternative in both these situations is to avoid the predictor formulation and
instead work directly with the IO-equation. After reparameterization, the key point is
that the result is a linear regression, but now the structure is $\theta^T \varphi(t) = 0$, where, unlike
the predictor case, $\varphi(t)$ also contains unknown outputs. So instead of minimizing a
prediction error we here have to content ourselves to minimize an equation error. To
be able to do this we first need to normalize the equation, i.e., one of the $\theta_i$ must be
assigned a value. Notice that the choice of $\theta_i$ indirectly means that special assumptions
on the underlying noise process are made.

Before leaving the symbolic world, it should be pointed out that the choice of dis-
cretization method indirectly involves the choice of output. When using the Euler for-
ward method, $y(t)$ originates from the highest order derivative of the output. On the
other hand, using the filtering approach it is quite natural to directly take $y(t)$ from the
IO-equation as output. More to the point, it is here a philosophic question what signal
to consider as the output.

---

8Notice that the model is a predictor if and only if the signals used in the filters are known at time $t$. 
Selecting the Regression Model

The end-product of the symbolic effort is a model structure delivered either as a predictor \( y(t) = f(\eta)^T \varphi(t) \), or as an equation of the form \( f(\eta)^T \varphi(t) = 0 \). Observe that the only approximation imposed this far lies in the numerical representation of the derivatives, which means that the structure of the original equations is preserved.

It is true that the parameters of \( f(\eta) \) can be estimated using iterative schemes (such as Equation (2.29)), but because of the drawbacks accounted for in Section 2.4 we would like to avoid this, though leaving it as an open possibility. Furthermore, by only accepting the “best” few regressors the inevitable by-product is that the physical interpretation of the parameters is lost. This motivates a more pragmatic view of the parameters, and makes the reparameterization \( \theta = f(\eta) \) to linear regression form a sound approach.

In the predictor case the parameters \( \theta \) can be estimated directly using the least-squares algorithm according to Equation (2.31). By (if necessary) fixing one of the parameters (to avoid the degenerate solution \( \theta = 0 \)), and regarding the corresponding term as the output, the same algorithm can be used in the equation error case.

An underlying assumption in the least-squares formulation is that the errors are confined to the output, i.e., the regressors \( \varphi(t) \) are assumed to be measured exactly. When errors, \( e_\varphi(t) \), are also considered to be present in the regressors, \( y(t) = \theta^T (\varphi(t) + e_\varphi(t)) + e(t), \) or \( 0 = \theta^T (\varphi(t) + e_\varphi(t)), \)

it is natural to change the optimization criterion to be the sum of squared prediction and regressor errors. This is commonly known as a total linear least-squares problem\(^1\), which can be solved via singular value decomposition as is demonstrated by Golub and Van Loan [35]. While this seldom is used in the predictor case, it is often the natural approach to follow in the equation error case. The reason is that if we assume that only \( y(t) \) is corrupted by noise, then, since \( \varphi(t) \) typically holds more than one term containing \( y(t) \), at least one of the regressors remaining after normalization will be affected by noise. In Example 7.22, e.g., there were two such terms, namely \( L^4 y(t) \) and \( L^2 y(t - 2)^2 y(t) \).

In other situations where there are correlations between the regressors, the corresponding least-squares problem will be ill-conditioned, thereby causing uncertain pa-

\(^1\)Sometimes referred to as an error-in-variables minimization problem.
rameter estimates. Notice that this is likely to occur when $\varphi(t)$ has many regressors of similar structure. The actual root of the problem lies in the awkwardness of computing the inverse of the regression matrix $R_N$. To reduce this problem so-called ridge regression can be used (see Draper and Smith [20]). In its simplest form, the idea is to make $R_N$ better conditioned by adding to it a positive matrix $\delta I$, thus giving the parameter estimate

$$\hat{\theta}_N = \left( \sum_{t=t^0}^{N} \varphi(t)\varphi^T(t) + \delta I \right)^{-1} \left( \sum_{t=t^0}^{N} \varphi(t)y(t) \right) = [R_N + \delta I]^{-1} f_N. \quad (8.3)$$

This modification means that the resulting model is made up of estimates that are not true least-squares - they are biased in a way determined by the positive user-tunable parameter $\delta$. The specific value of $\delta$ is often based on an inspection of a plot of the regression estimates against $\delta$ (a so-called ridge trace plot).

The point here is that even if a linear regression formulation is obtained, there still will be more options to consider, and again it is up to the user to decide which approach to follow. We will not further consider parameter estimation algorithms, but assume that a reasonable ensemble of such schemes is at hand. Observe that this is precisely what many commercial software packages, such as MATLAB [72] and its toolboxes, provide to us. For those wishing to go into this topic more thoroughly, a wealth of material is found in, e.g., Golub and Van Loan [35] and Dahlquist and Björk [16].

Having chosen an estimation algorithm, henceforth pure least-squares, we next discuss some statistical procedures for selecting the "best" regressors.

### 8.1 Model Selection Procedures

As we think of it, model selection is the procedure to pick out $m$ out of $n$ regressors that "best" can describe the true system. The obvious approach would here be to use a criterion for expressing what "best" means and then go through all possibilities. However, this is a rather cumbersome procedure which in general involves quite time-consuming computations. For example, if we are given 20 ($n$) regressors, but are willing to accept a model with at most 7 ($m$) parameters, then a total of 137,979 models have to be estimated and investigated. In practice, such amount of work cannot be tolerated, thus making some sort of selection procedure which shortens this task necessary.

The common idea behind such statistical screening procedures is to explore the most promising regressors in one way or another. This implies that only a small number of candidate models is considered, and consequently there is a risk of missing the best one. Also because of this, different selection strategies do not always lead to the same solution when applied to the same problem. These are matters one should not be too concerned with since several models, or indeed no model at all, may be acceptable for our purposes.

Most selection algorithms discussed in the literature operate in an automatic mode, i.e., once the algorithm is invoked the user can neither guide the search nor investigate intermediate models. This is often unsatisfactory as the selection criterion hardly ever is able to capture all modeling aspects; in particular it is hard to formalize and incorporate personal judgments. Furthermore, a lot of insight about the quality of the regressors can be gained by working in an iterative manner where estimation is followed by different validation tests. All in all, these observations call for some kinds of semi-automatic
consultative procedures which allow and encourage user interaction. Since there exist well-reputed automatic selection strategies, we suggest to modify a choice of these to meet our interactive needs.

Unless otherwise stated the strategies discussed below are special variants of algorithms found in, e.g., Draper and Smith [20]. As usual, we will assume that the data to be used have been appropriately polished (means are removed, measurements are scaled in suitable ways etc.).

- The **backward elimination** procedure is perhaps the natural one to start with, as the regressors originates from physical reasoning. It begins with the full regression equation, i.e., an equation with \( n \) parameters, and eliminates regressors one at a time, until (it is hoped) a reasonable model is found. At each stage the procedure suggests to remove the regressor that either decreases the model performance “least” or improves it “most”. In fact, all possibilities to remove a single regressor are considered in each iteration, hence giving rise to a ranking amongst the models. Notice though that it is still up to the user to follow this advice or not. The ranking of regressors is often based on hypothesis testing (F-test, t-test etc.), where the termination criterion is that the currently least wanted regressor is judged to be significant. When there is a large number of alternatives, it is important to have a simple criterion of goodness. We therefore suggest to use another quality measure, namely the loss function

\[
V_N(\hat{\theta}_N^m, Z^N) = \frac{1}{N} \sum_{t=t^0}^N \epsilon^2(t, \hat{\theta}_N^m),
\]

where the index \( m \) stress that only some of the regressors are present in the considered models. Observe that for this criterion to be fair, the data \( Z^N \) should not have been used for parameter estimation. By graphically viewing all computed loss functions in each step (see the next section), one can quickly get a crude overview of the quality of the remaining regressors.

The procedure is much more economical than searching all possible subsets. Starting, e.g., with 20 regressors, it terminates after having considered at most \( \sum_{k=2}^{20} k = 209 \) models (compare this with the 137,979 models investigated in the exhaustive case).

The most severe drawback with backward elimination is that if the problem is ill-conditioned, then the regressor proposed for removal may be a bad choice, caused by rounding errors. This is really serious since once a regressor has been eliminated it can never enter the model again.

- In the **forward selection** procedure the regressors are introduce one at a time, starting with no regressors at all. At each stage the regressors not yet in the model are ranked after the greatest reduction in, e.g., loss function. Based on this test and other hard to formalize considerations, new regressors are selected and added successively by the user, until finally a good enough model is found (hopefully). The amount of work is also here substantially decreased compared to the all possible subset scheme. For the 7 out of 20 selection problem detailed earlier, at most \( \sum_{k=1}^{14} k = 119 \) models have to be investigated with this method.
Compared to the backward strategy one advantage with forward selection is that it avoids working with more regressors than necessary while improving the model at every step. As a consequence, ill-conditioned problems are less troublesome here. However, when there is correlation between some of the regressors the ranking may not be that objective, especially not at an early stage of the procedure. This is bad news as regressors never are removed from the model. Furthermore, sensible judgment is as always required in the initial steps. For small sample intervals, the most likely first choice of regressor would be $y(t - 1)$ (if available). While $\hat{y}(t|\theta) = \theta_1 y(t - 1)$ often gives an excellent one-step-ahead prediction, the model, of course, is useless for simulation as there is no external input present. We leave this important matter for now and return to it in the next section.

- The combination of the backward and the forward schemes is commonly known as a stepwise regression procedure. Draper and Smith's [20] version of it is as follows. Start with the forward selection procedure, but once a new regressor has been included in the model, investigate the significance of every regressor in it. In other words, judge all regressors of the current model and not only the most recent entrant (which is the case when using forward selection). Based on the significance test either all regressors are kept, or alternatively, one or several regressors are removed. This select and take away pattern is repeated until the tests indicate that it is not worth adding yet another regressor, nor is it recommended to remove any of the chosen ones. For this do work in practice, cycling effects etc. must of course be detected and taken care of.

Here we will use the term stepwise in a wider context, namely whenever the backward and the forward methods are used together. Observe that a rather flexible stepwise procedure is obtained by allowing the user to interactively change strategy during the search process. We can, for example, choose to run the forward scheme 5 iterations, then run the backward procedure 2 iterations, and so on.

With the stepwise procedure, the computational effort increases slightly compared to the above described schemes. In general, however, it is much less demanding than doing an all possible subset search. The major advantage with stepwise regression is that a regressor can be entered or removed at different stages, thus circumventing the problems detailed for the pure backward and forward versions. But as with the other selection procedures, precautions in the initial selection steps are still required. Draper and Smith [20] strongly recommends the use of stepwise regression, not only because of its simplicity, but also because it has been successively applied in quite a few modeling situations. The experiments we have carried out so far certainly verifies the usefulness of working in this manner.

- In the presence of highly intercorrelated regressors, one alternative procedure is the so-called Principal Component Regression (PCR), which in idea differs from the methods discussed earlier, mainly in that it tries to analyze the correlation structure of the regression matrix. Algorithmically, one first computes the $n \times n$ regression matrix $R_N$, its eigenvalues, $\lambda_1, \ldots, \lambda_n$, and normalized eigenvectors, $\gamma_1, \ldots, \gamma_n$ (the normalization is such that $\gamma_j^T \gamma_j = 1$). Next, $n$ artificial regressors, enumerated $\Phi_j(t)$ where $j = 1, 2, \ldots, n$, are created from the original ones. $\varphi(t)$,
via the linear transformation

$$\Phi_j(t) = \sum_{i=1}^{n} \gamma_{i,j} \varphi_i(t), \quad j = 1, 2, \ldots, n, \quad t = t^0, \ldots, N.$$  \hfill (8.5)

Using vector notion, this means that \( n \) new data vectors, \( \Phi_j \) (each with \( N \) entries), are constructed in such a way that they are orthogonal to each other. The main point with this decomposition is that the \( \Phi_j \) corresponding to the largest \( \lambda_j \) (these are the principal components) explain the major part of the data set. Further \( \Phi_j \)'s influence smaller and smaller portions of the data, and hence a ranking of regressors is obtained. From a user's perspective, the only remaining issue to be settled is how many \( \Phi_j \)'s to retain in the model. Although there exist a few selection rules for automatically doing this, it is often worth testing other choices. Assuming that \( m \) out of \( n \) elements have been chosen from \( \Phi(t) \), we denote it \( \Phi_m(t) \), the corresponding least-squares estimate is

$$\hat{\theta}_N^m = \left[ \sum_{t=t^0}^N \Phi_m(t) \Phi_m^T(t) \right]^{-1} \left[ \sum_{t=t^0}^N \Phi_m(t) y(t) \right].$$  \hfill (8.6)

If desired, this model can be re-cast back into the original data representation, and then interpreted in terms of the \( \varphi(t) \)'s. However, this typically leads to models where all elements of \( \varphi(t) \) are present, and none is eliminated by the procedure at any stage. On the other hand, since the regression vectors are replaced by new ones that are orthogonal to each other, the inversion of the new regression matrix should cause no problem. Furthermore, by leaving out some of the "less" important components the procedure allows some noise reduction, but in doing so we also run the risk of discarding useful information. In our opinion, principal component regression is especially valuable as a means for checking the degree of correlation between regressors. It can also be used as a pre-selection step, indicating how many regressors to start with when, e.g., invoking a stepwise search.

- **Latent root regression** is a slight extension of principal component regression. The difference is that the data vector is augmented with the output \( y(t) \) (cf. with total linear least-squares). Otherwise the computations follow along the lines of PCR, but now eigenvalues and eigenvectors are computed for the extended regression matrix. The extra information gained hereby is a measure of the predictability of the response by each eigenvector. Small such values together with small eigenvalues indicates that the corresponding regressors can be dropped.

The method gives a more sophisticated selection criterion compared to that of PCR, but in large it shares the properties of principal component regression. For further details we refer to Draper and Smith [20].

- A procedure we have not yet tested but is worth mentioning in this context is **Partial Least-Squares** regression (PLS), which, e.g., has been successively applied to a wide range of chemical situations. Consult Wold and co-workers [105] for further details on this, as well as for the algorithm itself (actually a family of algorithms). Here, we will just emphasize two interesting features of PLS. A first note is that PLS accounts for errors in the regressors, so in this respect it is close
Chapter 8. Selecting the Regression Model

Step Ia. Suggest: remove $\varphi_4(t)$

Step Ib. Model with 5 regressors; $\varphi_4(t)$ is left out.

Figure 8.1: Step 1: Ranking of the 6 proposed solar-heated house regressors (left) and the simulation result of the 5th order model obtained when removing $\varphi_3(t)$ (right).

to total linear least-squares. Secondly, it removes correlation in the regressors by performing orthogonal projections into a new regressor space in which the actual estimation is carried out. This is indeed close to what is done in PCR, although PLS achieves this in a sequential manner (see algorithm PLS in Wold et al. [105]).

Needless to say, several other selection procedures exist. However, the above assortment represent what many statisticians would deem the best practical algorithms; see the opinions by Draper and Smith [20].

8.2 Selecting Regressors Interactively – An Example

We now finish off this chapter by showing how the second order solar-heated house model was obtained using a stepwise procedure. Recall first that the symbolic computations returned 6 regressors according to Structure (6.10), and that the corresponding simulation of the linear-in-the-parameters model was not that impressive (Figure 6.4).

Starting with all 6 regressors and running the backward elimination procedure 1 iteration gives the loss function ranking of Figure 8.1 (left). The ranking is based on validation data and shows the loss obtained when every regressor is eliminated from the current regressor list one at the time. By far, $\varphi_1(t) = y(t - 1)$ appears to be the most important regressor, because taking it away leads to a substantially increased loss (around 40, which should be compared to 6 for the second most important regressor). Using the same argument, $\varphi_3(t)$ is the least important regressor, and hence it is removed. The simulation shown in Figure 8.1 clearly elucidates the usefulness of doing this.

A second backward iteration step suggests to remove $\varphi_4(t)$, which, as is shown in Figure 8.2, leads to a further improvement of the fit between the measured and the simulated output. Another (the third) backward step reveals a rather interesting thing. According to Figure 8.3 (left) the regressor to be removed is $\varphi_2(t) = u(t - 1)f(t - 2)$, which from a simulation viewpoint is a fatal decision; see the right plot of Figure 8.3.

The reason is again that simulation and one-step-ahead prediction are quite different in nature. Here, the "one-step-ahead-based" ranking actually suggests to remove the
8.2 Selecting Regressors Interactively – An Example

Figure 8.2: Step 2: The second backward iteration step. Taking away \( \varphi_4(t) \) suggested by the ranking (left) yields a 4th order model whose simulated output is compared to the measured output (right).

Figure 8.3: Step 3: Ranking of remaining regressors and the simulation result when leaving out \( \varphi_2(t) \).

effective external input combination, thus, in principle, the test favors a model having time-series characteristics. As (delayed) measured outputs can be used to compensate for the input signals this makes perfect sense in the predictor case, while for simulation, where measured outputs cannot be used, the result is no good. To overcome this problem it can sometimes be worthwhile to adopt a selection criterion that looks \( k \) step ahead. In determining \( y(t) \), this means that measured signals up to time \( t - k \) can be used, whereas outputs after that time instance must be simulated (for pure simulation \( k = \infty \)). However, as was noted in Chapter 2 it can also be risky to use a too large \( k \), especially for models that are unstable or near to unstable. The bottom line here is that \( k \) must be carefully chosen in complicated modeling situations. Nevertheless, we stick to the loss function criterion and conclude that \( \varphi_2(t) \) (together with \( \varphi_1(t) \)) is a regressor to retain in the final model.
Chapter 8. Selecting the Regression Model

Step 4b. Model with 3 regressors, \( \varphi(t), \varphi(t) \) and \( \varphi(t) \) are left out.

Step 5b. Model with 2 regressors, \( \varphi(t) \) and \( \varphi(t) \) are kept in the model.

Figure 8.4: Step 4: Backward elimination. Keeping \( \varphi_2(t) \) and removing the second least wanted regressor, \( \varphi_6(t) \), yields the simulated third order model output to the right.

Figure 8.5: Step 5: Backward elimination. Ranking of remaining regressors (left) and simulated output of the model involving \( \varphi_1(t) \) and \( \varphi_2(t) \) only (right).

It now remains to evaluate the contribution of \( \varphi_5(t) \) and \( \varphi_6(t) \). From the simulations depicted in Figures 8.4 and 8.5 it is clear that these do not influence the model much, hence motivating the second order model

\[
\hat{y}(t|\hat{\theta}_N) = \hat{\theta}_1 y(t-1) + \hat{\theta}_2 u(t-1) I(t-2).
\] (8.7)

At this point, removing any of the remaining regressors is disastrous from a simulation point of view. Keeping \( \varphi_1(t) \) and \( \varphi_2(t) \) it is then interesting to see if any of the dropped regressors can reenter the model and improve its performance. The forward selection step detailed in Figure 8.6 (left) suggests that \( \varphi_3(t) \) is the "best" choice, but as can be seen in the same figure (right) this leads to a worse fit between the simulated and the measured output. The second best alternative is to include \( \varphi_4(t) \) instead. This results in a model with 3 estimated parameters, which shows the best fit between measured and
8.2 Selecting Regressors Interactively – An Example

Figure 8.6: Step 6: Ranking of the 4 regressors not currently in the model (left) and the simulation result of the third order model obtained when also \( \varphi_3(t) \) is included (right).

Figure 8.7: Step 7: Ranking of regressors (same as in Figure 8.6) (left) together with the simulation result of the least-squares fitted model obtained when choosing the regressors \( \varphi_1(t) \), \( \varphi_2(t) \) and \( \varphi_4(t) \) (right).

simulated output among all models that can be created in this way using any of the 6 regressors. See Figure 8.7. Since the improvement is not that striking, it is a matter of taste to choose between this model and the one with only 2 estimated parameters. We favor the latter model mainly because it involves fewer parameters than the former one.

In this case, the conclusion is that stepwise regression very well can be applied. As a matter of fact, we have not yet confronted a set of regressors for which a straightforward use of it results in a much worse model than the best conceivable one.

With this and the discussion of Chapter 7 as a departure point, we next concentrate on software needs in semi-physical modeling.
A Tool for Semi-Physical Modeling

In the previous three chapters we specified and discussed various algorithmic needs in semi-physical modeling. This chapter concerns software requirements in general, and SEMI—a prototype semi-physical modeling tool—in particular.

9.1 Software Requirements and Design Considerations

A computer-based tool for semi-physical modeling requires powerful software pieces of different kinds. On one hand the aim of this section is to identify and discuss these pieces, and on the other hand, to present and motivate the solution we have chosen.

The following tasks should be considered and supported in one way or another:

1. There must be some means for entering and representing prior system knowledge, i.e., the set of equations assumed to describe the underlying system. Thus we require some sort of modeling interface.

2. Computer algebra computations are needed to transform the set of equations into a form suitable (discussed in Chapter 7) for the available estimation algorithms.

3. Numerical and graphical routines for pre-treatment of data (detrending, removal of outliers, filtering etc.) should be offered.

4. There must be facilities for interactively selecting the most “important” regressors as well as schemes for parameter estimation (here different variants of the least-squares algorithm).

5. Validation by visually comparing predicted (with varying prediction horizon) and/or simulated model output with measured signals must be included. Different residual tests should also be offered.

6. It should be possible to work with several models in parallel. Furthermore, the route of modeling actions leading to one specific model should be traceable. These needs call for some sort of model data base responsible for keeping track of the most promising models and aiding in answering what has been done up to now.
7. Optionally, and on top of the other tools, there is a need for a more advanced help system which is able to assist the user throughout the modeling session. Such help systems can come in many shapes, yet a knowledge-base system is perhaps the most sophisticated and interesting alternative.

One pronounced trend in developing scientific software is to embody the theory into a high-level program package accessed via a Graphical User Interface (GUI); see, e.g., Ljung [64] and Van Overschee et al. [92]. With this and the specification list above in mind, it is not hard to imagine the considerable undertaking it would be to code the whole tool from scratch in a conventional programming language like C or Fortran.

Fortunately, this is not necessary, since numerous packages supporting various parts of the listed items are available on the market. So rather than reinventing the wheel, it seems more prudent to interface with some of these software libraries. This was the main decision made initially, and has ever since been the lodestar for all coding activities. Apart from the speed-up in implementation time, other decisive advantages with this approach include standardization, portability to other computer platforms and state of the art routines which are maintained and developed from one version to another. On the other hand, special designed tools force no compromises and are likely to execute faster because communication overheads can be avoided.

In adopting the library philosophy the obvious question is what packages to use. Since no package stand alone is able to fully resolve all listed requirements, several different packages must be combined, preferably in a transparent fashion. To achieve transparency (uniform syntax, usage etc.) and arrive at a common software environment there is a need for a supervisor who is administering the computations. This may very well be entrusted to the graphical user interface, which in addition can be designed to release the user from the burdensome task of entering some commands textually. The highly interactive and subjective process of sorting out a few regressors is a typical example which can be made more efficient this way. Moreover and perhaps more importantly, such an interface provides a suitable means for, at least partly, handling the "bookkeeping problem", i.e., item 6 above. The path to follow when choosing and designing a graphical user interface is tightly coupled to the choice of other computational services, and thus we consider these options first.

**Numerical Computations**

For numerical calculations there exist several efficient, well-reputed and wide spread software sources. One alternative is to directly interface with certain subroutine libraries, such as LINPACK [19] for linear algebra and EISPACK [29, 87] for eigenvalue computations. However, as these libraries constitute the numerical basis both for MATLAB [72] and MATRIX_X [49] a more sophisticated solution is to use one of these high-level tools instead. The additional bargain with this approach is that built-in graphical capabilities can be utilized, meaning that items 3, 4 and 5 from above can be covered.

We judge MATLAB and MATRIX_X to be equally good from a numerical point of view. Yet we favor MATLAB much because it is the standard numerical tool used at our department, but also because it is available on more computer platforms than MATRIX_X.
Symbolic Computations

Although the first commercial and general-purpose Computer Algebra Systems (CAS) emerged some twenty years ago\(^1\), it was not until recently symbolic computations became feasible. The development of the computer algebra systems of today started around 1980, taking advantage of new ideas in software engineering and avoiding shortcomings of earlier CAS. The most popular systems arising from these efforts were Maple [13], developed at the University of Waterloo, and Mathematica [106] from Wolfram Research, Inc.

As is the case with MATLAB, these packages are implemented in C [53]. In large, they also offer the same functionality. They, for example, both provide a Gröbner basis implementation, thus almost no coding is needed in this respect. Which system to choose is by no means obvious and again much subject to personal taste considerations. Nevertheless, two reasons motivate the use of Maple as the main symbolic engine. The first one is that the implementation of Ritt’s algorithm we started off from is that of Glad [33], written in the Maple language. So staying with Maple was quite natural, but mapping Ritt’s algorithm to Mathematica format should cause no trouble. The second reason is that Maple computations now are available within MATLAB through the Symbolic Math Toolbox. This at least indicates that MATLAB and Maple are able to work side by side.

Notice that advanced graphics and numerical computations are included in Maple and Mathematica as well. However, the developing effort of these systems mainly focus on symbolic algorithms, and consequently the need for high-performance numeric computation tools like MATLAB remains. In other words, it is foremost the second item of the specification list that can be handled through Maple.

Other Services

None of requirements 1, 6 and 7 have been addressed in any depth in this thesis. It is true that there exist general tools supporting each of them, but for our problem the question yet to be answered is what they can offer and how. Here we only put forward some remarks intended as an appetizer for future work. A further discussion on some of the topics can be found in Linkens’ collection [60].

Using Maple, the symbolic algorithms demand equations expressed in the internal Maple language. From a user’s standpoint this may not be the best way of entering the model (item 1). Perhaps should the equations be structured and entered graphically similarly to the way bond graphs are constructed. Another question is whether modern structuring concepts, such as hierarchical submodel decomposition, object-orientation etc., lately adopted by high-level CACE (Computer-Aided Control Engineering) languages for simulation, such as Dymola [22] and Omola [2, 73], can be used successively also here. Note that this might not be the case as the complex modeling situations Dymola and Omola primarily are intended for very well can be too much for the previously discussed symbolic algorithms.

Complicated modeling situations tend to generate a great deal of information. As was noted by several authors of [60], data base support (item 6) can help bringing order to the developed models. Graphical model browsers may also be used to quickly get an overview of estimated models as well as for documentation purposes.

\(^1\)REDUCE and MACSYMA were among the pioneering systems.
Intelligent help systems have already proved to be useful in an identification context, see for example Nagy [76]. One interesting possibility is to extend such a knowledge-base system (item 7) to also involve active help in the modeling phase. A second extension would be to apply heuristic reasoning to reduce the complexity of the symbolic computations.

Putting the Pieces Together

When integrating tools from different areas one usually talks about hybrid systems. To facilitate the use of such a system, we have earlier stressed that a suitable glue between the pieces is a graphical user interface, or more precisely, an interfacing language connecting the various tools through a graphical user interface.

Employing built-in user interface routines and the newly released Extended Symbolic Math Toolbox, MATLAB itself has become a very interesting GUI alternative. One particular intriguing solution is to construct the tool on top of the currently developed graphical user interface for the System Identification Toolbox [64, 62]. There are a couple of good reasons for pursuing this approach. Beyond inheriting portability and a uniform use, many of the numerical needs would be satisfied directly or with minor programming efforts. However, there are also some annoying drawbacks. One is that the capabilities of the symbolic toolbox are unknown. Preliminary tests indicate slow execution and a not yet stable implementation. More severely is that MATLAB’s environment at present is not as open as desirable. Integrating data base systems, knowledge-base systems etc. to MATLAB may thus be hard, or even impossible.

Desiring an open and easy-to-extend system we have chosen not to use MATLAB as a master computational server. Instead, we suggest to use it, Maple and the other packages as stand alone computational units, running as separate processes. Indeed, this is a concept widely accepted in many engineering contexts; Sorlie [89], e.g., discusses it from a grey box identification viewpoint. Besides being “open”, a further advantage here is that distributed computations, i.e., parallel processing is made more accessible, and perhaps this is really what is needed to solve more complicated symbolic problems. For this to be feasible in practice, though, more research on parallel symbolic algorithms is needed.

The C programming language seems to be a suitable interfacing language, principally because MATLAB, Maple and many other conceivable tools are C products. Admittedly, we have not yet addressed the important problem of designing a proper interfacing language. It was initially considered to be more important to quickly arrive at a prototype tool in order to investigate the potential of semi-physical modeling. In this spirit, we also abandoned portability by including the XView toolkit [45] (based on the X Window System and running on SUN computers) to support interactive graphics (windows, buttons, scroll bars, and so on). This leave us with a hybrid modeling tool, SEMI, structured as is depicted in Figure 9.1.

The present version of SEMI includes the packages marked grey in the figure. Besides compiled C and XView functions, SEMI contains a number of MATLAB m-files and a few Maple functions. If desired, these can be executed separately within the MATLAB and the Maple environments, respectively. Communication between the back end units and the C front end is provided using pipes. For MATLAB this is readily achieved through its engine library, which in addition supports network computations, even for machines of different architecture. Maple and many other possible tools lack such interfacing
libraries, thus leading to more coding. Since no data base manager is included yet, MATLAB is for the time being employed as a common repository for data, estimated models and shared structure information. Computationally this means that whenever a Maple output is available it is represented by a string and subsequently transmitted to MATLAB’s memory work space.

9.2 SEMI - A Tool for Doing Semi-Physical Modeling

Leaving further implementational details, the intention with this section is to give an overview of available services provided by SEMI. The interface window of SEMI is shown in Figure 9.2. Most computations are initiated either via pull-down menus or via buttons, selection lists etc. using the mouse.

The main SEMI screen image is partitioned after functionality as follows:

Main command area: This panel is divided into two parts: one row of pull-down menus for invoking the bulk of the commands and another row providing various status information. The pull-down menus are grouped accordingly:

File: The File menu contains commands for reading and saving model information; data, models, settings, and so on. Since all information is stored in MATLAB’s work space, some rudimentary MATLAB commands are utilized here. Printing of plots as well as a command trace facility is also provided.

Modeling: Commands for symbolic computations and functions for treatment of data are available from this menu via two subwindows: Symbolic and Data. The Symbolic command window provides interactive means for entering the a priori knowledge and running the algebraic algorithms of Chapter 7. As a first
Chapter 9. A Tool for Semi-Physical Modeling

Figure 9.2: The SEMI user interface. The system currently being investigated is the solar-heated house.

step, variables are introduced and classified: a variable can either be an input, an output or an internal non-measurable variable. The variables are also ordered after "simplicity", i.e., an ordering or a ranking amongst the variables is imposed. In the next step, the equations are entered textually as Maple expressions. After having specified the model to start with, the following Maple functions can be executed from the Symbolic window, preferably in the listed order:

1. \texttt{ode2ade(F)} Requirement 6.2. Converts a set of ordinary differential equations \( F \) to an "equivalent" set of algebraic differential equations \( ADE \). For the algorithm, see Section 7.3.

2. \texttt{complexity(ADE)} Requirement 6.3. Displays upper complexity bounds for the set of ADEs \( ADE \) according to the discussion of Section 7.4. This includes the maximum order of the resulting IO-expression, and in applicable cases, the maximum degree and number of terms in it.

3. \texttt{gbio(ADE,IO,X)} Requirement 6.1. Computes a Gröbner basis given the state space model \( ADE \), the ordered list of input-output variables \( IO \), and the ordered list of non-measurable variables \( X \). The last two entries are subsequently used to generate an appropriate monomial ordering. See Section 7.1 for further details.

4. \texttt{rittio(ADE,IO,X)} Requirement 6.1. Returns a characteristic set based on the set of ADEs \( ADE \). The concatenation of \( IO \) and \( X \), both defined above, gives the ranking. For the algorithm, see Section 7.2.

5. \texttt{lrform(IOeq,0,D)} Requirement 6.4. The IO-expression \( IOeq \) is first converted to a discrete-time "equivalent" using the discretization rule \( D \). Then
the function either returns a linear-in-the-parameters predictor structure
for $0$, or when this is impossible, it returns a linear-in-the parameters
$IO$-structure. See Section 7.5.

The use of some of these functions is exemplified in the next section. Fi-

nally, the regressors arrived at in the last step are sent to the "Available

regressors" window for use in the estimation procedure.

The Data subwindow provides functions for loading and polishing data. How-

ever, the present version of SEMI largely lacks the latter possibility, though

using MATLAB such services are rather easy to include.

Models: This is a dynamic menu for quick retrieval of models stored in the model
data base (see Model data base area below). In later versions, a graphical
model browser is going to be included and accessed from this menu.

Method: The Method menu accommodates the choice of estimation algorithms
and regressor selection strategies. Concerning the first category, the possible
choices are ordinary least-squares, total linear least-squares and ridge regres-
sion (see Chapter 8). The second category includes backward elimination,
forward selection and principal component regression strategies. A stepwise
regression procedure is readily obtained through a mixed usage of the first
two methods. Notice that no computations are carried out here; the adopted
settings are just passed to MATLAB for later use by the commands reachable
from the Estimation and validation area, discussed below.

Options: Various universal settings can be changed from this menu. The choice
of prediction horizon $k$ to use when ranking regressors is one such setting.
Otherwise, the common principle is that user's options are locally distributed
to windows responsible for the required service. In the Symbolic subwindow,
e.g., one can choose between Buchberger's or Ritt's algorithm.

Help: The current help facility is limited to a textual description of SEMI's func-
tionalilty. In principle, it is a somewhat more involved version of this section.
For example, one extra feature is that help texts associated with the included
MATLAB and Maple functions can be displayed.

Regressor selection area: The purpose with this and the following two panel areas
is to provide fast access to commands likely to be used often during the model
estimation phase. Through the first two buttons, regressors can be entered to
or eliminated from the current model. The New... button allows the user to
complement the available list of regressors with user-defined ones, i.e., regressors
not suggested by the symbolic machinery. This option is especially useful for
problems of the kind discussed in Example 6.1.

Model data base area: As was mentioned earlier, no advanced data base facility is
included at this point. However, the current model, embracing the regressors
displayed underneath "Selected regressors", is stored in MATLAB's work space
via the Store... button. Any previously stored model can later be recovered through
the Models menu. The Clear... button lastly allows the user to permanently delete
a model from this menu and from the memory work space.

Estimation and validation area: Most numerical computations are initiated from
this area. From top to bottom the following MATLAB m-files are called:
Chapter 9. A Tool for Semi-Physical Modeling

Model \rightarrow \text{semimodel}(\ldots) \text{ Estimates a model using the least-squares, the total linear least-squares or the ridge regression procedure. The estimated model along with parameter variances are subsequently shown in a separate text window.}

Loss \rightarrow \text{semiranking}(\ldots) \text{ This is the main procedure for selecting regressors. Depending on strategy, a number of models is first estimated. These models are then ordered after loss function value when applied to validation data, whereupon the output (the ranking) is shown in a separate MATLAB window of the form detailed in Section 8.2.}

Predict \rightarrow \text{semipredict}(\ldots) \text{ Displays measured and predicted model output in a plot window. Notice that the prediction horizon can be chosen between 1 and } \infty, \text{ and hence pure simulation is possible.}

Residuals \rightarrow \text{semiresiduals}(\ldots) \text{ Computes and plots residuals } e(t), \hat{R}_e^N(t) \text{ and } \hat{R}^N_{eu}(t) \text{ in the residual plot window. Here, } \hat{R}_e^N(t) \text{ and } \hat{R}^N_{eu}(t) \text{ are computed according to Equations (2.32) and (2.33), respectively.}

Regressors and ranking lists: The major part of the screen is dedicated to the regressor selection problem discussed in Chapter 8. Regressors, which either are automatically generated by the symbolic package or entered by hand using New..., are lined up in the leftmost subwindow ("Available regressors"). Any of these regressors can be selected to enter the model using the Add button. The regressors of the current model are listed in the "Selected regressors" subwindow. If desired, any of these can be eliminated from the model via the Delete button. Given a regressor selection strategy, the "Best calculated regressors" subwindow shows a ranking among regressors, computed and presented when the Loss button is pushed. Thus it is a textual version of what is expressed in the corresponding plot window. Notice that this ranking is strategy dependent; using backward elimination as in Figure 9.2, the first entry is the regressor suggested for removal, while in the forward selection case it would have been the regressor to add.

Information area: This is a text field showing ongoing computations or, when waiting for new commands as in Figure 9.2, the process file presently being investigated.

9.3 A Modeling Scenario with SEMI

In this last section we will show how a typical modeling session with SEMI can look like. The process to be modeled is the immersion heater system depicted in Figure 9.3. In large, this is the system from Example 7.16 with the major difference being that the heater here is controlled by the current \( i(t) \). Another visible difference is that the flow through the tank is taken into account in the initial modeling phase, though only the zero flow case is going to be studied in detail later on. To further complicate the situation, the resistance of the heater element is assumed to be temperature dependent:

\[
R(t) = R_0 \left(1 - \alpha_1 T_R(t) - \alpha_2 T_R(t)^2 \right), \quad \alpha_1, \alpha_2 \in \mathbb{R}^+,
\]

where \( R_0 \) is a nominal resistance and \( R(t) \) is varying between 25 and 15 \( \Omega \) when \( T_R(t) \) spans the operating region of 20 to 150° C (increasing \( T_R(t) \) leads to decreasing \( R(t) \)).
Applying the principle of conservation of energy, a bit of straightforward modeling gives

\[
\begin{align*}
\frac{d(WT_R(t))}{dt} &= R(t)i(t)^2 - K_1 (T_R(t) - T(t)) \\
\frac{d(\rho Ah(t)cT(t))}{dt} &= \rho Q_i(t)cT_i(t) - \rho Q_o(t)cT(t) + \\
&\quad K_1 (T_R(t) - T(t)) - K_2 (T(t) - T_s(t)).
\end{align*}
\] (9.2) (9.3)

where \(W, K_1, K_2, \rho, A\) and \(c\) are parameters, which except for the area \(A\) are unknown. Assuming further that the pump generates an inlet flow, \(Q_i(t) = ku(t)\), the total mass of the liquid in the tank evolves as

\[
\frac{d(\rho Ah(t))}{dt} = \rho Q_i(t) - \rho Q_o(t).
\] (9.4)

where the outlet flow is described by Bernoulli's law:

\[
Q_o(t) = a\sqrt{2gh(t)},
\] (9.5)

with \(a\) being the cross-section area of the outlet hole and \(g\) the acceleration due to gravity.

As before the modeling aim is to describe the liquid temperature \(T(t)\) when the current \(i(t)\) is acting as an input signal. To simplify the estimation procedure we will restrict the study to the case where the liquid level is constant \(h\), the pump is off \((u(t) = 0)\), the valve is closed \((Q_o(t) = 0)\), and the temperature of the surroundings \(T_s(t)\) is constant and equal to 20° C. With the mathematical model above, it should be noted that further signals have to be measured when there is a flow through the tank. In such a case not only \(u(t)\) (or \(Q_i(t)\)) must be measured but also the liquid level \(h(t)\) (or the outlet flow \(Q_o(t)\)).

The first step when using SEMI is to enter the variables and equations assumed to govern the heater system. As was said earlier, variables as well as equations are textually entered as Maple expressions in the Symbolic subwindow, partly seen in Figure 9.4. Having specified the model structure to begin with, the symbolic computations basically
follow along the lines of the script below, which illustrates what is going on behind the screen. Notice especially the syntax used in steps 1 and 2 to represent variables and equations.

SYMBOLIC COMPUTATIONS: IMMERSION HEATER.

> ### 1. Variables and indirectly their ranking (ordering) defined automatically by SEMI.
> INPUTS := \( i(t) \):
> OUTPUTS := \( T(t) \):
> UNKNOWNS := \( T[R](t), R(t), Q[i](t), Q[o](t), u(t), h(t), T[i](t), T[S](t) \):
> IO := \( \{ \text{op(INPUTS)}, \text{op(OUTPUTS)} \} \);

> ### 2. Equations from SEMI, compiled in the list \( F \).
> \( F := \{ \) \( \) \( \frac{\Delta}{\Delta t} W\left[T[R](t)\right] = R(t)\cdot i(t)^2 - K[1]\cdot \left[T[R](t) - T(t)\right], \right. \)
> \( \) \( \frac{\Delta}{\Delta t} (\rho\cdot A\cdot h(t)\cdot c\cdot T(t)) = \left( \rho\cdot Q[i](t)\cdot c\cdot T[i](t) + K[1]\cdot \left[T[R](t) - T(t)\right] - \rho\cdot Q[o](t)\cdot c\cdot T(t) + K[2]\cdot \left[T(t) - T[S](t)\right] \right), \)
> \( \) \( R(t) = R[0]\cdot (1 + \alpha[1]\cdot T[R](t) + \alpha[2]\cdot T[R](t)^2), \)
> \( \) \( \frac{\Delta}{\Delta t} \rho\cdot A\cdot h(t) = \rho\cdot Q[i](t) - \rho\cdot Q[o](t), \)
> \( \) \( Q[o](t) = a\cdot \sqrt{2\cdot g\cdot h(t)}, Q[i](t) = k\cdot u(t) \) \( \) \( \} \);

> ### 3. Simplify the problem by assigning values to some of the variables:
> u(t)=0, \( T[S](t)=20 \), h(t)=h, \( Q[o](t)=0 \).
> \( F := \text{simplify(subs(} \{ u(t)=0, T[S](t)=20, h(t)=h, Q[o](t)=0 \}, F) \);

> ### 4. ADE formulation (the first of the four entries in the list).
> The input is already in polynomial form, but the function simplifies and rearranges the equations.
> ADE := \( \text{ode2ade(} F) \);
9.3 A Modeling Scenario with SEMI

### 5. Check the complexity of the problem.

```plaintext
complexity(ADE[1]);
```

The order of the IO-expression will be 2.

### 6. Eliminate unknown variables. Since the structure in principle is in state space form, either Buchberger's or Ritt's algorithm can be used. Let us try the latter one:

```plaintext
IOeq := rittio(ADE[1], IO, [op(UNKNOWNWS), op(ADE[4])]);
```

\[
\begin{align*}
\text{IOeq} & := \left[ K_1 i(t)^2 R_0 + K_1^2 \alpha_2 R_0 i(t)^2 T(t)^2 + K_1^2 \alpha_1 R_0 i(t)^2 T(t) \\
& + 20 K_1^2 K_2 - W K_1^2 \left( \frac{\partial}{\partial t} T(t) \right) - K_1^2 \rho A h c \left( \frac{\partial}{\partial t} T(t) \right) \\
& - K_1^2 K_2 T(t) + 2 K_1 T(t)^2 i(t)^2 R_0 \alpha_2 K_2 \\
& - 40 K_1 i(t)^2 R_0 \alpha_2 K_2 T(t) + K_1 \rho A h c \left( \frac{\partial}{\partial t} T(t) \right) i(t)^2 R_0 \alpha_1 \\
& + 2 K_1 \rho A h c \left( \frac{\partial}{\partial t} T(t) \right) i(t)^2 R_0 \alpha_2 T(t) - K_1 W K_2 \left( \frac{\partial}{\partial t} T(t) \right) \\
& + K_1 K_2 T(t) i(t)^2 R_0 \alpha_1 - K_1 W \rho A h c \left( \frac{\partial^2}{\partial t^2} T(t) \right) \\
& - 20 K_1 K_2 i(t)^2 R_0 \alpha_1 + K_1^2 T(t)^2 i(t)^2 R_0 \alpha_2 \\
& - 40 K_2^2 T(t) i(t)^2 R_0 \alpha_2 + 2 \rho A h c \left( \frac{\partial}{\partial t} T(t) \right) i(t)^2 R_0 \alpha_2 K_2 T(t) \\
& + 400 i(t)^2 R_0 \alpha_2 K_2^2 + \rho^2 A^2 h^2 c^2 \left( \frac{\partial}{\partial t} T(t) \right)^2 i(t)^2 R_0 \alpha_2 \\
& - 40 \rho A h c \left( \frac{\partial}{\partial t} T(t) \right) i(t)^2 R_0 \alpha_2 K_2 \right], \{i(t)\}, \text{ptable}, [
\left[ 10,8,16,4 \right], \left[ 6,1,3,4,5,2 \right], \left[ 7 \right], \left[ 8,6,1,3,4,5,2 \right], \left[ 9 \right], \left[ 10,8,6,1,3,4,5,2,7,9,11,12,13,14,15 \right], \left[ 10,8,6,1,3,4,5,2,9,11,7,12,13,14 \right]]
\end{align*}
\]

The entries of the list IOeq is as follows:

1. The IO-expression searched for.
2. Other IO-expressions to consider.
3. Table of generated polynomials.
4. Structure information. Polynomials 10, 8, 16 and 4 of ptable form the characteristic set.

### 7. Use the Euler forward approximation to compute a discrete-time predictor structure:

```plaintext
LR := lrform(IOeq[1][1], OUTPUTS[1]);
eta := LR[1][1];
phi := LR[2][1];
```

\[
\phi := \left[ i(t^2 -2)^2 T(t-1)^2, i(t-2)^2 T(t-1) T(t-2), T(t-1) i(t-2)^2, T(t), i(t-2)^2, T(t-2) i(t-2)^2, T(t-2), i(t-2)^2, 1 \right]
\]
After that \texttt{lrform} has been executed the regressors are listed in the "Available regressors" subwindow according to Figure 9.4.

The next step is to load estimation and validation data from the Data subwindow, also shown in Figure 9.4. It should here be emphasized that the data sets were artificially generated from Equations (9.1)-(9.5) using MATLAB. The parameter values of these equations were chosen somewhat arbitrarily without paying much attention to physical considerations. To anyway make the estimation problem a bit more realistic, white noise was added to the liquid temperature $T(t)$, thereby giving the input-output behaviors of Figure 9.5.

Before proceeding any further, we can see what happens if a linear model structure is fitted to the data. One of the best candidates found in this category is a sixth order ARX model

$$\hat{T}(t) = -\hat{a}_1 T(t-1) - \hat{a}_2 T(t-2) - \hat{a}_3 T(t-3) + \hat{b}_1 i(t-1) + \hat{b}_2 i(t-2) + \hat{b}_3 i(t-3),$$

whose simulated output is detailed in Figure 9.6. The fit is not that bad, yet unacceptable for certain operating regions which show an up to $5^\circ\text{C}$ discrepancy between measured and model output. By realizing that it is the heater power rather than the current that causes the temperature to change a second plausible attempt is to test a model of the
Figure 9.5: Input-output data generated from the heater equations. The left plot shows the data used for estimation, while the right plot shows validation data.

Figure 9.6: Simulated output of the sixth order linear model (Structure 9.6) compared with the measured output (left). In the right plot the linear model is replaced by a nonlinear least-squares fitted model corresponding to Structure 9.7.

following kind:

\[
\hat{T}(t) = -\hat{a}_1 T(t-1) - \hat{a}_2 T(t-2) - \hat{a}_3 T(t-3) + \hat{b}_1 i(t-1)^2 + \hat{b}_2 i(t-2)^2 + \hat{b}_3 i(t-3)^2.
\]  

However, which is illustrate in Figure 9.6, this does not improve the fit, rather it becomes worse. Marrying together these model structures the mean temperature discrepancy can be reduced to 1, although for certain input-output combinations the deviation is still unacceptably large. The conclusion to be drawn here is that the assumed temperature dependency of the heater resistance must be taken into account in the modeling work.

This observation brings us back to the regressors from the symbolic modeling procedure. Including all 9 regressors in the model structure and using the ordinary least-squares algorithm gives a model having a mean error between measured and simulated output of around 0.4. Moreover, the discrepancies are spread out over all data points and not to some particular regions, which firstly and as expected indicates that the important nonlinearities are captured, and secondly that the Euler forward approximation
is sufficient in this case.

Experimenting with the selection strategies reveals that some of the regressors can be eliminated. In fact, there exist a couple of models of lower order which seems to be equally good. The model we finally decided on is the following fifth order one

\[
\hat{T}(t) = \hat{\theta}_1 + \hat{\theta}_2 T(t - 1) + \hat{\theta}_3 i(t - 2)^2 + \\
\hat{\theta}_4 i(t - 2)^2 T(t - 1)^2 + \hat{\theta}_5 i(t - 2)^2 T(t - 1)T(t - 2),
\]

with parameter values shown in the Model subwindow of Figure 9.7. Notice in particular how the temperature dependency of the heater element is expressed through the last two regressors. The structure and importance of these regressors are not that easily recognized from the original equations, thus indicating the usefulness of the suggested machinery. Another observation is that the parameter values ranges over a pretty large region due to that the regressors have different total degrees (1, 2 or 4). This may lead to numerical difficulties for which certain scaling techniques must be applied to get a well-posed problem. Nevertheless, the SEMI generated simulation and residual plots of Figure 9.7 clearly shows the performance improvement compared to the previously discussed models. Thus we conclude that the modeling procedure supported by SEMI can aid in finding suitable model structures and models.
Conclusions and Future Extensions

To succeed in many modeling situations it is of utmost importance that prior system knowledge are taken into account. In this thesis, we have from a system identification viewpoint studied two different ways of utilizing such prior knowledge.

Modeling Using Laguerre and Kautz Filters

In the first approach we confined to linear model structures, where we used a priori knowledge about the systems' dominating time constants and resonance frequencies. The idea was to generalize FIR models by replacing the delay operator with special designed so-called discrete Laguerre or Kautz filters. By this, the advantages of the FIR model structure are maintained, while its major drawback of sometimes requiring a large number of parameters is mastered through the prior. Because the a priori information used most often is accessible in any identification context, we foremost consider the methods to be an alternative to more traditional FIR, ARX, OE or AR, ARMA modeling.

The basic Laguerre and Kautz estimation procedure was outlined in Chapter 4. It was later extended to handle effects of unknown initial conditions and situations where the prior given was poor or uncertain. In the second extension, however, we had to leave the linear regression framework and instead depend upon nonlinear numerical search methods. The usefulness of the suggested algorithms was finally illustrate in a number of simulation examples, and in Chapter 5, in two industrial applications. To extend and simplify the use of these methods a package of MATLAB m-files, fully compatible with the System Identification Toolbox [62], has been developed.

As was recognized in Chapter 5 an issue to study in the future is recursive variants of the proposed identification algorithms. The potential of Laguerre and Kautz models in real applications should also be explored more thoroughly. This is a task that already has been started; human lung mechanics, ear dynamics as well as recovery boiler data from the paper and pulp industry are currently being investigated. Furthermore, discussions with users of the developed software package have emphasized the need for better means for choosing a priori filter parameters. In future software releases we are planning to provide such aids in form of an add-on package to the graphical user interface for the System Identification Toolbox [62, 64].
Semi-Physical Modeling

Semi-physical modeling is more involved than the previous approach, yet less “ambitious” than traditional physical modeling. This is due to that no complete physical structure is sought, just combinations of inputs and outputs that can be subjected to more or less standard model structures, such as linear regressions. Using simple physical principles, the prior knowledge used is a set of static and dynamic equations which are believed to describe the true system reasonably well.

The first step of the procedure is to employ symbolic algebra (commutative and differential algebra) to come up with a set of combinations of input and output signals based on the given physical insight. A subset of these regressors is then selected and a linear-in-the-parameters model is estimated using a least-squares algorithm. The discussed symbolic and numerical algorithms have been implemented in a software tool, SEMI, which was used to illustrate the procedure on two target processes: a solar-heated house and an immersion heater system.

In both these examples the procedure worked rather smoothly. However, for larger systems the complexity of the symbolic machinery is likely to be a decisive limiting factor. The question then is whether other and less demanding algorithms can be devised for these cases. Since the regressors always are tested against data, completeness of the algorithms is perhaps a property that can be relaxed. Another bothering complexity issue is that even though the given equations are few and compact, the number of regressors may still be very large. A further observation is that a complicated implicit IO-expression not too seldom is returned. Admittedly, this has not been treated in detail here, and for the time being we just add it to the list of open problems.

The SEMI implementation is of prototype character and leaves several questions unanswered: Can we benefit from data base and knowledge-base systems? Can the computational complexity be managed by letting the user monitor and guide the symbolic calculations?

Finally, semi-physical modeling needs further corroboration, preferably through case studies of real-world applications. The procedure should also be compared with other grey box methods, both concerning time to arrive at a model, and concerning different quality aspects of the derived models.
Bibliography


Subject Index

A
ADE (Algebraic Differential Equation) 85
algorithm 88
application
   aircraft flight flutter 49
   neutron flux noise 53
   solar-heated house 60, 100
approximation theory 19
ARMAX
   model 53
   model structure 10, 35
ARMA
   model 55
   model structure 10, 36
ARX
   model 53, 61
   model structure 10
AR
   model 55
   model structure 10, 22, 33
autoreduced set 80
a priori knowledge 4, 17, 36, 51, 54, 59
B
backward elimination 97
BJ
   model structure 10
BondToolM 14
Buchberger's algorithm 75
C
characteristic set 66, 81
combined Laguerre/Kautz model 32, 52
comutative algebra 66, 71
computational complexity 90
D
decay ratio 53
differential algebra 66, 78
Dymola 107
dynamic system 4
E
ETFE 50
Euler forward approximation 63, 93
F
field 72
differential 79
FIR
   model 28, 32
   model structure 10, 19
forward selection 97
G
Gauss-Newton algorithm 46
GMDH 68
Graphical User Interface (GUI) 108
grey box modeling 59
Gröbner basis 66, 71, 75
H
Hammerstein model 68
Hessian 45
<table>
<thead>
<tr>
<th>Subject Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
</tr>
<tr>
<td>ideal 72</td>
</tr>
<tr>
<td>differential 79</td>
</tr>
<tr>
<td>prime 78</td>
</tr>
<tr>
<td>radical differential 84</td>
</tr>
<tr>
<td>IDKIT 13, 69</td>
</tr>
<tr>
<td>IDPAC 13</td>
</tr>
<tr>
<td>ID TOOL EXPERT 14</td>
</tr>
<tr>
<td>IFS 14</td>
</tr>
<tr>
<td>impulse response 18, 44</td>
</tr>
<tr>
<td>initial 81</td>
</tr>
<tr>
<td>initial conditions 43, 47</td>
</tr>
<tr>
<td>K</td>
</tr>
<tr>
<td>Kautz</td>
</tr>
<tr>
<td>filter coefficients 45</td>
</tr>
<tr>
<td>functions 30</td>
</tr>
<tr>
<td>model 31, 32, 47, 54</td>
</tr>
<tr>
<td>network 31</td>
</tr>
<tr>
<td>state space structure 39</td>
</tr>
<tr>
<td>Kronecker delta 19</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>Laguerre</td>
</tr>
<tr>
<td>filter coefficients 45</td>
</tr>
<tr>
<td>functions 26</td>
</tr>
<tr>
<td>model 28, 32, 46</td>
</tr>
<tr>
<td>network 26</td>
</tr>
<tr>
<td>state space structure 37</td>
</tr>
<tr>
<td>latent root regression 99</td>
</tr>
<tr>
<td>leader 80</td>
</tr>
<tr>
<td>leading coefficient 73</td>
</tr>
<tr>
<td>leading monomial 73</td>
</tr>
<tr>
<td>leading term 73</td>
</tr>
<tr>
<td>leading variable 80</td>
</tr>
<tr>
<td>least-squares algorithm 20, 34, 44, 95</td>
</tr>
<tr>
<td>lexicographic monomial ordering 73</td>
</tr>
<tr>
<td>Lie derivative 76</td>
</tr>
<tr>
<td>limiting estimate 27</td>
</tr>
<tr>
<td>linear regression 11, 20, 33, 44, 63, 95</td>
</tr>
<tr>
<td>M</td>
</tr>
<tr>
<td>Maple 91, 107</td>
</tr>
<tr>
<td>Markov parameters 18</td>
</tr>
<tr>
<td>mathematical model 4</td>
</tr>
<tr>
<td>Mathematica 107</td>
</tr>
<tr>
<td>MATLAB 13, 106</td>
</tr>
<tr>
<td>MATRIX 13, 106</td>
</tr>
<tr>
<td>model structure 7</td>
</tr>
<tr>
<td>model validation 12</td>
</tr>
<tr>
<td>monomial 72</td>
</tr>
<tr>
<td>multidegree 73</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>neural network 69</td>
</tr>
<tr>
<td>Newton-Raphson algorithm 45</td>
</tr>
<tr>
<td>nonlinear identification 68</td>
</tr>
<tr>
<td>normal equations 11, 34</td>
</tr>
<tr>
<td>O</td>
</tr>
<tr>
<td>OE model 46, 47, 53</td>
</tr>
<tr>
<td>model structure 10, 36</td>
</tr>
<tr>
<td>Omola 107</td>
</tr>
<tr>
<td>orthonormal 18</td>
</tr>
<tr>
<td>basis 19</td>
</tr>
<tr>
<td>functions 18</td>
</tr>
<tr>
<td>set 19</td>
</tr>
<tr>
<td>oscillatory system 28</td>
</tr>
<tr>
<td>overfit 64</td>
</tr>
<tr>
<td>P</td>
</tr>
<tr>
<td>parsimony 69</td>
</tr>
<tr>
<td>Partial Least-Squares (PLS) 99</td>
</tr>
<tr>
<td>periodogram estimate 54</td>
</tr>
<tr>
<td>persistently exciting 35</td>
</tr>
<tr>
<td>polynomial 72</td>
</tr>
<tr>
<td>differential 78</td>
</tr>
<tr>
<td>polynomial ring 72</td>
</tr>
<tr>
<td>differential 79</td>
</tr>
<tr>
<td>Prediction Error Method (PEM) 11</td>
</tr>
<tr>
<td>prefiltering 34</td>
</tr>
<tr>
<td>primeness 84</td>
</tr>
<tr>
<td>Principal Component Regression (PCR) 98</td>
</tr>
<tr>
<td>Q</td>
</tr>
<tr>
<td>quasi-stationary 35</td>
</tr>
<tr>
<td>R</td>
</tr>
<tr>
<td>ranking 79</td>
</tr>
<tr>
<td>rate of convergence 21, 26</td>
</tr>
<tr>
<td>reducedness 80</td>
</tr>
<tr>
<td>remainder 74, 83</td>
</tr>
<tr>
<td>partial 82</td>
</tr>
<tr>
<td>residual 12</td>
</tr>
<tr>
<td>ridge regression 96</td>
</tr>
</tbody>
</table>
ring-homomorphism 76
Ritt's algorithm 81, 83

S
S-polynomial 74
sampling 21
semi-physical modeling 59
  Requirement 6.1 66
  Requirement 6.2 67
  Requirement 6.3 67
  Requirement 6.4 67
  Requirement 6.5 68
SEMI 108, 109
separant 81
SITB 13, 108
SNR 46, 49
space
  $L^2(\cdot,\cdot)$ 19
spectral estimate 50, 54
stepwise regression 98

T
time series 21, 33, 53
Toeplitz structure 35
total degree 92
total linear least-squares 95
transfer function 18
transients 34, 43

W
well-damped system 25


