SUBSPACE IDENTIFICATION FOR LINEAR SYSTEMS

Theory - Implementation - Applications

SUBSPACE IDENTIFICATION FOR LINEAR SYSTEMS

Theory - Implementation - Applications

Peter VAN OVERSCHEE
Bart DE MOOR
Katholieke Universiteit Leuven
Belgium

KLUWER ACADEMIC PUBLISHERS
Boston/London/Dordrecht

CONTENTS

PREFACE

Ceci n'est pas une pipe.
René Magritte, Belgian painter, 1898-1967.

The last 30 years or so, system identification has matured from Eykhoff's '*bag of tricks*', over the impressive *Ljungian theory for the user* of so-called *prediction-error methods*, to Willems' *behavioral framework*. Many papers have been written, several excellent textbooks have appeared and hundreds of workshops and conferences have been organized. Specifically for the identification of linear dynamic time-invariant models from given input-output data, the collection of available methods has become immense.

So why write yet another book about this, by now, almost classical problem? Well, to start with, the problem is important! There is a growing interest in manageable mathematical models for all kinds of applications, such as simulation, prediction, fault diagnosis, quality and safety monitoring, state estimation, signal processing (direction-of-arrival algorithms (SDMA)) and last but not least, model-based control system design. And sure enough, *linear* models are very popular because of their utmost simplicity (at least at first sight).

In this book, we do not really solve a new problem. Indeed, the goal is to find dynamical models from input-output data that were generated by so-called combined deterministic-stochastic linear systems. Said in other words, data that are generated by a linear, time-invariant, finite-dimensional, dynamic system, with both deterministic and stochastic input signals (including several special cases).

What is new in this book, are the methods and algorithms for solving this 'classical' problem. The insights that will be developed, originate in a mixture of ideas, facts and algorithms from system theory, statistics, optimization theory and (numerical) linear algebra. They culminate in so-called 'subspace' methods, the name of which reflects the fact that linear models can be obtained from row and column spaces of certain matrices, calculated from input-output data. Typically, the column space of such data matrices contains information about the model, while the row spaces allow to obtain a (Kalman filter) state sequence, *directly from input-output data* (i.e.without knowing the model a priori)[1]. Another important aspect of this book is the development of a *unifying* framework, in which almost all existing subspace methods that have appeared in the literature of the last 10 years or so, have found their place.

Apart from these *conceptual* contributions, there are other advantages to subspace methods. For instance, there is no need for an explicit model parametrization, which, for multi-output linear systems is a rather complicated matter. A second numerical advantage is the elegance and computational efficiency of subspace algorithms. The dimension and numerical representation of the subspaces mentioned before, are calculated using the QR- and the singular value decomposition. These are well-understood techniques from numerical linear algebra, for which numerically robust and efficient algorithms are widely available.

Of course, we should *never* forget that *a (mathematical) model is not the real system* (think of Magritte). Even though there are still missing links in the question of guaranteed optimality of subspace methods, it is now widely accepted that they prove very useful in many applications, in which they often provide excellent models and because of their utmost user-friendliness (limited number of user-choices to deal with). They also provide (often excellent) initial guesses for non-linear iterative optimization algorithms which are used in prediction-error methods, L_2-optimal system identification, neural nets, etc...

Finally, we have paid special attention to the development of easy accessible and user-friendly software packages, which are described in Chapter 6 (Xmath[2] ISID II) and Appendix B (which describes Matlab files and several demos). This book goes with a diskette that contains all of these .m files and examples.

[1] Have a look at Theorems 2, 8 and 12 of this book.
[2] A product of Integrated Systems Incorporated, Santa Clara, CA, USA.

Mister Data, there's a subspace communication for you.
Quote from Star Trek, the Next Generation.

This book emanates from the authors' PhD theses at ESAT, the department of Electrical Engineering of the Katholieke Universiteit Leuven in Belgium. Bart's 1988 thesis contained the initial concepts and ideas for subspace identification (of course inspired by the work of many others), linking ideas from system theory (realization algorithms), linear algebra (orthogonal projections and intersections of subspaces) to numerical issues (QR and singular value decompositions). Peter's 1995 thesis, which forms the basis of this book, contains the detailed unification of all these insights, culminating in some robust subspace identification methods, together with other results such as model reduction issues, relations with other identification algorithms, etc...

The work reported on in this book would have been impossible without the support, both financial and moral, from many institutions and people.

We would like to mention the financial support from the Flemish Government (Concerted Action GOA-MIPS *Model-Based Information Processing Systems*), the National Fund for Scientific Research, the Federal Government (Interuniversity Attraction Poles IUAP-17 *Modeling and Control of Dynamic Systems* and IUAP-50 *Automation in Design and Production*) and the European Commission (Human Capital and Mobility SIMONET *System Identification and Modeling Network*).

Our gratitude also goes to the many people, who, in one way or another, directly or indirectly, have contributed to this work: Lennart Ljung and Tomas McKelvey (Linköping University, Sweden), Stephen Boyd, Thomas Kailath and Gene Golub (Stanford University, USA), Björn Ottersten, Bo Wahlberg and Anders Lindquist (Royal Institute of Technology, Stockholm), Mats Viberg (Chalmers University of Technology, Sweden), Okko Bosgra, Paul Van den Hof (Technical University Delft, The Netherlands), Manfred Deistler (Technical University of Vienna, Austria), Jan Willems (Rijksuniversiteit Groningen, The Netherlands), Jan Maciejowski (Cambridge University, England), Wally Larimore (ADAPTX, USA), Vasile Sima (Research Institute for Informatics, Romania), Torsten Söderström and Petre Stoica (Uppsala University, Sweden), Giorgio Picci (University of Padua, Italy), Jan Van Schuppen (Centrum voor Wiskunde en Informatica, The Netherlands), Michel Verhaegen (Delft University of Technology, The Netherlands) and last but not least, *les amis* of our sister Université Catholique de Louvain, Michel Gevers, Georges 'Jojo' Bastin *et les autres*. To all our colleagues and friends of our home university, including the several generations of PhD and Master students, thank you!

The constructive feedback of our industrial partners, Henk Aling and Robert Kosut (Integrated Systems Inc., CA, USA) and Ton Backx, Jobert Ludlage and Yu-Cai Zhu (Setpoint-IPCOS, the Netherlands), was instrumental in changing our view on system identification from *practical* 'real' data. To Alexandra Schmidt, we are especially in debt for her clear advise and great friendship.

Last but not least, we thank our family for their support: Our parents, parents in law, brothers and sisters, our wives Annelies (Peter's) and Hilde (Bart's), Bart's children Thomas and Hannah (for sharing with us their highly original opinions) and Peter's soon to be born child X.

It's to them all that we dedicate this book.

Peter Van Overschee
Bart De Moor

Leuven, January 1, 1996.

1

INTRODUCTION, MOTIVATION AND GEOMETRIC TOOLS

"The development of Subspace Methods
is the most exciting thing
that has happened to system identification
the last 5 years or so . . . "
Professor Lennart Ljung from Linköping, Sweden
at the second European Research Network
System Identification workshop
Louvain-la-Neuve, October 2, 1993.

In this Chapter, we summarize the main contributions of the book. In Section 1.1, we first give a short motivation for dealing with the multivariable system identification problem. In Section 1.2, we discuss in some more detail the main contributions which make that subspace identification algorithms are excellent tools to work with in an industrial environment. We also provide some historical background and compare our achievements to previously existing approaches to find black box mathematical models of systems. Notes on the organization of the book and a Chapter by Chapter overview can be found in Section 1.3. Finally, Section 1.4 introduces the main geometric and statistical tools, used for the development of, and the insights in subspace identification algorithms.

1.1 MODELS OF SYSTEMS AND SYSTEM IDENTIFICATION

A dynamic model, pictorially described in Figure 1.1, covers almost all physical, economical, biological, industrial, technical, etc. . .phenomena. One could distinguish

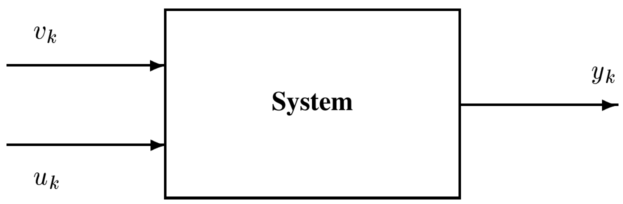

Figure 1.1 A dynamic system with deterministic inputs u_k, outputs y_k and disturbances v_k (see below). All arrows represent vector signals and k is the discrete time index. The user can control u_k but not v_k. In some applications, either u_k or v_k can be missing. The measured (input and) output signals provide useful information about the unknown system.

between mental, intuitive or verbal models, or graphically oriented approaches such as graphs and tables, but we will mainly be interested in mathematical models. Such models are described as differential (continuous time) or difference (discrete time) equations. They describe the dynamic behavior of a system as a function of time. Mathematical models exist in all scientific disciplines, and, as a matter of fact, form the heart of scientific research itself. They are used for simulation, operator training, analysis, monitoring, fault detection, prediction, optimization, control system designs, quality control, *etc.*... Typically, models are highly useful in those situations in which experimenting with the real system is too expensive, too dangerous, too difficult or merely impossible. Last but not least, mathematical models are used for *control* and *feedback*.

Basically, there are two main roads to construct a mathematical model of a dynamic system. Physicists will be interested in models (physical laws) that carefully explain the underlying essential mechanisms of observed phenomena and that are not *falsified* by the available experiments. The necessary mathematical equipment is that of *non-linear partial differential equations*. This is the analytic approach, which rigorously develops the model from *first principles*.

For engineers however, this framework is often much too involved to be really *useful*. The reason is that engineers are not really interested in the exact model *as such*, but more in the potential engineering applications of models. In this perspective, a mathematical model is only one step in the global design of a system. The quality of a model is dictated by the ultimate goal it serves. Model uncertainty is allowed as long as the robustness of the overall system is ensured. Engineers -in contrast with mathematical physicists- are prepared to *trade-off model complexity versus accuracy*.

A complex model will lead to a complex design, while a simplistic model will deteriorate overall performance and robustness of the final implementation. As an example, the best model for simulation (for instance a set of partial differential equations which accurately models the system's behavior) is not the best one for control, because, as a generic property of control system design, the complexity of the controller and the degree of difficulty associated with its implementation, are proportional to the model complexity. Therefore engineers will typically use *system identification* techniques to build their models. This is the field of modeling dynamical systems from experimental data: Experiments are performed on a system, a certain parameterized model class is predefined by the user and suitable numerical values are assigned to the parameters so as to fit as closely as possible the recorded data. In this sense, system identification is the dynamic extension of curve fitting. Finally there is a validation step, in which the model is tried out on experimental data which were not used in the system identification experiment.

In Chapter 6, we describe an industrial process which perfectly illustrates the fundamentally different point of view between the two modeling approaches. The glass-tube manufacturing process described there could in principle be characterized completely using the laws of physics (in this case the laws that govern the behavior of solidifying glass). But, not only would this be a formidable task -if practically possible at all-, but even if there was such a model, it would be impossible to derive an appropriate control action to regulate the system, because of the complexity of the model. However, in Chapter 6, it will be shown how a relatively simple state space model, obtained from measurements as in Figure 1.2 and by application of the mathematical methods described in this book, allows for the design of a high quality minimum variance controller. The quality improvement induced by this controller is illustrated in Figure 1.3.

The message is that system identification provides a meaningful engineering alternative to physical modeling. Compared to models obtained from physics, system identification models have a limited validity and working range and in some cases have no direct physical meaning. But, they are relatively easy to obtain and use and even more importantly, these models are simple enough to make model-based control system design mathematically (and also practically) tractable. Of course, there are still problems such as the choice of an appropriate model structure, the fact that many systems are time-varying and the often largely underestimated measurement problems (appropriate sensors, sampling times, filters, outlier detection, etc...).

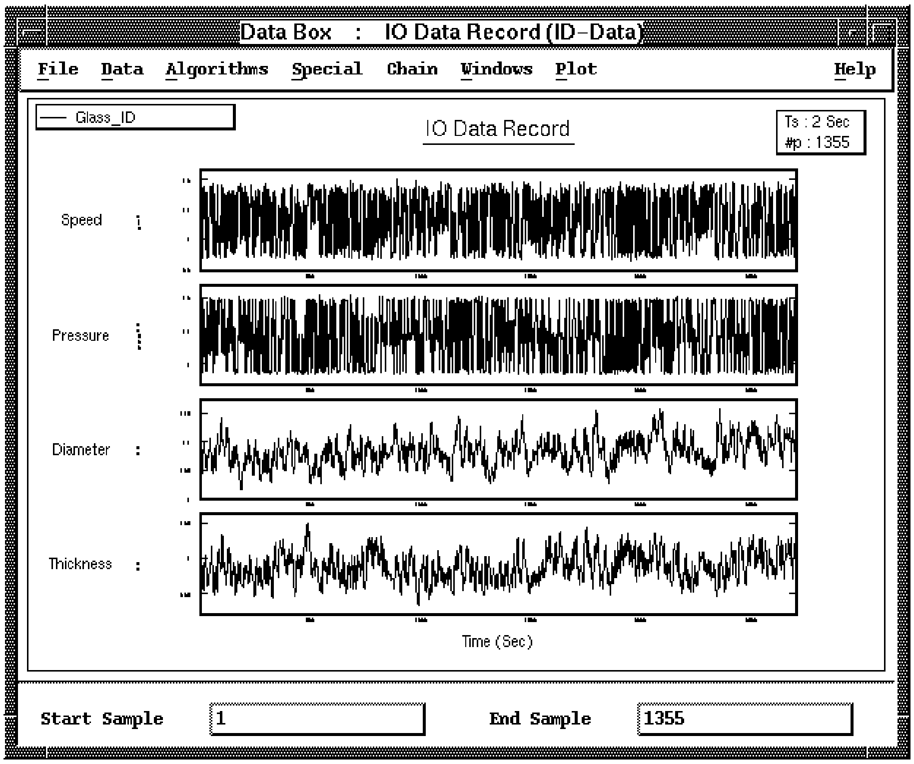

Figure 1.2 Data set used for the identification of the glass tube production process of Chapter 6. The process is excited using pseudo-binary noise sequences as inputs (top two signals). The diameter and the thickness of the produced glass tubes (bottom two signals) are recorded. Solely based on this information, and using the subspace identification algorithms described in this book, a mathematical model of the glass-tube manufacturing process is derived. This mathematical model is then used to design an optimal control strategy.

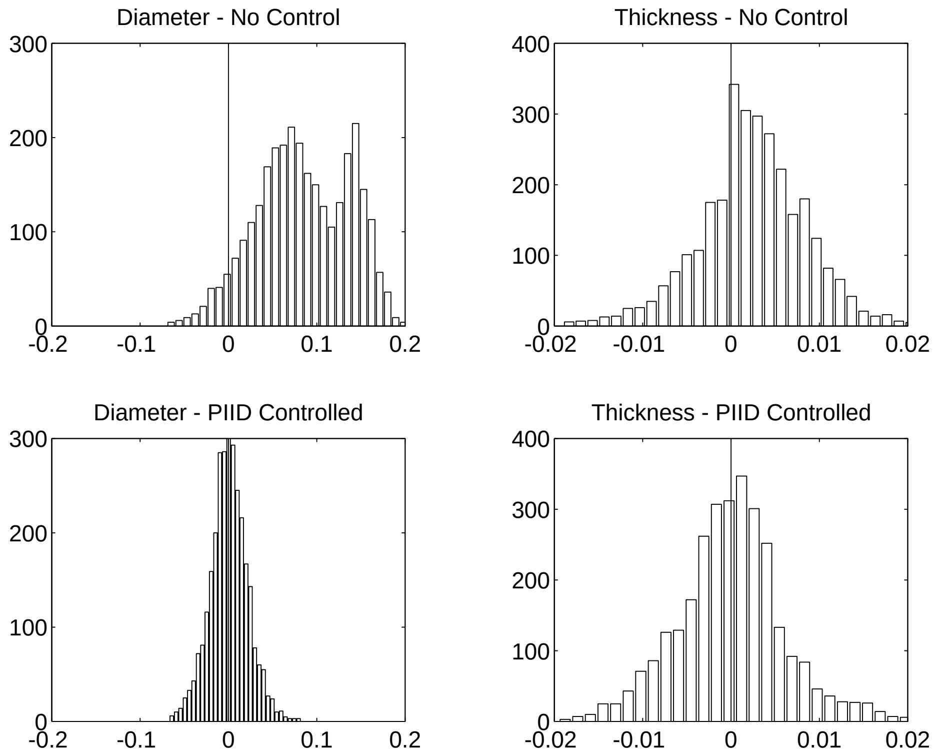

Figure 1.3 Illustration of the quality improvement. The top two figures show a histogram of the deviation from the setpoint for the tube diameter and thickness without the optimal controller installed. The reference setpoint for production corresponds to zero (the vertical line). Clearly, both diameter and thickness are too large (on average). Especially the diameter does not satisfy the production specifications. The bottom two figures show the histograms of the controlled system. The variance on the diameter is a factor two smaller. The mean diameter is exactly at its reference. The variance of the thickness is not reduced (not important in the specifications). However the mean value is right at the specification now. This Figure illustrates the benefits of subspace identification and of model-based control design.

1.2 A NEW GENERATION OF SYSTEM IDENTIFICATION ALGORITHMS

This Section contains a description of the central ideas of this book. First of all, in Subsection 1.2.1, we describe the central importance of state space models*, which is the type of models that is delivered by subspace identification algorithms. In Subsection 1.2.2 we explain how subspace identification algorithms work. In Subsection 1.2.3, we highlight the main innovations of subspace identification algorithms with respect to existing "classical" approaches. Subsection 1.2.4 situates the development of subspace identification algorithms in an historical context by indicating that some of the concepts used in their development are more than 100 years old (besides more modern insights of course).*

1.2.1 State space models are good engineering models

It goes without saying that there is an infinite collection of mathematical models. In this book, we have restricted ourselves to discrete time, linear, time-invariant, state space models. From the number of epitheta used, this might seem like a highly restricted class of models (especially the fact they are linear), but, surprisingly enough, many industrial processes can be described very accurately by this type of models. Moreover, by now, the number of control system design tools that are available to build a controller based on this type of models, is almost without bound (for example [BB 91] [FPW 90]). Especially for this reason, this model class is a very interesting one.

Mathematically, these models are described by the following set of difference equations[1]:

$$x_{k+1} = Ax_k + Bu_k + w_k \ , \tag{1.1}$$
$$y_k = Cx_k + Du_k + v_k \ , \tag{1.2}$$

with

$$\mathbf{E}[\begin{pmatrix} w_p \\ v_p \end{pmatrix} \begin{pmatrix} w_q^T & v_q^T \end{pmatrix}] = \begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} \delta_{pq} \quad \geq 0 \ . \tag{1.3}$$

In this model, we have

vectors: The vectors $u_k \in \mathbb{R}^m$ and $y_k \in \mathbb{R}^l$ are the measurements at time instant k of respectively the m inputs and l outputs of the process. The vector $x_k \in \mathbb{R}^n$ is the state vector of the process at discrete time instant k and contains the numerical

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

values of n states. These states do not necessarily have a direct physical interpretation but they have a conceptual relevance. Of course, if the system states would have some physical meaning, one could always find a similarity transformation of the state space model to convert the states to physically meaningful ones. $v_k \in \mathbb{R}^l$ and $w_k \in \mathbb{R}^n$ are unmeasurable vector signals. It is assumed that they are zero mean, stationary, white noise vector sequences.

matrices: $A \in \mathbb{R}^{n\times n}$ is called the (dynamical) system matrix. It describes the dynamics of the system (as completely characterized by its eigenvalues). $B \in \mathbb{R}^{n\times m}$ is the input matrix which represents the linear transformation by which the deterministic inputs influence the next state. $C \in \mathbb{R}^{l\times n}$ is the output matrix which describes how the internal state is transferred to the outside world in the measurements y_k. The term with the matrix $D \in \mathbb{R}^{l\times m}$ is called the direct feedthrough term. In continuous time systems this term is most often 0, which is not the case in discrete time systems due to the sampling. The matrices $Q \in \mathbb{R}^{n\times n}$, $S \in \mathbb{R}^{n\times l}$ and $R \in \mathbb{R}^{l\times l}$ are the covariance matrices of the noise sequences w_k and v_k. The matrix pair $\{A, C\}$ is assumed to be observable, which implies that all *modes* in the system can be observed in the output y_k and can thus be identified. The matrix pair $\{A, [\ B \quad Q^{1/2}\ ]\}$ is assumed to be controllable, which in its turn implies that all *modes* of the system are excited by either the deterministic input u_k and/or the stochastic input w_k.

A graphical representation of the system can be found in Figure 1.4. Let us comment in some detail why it is often a good idea to try to fit experimental (industrial) process data to the model just described.

- First of all, for multiple-input, multiple output systems, the state space representation is the only model that is convenient to work with in *computer aided control system design* (**CACSD**). Most optimal controllers can be effectively computed in terms of the state space model, while for other system representations (such as e.g. matrix fractional forms [Kai 80]) the calculations are not so elegant.

- Observe that we have collected *all* dynamics in one matrix A, that is to say that the eigenvalues of the matrix A will describe all the dynamical modes that have been measured, whether they come from the real system, from stochastic dynamic disturbances, from measurement sensors or the dynamics of the input actuators. This is quite unusual as compared to approaches that are described in the literature, in which one always carefully distinguishes between e.g. deterministic models (such as models for the "real" system and sensor and actuator dynamics) and noise models for stochastic disturbances (as is for instance the case in the Box-Jenkins approach [BJ 76]). The point here is that more often than not, we do not care

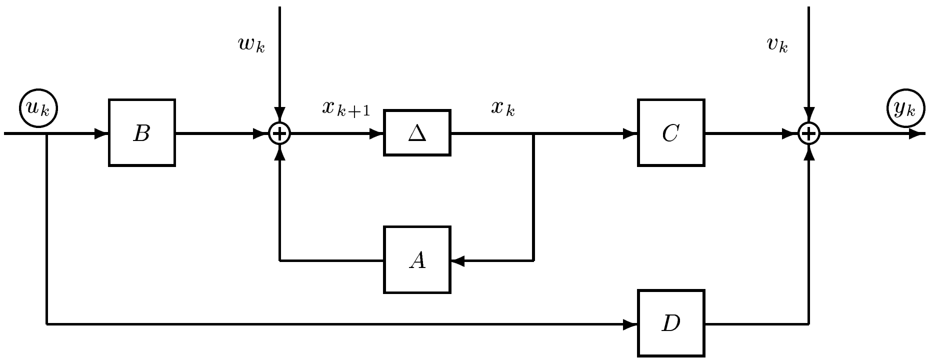

Figure 1.4 This picture is the same as the one in Figure 1.1. But here, we have restricted ourselves to finite dimensional linear time invariant systems to be identified. The (circled) vector signals u_k and y_k are available (measured) while v_k, w_k are unknown disturbances. The symbol Δ represents a delay. Note the inherent feedback via the matrix A (which represents the dynamics). Sensor or actuator dynamics are completely contained in A too. It is assumed that u_k is available without measurement noise.

about the precise origin of the dynamic modes, since, if they are important, they will certainly influence the controller action, independent of their origin. There is a modern trend in **CACSD** to define what is called a *standard plant* (see e.g. [BB 91]), which contains the model of all disturbances, all sensors and the system model in one general state space description, which exactly corresponds to the model we will use.

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

We are now ready to state the main mathematical problem of this book:

> Given input and output measurements $u_1, \ldots, u_s$, and $y_1, \ldots, y_s$. Find an appropriate order n and the system matrices A, B, C, D, Q, R, S.

1.2.2 How do subspace identification algorithms work ?

The goal of this Subsection is to provide a verbal description of the main principles on which subspace identification algorithms are based. The fine mathematical details and proofs will be treated in the next Chapters.

Subspace identification algorithms are based on concepts from system theory, (numerical) linear algebra and statistics, which is reflected in the following table that summarizes the main elements:

System	Geometry	Algorithm
High order state sequence	Projection (orthogonal or oblique)	QR-decomposition
Low order state sequence	Determine finite dimensional subspace	(Generalized) singular value decomposition
System matrices	Linear relations	Least squares

The main conceptual novelties in subspace identification algorithms are:

- The *state of a dynamical system* is emphasized in the context of system identification, whereas "classical" approaches are based on an input-output framework. The difference is illustrated pictorially in Figure 1.5. This relatively recent introduction of the state into the identification area may come as a surprise since in control theory and the analysis of dynamical systems, the importance of the concept of state has been appreciated for quite some time now. So an important achievement of the research in subspace identification is to demonstrate how the Kalman filter states can be obtained from input-output data using linear algebra tools (QR and singular value decomposition). An important consequence is that, once these states are known, the identification problem becomes a linear least squares problem in the unknown system matrices. This implies that one possible

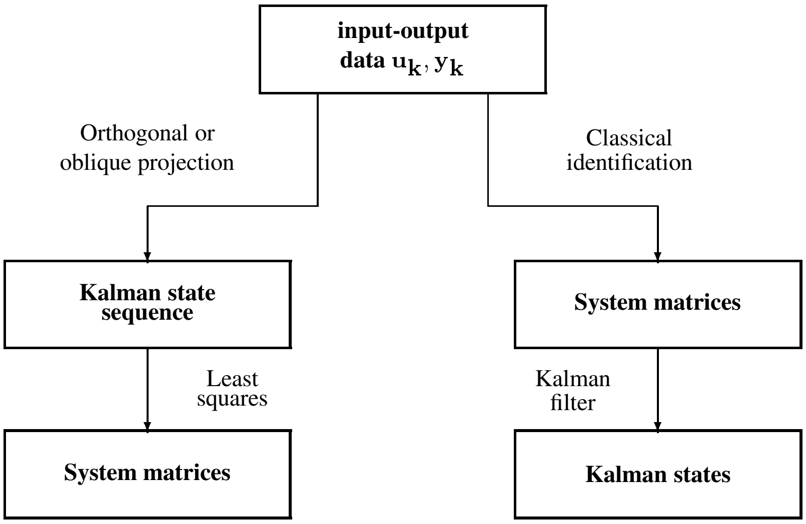

Figure 1.5 System identification aims at constructing state space models from input-output data. The left hand side shows the subspace identification approach : first the (Kalman filter) states are estimated directly from input-output data, then the system matrices can be obtained. The right hand side is the classical approach : first obtain the system matrices, then estimate the states.

interpretation of subspace identification algorithms is that they *conditionally linearize* the problem, which, when written in the "classical" form of prediction error methods [Lju 87], is a highly nonlinear optimization problem. Yet another point of view is that subspace identification algorithms do not identify *input-output* models, but they identify *input-state-output* models.

- The subspace system identification approach of this book makes full use of the by now well developed body of *concepts and algorithms from numerical linear algebra*. While classical methods are basically inspired by least squares, our methods use "modern" algorithms such as the QR - decomposition, the singular value decomposition and its generalizations, and angles between subspaces.

- Our approach provides a *geometric framework*, in which seemingly different models are treated in a unified manner. As will be illustrated at the end of Chapter 4, the deterministic (Chapter 2), stochastic (Chapter 3) and combined deterministic-stochastic (Chapter 4) system identification problem can all be treated with the same geometric concepts and algorithm. We think that the conceptual and algorithmic simplicity of subspace identification algorithms is a major advantage over the classical prediction error approach [Lju 87].

- The conceptual straightforwardness of subspace identification algorithms translates into *user-friendly software implementations*. To give only one example: Since there is no explicit need for parametrizations in our geometric framework, the user is not confronted with highly technical and theoretical issues such as canonical parametrizations, and hence, at the level of possible choices to be offered by the software. This will be illustrated in Chapter 6, where we describe the graphical user interface (**GUI**) software **ISID** that was developed by the authors of this book. It will also become clear from the Matlab files which implement the algorithms of this book.

1.2.3 What's new in subspace identification ?

The mathematical engineering field of system identification has begun to blossom some 15 years ago with the work of Åström [AE 71] [AW 84], Box & Jenkins [BJ 76], Eykhoff [Eyk 74], Ljung [Lju 87] (and many others, see e.g. [Nor 86] [SS 89]). So it is a relatively young branch of research, the industrial spin-offs of which become only gradually visible now. In this Subsection, we confront the innovations in subspace identification with the properties of these "classical' approaches".

Parametrizations: When viewed as a data fitting problem, it becomes clear that system identification algorithms require a certain user-specified parametrization. In subspace identification algorithms we use full state space models and the only "parameter" is the order of the system. For classical algorithmic approaches however, there has been an extensive amount of research to determine so-called *canonical* models, i.e. models with a minimum number of parameters (see e.g. [GW 74] [Gui 75] [Gui 81] [HD 88] [Kai 80] [Lue 67] [VOL 82]). There are however many problems with these minimal parametrizations:

- They can lead to numerically ill-conditioned problems, meaning that the results are extremely sensitive to small perturbations.
- There is a need for *overlapping* parametrizations, since none of the existing parametrizations can cover all possible systems.
- Only minimal state space models are really feasible in practice. If there are for instance uncontrollable but observable (deterministic) modes, this requires special parametrizations.

The subspace identification approach does not suffer from any of these inconveniences. The only parameter to be user-specified is the order of the model, which can be determined by inspection of certain singular values.

Convergence: When implemented *correctly*, subspace identification algorithms are fast, despite the fact that they are using QR and singular value decompositions. As a matter of fact, they are faster than the "classical" identification methods, such as Prediction Error Methods, because they are not iterative (see also the applications in Section 6.4). Hence there are also no *convergence* problems. Moreover, numerical robustness is guaranteed precisely because of these well-understood algorithms from numerical linear algebra. As a consequence, the user will never be confronted with hard-to-deal-with-problems such as lack of convergence, slow convergence or numerical instability.

Model reduction: Since one of our main interests lies in using the models in a computer aided control system design environment and because, when using linear theories, the complexity of the controller is proportional to the order of the system, one is always inclined to obtain models with as low an order as possible. In subspace identification, the reduced model can be obtained *directly*, without having to compute first the high order model, and this directly from input-output data. This is illustrated in Figure 1.6. The interpretation is straightforward within Enns's [Enn 84] weighted balanced reduction framework as will be shown in Chapter 5.

We would like to end this Subsection with a note of Ljung [Lju 91a] "...it remains to be established what these signal subspace methods have to offer and how they compare to conventional approaches ...". We hope that with this book we have bridged a little bit of this gap, a hope which is partially confirmed by the 1993 quote at the beginning of this Chapter.

1.2.4 Some historical elements

In this Subsection, we give an historical survey of the several concepts that are present in subspace identification and that make it to be one of the most powerful and sophisticated identification frameworks that is presently available.

Table 1.1 summarizes in a schematic way the different hallmark contributions and mathematical elements that have lead to and/or are incorporated in some way or another in subspace identification[2]. The idea is twofold: First of all, this table teaches

[2]We apologize a priori for omissions (and mistakes) in this table. It is not always easy to find the "historical truth".

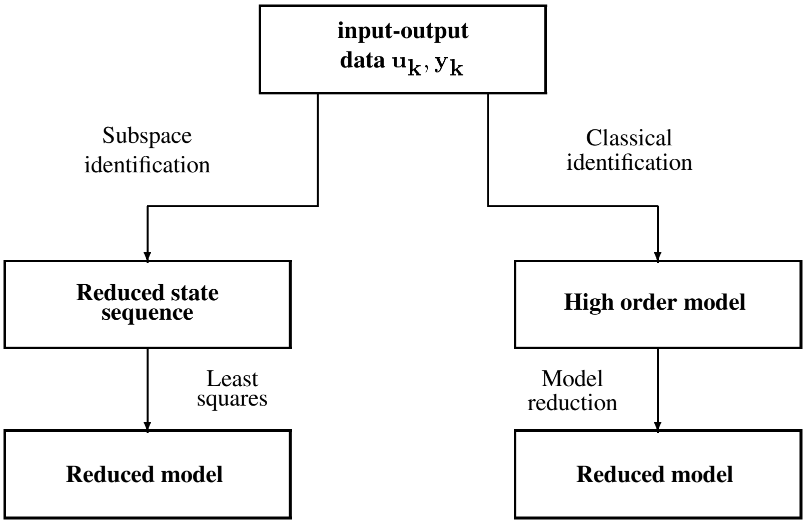

Figure 1.6 System identification aims at constructing state space models from input-output data. When a reduced order model is required, in some classical approaches (to the right), one first identifies a high order model and then applies a model reduction technique to obtain a low order model. The left hand side shows the subspace identification approach: Here, we first obtains a "reduced" state sequence, after which one can identify directly a low order model.

us that certain concepts, such as e.g. angles between subspaces (Jordan, 1875) or the singular value decomposition (Beltrami, Jordan, Sylvester, 1880's) need a long *incubation* period before they are applied in mathematical engineering. Secondly, it shows how clever combinations of seemingly unrelated concepts and techniques may lead to powerful algorithms, such as subspace identification. Of course, space does not permit us here to discuss these contributions in detail.

Let us now summarize the main direct sources of inspiration for this work on subspace identification . First of all, subspace identification algorithms are the input-state-output generalizations of the classical realization theory and algorithms of the seventies, which identify a state space model from impulse responses (Markov parameters), such as [DMK 74a] [DKM 74b] [HK 66] [Kun 78] [Lju 91b] [Moo 81] [MR 76] [Sil 71] [ZM 74]. The insights obtained in these works have really enhanced the understanding of the *structure* of linear systems and their identification. The first papers on obtaining models from input-output data which have influenced this work are [Bud 71] [Gop 69] [LS 77] but more recently, also the work by Willems [Wil 86] was influential for the deterministic parts. Meanwhile, there were other insights obtained in a more statistically oriented context, such as the work by Akaike [Aka 74] [Aka 75], which introduced canonical correlations in the stochastic realization framework. Other influential work was done in [Aok 87] [AK 90] [Cai 88] [DP 84] [DKP 85] [Fau 76]. Related ideas on the combined deterministic-stochastic problem can be found in [Lar 90] [Lar 83] [VD 92] [Ver 91].

Good recent overview papers that contain on overview of the whole class of subspace algorithms (more than is presented in this book) are [VDS 93] [RA 92] [Vib 94].

Year	Name	Contribution	Discipline	Refs.
1809	Gauss	Least Squares	Statistics	[Gau 1857]
1873	Beltrami	SVD	Algebra	[Bel 1873]
1874	Jordan	SVD	Algebra	[Jor 1874]
1875	Jordan	Angles between subspaces	Algebra	[Jor 1875]
1883	Gram	QR	Algebra	[Gra 1883]
1885	Sylvester	SVD	Algebra	[Syl 1889]
1907	Schmidt	QR	Algebra	[Sch 07]
1913	Autonne	SVD	Algebra	[Aut 13]
1936	Eckart	SVD	Physics (!)	[EY 36]
1936	Hotelling	Canonical correlations	Statistics	[Hot 36]
1960	Kalman	Kalman Filter	System Theory	[Kal 60]
1965	Golub/Kahan	SVD-algorithms	Numerical lin.alg.	[GVL 89]
1966	Ho/Kalman	Realization	System Theory	[HK 66]
1974	Zeiger/McEwen	SVD & Realization	System Theory	[ZM 74]
1974	Akaike	Stochastic Realization	Statistics	[Aka 74,75]
1976	Box-Jenkins	Box-Jenkins models	Statistics	[BJ 76]
1976	Faure	Stochastic linear systems	System Theory	[Fau 76]
1978	Kung	Realization theory	System theory	[Kun 78]
1986	Willems	Behavioral framework	System Theory	[Wil 86]
1987	Ljung	Prediction Error	System Theory	[Lju 87]

Table 1.1 Schematic summary of the different hallmark contributions and mathematical elements that have lead to and/or are incorporated in some way or another in subspace identification. This table teaches us that certain concepts, such as e.g. angles between subspaces (Jordan, 1875) or the singular value decomposition (Beltrami, Jordan, Sylvester, 1880's) need a long *incubation* period before they are applied in mathematical engineering. It also shows how clever combinations of seemingly unrelated concepts and techniques may lead to powerful subspace algorithms.

This book came about as the logical consequence of the evolution of subspace identification algorithms from a purely deterministic context [DMo 88] [MDMVV 89], to the purely stochastic problem [VODM 93a]. In this book we combine the two approaches in one unifying combined deterministic-stochastic framework [VODM 95a]. Note also that this book has lead to software implementations [AKVODMB 93] [VODMAKB 94], which have been applied to real industrial processes [FVOMHL 94] [DMVO 94] [VVDVOV 94] [VODM 93c] [ZVODML 94] [VODMAKB 94] [VO 94].

1.3 OVERVIEW

When confronted with a sizeable amount of research material and results, there are different ways of organizing it. In this Section we motivate the organization of this book[3]*. A Chapter by Chapter overview is also given.*

- A first possible organization is to start with the most general and thus most complicated system identification algorithm. The simpler identification problems are then presented as special cases of the general problem. The advantage of this organization is that the overlap between different Chapters is minimal. The major disadvantage however, is that the reader is immediately confronted with the most complicated case, which can be rather confusing.
- The second (chronological) organization consists of a gradual increase of complexity of the problem to be solved in each Chapter. In this way, the reader is introduced slowly to the concepts and has the time to assimilate them before the more complicated cases are treated. This is also the (natural) way the research work came about. The disadvantage of this order of presentation is that there will always be a certain amount of overlap between the different Chapters. However, we found the advantage of increased readability to outweigh this disadvantage, and thus have chosen the chronological presentation.

The Chapters of this book are organized as follows (see also Figure 1.7):

Chapter 1

contains the introduction and the motivation for linear system identification in general and for subspace identification more specifically. This Chapter also contains the origin and the innovative features of subspace identification algorithms. Finally the geometric and statistical tools are introduced.

[3] We believe this motivation increases the readability.

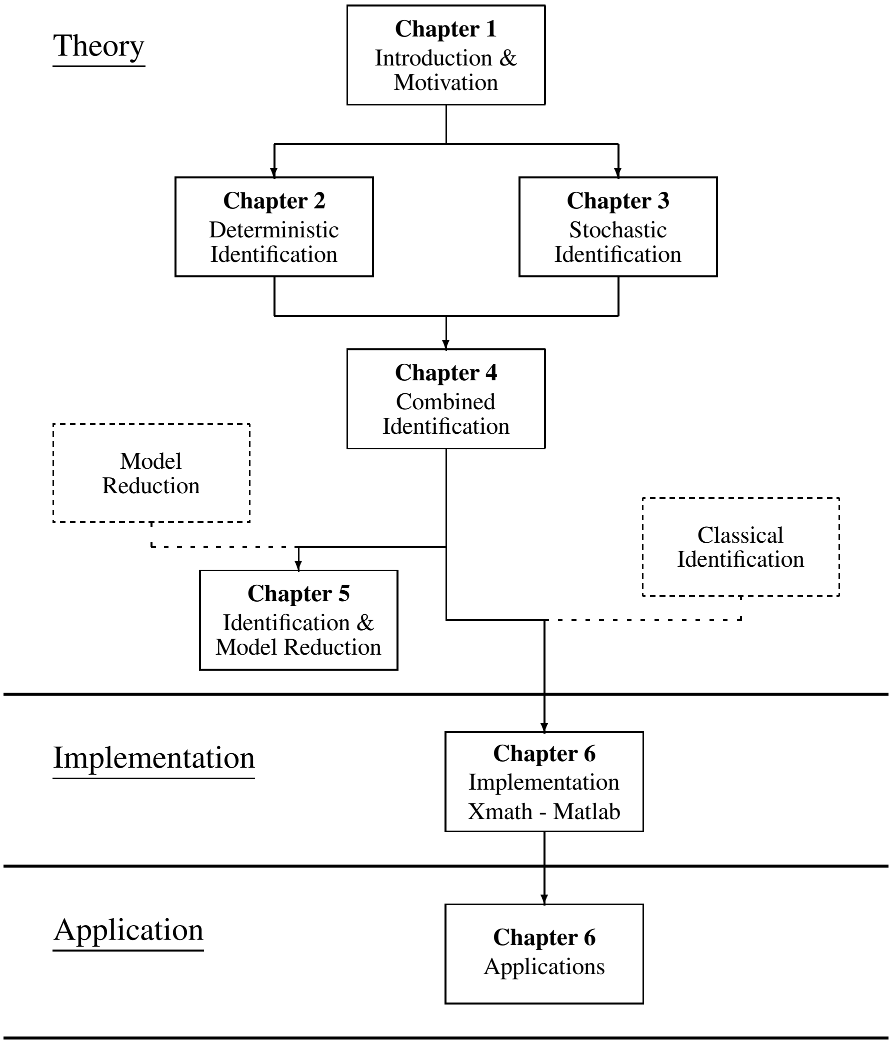

Figure 1.7 Chapter by Chapter overview of the book: Theory - Implementation - Application. The dotted boxes indicate related research work contributing to the results of certain Chapters.

Chapter 2

introduces the (simple) problem of the subspace identification of deterministic systems, where both the process noise w_k and the measurement noise v_k are identically zero:

$$\begin{aligned} w_k &\equiv 0 \,, \\ v_k &\equiv 0 \,. \end{aligned}$$

Even though many results had been obtained in this area already, we treat this problem for two reasons:

- Most of the conceptual ideas and geometric concepts, which will also be used in the Chapters to follow, are introduced by means of this simple identification problem.
- We treat the problem from a different point of view as in the literature, which makes it easier to assimilate it as a special case in the Chapters to follow. Similarities between the presented algorithm and the literature are indicated.

The core of this Chapter is a main Theorem indicating how the states can be recovered from given input-output data.

Chapter 3

treats the case of the subspace identification of stochastic systems, with no *external* input u_k:

$$u_k \equiv 0 \,.$$

The properties of stochastic systems are summarized and are then used to device stochastic subspace system identification algorithms. The main Theorem indicates how the Kalman filter states can be recovered from the given output data. By means of this Theorem, three identification algorithms are presented. The connections with existing algorithms are indicated. Finally, the important problem of positive real covariance sequences is addressed and solved.

Chapter 4

treats the general problem of the subspace identification of combined deterministic-stochastic systems. The central part of this Chapter is again a main Theorem showing how the Kalman filter states can be recovered from the given input-output data. The Theorem leads to two algorithms, of which one is simple but asymptotically biased and the other more complicated but asymptotically unbiased. This last algorithm is further refined to make it suitable and robust for practical (industrial) applications. We also show how the presented theory ties in with the results in the literature.

Each of the preceding Chapters contains one *main Theorem*. These main Theorems state for each problem how the (Kalman filter) states can be recovered from the (input)-output data. At the end of Chapter 4 we indicate how the stochastic and deterministic Theorems can be considered as special cases of the combined deterministic-stochastic Theorem.

Chapter 5

treats the connections between subspace identification algorithms and model reduction. It is shown how the state space basis in which the models are calculated is uniquely determined by the spectrum of the inputs and by user defined weights. The main observation in this Chapter is that there exists a connection between subspace system identification and frequency weighted model reduction. The interpretation of the results leads to more insight in the behavior of subspace identification algorithms, and has consequences for the low order identification problem.

Chapter 6

treats the numerical implementation of the subspace identification algorithms. This implementation has been carried out in a graphical user interface environment. The concepts and new ideas behind this implementation are briefly sketched. The resulting commercial toolbox contains, apart from the subspace identification algorithms, a whole scale of processing, classical identification and validation algorithms.

The toolbox is used to identify an industrial glass tube manufacturing process. Based on the resulting model, an optimal controller for the process is designed, which significantly reduces the variations of the production parameters of the process.

Finally, we give an overview of the application of the Matlab files accompanying this book (implementing the subspace identification algorithms) to ten different practical (industrial) examples. These results show that the developed algorithms work well in practice.

Chapter 7

contains the conclusions of the presented work. Since the research in subspace identification algorithms is far from being a closed area, the major open problems that were spotted during our research are also listed.

1.4 GEOMETRIC TOOLS

Subspace identification algorithms are often based on geometric concepts: As will be shown in the Chapters to follow, some system characteristics can be revealed by geometric manipulation of the row spaces of certain matrices. In Subsections 1.4.1 through 1.4.3 we introduce the main geometric tools used throughout the book. They are described from a linear algebra point of view, independently of the system identification framework we will be using in the next Chapters. Subsection 1.4.4 gives a geometric interpretation of the major statistical assumption used in subspace identification, while Subsection 1.4.5 (re-)defines the geometric operations in a statistical framework.

In the following Subsections we assume that the matrices $A \in \mathbb{R}^{p\times j}, B \in \mathbb{R}^{q\times j}$ and $C \in \mathbb{R}^{r\times j}$ are given. The elements of a row of one of the given matrices can be considered as the coordinates of a vector in the j-dimensional ambient space. The rows of each matrix A, B, C thus define a basis for a linear vector space in this ambient space. In Subsection 1.4.1 through 1.4.3 we define three different geometric operations that can be performed with these row spaces. It should be noted that these geometric operations can be easily implemented using an RQ decomposition. We will not pursue this any further in this Chapter, but refer the reader to Section 6.1 for the numerical implementation issues.

1.4.1 Orthogonal projections

Π_B denotes the operator that projects the row space of a matrix onto the row space of the matrix $B \in \mathbb{R}^{q\times j}$:

$$\Pi_B \stackrel{\text{def}}{=} B^T.(BB^T)^\dagger.B \ ,$$

where $\bullet^\dagger$ denotes the Moore-Penrose pseudo-inverse of the matrix $\bullet$. $A/\boldsymbol{B}$ is shorthand for the projection of the row space of the matrix $A \in \mathbb{R}^{p\times j}$ on the row space of the matrix B:

$$\begin{aligned} A/\boldsymbol{B} &\stackrel{\text{def}}{=} A.\Pi_B \\ &= AB^T.(BB^T)^\dagger.B \ . \end{aligned}$$

The projection operator can be interpreted in the ambient j-dimensional space as indicated in Figure 1.8. The RQ decomposition is the natural numerical tool for this orthogonal projection as will be shown in Section 6.1.

Note that in the notation $A/\boldsymbol{B}$ the matrix B is printed bold face, which indicates that the result of the operation $A/\boldsymbol{B}$ lies in the row space of B. We will adhere to this

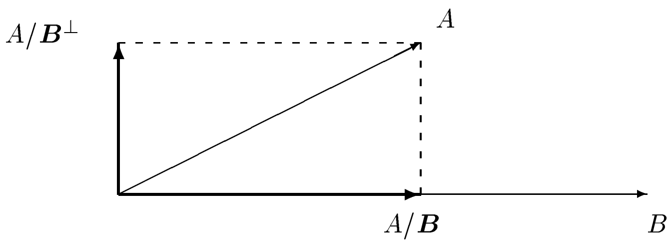

Figure 1.8 Interpretation of the orthogonal projection in the j-dimensional space ($j = 2$ in this case). A/B is formed by projecting the row space of A on the row space of B. $A/B^\perp$ on the other hand is formed by projecting the row space of A on the orthogonal complement of the row space of B. Note the boldface notation for the row space onto which one projects.

convention, also for the geometric operations to follow, which improves the readability of the formulas.

$\Pi_{B^\perp}$ is the geometric operator that projects the row space of a matrix onto the orthogonal complement of the row space of the matrix B:

$$A/\boldsymbol{B}^\perp \stackrel{\text{def}}{=} A.\Pi_{B^\perp} \ ,$$

where:

$$\Pi_{B^\perp} = I_j - \Pi_B \ .$$

Once again these projections can be interpreted in the j-dimensional space as indicated in Figure 1.8. The combination of the projections Π_B and $\Pi_{B^\perp}$ decomposes a matrix A into two matrices of which the row spaces are orthogonal:

$$A = A.\Pi_B + A.\Pi_{B^\perp} \ .$$

Alternatively, the projections decompose the matrix A as linear combination of the rows of B and of the rows of the orthogonal complement of B. With:

$$\begin{aligned} L_B.B &\stackrel{\text{def}}{=} A/\boldsymbol{B} \ , \\ L_{B^\perp}.B^\perp &\stackrel{\text{def}}{=} A/\boldsymbol{B}^\perp \ , \end{aligned}$$

where $B^\perp$ is a basis for the orthogonal complement of the row space of B, we find:

$$A = L_B.B + L_{B^\perp}.B^\perp \ ,$$

which is indeed a decomposition of A into a sum of linear combinations of the rows of B and of $B^\perp$.

1.4.2 Oblique projections

Instead of decomposing A as linear combinations of two orthogonal matrices (B and $B^\perp$), it can also be decomposed as linear combinations of two non-orthogonal matrices B and C and of the orthogonal complement of B and C. This is illustrated in Figure 1.9. The rows of a matrix A are decomposed as linear combinations of the rows of B and C and of the rows of a third matrix orthogonal to B and C. This can be written as:

$$A = L_B.B + L_C.C + L_{B^\perp,C^\perp}.\begin{pmatrix} B \\ C \end{pmatrix}^\perp .$$

The matrix $L_C.C$ is defined[4] as the oblique projection of the row space of A along the row space of B on the row space of C:

$$A/_B\boldsymbol{C} \stackrel{\text{def}}{=} L_C.C .$$

Figure 1.9 illustrates the oblique projection in the j-dimensional space. The name *oblique* refers to the non-orthogonal projection direction. The oblique projection can also be interpreted through the following recipe: Project the row space of A orthogonally on the joint row space of B and C; and decompose the result along the row space of C. Mathematically, the orthogonal projection of the row space of A on the joint row space of B and C can be stated as:

$$A/\begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{B} \end{pmatrix} = A\begin{pmatrix} C^T & B^T \end{pmatrix}.\begin{pmatrix} CC^T & CB^T \\ BC^T & BB^T \end{pmatrix}^\dagger.\begin{pmatrix} C \\ B \end{pmatrix} .$$

Decomposing this expression along the row spaces of B and C leads to the following definition for the oblique projection:

Definition 1 Oblique projections

The oblique projection of the row space of $A \in \mathbb{R}^{p\times j}$ along the row space of $B \in \mathbb{R}^{q\times j}$ on the row space of $C \in \mathbb{R}^{r\times j}$ is defined as:

$$A/_B\boldsymbol{C} \stackrel{def}{=} A\begin{pmatrix} C^T & B^T \end{pmatrix}.\left[\begin{pmatrix} CC^T & CB^T \\ BC^T & BB^T \end{pmatrix}^\dagger\right]_{\textit{first r columns}}.C . \qquad (1.4)$$

[4]Note that this intuitive definition of L_B and L_C is only unique when B and C are of full row rank and when the intersection of the row spaces of B and C is empty. A unique definition is presented in Definition 1.

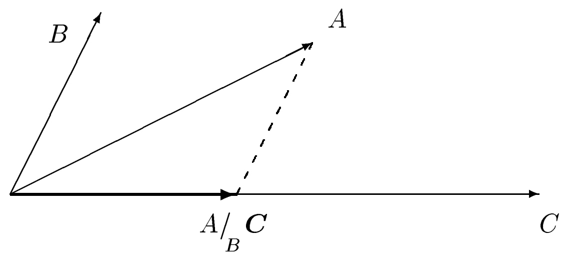

Figure 1.9 Interpretation of the oblique projection in the j-dimensional space ($j = 2$ in this case). The oblique projection is formed by projecting the row space of A along the row space of B on the row space of C.

Some properties of the oblique projection are:

$$B/_{B}\boldsymbol{C} = 0 , \tag{1.5}$$
$$C/_{B}\boldsymbol{C} = C . \tag{1.6}$$

Actually, as indicated in [BK 79], these two properties can be used to define the oblique projection i.e. any operation that satisfies (1.5)-(1.6) is an oblique projection. From (1.5)-(1.6) it can now be easily shown that an equivalent definition of the oblique projection is:

Corollary 1 Oblique projections

The oblique projection of the row space of $A \in \mathbb{R}^{p\times j}$ along the row space of $B \in \mathbb{R}^{q\times j}$ on the row space of $C \in \mathbb{R}^{r\times j}$ can also be defined as:

$$A/_{B}\boldsymbol{C} = \left[A/\boldsymbol{B}^{\perp}\right] . \left[C/\boldsymbol{B}^{\perp}\right]^{\dagger} .C . \tag{1.7}$$

Note that when $B = 0$ or when the row space of B is orthogonal to the row space of C ($B.C^T = 0$) the oblique projection reduces to an orthogonal projection:

$$A/_{B}\boldsymbol{C} = A/\boldsymbol{C} .$$

This fact will be used when unifying the three main Theorems of Chapter 2, 3 and 4 in Section 4.5.

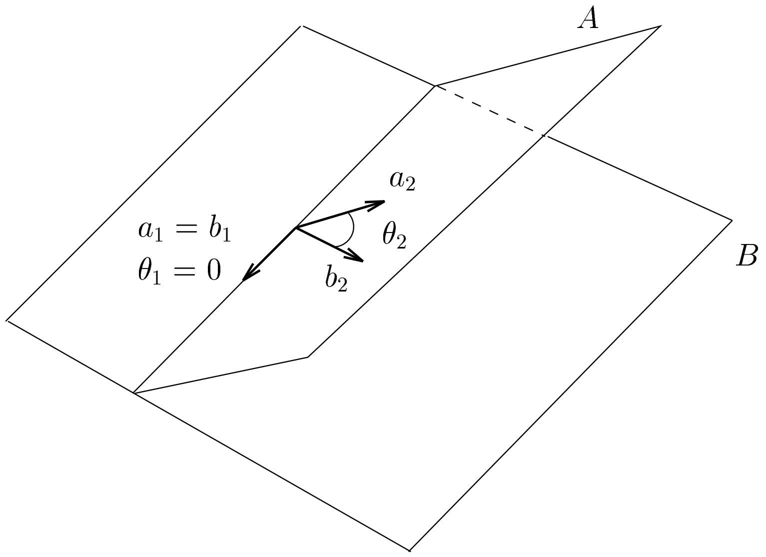

Figure 1.10 Interpretation of principal angles and directions in the j-dimensional space ($j = 3$ in this case). The unit vectors $a_1 = b_1$ indicate the first principal directions. The angle θ_1 between them is the first principal angle and is in this case equal to zero ($\theta_1 = 0$). Zero principal angles imply a non-empty intersection between the row spaces of A and B. The corresponding principal directions can be chosen as a basis for this intersection. The second principal angle θ_2 is the angle between the unit vectors a_2 and b_2. Since a_2 and b_2 have to be orthogonal to $a_1 = b_1$, these vectors lie in a plane orthogonal to the intersection of the planes A and B.

1.4.3 Principal angles and directions

The principal angles between two subspaces are a generalization of an angle between two vectors as illustrated in Figure 1.10 (the concept goes back to Jordan [Jor 1875]). Suppose we are given two matrices $A \in \mathbb{R}^{p \times j}$ and $B \in \mathbb{R}^{q \times j}$. The first principal angle θ_1 (the smallest one) is obtained as follows: Choose unit vectors $a_1 \in$ row space A and $b_1 \in$ row space B and minimize the angle between them. This is the first principal angle and the unit vectors a_1 and b_1 are the first principal directions. Next choose a unit vector $a_2 \in$ row space A orthogonal to a_1 and $b_2 \in$ row space B orthogonal to b_1 and minimize the angle θ_2 between them. These are the second principal angle and directions. Continue in this way until $\min(p, q)$ angles have been found. Figure 1.10 illustrates this in a three dimensional space. This informal description can also be formalized:

Definition 2 Principal angles and directions

The principal angles $\theta_1 \leq \theta_2 \leq \ldots \leq \pi/2$ *between the row spaces of* A *and* B *of two matrices* $A \in \mathbb{R}^{p\times j}$ *and* $B \in \mathbb{R}^{q\times j}$ *and the corresponding principal directions* $a_i \in$ *row space* A *and* $b_i \in$ *row space* B *are defined recursively as:*

$$\begin{aligned}\cos\theta_k &= \max_{a\,\in \text{row space } A\ ,\ b\in \text{row space } B} a^T.b \\ &= a_k^T.b_k\ ,\end{aligned}$$

subject to $\|a\| = \|b\| = 1$ *and for* $k > 1$, $a^T.a_i = 0$ *for* $i = 1,\ldots,k-1$ *and* $b^T.b_i = 0$ *for* $i = 1,\ldots,k-1$.

In the following we present two alternative definitions for the principal angles and directions. These definitions are a little more practical since they allow for an easy computation of the angles and directions.

Definition 3 Principal angles and directions

Given two matrices $A \in \mathbb{R}^{p\times j}$ *and* $B \in \mathbb{R}^{q\times j}$ *and the singular value decomposition:*

$$A^T.(AA^T)^\dagger.AB^T.(BB^T)^\dagger.B = USV^T\ ,$$

then the principal directions between the row spaces of A *and* B *are equal to the rows of* U^T *and the rows of* V^T. *The cosines of the principal angles between the row spaces of* A *and* B *are defined as the singular values (the diagonal of* S*). The principal directions and angles between the row spaces of* A *and* B *are denoted as :*

$$\begin{aligned}[\boldsymbol{A} \sphericalangle B] &\overset{def}{=} U^T\ ,\\ [A \sphericalangle \boldsymbol{B}] &\overset{def}{=} V^T\ ,\\ [A \sphericalangle B] &\overset{def}{=} S\ .\end{aligned}$$

An alternative definition to compute the principal angles and directions is given in [AK 90] [Pal 82]:

Definition 4 Principal angles and directions

The principal angles and directions between the row spaces of A *and* B *of two matrices* $A \in \mathbb{R}^{p\times j}$ *and* $B \in \mathbb{R}^{q\times j}$ *can be found through the singular value decomposition of* *(where $\bullet^{1/2}$ denotes any square root of a matrix):*

$$(AA^T)^{-1/2}.(AB^T).(BB^T)^{-1/2} = USV^T \ , \tag{1.8}$$

as:

$$\begin{aligned}
[\boldsymbol{A} \lhd B] &= U^T.(AA^T)^{-1/2}.A \ , \\
[A \lhd \boldsymbol{B}] &= V^T.(BB^T)^{-1/2}.B \ , \\
[A \lhd B] &= S \ .
\end{aligned}$$

This definition is very appealing, since when A and B are just vectors ($p = q = 1$), equation (1.8) reduces to:

$$\frac{A.B^T}{\sqrt{AA^T}.\sqrt{BB^T}} = S \ ,$$

which is the classical definition of the cosine between two row vectors.

1.4.4 Statistical tools

In this Subsection we relate statistical assumptions to geometric properties. These properties will be used throughout the book in the proves of, and insights in the main Theorems. They lie at the heart of the reason why subspace identification algorithms work well for large data sets.

Consider two given sequences $a_k \in \mathbb{R}^{n_a}$ and $e_k \in \mathbb{R}^{n_e}$, $k = 0, 1 \ldots, j$. The sequence e_k is a zero mean sequence, independent of a_k:

$$\begin{aligned}
\mathbf{E}[e_k] &= 0 \ , \\
\mathbf{E}[a_k e_k^T] &= 0 \ .
\end{aligned}$$

In subspace identification we typically assume that there are long time series of data available ($j \to \infty$), and that the data is ergodic. Due to ergodicity and the infinite number of data at our disposition, we can replace the expectation operator $\mathbf{E}$ (average over an infinite number of experiments) with the different operator $\mathbf{E_j}$ applied to the sum of variables (average over one, infinitely long, experiment). For instance for the correlation between a_k and e_k we get:

$$\begin{aligned}
\mathbf{E}[a_k e_k^T] &= \lim_{j\to\infty} [\frac{1}{j} \sum_{i=0}^{j} a_i e_i^T] \\
&= \mathbf{E_j}[\sum_{i=0}^{j} a_i e_i^T] \ ,
\end{aligned}$$

with an obvious definition of $\mathbf{E_j}$:

$$\mathbf{E_j}[\bullet] \stackrel{\text{def}}{=} \lim_{j\to\infty} \frac{1}{j}[\bullet] \ .$$

These formulas lie at the heart of the subspace approach. Consider for instance a_k to be the sequence of inputs u_k and e_k to be a disturbance. If we now assume that we have an infinite number of data available (a large set of data samples) and that the data are ergodic and that u_k and e_k are independent, we find that:

$$\mathbf{E_j}[\sum_{i=0}^{j} u_i e_i^T] = 0 \ . \tag{1.9}$$

Putting the data into row matrices:

$$\begin{array}{rcl} u & \stackrel{\text{def}}{=} & \begin{pmatrix} u_0 & u_1 & \dots & u_j \end{pmatrix} \ , \\ e & \stackrel{\text{def}}{=} & \begin{pmatrix} e_0 & e_1 & \dots & e_j \end{pmatrix} \ , \end{array}$$

we find with (1.9) that:

$$\mathbf{E_j}[u.e^T] = 0 \ ,$$

which implies that the input vector u is perpendicular to the noise vector e. So geometrically (and for $j \to \infty$) we can state that the row vectors of disturbances are perpendicular to row vectors of inputs (and to other variables not correlated with the noise). This property is used in subspace identification algorithms to get rid of the noise effects. For instance by projecting the noise on the input[5], the noise is annihilated:

$$\mathbf{E_j}[\ \|e/\boldsymbol{u}\|\ ] = 0 \ .$$

This last formula is illustrated numerically in Figure 1.11. From this it should be clear that subspace identification algorithms are, in general, not well suited for short data records.

More ideas about geometric properties of noise can be found in [DMo 93]. Note also that the idea is closely related to the instrumental variable approach [Lju 87], as also indicated in the papers of Viberg & Ottersten [VOWL 93] [Vib 94].

[5] See Section 1.4.5 for a statistical definition of the projection.

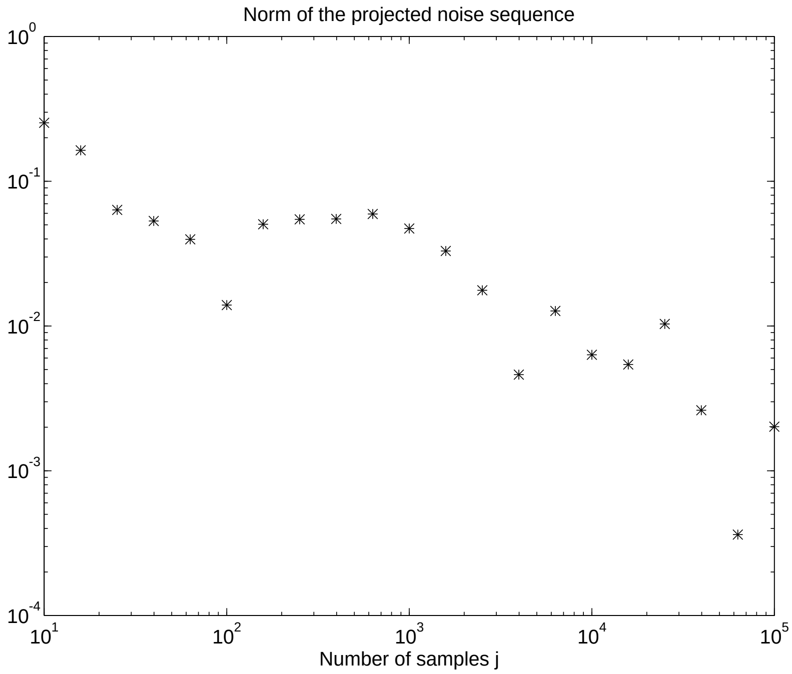

Figure 1.11 Norm of the projected noise sequence e/u, where e and u are vectors with j samples. The norm goes to zero with a factor $1/\sqrt{j}$ as can be seen from the Figure. When j is large enough and e is a zero mean white noise sequence, e and u can be considered perpendicular to each other ($\| e/u\| = 0$). This illustrates why subspace algorithms work well, even in the presence of noise (when a large number of samples is available).

1.4.5 Geometric tools in a statistical framework

In the statistical (stochastic) framework we define the covariance $\Phi_{[A,B]}$ between two matrices $A \in \mathbb{R}^{p\times j}$ and $B \in \mathbb{R}^{q\times j}$ as:

$$\Phi_{[A,B]} \stackrel{\text{def}}{=} \mathbf{E_j}[A.B^T] \ .$$

We can now extend the geometric tools introduced above in the deterministic framework to the stochastic framework, i.e. for ease of notation and to facilitate the theoretical derivations we re-define the geometric operations in a stochastic context. This re-definition simply consists of the following substitution in all definitions:

$$A.B^T \ \leftarrow \ \Phi_{[A,B]} \ .$$

In a statistical framework, we thus get:

$$\begin{aligned}
A/\boldsymbol{B} &= \Phi_{[A,B]}.\Phi^{\dagger}_{[B,B]}.B \ , \\
A/\boldsymbol{B}^{\perp} &= A - \Phi_{[A,B]}.\Phi^{\dagger}_{[B,B]}.B \ , \\
A/_{B}\boldsymbol{C} &= \begin{pmatrix} \Phi_{[A,C]} & \Phi_{[A,B]} \end{pmatrix} . \left[\begin{pmatrix} \Phi_{[C,C]} & \Phi_{[C,B]} \\ \Phi_{[B,C]} & \Phi_{[B,B]} \end{pmatrix}^{\dagger} \right]_{\text{first r columns}} .C \\
&= [A/\boldsymbol{B}^{\perp}].[C/\boldsymbol{B}^{\perp}]^{\dagger}.C \ ,
\end{aligned}$$

and the principal angles and directions from the **SVD** of:

$$\Phi^{-1/2}_{[A,A]}.\Phi_{[A,B]}.\Phi^{-1/2}_{[B,B]} \ , \tag{1.10}$$

as:

$$[\boldsymbol{A} \sphericalangle B] = U^T.\Phi^{-1/2}_{[A,A]}.A \ , \tag{1.11}$$

$$[A \sphericalangle \boldsymbol{B}] = V^T.\Phi^{-1/2}_{[B,B]}.B \ , \tag{1.12}$$

$$[A \sphericalangle B] = S \ . \tag{1.13}$$

We use the same notation for the deterministic and stochastic geometric operations since, when implementing the algorithms the number of measurements will always be finite[6] ($j \neq \infty$) and we approximate $\Phi_{[A,B]}$ as:

$$\Phi_{[A,B]} \simeq \frac{1}{j} \ AB^T \ .$$

Thus the two slightly different definitions in the deterministic and stochastic framework coincide. For instance, for the orthogonal projection:

$$\begin{aligned}
A/\boldsymbol{B} &= \Phi_{[A,B]}.\Phi^{\dagger}_{[B,B]}.B \\
&= [\frac{1}{j} \ AB^T].[\frac{1}{j} \ BB^T]^{\dagger}B \\
&= AB^T.[BB^T]^{\dagger}.B \ ,
\end{aligned}$$

which is exactly the same definition as in the deterministic setting. It should be clear from the context which definition is implied, however the deterministic definition is typically used in Chapter 2, while in Chapters 3 and 4 the stochastic definition is implied.

[6]For many *theoretical* derivations in the stochastic framework, we will however assume that $j \to \infty$.

1.5 CONCLUSIONS

In this Chapter we have motivated the need for mathematical models. We have also provided an overview of the advantages of subspace identification algorithms, as an overview of the major differences with classical identification methods. The mathematical tools have been put in an historical perspective. Finally, we have given an overview of the different Chapters.

In the last Section, we have introduced some geometric tools: orthogonal and oblique projections and principal angles and directions. To solve the stochastic and combined deterministic-stochastic identification problem, these concepts have been extended to the statistical framework.

2

DETERMINISTIC IDENTIFICATION

In this Chapter we treat the subspace identification of purely deterministic systems, with no measurement nor process noise ($v_k \equiv w_k \equiv 0$ in Figure 1.4). We treat this problem for two reasons:

- *Most of the conceptual ideas and geometric concepts, which will also be used in the Chapters to follow, are introduced by means of this simple identification problem.*
- *We treat the problem from a different point of view as in the literature, which makes it easier to assimilate it as a special case in the Chapters to follow. Similarities between the presented algorithm and the literature are pointed out.*

In Section 2.1 we state the deterministic (subspace) identification problem mathematically and introduce the notation. Section 2.2 contains the main Theorem, which is the backbone of this Chapter. The Theorem allows for the extraction of the states directly from input-output data. Relations to other algorithms in the literature are treated in Section 2.3. Finally, Section 2.4 describes how the results of the main Theorem lead to two algorithms that compute the system matrices. Section 2.5 contains the conclusions. Appendix B describes the software implementation of the algorithms in this Chapter.

Before we start the treatment of purely deterministic system identification, it should be noted that most real-life measurements are corrupted by noise. This makes the identification of deterministic systems a fairly academic issue. We refer the reader to Subsection 2.3.3 for a short treatment of the noise effects and to Chapter 4 for the more practical combined deterministic-stochastic identification problem, where the measurements are corrupted by additive noise.

Deterministic identification problem:

Given: s measurements of the input $u_k \in \mathbb{R}^m$ and the output $y_k \in \mathbb{R}^l$ generated by the unknown deterministic system of order n:

$$x_{k+1}^d = Ax_k^d + Bu_k \ , \quad (2.1)$$

$$y_k = Cx_k^d + Du_k \ . \quad (2.2)$$

Determine:

- The order n of the unknown system
- The system matrices $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}, C \in \mathbb{R}^{l\times n}, D \in \mathbb{R}^{l\times m}$ (up to within a similarity transformation).

Figure 2.1 The deterministic subspace identification problem.

2.1 DETERMINISTIC SYSTEMS

2.1.1 Problem description

Deterministic subspace identification algorithms compute state space models from given input-output data. Figure 2.1 states the deterministic[1] (subspace) identification problem. The unknown deterministic system is represented in Figure 2.2.

Several solutions to this problem have been presented in the literature [DMV 87] [DMo 88] [DMVMVVM 88] [DMMVV 88a] [DMMVV 88b] [DMMVV 88c] [MDMVV 89] [MDMV 91] [MDM 92] [VODMS 91] [VD 92]. In this Chapter we present a new Theorem that unifies all these approaches into one general framework. As will be shown in Section 2.3, all methods are closely related. The reason for presenting a new Theorem, is that it fits nicely into the framework of combined system identification of Chapter 4, and thus adds to the general consistency of the work presented in this book.

[1] Note that the superscript "d" in all the variables stands for "deterministic".

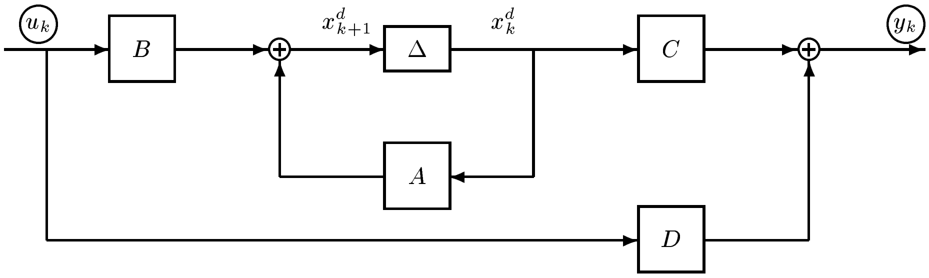

Figure 2.2 A linear time-invariant deterministic system with inputs u_k, outputs y_k and states x_k^d, described by four matrices A, B, C and D. The symbol Δ represents a delay. Note the inherent feedback via the matrix A (representing the dynamics). In the deterministic identification problem, the circled signals (input u_k and output y_k) are known. The state is unknown, but will be determined as an intermediate result in the subspace identification algorithms.

2.1.2 Notation

Block Hankel matrices and state sequences

In this Subsection we introduce the notation for block Hankel matrices and for system related matrices.

Block Hankel matrices play an important role in subspace identification algorithms. These matrices can be easily constructed from the given input-output data. Input block Hankel matrices are defined as:

$$
\begin{array}{rcl}
U_{0|2i-1} \stackrel{\text{def}}{=} & \begin{array}{c} \overset{\text{j}}{\longleftrightarrow} \end{array} & \\
\begin{array}{c} \text{i} \\[5em] \text{i} \end{array} &
\left(\begin{array}{ccccc}
u_0 & u_1 & u_2 & \dots & u_{j-1} \\
u_1 & u_2 & u_3 & \dots & u_j \\
\dots & \dots & \dots & \dots & \dots \\
u_{i-1} & u_i & u_{i+1} & \dots & u_{i+j-2} \\
\hline
u_i & u_{i+1} & u_{i+2} & \dots & u_{i+j-1} \\
u_{i+1} & u_{i+2} & u_{i+3} & \dots & u_{i+j} \\
\dots & \dots & \dots & \dots & \dots \\
u_{2i-1} & u_{2i} & u_{2i+1} & \dots & u_{2i+j-2}
\end{array}\right) &
\begin{array}{c} \text{"past"} \\[5em] \text{"future"} \end{array}
\end{array}
$$

$$
\stackrel{\text{def}}{=} \left(\frac{U_{0|i-1}}{U_{i|2i-1}}\right) \stackrel{\text{def}}{=} \left(\frac{U_p}{U_f}\right)
$$

$$
\begin{array}{rcl}
 & \overset{\text{j}}{\longleftrightarrow} & \\
\begin{array}{c} \text{i+1} \\ \stackrel{\text{def}}{=} \\[4em] \text{i-1} \end{array} &
\left(\begin{array}{ccccc}
u_0 & u_1 & u_2 & \dots & u_{j-1} \\
u_1 & u_2 & u_3 & \dots & u_j \\
\dots & \dots & \dots & \dots & \dots \\
u_{i-1} & u_i & u_{i+1} & \dots & u_{i+j-2} \\
u_i & u_{i+1} & u_{i+2} & \dots & u_{i+j-1} \\
\hline
u_{i+1} & u_{i+2} & u_{i+3} & \dots & u_{i+j} \\
\dots & \dots & \dots & \dots & \dots \\
u_{2i-1} & u_{2i} & u_{2i+1} & \dots & u_{2i+j-2}
\end{array}\right) &
\begin{array}{c} \text{"past"} \\[5em] \text{"future"} \end{array}
\end{array}
$$

$$
\stackrel{\text{def}}{=} \left(\frac{U_{0|i}}{U_{i+1|2i-1}}\right) \stackrel{\text{def}}{=} \left(\frac{U_p^+}{U_f^-}\right)
$$

where:

- The number of block rows (i) is a user-defined index which is large enough i.e. it should at least be larger than the maximum order[2] of the system one wants to identify (see below). Note that, since each block row contains m (number of inputs) rows, the matrix $U_{0|2i-1}$ consists of $2mi$ rows.
- The number of columns (j) is typically equal to $s - 2i + 1$, which implies that all given data samples are used. Throughout the book, for statistical reasons (see

[2]Theoretically, the number of block rows should only be larger than the largest observability index, but since this index is unknown we assume that $i > n$.

also Subsection 1.4.4), we will often assume that $j, s \to \infty$. For deterministic (noiseless) systems this will not be necessary.

- The subscripts of $U_{0|2i-1}, U_{0|i-1}, U_{0|i}$ denote the subscript of the first and last element of the first column in the block Hankel matrix. The subscript "p" stands for "past" and the subscript "f" for "future". The matrices U_p (the past inputs) and U_f (the future inputs) are defined by splitting $U_{0|2i-1}$ in two equal parts of i block rows. The matrices U_p^+ and U_f^- on the other hand are defined by shifting the border between past and future one block row down[3].

- Note that the distinction between past and future is somewhat loose, since both the matrices U_p and U_p^+ are denoted by "past inputs". These loose notations are however useful when explaining concepts intuitively. Note also that the past and future inputs have many elements in common. For instance the input u_i can be found in U_p as in U_f. However, the corresponding columns of U_p and U_f have no elements in common, and thus the distinction between past and future.

The output block Hankel matrices $Y_{0|2i-1}, Y_p, Y_f, Y_p^+, Y_f^-$ are defined in a similar way. Following the notation of Willems [Wil 86], we define the block Hankel matrices consisting of inputs *and* outputs as $W_{0|i-1}$:

$$\begin{aligned} W_{0|i-1} &\stackrel{\text{def}}{=} \begin{pmatrix} U_{0|i-1} \\ Y_{0|i-1} \end{pmatrix} \\ &= \begin{pmatrix} U_p \\ Y_p \end{pmatrix} \\ &= W_p \ . \end{aligned}$$

Similarly as before, W_p^+ is defined as:

$$W_p^+ = \begin{pmatrix} U_p^+ \\ Y_p^+ \end{pmatrix} \ .$$

State sequences play an important role in the derivation and interpretation of subspace identification algorithms. The (deterministic) state sequence X_i^d is defined as:

$$X_i^d \stackrel{\text{def}}{=} \begin{pmatrix} x_i^d & x_{i+1}^d & \dots & x_{i+j-2}^d & x_{i+j-1}^d \end{pmatrix} \quad \in \mathbb{R}^{n \times j} \ ,$$

where the subscript i denotes the subscript of the first element of the state sequence.

[3]The superscript "+" stands for "add one block row" while the superscript "$-$" stands for "delete one block row".

Analogous to the past inputs and outputs, we denote the past state sequence by X_p^d and the future state sequence by X_f^d:

$$X_p^d = X_0^d \qquad , \qquad X_f^d = X_i^d \; . \tag{2.3}$$

System related matrices

Subspace identification algorithms make extensive use of observability and controllability matrices and of their structure. The extended ($i > n$) observability matrix Γ_i (where the subscript i denotes the number of block rows) is defined as:

$$\Gamma_i \stackrel{\text{def}}{=} \begin{pmatrix} C \\ CA \\ CA^2 \\ \dots \\ CA^{i-1} \end{pmatrix} \quad \in \mathbb{R}^{li \times n} \; . \tag{2.4}$$

We assume the pair $\{A, C\}$ to be observable, which implies (see for instance [Kai 80]) that the rank of Γ_i is equal to n. The reversed extended controllability matrix Δ_i^d (where the subscript i denotes the number of block columns) is defined as:

$$\Delta_i^d \stackrel{\text{def}}{=} \begin{pmatrix} A^{i-1}B & A^{i-2}B & \dots & AB & B \end{pmatrix} \quad \in \mathbb{R}^{n \times mi} \; .$$

We assume the pair $\{A, B\}$ to be controllable. The controllable modes can be either stable or unstable. The lower block triangular Toeplitz matrix H_i^d is defined as:

$$H_i^d \stackrel{\text{def}}{=} \begin{pmatrix} D & 0 & 0 & \dots & 0 \\ CB & D & 0 & \dots & 0 \\ CAB & CB & D & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ CA^{i-2}B & CA^{i-3}B & CA^{i-4}B & \dots & D \end{pmatrix} \quad \in \mathbb{R}^{li \times mi} \; .$$

2.2 GEOMETRIC PROPERTIES OF DETERMINISTIC SYSTEMS

In this Section we investigate the geometric properties of deterministic systems. In a first Subsection 2.2.1 we introduce the matrix input-output equations, which lead to the main Theorem for deterministic system identification in Subsection 2.2.2. The Theorem allows the extraction of the state sequence directly from given input-output data. The geometrical interpretation is given in Subsection 2.2.3.

2.2.1 Matrix input-output equations

The following Theorem states how the linear state space relations of formula (2.1)-(2.2) can be reformulated in a matrix form. The Theorem was introduced in [DMo 88], and is very useful in many proofs of, and insights in subspace identification algorithms.

Theorem 1 Matrix input-output equations

$$Y_p = \Gamma_i X_p^d + H_i^d U_p \ , \tag{2.5}$$
$$Y_f = \Gamma_i X_f^d + H_i^d U_f \ , \tag{2.6}$$
$$X_f^d = A^i X_p^d + \Delta_i^d U_p \ . \tag{2.7}$$

The proof follows directly from the state space equations (2.1)-(2.2). The geometric interpretation of equation (2.5) is illustrated in Figure 2.3.

2.2.2 Main Theorem

Before stating the main deterministic identification Theorem, the following remark that emphasizes the symmetry between the different Chapters is in order: For each of the separate identification problems (Chapter 2, 3 and 4) we present a main Theorem which states how the state sequence and the extended observability matrix can be extracted from the given input-output data. After having treated the three Theorems for the three different cases (deterministic, stochastic and combined deterministic-stochastic identification), it will become clear that they are very similar[4]. These similarities will be treated in Section 4.5. We mention this fact early in the book, before the Theorems

[4]Note that these similarities are no coincidence. Chapter 2 and 3 are written with "fore-sight" towards Chapter 4.

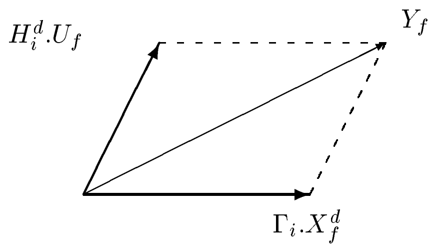

Figure 2.3 Vectors in the row space of the block Hankel matrix Y_f are obtained as a sum of linear combinations of vectors in the row space of the state sequence X_f^d and linear combinations of vectors in the row space of the block Hankel matrix U_f.

are introduced, so that the synthesis in Section 4.5 will be anticipated by the attentive reader.

The consequences of the deterministic identification Theorem are twofold:

- The state sequence X_f^d can be determined directly from the given data u_k and y_k, without knowledge of the system matrices A, B, C, D.
- The extended observability matrix Γ_i can be determined directly from the given input-output data.

In Section 2.4, we will then describe how the the system matrices A, B, C, D can be extracted from these intermediate results X_f^d and Γ_i. An overview of the overall deterministic identification procedure is presented in Figure 2.4.

In the main deterministic identification Theorem, we introduce two weighting matrices W_1 and W_2. The interpretation and significance of these matrices is postponed until Section 2.3 and Chapter 5. Suffices to state here that specific choices of the matrices lead to different identification algorithms of the literature and that the choice of the weights determines the state space basis in which the final model is obtained (see Chapter 5).

In the main Theorem we will also use the concept of persistency of excitation. We adopt the definition of Ljung [Lju 87]:

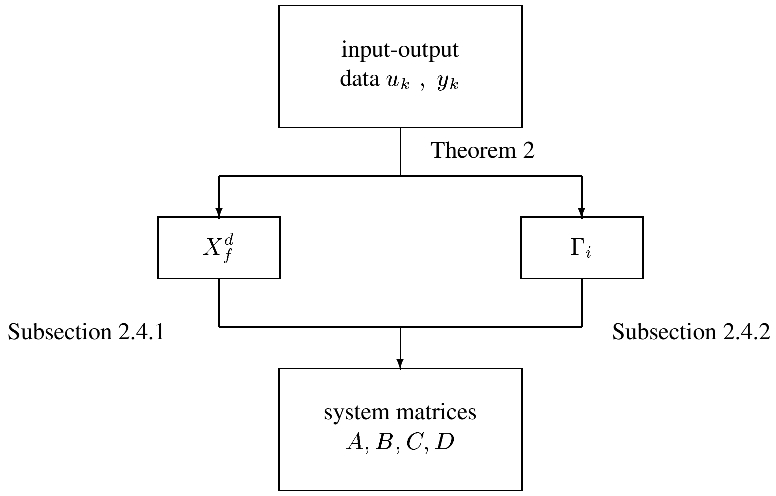

Figure 2.4 An overview of the deterministic subspace identification procedure. Through the main Theorem 2 the state sequence X_f^d and the extended observability matrix Γ_i are determined. The system matrices are then extracted using any of the two algorithms described in Sections 2.4.1 or 2.4.2.

Definition 5 Persistency of excitation

The input sequence $u_k \in \mathbb{R}^m$ is persistently exciting of order $2i$ if the input covariance matrix

$$R^{uu} \stackrel{def}{=} \Phi_{[U_{0|2i-1},U_{0|2i-1}]}$$

has full rank, which is $2.m.i$.

Theorem 2 Deterministic identification

Under the assumptions that:

1. *The input u_k is persistently exciting of order $2i$ (Definition 5).*
2. *The intersection of the row space of U_f (the future inputs) and the row space of X_p^d (the past states) is empty.*

3. *The user-defined weighting matrices $W_1 \in \mathbb{R}^{li \times li}$ and $W_2 \in \mathbb{R}^{j \times j}$ are such that W_1 is of full rank and W_2 obeys: rank $(W_p) =$ rank $(W_p.W_2)$, where W_p is the block Hankel matrix containing the past inputs and outputs.*

And with $\mathcal{O}_i$ defined as the oblique projection:

$$\mathcal{O}_i \stackrel{def}{=} Y_f /_{U_f} \boldsymbol{W}_p \ , \tag{2.8}$$

and the singular value decomposition:

$$\begin{aligned} W_1 \mathcal{O}_i W_2 &= \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} S_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} & (2.9) \\ &= U_1 S_1 V_1^T \ , & (2.10) \end{aligned}$$

we have:

1. *The matrix $\mathcal{O}_i$ is equal to the product of the extended observability matrix and the states:*
$$\mathcal{O}_i = \Gamma_i . X_f^d \ . \tag{2.11}$$

2. *The order of the system (2.1)-(2.2) is equal to the number of singular values in equation (2.9) different from zero.*

3. *The extended observability matrix Γ_i is equal to*[5]*:*
$$\Gamma_i = W_1^{-1} U_1 S_1^{1/2} . T \ . \tag{2.12}$$

4. *The part of the state sequence X_f^d that lies in the column space of W_2 can be recovered from:*
$$X_f^d W_2 = T^{-1} . S_1^{1/2} V_1^T \ . \tag{2.13}$$

5. *The state sequence X_f^d is equal to:*
$$X_f^d = \Gamma_i^{\dagger} . \mathcal{O}_i \ . \tag{2.14}$$

[5]With $T \in \mathbb{R}^{n \times n}$ an arbitrary non-singular similarity transformation.

The proof of the Theorem is fairly simple and leads to some insight in how subspace identification results are typically derived. That is the reason why it is presented in the main body of this book and not in an Appendix[6]. We will first present the proof, after which interpretations of the fine details of the Theorem will be stated as a series of remarks.

Proof

From formulas (2.5) and (2.7), we find that X_f^d can be written as a linear combination of the past inputs U_p and outputs Y_p as follows:

$$\begin{aligned} X_f^d &= A^i X_p^d + \Delta_i^d U_p \\ &= A^i \, . \, [\, \Gamma_i^\dagger Y_p - \Gamma_i^\dagger H_i^d U_p \,] + \Delta_i^d U_p \\ &= [\, \Delta_i^d - A^i \Gamma_i^\dagger H_i^d \,] \, U_p + [\, A^i \Gamma_i^\dagger \,] \, Y_p \\ &= L_p \, . \, W_p \, , \end{aligned} \tag{2.15}$$

with:

$$L_p = \left(\begin{array}{c|c} \Delta_i^d - A^i \Gamma_i^\dagger H_i^d & A^i \Gamma_i^\dagger \end{array} \right) \, .$$

With (2.15), formula (2.6) can be rewritten as:

$$Y_f = \Gamma_i . L_p . W_p + H_i^d . U_f \, .$$

From this formula and using (1.7), the first claim of the Theorem can be proven as follows:

$$\begin{aligned} Y_f &= \Gamma_i . L_p . W_p + H_i^d . U_f \, , \\ Y_f \Pi_{U_f^\perp} &= \Gamma_i . L_p . W_p . \Pi_{U_f^\perp} + H_i^d . \underbrace{U_f \Pi_{U_f^\perp}}_{=0} \, , \\ Y_f / \boldsymbol{U}_f^\perp &= \Gamma_i . L_p . W_p / \boldsymbol{U}_f^\perp \, , \\ \underbrace{\left[Y_f / \boldsymbol{U}_f^\perp \right] . [W_p / \boldsymbol{U}_f^\perp]^\dagger \, . W_p}_{=\mathcal{O}_i} &= \Gamma_i . \underbrace{L_p . W_p}_{=X_f^d} \, , \\ \mathcal{O}_i &= \Gamma_i . X_f^d \, , \end{aligned}$$

where we have used the fact that $[W_p/\boldsymbol{U}_f^\perp].[W_p/\boldsymbol{U}_f^\perp]^\dagger.W_p = W_p$. This is not trivial, since $W_p/\boldsymbol{U}_f^\perp$ is rank deficient for purely deterministic systems (see for instance [MDMVV 89]) which implies that $[W_p/\boldsymbol{U}_f^\perp].[W_p/\boldsymbol{U}_f^\perp]^\dagger$ is different from the identity.

[6] Future proofs will be presented in the Appendices.

However, by imposing the first two conditions of Theorem 2, it is possible to prove that (see Appendix A.1):

$$[W_p/\boldsymbol{U}_f^\perp].[W_p/\boldsymbol{U}_f^\perp]^\dagger.W_p = W_p \ . \tag{2.16}$$

The other claims of Theorem 2 are easy to prove: The second claim follows from the fact that the matrix $W_1\mathcal{O}_iW_2$ is equal to the product of two matrices $W_1\Gamma_i$ (n columns) and $X_f^dW_2$ (n rows). Since both matrices are of rank n (W_1 is of full rank and the product $X_f^d.W_2$ is of rank n due to assumption 3 of the Theorem), their product is also of rank n. Equation (2.10) can be split into two parts (where $T \in \mathbb{R}^{n\times n}$ is an arbitrary non-singular matrix representing a similarity transformation):

$$\begin{aligned} W_1\Gamma_i &= U_1S_1^{1/2}.T \ , \\ X_f^dW_2 &= T^{-1}.S_1^{1/2}V_1^T \ , \end{aligned}$$

which leads to claim 3 and 4 of the Theorem. Claim 5 easily follows from the first claim.

□

Remarks & comments

1. The important formula (2.15) (see also for instance [MDMVV 89]) shows that the states X_f^d are lying in the row space of the past input and outputs W_p . It is worth noting, even at this stage, that a similar observation will be made in the next two Chapters for the stochastic and combined identification problems, where the states will also be found as linear combinations of the "past". Theorem 2 can algebraically be summarized as follows:

$$\boxed{\begin{aligned} \text{rank } (Y_f/_{U_f}\, \boldsymbol{W}_p) &= n \\ \text{row space } (Y_f/_{U_f}\, \boldsymbol{W}_p) &= \text{row space } (X_f^d) \\ \text{column space } (Y_f/_{U_f}\, \boldsymbol{W}_p) &= \text{column space } (\Gamma_i) \end{aligned}}$$

   This summary is the essence of why these algorithms are called *subspace* algorithms: they retrieve system related matrices as subspaces of projected data matrices.

2. Note that the Theorem is very similar to the unifying Theorem presented in [VODM 94b]. It is however different, since in [VODM 94b] purely deterministic systems are excluded, and only combined deterministic-stochastic systems with a strictly non-zero stochastic component are discussed.

3. Note also that the reader who is familiar with the literature of deterministic system identification, may find the presentation of this Theorem a little awkward. The results for deterministic system identification can indeed be presented in many different ways. We choose this (new) presentation, since it generalizes nicely to the stochastic and the combined deterministic-stochastic identification of the next two Chapters. In Section 2.3.1 and 2.3.2 we will indicate how the algorithms of the literature tie in with Theorem 2.

4. The study of the effect of the weighting matrices W_1 and W_2 is postponed to Section 2.3 and Chapter 5. Suffices to say, that both W_1 and W_2 determine the state space basis in which the identified model will be identified.

5. Some comments on the similarity transformation T. It is introduced to make the recovered observability matrix Γ_i and state sequence X_f^d exactly (number-wise) equal to the original X_f^d (2.3) and Γ_i (2.4). The similarity transformation is a function of W_1 and W_2, so we should write $T(W_1, W_2)$ but this would overload the notation. However it is not necessary to recover number-wise the original matrices A, B, C, D from which the input-output data was generated, as long as the identified set of state space matrices is equivalent within a similarity transformation to the original set of state space matrices. This implies that we can just as well put the similarity transformation T equal to I_n. In doing so, when using different weighting matrices W_1 and W_2, "different[7]" observability matrices and state space sequences will be obtained from Theorem 2. However, each set of quantities will lead to a set of state space matrices that is equivalent (up to a similarity transformation) to the original set of system matrices.

6. A Final note concerns the case where the input u_k is zero everywhere, except at time $i-1$ where it is equal to 1: $u_{i-1} = 1$. The state x_k is also zero for $k = 0, \dots, i-1$. This implies that y_k is an impulse response shifted over i time steps. Consider for simplicity a single-input system. In that case, we find that the state sequence X_f^d is equal to:

$$X_f^d = \begin{pmatrix} B & AB & A^2B & \dots & A^{j-1}B \end{pmatrix} .$$

[7] In a sense that the numbers in the matrices are different, because the choice of basis in the state space is different.

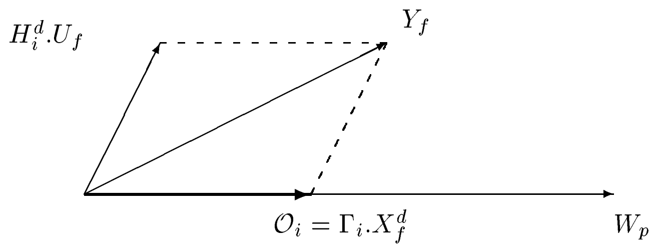

Figure 2.5 Graphical illustration of Theorem 2. The oblique projection decomposes the future outputs Y_f along the future inputs U_f and the past inputs and outputs W_p.

So, the oblique projection (2.8) then reduces to:

$$\mathcal{O}_i = \begin{pmatrix} C \\ CA \\ \ldots \\ CA^{i-1} \end{pmatrix} \begin{pmatrix} B & AB & A^2B & \ldots & A^{j-1}B \end{pmatrix} ,$$

which shows that in this case the deterministic identification problem becomes equivalent to the realization problem [Kun 78] [ZM 74], with i block rows and j columns.

2.2.3 Geometric interpretation

Figure 2.5 shows a graphical interpretation of Theorem 2. From the Figure we see that the oblique projection decomposes the future outputs (Y_f) into its two components: One part is due to the future inputs ($H_i^d.U_f$); the other part is due to the past inputs and outputs. This part is rank deficient due to the finite dimensional system that generated the data: Only an n-dimensional subspace of the row space spanned by W_p is needed to reconstruct the future.

2.3 RELATION TO OTHER ALGORITHMS

In this Section we investigate the similarities between deterministic subspace identification algorithms of the literature and the new deterministic identification Theorem

2. Two classes of deterministic algorithms are treated: intersection and projection algorithms.

2.3.1 Intersection algorithms

A first class of identification algorithms are the *intersection* algorithms of [DMo 88] [DMVMVVM 88] [DMMVV 88a] [DMMVV 88b] [DMMVV 88c] [MDMVV 89] [MDMV 91] [MDM 92] [Wil 86], where the row space of the state sequence X_f^d is computed as the intersection between the row space of the past inputs and outputs *and* the row space of the future inputs and outputs:

$$\text{row space } X_f^d = \text{row space } \begin{pmatrix} U_p \\ Y_p \end{pmatrix} \cap \text{ row space } \begin{pmatrix} U_f \\ Y_f \end{pmatrix} .$$

In [MDMVV 89] it is shown that this intersection is n dimensional, and indeed represents a valid state sequence. It can easily be shown that the state sequence X_f^d computed in Theorem 2 (2.14) lies in the same intersection. Indeed, from (2.8) and (2.14), we find that the state sequence is formed as a linear combination of the rows of W_p:

$$X_f^d = \Gamma_i^\dagger . Y_f /_{U_f} \boldsymbol{W}_p \; .$$

From (2.6), we find that the state sequence X_f^d can also be written as:

$$X_f^d = \Gamma_i^\dagger . Y_f - \Gamma_i^\dagger . H_i^d U_f \; ,$$

which proves that the state sequence X_f^d is also lying in the sum of the row space of the future inputs (U_f) and outputs (Y_f). The state sequence of the intersection algorithms thus corresponds to the state sequence computed in Theorem 2.

Different ways to compute the intersection have been proposed. A first way, presented in [MDMVV 89] [MDMV 91] [MDM 92], is by making use of a singular value decomposition of a concatenated Hankel matrix:

$$\begin{pmatrix} U_{0|2i-1} \\ Y_{0|2i-1} \end{pmatrix} .$$

A second way [DMo 88] [DMMVV 88b] is by taking as a basis for the intersection the principal directions between the row space of the past inputs and outputs *and* the row space of the future inputs and outputs. Indeed, from Figure 1.10 it can be seen that a non-empty intersection between two subspaces is characterized by a number of principal angles equal to zero, and that the principal directions corresponding to these zero angles form a basis for the row space of the intersection.

The Matlab function `intersec.m` contains a Matlab implementation of this algorithm. See also Section 6.1 and Appendix B.

2.3.2 Projection algorithms

In the literature, several *projection* algorithm have been described [CXK 94a] [DMV 87] [DMo 88] [Liu 92] [SROK 92] [VODMS 91] [VD 92]. An outline of these algorithms is very simple: When multiplying equation (2.6):

$$Y_f = \Gamma_i . X_f^d + H_i^d . U_f$$

from the right with $\Pi_{U_f^\perp}$, we find:

$$Y_f / \boldsymbol{U}_f^\perp = \Gamma_i . X_f^d / \boldsymbol{U}_f^\perp \ . \tag{2.17}$$

The main observation of the projection algorithms is that the system order and the extended observability matrix can be extracted from a singular value decomposition of (2.17): Its rank[8] is equal to n and its column space coincides with that of Γ_i. Note that to compute the left hand side of (2.17), only input-output data is required. This class of algorithms can be completely incorporated into the framework of Theorem 2. It suffices to take:

$$\begin{aligned} W_1 &= I_{li} \ , \\ W_2 &= \Pi_{U_f^\perp} \ , \end{aligned}$$

which leads, with formula (2.9) and (1.7) to the singular value decomposition of:

$$\begin{aligned} W_1 \mathcal{O}_i W_2 &= [Y_f / \boldsymbol{U}_f^\perp] . [W_p / \boldsymbol{U}_f^\perp]^\dagger . [W_p / \boldsymbol{U}_f^\perp] \\ &= [Y_f / \boldsymbol{U}_f^\perp] . [W_p / \boldsymbol{U}_f^\perp]^T [(W_p / \boldsymbol{U}_f^\perp) . (W_p / \boldsymbol{U}_f^\perp)^T]^\dagger . [W_p / \boldsymbol{U}_f^\perp] \\ &= [Y_f / \boldsymbol{U}_f^\perp] . \Pi_{[W_p / U_f^\perp]} \qquad (2.18) \\ &= [Y_f / \boldsymbol{U}_f^\perp] \ . \qquad (2.19) \end{aligned}$$

The last step is due to the fact:

$$Y_f / \boldsymbol{U}_f^\perp = \Gamma_i X_f^d / \boldsymbol{U}_f^\perp = \Gamma_i . L_p . W_p / \boldsymbol{U}_f^\perp \ ,$$

which indicates that the row space of $Y_f / \boldsymbol{U}_f^\perp$ is a subspace of the row space of $W_p / \boldsymbol{U}_f^\perp$. Projecting the row space of $Y_f / \boldsymbol{U}_f^\perp$ on the row space of $W_p / \boldsymbol{U}_f^\perp$ will thus have no

[8] Note that through assumptions 1 and 2 of Theorem 2 it is guaranteed that $X_f^d / U_f^\perp$ is of rank n.

effect, and results in $Y_f/\boldsymbol{U}_f^\perp$. We thus conclude that the singular value decompositions of (2.17) used in the projection algorithms and of (2.18) used in Theorem 2 are singular value decompositions of the same matrix, which illustrates that there is no difference between the classical formulation of the projection algorithms and the formulation of Theorem 2.

The Matlab function `project.m` contains a Matlab implementation of this algorithm. See also Section 6.1 and Appendix B.

2.3.3 Notes on noisy measurements

In this Subsection we will shortly treat the behavior of the different algorithms in the presence of noise. The main question is the one of asymptotic unbiasedness, i.e. when given an infinite amount of noisy data generated by an unknown linear system, does the algorithm compute the exact linear system ?

For the intersection algorithm of Subsection 2.3.1 it was proven in [MDMVV 89] that the exact linear system is retrieved when both the inputs and outputs are corrupted by additive spatially and temporary white noise sequences of equal covariance[9]. When this assumption is violated, it is possible to alter the algorithm by introducing weights based on the knowledge of the noise correlation. This is described in [MV 90]. However the a-priori knowledge of the noise correlation is a severe restriction on the application of this algorithm (since in most practical cases the noise correlation is *not* known).

For the projection algorithms of Subsection 2.3.2, it was proven in [VD 92] that the algorithms are asymptotically unbiased when the outputs are corrupted by additive spatially and temporary white noise. In [SROK 92] it was described how to reduce the finite sample variance by introducing extra weights.

Finally, for the algorithms based on the results of Theorem 2, it will be proven in Chapter 4, that the algorithms compute asymptotically unbiased estimates when the outputs are corrupted by additive (white or) colored noise.

So, even though the intersection algorithms, the projection algorithms and the algorithms based on Theorem 2 have been proven to be equivalent in the purely deterministic case, they behave differently in the presence of noise. This short discussion should provide an extra motivation for the algorithms presented in this book, since most of

[9] A sequence is spatially white when there is no correlation between the different channels. It is temporary white when there is no correlation between different time samples.

the practical identification problems can be brought back to the case where the outputs are corrupted by additive colored noise.

We illustrate the differences between the different algorithms with a small example. Consider the system:

$$\begin{aligned} x_{k+1} &= 0.85x_k + 0.5u_k \ , \\ y_k &= -0.3x_k + 0.1u_k \ . \end{aligned}$$

The input is the sum of a zero mean white noise sequence (variance 1) filtered with a second order Butterworth filter (cutoff 0.3 times the Nyquist frequency, sampling time $T = 1$) and a zero mean white noise sequence of variance 0.01. We performed 200 Monte Carlo experiments ($j = 1000, i = 4$) with the same input. Three different algorithms were considered to retrieve the matrix A from input-output data:

Intersection: The algorithm as described in Subsection 2.3.1 and [MDMVV 89].

Projection: The algorithm as described in Subsection 2.3.2 and [VD 92]. The matrix A is determined from Γ_i in the "least squares" way as described on page 53.

Theorem 2: The algorithm based on the results of Theorem 2 as described in Figure 2.8 (see further).

The data were distorted in three different ways:

White noise on outputs: A white noise sequence of variance 0.01 was added to the output.

White noise on inputs and outputs: Both inputs and outputs were corrupted by adding an independent white noise sequence of variance 0.01 to them.

Colored noise on outputs: The outputs were corrupted by a colored noise sequence that was generated by sending a white noise sequence e_k of variance 1 through the linear filter:

$$H(z) = \frac{0.02z - 0.041}{z - 0.85} \ .$$

The Matlab file `det_sim1.m` contains a Matlab implementation of this example.

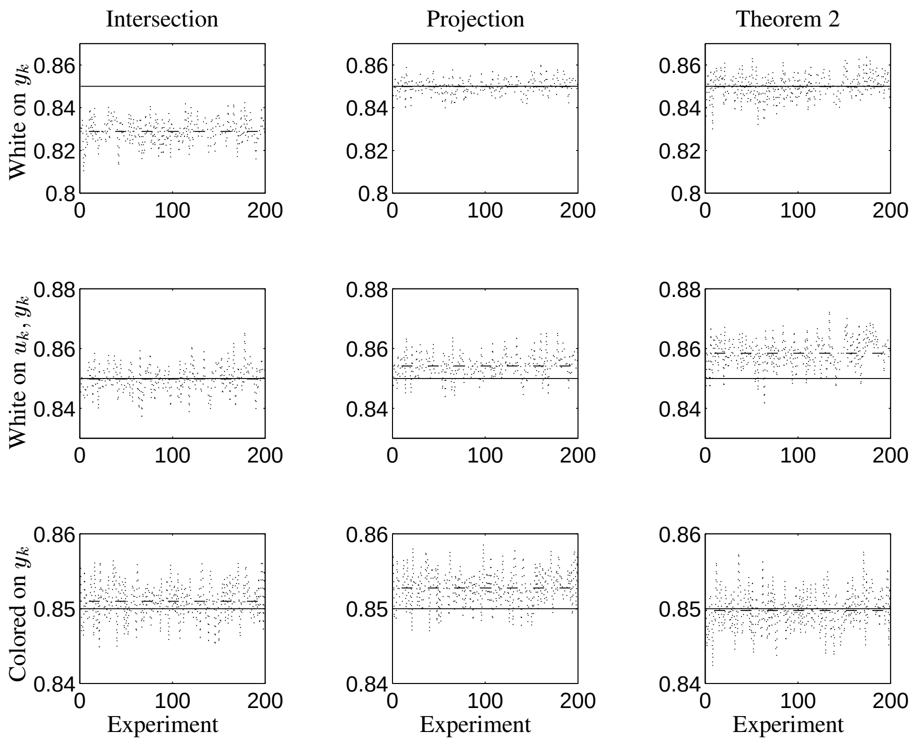

Figure 2.6 Each of the plots shows as a dotted line the 200 Monte Carlo estimates of the A matrix. The mean of these 200 experiments is plotted as a dashed dotted horizontal line, while the exact value of the A matrix (0.85) is plotted as a full horizontal line. The three columns represent the three algorithms that were used: intersection, projection and the algorithm based on Theorem 2. The three rows represent the different experimental conditions: In the first row white noise is added to the outputs, in the second row white noise is added to the inputs *and* outputs and in the third row colored noise is added to the outputs. From this Figure it can be clearly seen which algorithms compute asymptotically unbiased estimates for which experimental conditions (full line and dashed dotted line coincide). The intersection algorithms are unbiased for white noise distorted inputs *and* outputs, the projection algorithms are unbiased for white noise distorted outputs, while the algorithms based on Theorem 2 are unbiased for white *and* colored noise distorted outputs. This Figure was generated using the Matlab file det_sim1.m.

2.4 COMPUTING THE SYSTEM MATRICES

In this Section, we explain how the system matrices A, B, C and D can be computed from the results of Theorem 2 in two different ways. A schematic overview of both algorithms can be found in Figures 2.7 and 2.8. Note that the first steps of both algorithms are the same and coincide with the steps described in Theorem 2. For the implementation related issues we refer to Section 6.1.

2.4.1 Algorithm 1 using the states

From Theorem 2, we find:

- The order of the system from inspection of the singular values of equation (2.9).
- The extended observability matrix Γ_i from equation (2.12).
- The state sequence X_i^d $(= X_f^d)$ from equation (2.14).

Through a similar reasoning and proof as in Theorem 2, it is easy to show that the following holds:

$$\begin{aligned} \mathcal{O}_{i-1} &\stackrel{\text{def}}{=} Y_f^- /_{U_f^-} \boldsymbol{W}_p^+ \\ &= \Gamma_{i-1} . X_{i+1}^d \ . \end{aligned}$$

It is also easy to check that if we strip the last l (number of outputs) rows of Γ_i (calculated from 2.12), we find Γ_{i-1}:

$$\Gamma_{i-1} = \underline{\Gamma_i} \ ,$$

where $\underline{\Gamma_i}$ denotes the matrix Γ_i without the last l rows. Now X_{i+1}^d can be calculated as:

$$X_{i+1}^d = \Gamma_{i-1}^\dagger \mathcal{O}_{i-1} \ .$$

At this moment, we have calculated X_i^d and X_{i+1}^d, using only input-output data. The matrices A, B, C, D can be solved from:

$$\underbrace{\begin{pmatrix} X_{i+1}^d \\ Y_{i|i} \end{pmatrix}}_{\text{known}} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \underbrace{\begin{pmatrix} X_i^d \\ U_{i|i} \end{pmatrix}}_{\text{known}} \ , \tag{2.20}$$

where $U_{i|i}, Y_{i|i}$ are block Hankel matrices with only one block row of inputs respectively outputs. This set of equations can be easily solved in a linear least squares sense[10]. Note the symmetry in the way A, B, C and D appear in equation (2.20), i.e. all matrices are solved from the set of equations in one step. This is in contrast with the second algorithm (see below) where first A and C are determined, after which B and D are determined in a separate step. Figure 2.7 summarizes the steps of this first deterministic algorithm.

The Matlab function `det_stat.m` contains a Matlab implementation of this algorithm. See also Section 6.1 and Appendix B.

2.4.2 Algorithm 2 using the extended observability matrix

The system matrices are determined in two separate steps: As a first step, A and C are determined from Γ_i; In a second step B and D are computed. From Theorem 2, we find:

- The order of the system from inspection of the singular values of equation (2.9).
- The extended observability matrix Γ_i from equation (2.12).

Determination of A and C

The matrices A and C can now be determined from the extended observability matrix in different ways. All the methods, make use of the shift structure of the matrix Γ_i, which implies that (see [Kun 78]):

$$\underline{\Gamma_i}.A = \overline{\Gamma_i} \ ,$$

where $\overline{\Gamma_i}$ denotes Γ_i without the first l (number of outputs) rows. This equation can be solved in many different ways:

Least squares: [11]

$$A = \underline{\Gamma_i}^\dagger.\overline{\Gamma_i} \ .$$

[10]Note that when there is no noise, this set of equations is consistent and there is no need for a least squares solution.

[11]The names "least squares" and "total least squares" are not very meaningful in the case of purely deterministic systems, since there is no noise. However, in the next Chapters (where the data are corrupted by noise), we will refer to these methods again. At that point, the names *do* make sense, and that is why we already introduce them here.

Deterministic algorithm 1:

1. Calculate the oblique projections:

$$\mathcal{O}_i = Y_f /_{U_f} \boldsymbol{W}_p \ ,$$
$$\mathcal{O}_{i-1} = Y_f^- /_{U_f^-} \boldsymbol{W}_p^+ \ .$$

2. Calculate the **SVD** of the weighted oblique projection:

$$W_1 \mathcal{O}_i W_2 = U S V^T \ .$$

3. Determine the order by inspecting the singular values in S and partition the **SVD** accordingly to obtain U_1 and S_1.

4. Determine Γ_i and Γ_{i-1} as:

$$\Gamma_i = W_1^{-1} U_1 S_1^{1/2} \quad , \quad \Gamma_{i-1} = \underline{\Gamma_i} \ .$$

5. Determine X_i^d and X_{i+1}^d as:

$$X_i^d = \Gamma_i^\dagger \mathcal{O}_i \quad , \quad X_{i+1}^d = \Gamma_{i-1}^\dagger \mathcal{O}_{i-1} \ .$$

6. Solve the set of linear equations for A, B, C and D:

$$\begin{pmatrix} X_{i+1}^d \\ Y_{i|i} \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} X_i^d \\ U_{i|i} \end{pmatrix} \ .$$

Figure 2.7 A schematic overview of the first deterministic identification algorithm. See Section 6.1 for implementation issues. This algorithm has been implemented in the Matlab function det_stat.m.

Total least squares: By making use of the singular value decomposition of the concatenated matrix [DMo 88]

$$\begin{pmatrix} \overline{\Gamma_i} & -\underline{\Gamma_i} \end{pmatrix} = USV^T ,$$

and by partitioning the matrix $V \in \mathbb{R}^{2n \times 2n}$ as:

$$V = \begin{array}{c} \\ n \\ n \end{array} \!\!\begin{array}{c} \begin{array}{cc} n & n \end{array} \\ \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \end{array} ,$$

we find the total least squares solution as:

$$A = V_{22}.V_{12}^{-1} .$$

Stable A: In many applications, it is known in advance that the dynamical system matrix A should be stable. As was shown in [Mac 94], the following procedure will always compute a stable matrix A_{stable}:

$$A_{\text{stable}} = \Gamma_i^{\dagger}.\begin{pmatrix} \overline{\Gamma_i} \\ 0 \end{pmatrix} ,$$

where 0 represent l rows of zeros. This trick however introduces a bias in the solution for A. The dynamical system matrix will only be recovered exactly when A is indeed stable and when $i \to \infty$. In this case, the extra zeros added at the bottom do not introduce an error ($\lim_{i\to\infty} CA^i = 0$). In practice this trick should be used with care, especially when the system has lightly damped modes (or unstable modes). Indeed, the poles associated with these modes will be "pushed" closer to the origin in the $\mathcal{Z}$-plane to preserve stability. This can lead to significant errors in the identified models.

Optimally: In [OV 94] it is described how the poles of the A matrix can be determined trough fitting a matrix containing the coefficients of the characteristic polynomial of A to the null-space of Γ_i. With this method it is also possible to take the finite sample statistics of Γ_i into account, and thus obtain a statistical optimal estimate of A. We refer to [OV 94] for more details.

The matrix C can be determined from the first l rows of Γ_i.

Determination of B and D

After the determination of A and C, the system matrices B and D have to be computed. Here we will only sketch one possible way to do so, since for the purely deterministic case, independent of the method used, the resulting B and D are exactly the same[12]. In Chapter 4, we will indicate how to extract B and D in the case of a finite number of noisy data[13].

From the input-output equation (2.6), we find that:

$$\Gamma_i^{\perp}.Y_f = \Gamma_i^{\perp}.H_i^d.U_f \ , \tag{2.21}$$

where $\Gamma_i^{\perp} \in \mathbb{R}^{(li-n)\times li}$ is a full row rank matrix satisfying $\Gamma_i^{\perp}.\Gamma_i = 0$. Observe that with known matrices $A, C, \Gamma_i^{\perp}, U_f$ and Y_f this equation is linear in B and D. To enable an easy extraction of the matrices B and D, we multiply (2.21) with $U_f^{\dagger}$ from the right hand side. This leads to:

$$\underbrace{\Gamma_i^{\perp}.Y_f.U_f^{\dagger}}_{\in\mathbb{R}^{(li-n)\times mi}} = \underbrace{\Gamma_i^{\perp}}_{\in\mathbb{R}^{(li-n)\times li}} . \underbrace{H_i^d}_{\in\mathbb{R}^{li\times mi}} .$$

For simplicity of notation, we denote the left hand side of the equation with $\mathcal{M}$ and $\Gamma_i^{\perp}$ with $\mathcal{L}$. This equation can then be rewritten as:

$$\begin{aligned} \begin{pmatrix} \mathcal{M}_1 & \mathcal{M}_2 & \dots & \mathcal{M}_i \end{pmatrix} &= \begin{pmatrix} \mathcal{L}_1 & \mathcal{L}_2 & \dots & \mathcal{L}_i \end{pmatrix} \\ &\times \begin{pmatrix} D & 0 & 0 & \dots & 0 \\ CB & D & 0 & \dots & 0 \\ CAB & CB & D & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ CA^{i-2}B & CA^{i-3}B & CA^{i-4}B & \dots & D \end{pmatrix} , \end{aligned}$$

where $\mathcal{M}_k \in \mathbb{R}^{(li-n)\times m}$ and $\mathcal{L}_k \in \mathbb{R}^{(li-n)\times l}$. This can be rewritten as:

$$\underbrace{\begin{pmatrix} \mathcal{M}_1 \\ \mathcal{M}_2 \\ \vdots \\ \mathcal{M}_i \end{pmatrix}}_{\in\mathbb{R}^{i(li-n)\times m}} = \underbrace{\begin{pmatrix} \mathcal{L}_1 & \mathcal{L}_2 & \dots & \mathcal{L}_{i-1} & \mathcal{L}_i \\ \mathcal{L}_2 & \mathcal{L}_3 & \dots & \mathcal{L}_i & 0 \\ \mathcal{L}_3 & \mathcal{L}_4 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \mathcal{L}_i & 0 & \dots & 0 & 0 \end{pmatrix}}_{\in\mathbb{R}^{i(li-n)\times li}} \underbrace{\begin{pmatrix} I_l & 0 \\ 0 & \underline{\Gamma_i} \end{pmatrix}}_{\in\mathbb{R}^{li\times(l+n)}} \begin{pmatrix} D \\ B \end{pmatrix} ,$$

[12] Once A and C are fixed, the state space basis of the system is fixed and thus B and D are uniquely defined.

[13] In this case, different methods give different results, and it is thus important to choose the right method.

which is a set of linear equations in the unknowns B and D which is typically overdetermined (when $i(li - n) \geq (l + n)$). It could for instance be solved using least squares[14].

The Matlab function `det_alt.m` contains a Matlab implementation of this algorithm. See also Section 6.1 and Appendix B.

2.5 CONCLUSIONS

In this Chapter we have treated the subspace identification of purely deterministic systems. By using geometric operations on the input-output block Hankel matrices, a new general deterministic identification Theorem has been derived. This enables the computation of the state sequences and of the extended observability matrix through an oblique projection, directly from input-output data. Connections with published deterministic subspace algorithms have been indicated. Finally, two complete algorithms have been presented.

There are many more interesting properties of block-Hankel matrices and deterministic identification (see [DMo 88] [MDMVV 89] for instance), which we have not summarized since we only wanted to stress the results that are important in the context of the following Chapters.

[14] When there is no noise, this set of equations is consistent, and no least squares solution is needed.

Deterministic algorithm 2:

1. Calculate the oblique projection:

$$\mathcal{O}_i = Y_f /_{U_f} \boldsymbol{W}_p \ .$$

2. Calculate the **SVD** of the weighted oblique projection:

$$W_1 \mathcal{O}_i W_2 = U S V^T \ .$$

3. Determine the order by inspecting the singular values in S and partition the **SVD** accordingly to obtain U_1, U_2 and S_1.

4. Determine Γ_i and $\Gamma_i^\perp$ as:

$$\Gamma_i = W_1^{-1} U_1 S_1^{1/2} \quad , \quad \Gamma_i^\perp = U_2^T W_1 \ .$$

5. Determine A from Γ_i as $A = \underline{\Gamma_i}^\dagger \overline{\Gamma_i}$ or from any other method described on page 53. Determine C as the first l rows of Γ_i.

6. With:

$$\begin{aligned} \begin{pmatrix} \mathcal{M}_1 & \mathcal{M}_2 & \dots & \mathcal{M}_i \end{pmatrix} &= \Gamma_i^\perp . Y_f . U_f^\dagger \ , \\ \begin{pmatrix} \mathcal{L}_1 & \mathcal{L}_2 & \dots & \mathcal{L}_i \end{pmatrix} &= \Gamma_i^\perp \ , \end{aligned}$$

   solve B and D from:

$$\begin{pmatrix} \mathcal{M}_1 \\ \mathcal{M}_2 \\ \vdots \\ \mathcal{M}_i \end{pmatrix} = \begin{pmatrix} \mathcal{L}_1 & \mathcal{L}_2 & \dots & \mathcal{L}_{i-1} & \mathcal{L}_i \\ \mathcal{L}_2 & \mathcal{L}_3 & \dots & \mathcal{L}_i & 0 \\ \mathcal{L}_3 & \mathcal{L}_4 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \mathcal{L}_i & 0 & \dots & 0 & 0 \end{pmatrix} \times \begin{pmatrix} I_l & 0 \\ 0 & \underline{\Gamma_i} \end{pmatrix} \begin{pmatrix} D \\ B \end{pmatrix} .$$

Figure 2.8 A schematic overview of the second deterministic identification algorithm. See Section 6.1 for implementation issues. This algorithm has been implemented in the Matlab function det_alt.m

3

STOCHASTIC IDENTIFICATION

In this Chapter, we treat the subspace identification of purely stochastic systems with no external input ($u_k \equiv 0$). The stochastic identification problem thus consists of computing the stochastic system matrices from given output data only. We show how this can be done using geometric operations.

This Chapter is organized as follows. In Section 3.1 we mathematically state the stochastic (subspace) identification problem and introduce the major properties and notation. Forward and backward innovation models play an important role, as does the concept of positive real sequences. One key result is the non-iterative formula for the non-steady state Kalman filter state estimate. Section 3.2 contains the main Theorem for stochastic subspace identification. It allows for the extraction of the non-steady state Kalman filter states directly from the output data. Relations to other algorithms are discussed in Section 3.3. Finally, Section 3.4 shows how the system matrices can be computed in three different ways. Section 3.5 contains the conclusions. Appendix B describes the software implementation of the algorithms in this Chapter.

3.1 STOCHASTIC SYSTEMS

3.1.1 Problem description

Stochastic subspace identification algorithms compute state space models from given output data. Figure 3.1 states the stochastic (subspace) identification problem. The unknown stochastic system is represented in Figure 3.2.

Stochastic identification problem:

Given: s measurements of the output $y_k \in \mathbb{R}^l$ generated by the unknown stochastic system of order n:

$$x_{k+1}^s = Ax_k^s + w_k \ , \tag{3.1}$$
$$y_k = Cx_k^s + v_k \ , \tag{3.2}$$

with w_k and v_k zero mean, white vector sequences with covariance matrix:

$$\mathbf{E}[\begin{pmatrix} w_p \\ v_p \end{pmatrix} \begin{pmatrix} w_q^T & v_q^T \end{pmatrix}] = \begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} \delta_{pq} \ . \tag{3.3}$$

Determine:

- The order n of the unknown system
- The system matrices $A \in \mathbb{R}^{n\times n}, C \in \mathbb{R}^{l\times n}$ up to within a similarity transformation and $Q \in \mathbb{R}^{n\times n}, S \in \mathbb{R}^{n\times l}, R \in \mathbb{R}^{l\times l}$ so that the second order statistics of the output of the model *and* of the given output are equal.

Figure 3.1 The stochastic subspace identification problem.

The contributions of this book to the solution of the stochastic identification problem are the following (see also [VODM 91a] [VODMS 91] [VODM 91b] [VODM 93a]):

- Since the pioneering papers by Akaike [Aka 75], canonical correlations (which were first introduced by Jordan [Jor 1875] in linear algebra and then by Hotelling [Hot 36] in the statistical community) have been used as a mathematical tool in the stochastic realization problem. We have shown how the approach by Akaike [Aka 75] and others (e.g.[AK 90] [Lar 83] [Lar 90]) boils down to applying canonical correlation analysis to two matrices that are (implicitly assumed to be) double infinite (i.e. have an infinite number of rows and columns). A careful analysis reveals the nature of this double infinity and we manage to reduce the canonical correlation approach to a semi-infinite matrix problem, i.e. only the number of columns needs to be very large while the number of block rows remains sufficiently small. This observation is extremely relevant with respect

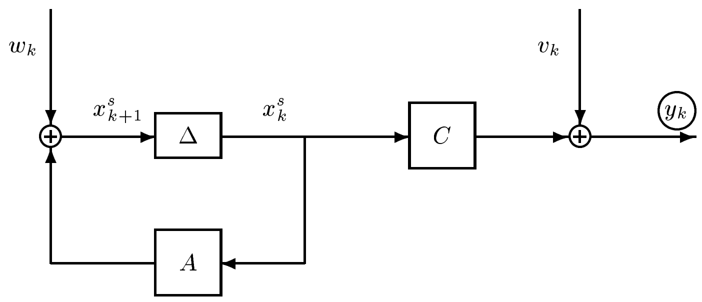

Figure 3.2 A linear time-invariant stochastic system with outputs y_k and states x^s_k, described by the matrices A, C and the covariance matrices Q, S, R. The symbol Δ represents a delay. In the stochastic identification problem, only the output is measured. The state is unknown, but will be determined as an intermediate result of the subspace identification algorithms.

to (for instance) the use of updating techniques. We have also shown how the canonical correlation algorithm is just a special case of a more general class of algorithms.

- In order to find the state space model, we derive a finite-dimensional vector sequence which, in the case of double infinite block Hankel matrices, would be a valid state sequence of the stochastic model. This sequence would correspond to the outputs of an infinite number of steady state Kalman filters with an infinite number of output measurements as inputs. For the semi-infinite matrix problem, the sequence corresponds to the output of an infinite number of *non-steady* state Kalman filters that have only used a finite number of output data as input. These state sequences are obtained directly from the output data, without any need for the state space model. The state space model is then derived from these sequences by solving a least squares problem. Figure 1.5 illustrates the difference between this approach and the classical one.

- A largely underestimated problem in stochastic subspace system identification is that of positive real sequences. Indeed, for an identified covariance sequence to be physically meaningful, it should be a positive real sequence. Almost all subspace algorithms presented in the literature [Aka 75] [Aok 87] [AK 90] do not guarantee this property, which implies that the spectral factor of the identified covariance sequence does not exist. We recognize this problem and present a variation to one of the algorithms so it computes a slightly asymptotically biased solution (the bias decreases when the number of block rows increases), but the positive realness of the solution is guaranteed (if the identified system matrix A is stable).

- We have derived a numerically robust square root algorithm (i.e. it does not square up the matrices), that mainly uses the QR-decomposition and the Quotient Singular Value Decomposition (**QSVD**) of the triangular factors and is completely data driven instead of covariance driven. The fact that only one of the dimensions of the matrices involved needs to go to infinity is very important since then updating techniques can be used. This algorithm is described in [VODM 93a].

3.1.2 Properties of stochastic systems

In this section, we summarize the main properties of linear time invariant stochastic processes, including the non-uniqueness of the state space description.

It is assumed that the stochastic process is stationary, i.e.:

$$\begin{aligned} \mathbf{E}[x_k^s] &= 0 \,, \\ \mathbf{E}[x_k^s (x_k^s)^T] &\stackrel{\text{def}}{=} \Sigma^s \,, \end{aligned} \tag{3.4}$$

where the state covariance matrix Σ^s is independent of the time k. This implies that A is a stable matrix (all of its poles are strictly inside the unit circle). There are many representations of stochastic state space models. All of the representations are equivalent, in the sense that the second order statistics of the output generated by the models is the same, i.e. the covariance sequence of the output is identical. We introduce the *forward model*, the *backward model*, the *forward innovation model* and the *backward innovation model*.

Forward model

First we will develop some (well-known) structural relations for linear time-invariant stochastic processes. Since w_k and v_k are zero mean white noise vector sequences, independent of x_k^s, we know that:

$$\begin{aligned} \mathbf{E}[x_k^s v_k^T] &= 0 \,, \\ \mathbf{E}[x_k^s w_k^T] &= 0 \,. \end{aligned}$$

Then we find the Lyapunov equation for the state covariance matrix Σ^s (3.4):

$$\Sigma^s = \mathbf{E}[x_{k+1}^s . (x_{k+1}^s)^T]$$

$$
\begin{aligned}
&= \mathbf{E}[(Ax_k^s + w_k).(Ax_k^s + w_k)^T] \\
&= A\mathbf{E}[x_k^s(x_k^s)^T]A^T + \mathbf{E}[w_k w_k^T] \\
&= A\Sigma^s A^T + Q \ . \qquad (3.5)
\end{aligned}
$$

Defining the output covariance matrices as:

$$\Lambda_i \stackrel{\text{def}}{=} \mathbf{E}[y_{k+i}.y_k^T] \ ,$$

we find for Λ_0:

$$
\begin{aligned}
\Lambda_0 &= \mathbf{E}[y_k.y_k^T] \\
&= \mathbf{E}[(Cx_k^s + v_k).(Cx_k^s + v_k)^T] \\
&= C\mathbf{E}[x_k^s(x_k^s)^T]C^T + \mathbf{E}[v_k v_k^T] \\
&= C\Sigma^s C^T + R \ . \qquad (3.6)
\end{aligned}
$$

Defining

$$
\begin{aligned}
G &\stackrel{\text{def}}{=} \mathbf{E}[x_{k+1}^s.y_k^T] \\
&= \mathbf{E}[(Ax_k^s + w_k).(Cx_k^s + v_k)^T] \\
&= A\mathbf{E}[x_k^s(x_k^s)^T]C^T + \mathbf{E}[w_k v_k^T] \\
&= A\Sigma^s C^T + S \ , \qquad (3.7)
\end{aligned}
$$

we get (for $i = 1, 2, \ldots$):

$$\Lambda_i = CA^{i-1}G \ , \qquad (3.8)$$

$$\Lambda_{-i} = G^T(A^{i-1})^T C^T \ . \qquad (3.9)$$

This last observation, indicates that the output covariances can be considered as Markov parameters of the deterministic linear time invariant system A, G, C, Λ_0. This is an important observation, that will play a major role in the derivation of stochastic subspace identification algorithms. The properties of the forward model are summarized in Figure 3.3.

Forward innovation model

The model (3.1)-(3.2) can be converted into a so-called *forward innovation model* (see e.g. [Pal 82]). The forward innovation model is obtained by applying a Kalman filter to the stochastic system (3.1)-(3.2):

$$
\begin{aligned}
x_{k+1}^f &= Ax_k^f + K^f e_k^f \ , \\
y_k &= Cx_k^f + e_k^f \ ,
\end{aligned}
$$

The forward stochastic model:

$$\begin{aligned} x_{k+1}^s &= Ax_k^s + w_k \ , \\ y_k &= Cx_k^s + v_k \ , \end{aligned}$$

$$\mathbf{E}[\begin{pmatrix} w_p \\ v_p \end{pmatrix} \begin{pmatrix} w_q^T & v_q^T \end{pmatrix}] = \begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} \delta_{pq} \ .$$

$$\begin{aligned} \mathbf{E}[x_k^s (x_k^s)^T] &= \Sigma^s &= A\Sigma^s A^T + Q \ , \\ \mathbf{E}[y_k y_k^T] &= \Lambda_0 &= C\Sigma^s C^T + R \ , \\ \mathbf{E}[x_{k+1}^s y_k^T] &= G &= A\Sigma^s C^T + S \ . \end{aligned}$$

Figure 3.3 The forward stochastic model.

$$\mathbf{E}[e_k^f (e_k^f)^T] = (\Lambda_0 - CPC^T) \ .$$

Here K^f is the (forward) Kalman gain:

$$K^f = (G - APC^T)(\Lambda_0 - CPC^T)^{-1} \ ,$$

and P is the forward state covariance matrix, which can be determined as the stabilizing solution of the forward Riccati equation:

$$P = APA^T + (G - APC^T)(\Lambda_0 - CPC^T)^{-1}(G - APC^T)^T \ . \tag{3.10}$$

In order to solve this Riccati equation, we have the following Theorem.

Theorem 3 Forward Riccati equation

The solution to the Riccati equation (3.10) can be found from the generalized eigenvalue problem

$$\begin{pmatrix} A^T - C^T\Lambda_0^{-1}G^T & 0 \\ -G\Lambda_0^{-1}G^T & I_n \end{pmatrix} \begin{pmatrix} W_1 \\ W_2 \end{pmatrix} = \begin{pmatrix} I_n & -C^T\Lambda_0^{-1}C \\ 0 & A - G\Lambda_0^{-1}C \end{pmatrix} \begin{pmatrix} W_1 \\ W_2 \end{pmatrix} \Lambda \ , \tag{3.11}$$

as $P = W_2 W_1^{-1}$. Λ contains the n stable (i.e. inside the unit circle) eigenvalues of the generalized eigenvalue pencil.

Laub [Lau 79] presented a robust algorithm for solving the Riccati equation based on the Schur decomposition. The properties of the forward innovation model are summarized in Figure 3.4.

The Matlab function `solvric.m` contains a Matlab implementation of this Riccati solver. See also Appendix B.

The forward innovation model:

$$\begin{aligned} x^f_{k+1} &= A x^f_k + K^f e^f_k \ , \\ y_k &= C x^f_k + e^f_k \ , \end{aligned}$$

$$\begin{aligned} \mathbf{E}[e^f_k (e^f_k)^T] &= (\Lambda_0 - CPC^T) \ , \\ \mathbf{E}[x^f_k (x^f_k)^T] &= P \ , \end{aligned}$$

$$P = APA^T + (G - APC^T)(\Lambda_0 - CPC^T)^{-1}(G - APC^T)^T \ ,$$

$$K^f = (G - APC^T)(\Lambda_0 - CPC^T)^{-1} \ .$$

Figure 3.4 The forward innovation stochastic model.

Backward model

Associated with every forward model (3.1)-(3.2), there is a backward[1] model with the same second order statistics as the forward model. This backward model can be obtained as follows. Define the estimate[2] of x^s_k based on x^s_{k+1} as:

$$\Pi(x^s_k | x^s_{k+1}) \stackrel{\text{def}}{=} \mathbf{E}[x^s_k (x^s_{k+1})^T](\mathbf{E}[x^s_{k+1}(x^s_{k+1})^T])^{-1} x^s_{k+1} \ .$$

We now have:

$$\begin{aligned} x^s_k &= \Pi(x^s_k|x^s_{k+1}) + (x^s_k - \Pi(x^s_k|x^s_{k+1})) \\ &= \mathbf{E}[x^s_k(x^s_{k+1})^T](\mathbf{E}[x^s_{k+1}(x^s_{k+1})^T])^{-1} x^s_{k+1} + (x^s_k - \Pi(x^s_k|x^s_{k+1})) \\ &= \mathbf{E}[x^s_k(x^s_{k+1})^T](\Sigma^s)^{-1} x^s_{k+1} + (x^s_k - \Pi(x^s_k|x^s_{k+1})) \\ &= \mathbf{E}[x^s_k((x^s_k)^T A^T + w^T_k)](\Sigma^s)^{-1} x^s_{k+1} + (x^s_k - \Pi(x^s_k|x^s_{k+1})) \\ &= \Sigma^s A^T (\Sigma^s)^{-1} x^s_{k+1} + (x^s_k - \Pi(x^s_k|x^s_{k+1})) \ . \end{aligned}$$

[1] Backward means that the iterative state space formulas of the model are running backward in time.

[2] For Gaussian signals this is the minimum variance estimate [Pap 84].

Now define the backward state $z^s_{k-1} \stackrel{\text{def}}{=} (\Sigma^s)^{-1} x^s_k$, then

$$z^s_{k-1} = A^T z^s_k + w^b_k \tag{3.12}$$

where:

$$w^b_k \stackrel{\text{def}}{=} (\Sigma^s)^{-1}(x^s_k - \Pi(x^s_k | x^s_{k+1})) \ .$$

Note that w^b_k is independent of x^s_{k+1} and thus also of z^s_k. Note also that the covariance of the state of the backward model is given as:

$$\begin{aligned} \mathbf{E}[z^s_{k-1}(z^s_{k-1})^T] &= (\Sigma^s)^{-1}\mathbf{E}[x^s_k(x^s_k)^T](\Sigma^s)^{-1} \\ &= (\Sigma^s)^{-1} \ . \end{aligned}$$

For the output equation, we obtain in a similar way:

$$\begin{aligned} y_k &= \Pi(y_k|x^s_{k+1}) + (y_k - \Pi(y_k|x^s_{k+1})) \\ &= \mathbf{E}[y_k(x^s_{k+1})^T](\mathbf{E}[x^s_{k+1}(x^s_{k+1})^T])^{-1}x^s_{k+1} + (y_k - \Pi(y_k|x^s_{k+1})) \\ &= \mathbf{E}[(Cx^s_k + v_k)((x^s_k)^T A^T + w^T_k)](\Sigma^s)^{-1}x^s_{k+1} + (y_k - \Pi(y_k|x^s_{k+1})) \\ &= (C\Sigma^s A^T + S^T)(\Sigma^s)^{-1}x^s_{k+1} + (y_k - \Pi(y_k|x^s_{k+1})) \\ &= G^T z^s_k + v^b_k \ , \end{aligned}$$

with $v^b_k \stackrel{\text{def}}{=} (y_k - \Pi(y_k|x^s_{k+1}))$. Once again v^b_k is independent of x^s_{k+1} and thus also of z^s_k. Figure 3.5 summarizes the remaining properties of the backward model which can be derived in a similar way as the corresponding properties of the forward model.

Backward innovation model

Associated with the general backward model (3.12)-(3.12) is the *backward innovation model*, which is obtained by applying a (backward) Kalman filter to the system (3.12)-(3.12):

$$\begin{aligned} z^b_{k-1} &= A^T z^b_k + K^b e^b_k \ , \\ y_k &= G^T z^b_k + e^b_k \ , \\ \mathbf{E}[e^b_k(e^b_k)^T] &= (\Lambda_0 - G^T N G) \ . \end{aligned}$$

Here K^b is the (backward) Kalman gain:

$$K^b = (C^T - A^T N G)(\Lambda_0 - G^T N G)^{-1} \ ,$$

and N is the backward state covariance matrix, which can be determined from the backward Riccati equation:

$$N = A^T N A + (C^T - A^T N G)(\Lambda_0 - G^T N G)^{-1}(C^T - A^T N G)^T \ . \tag{3.13}$$

In order to solve this Riccati equation, we have the following Theorem.

The backward stochastic model:

$$
\begin{aligned}
z^s_{k-1} &= A^T z^s_k + w^b_k \;, \\
y_k &= G^T z^s_k + v^b_k \;,
\end{aligned}
$$

$$
\mathbf{E}[\begin{pmatrix} w^b_p \\ v^b_p \end{pmatrix} \begin{pmatrix} (w^b_q)^T & (v^b_q)^T \end{pmatrix}] = \begin{pmatrix} Q^b & S^b \\ (S^b)^T & R^b \end{pmatrix} \delta_{pq} \;,
$$

$$
\begin{aligned}
\mathbf{E}[z^s_k (z^s_k)^T] &= (\Sigma^s)^{-1} &&= A^T(\Sigma^s)^{-1}A + Q^b \;, \\
\mathbf{E}[y_k y_k^T] &= \Lambda_0 &&= G^T(\Sigma^s)^{-1}G + R^b \;, \\
\mathbf{E}[z^s_{k-1} y_k^T] &= C^T &&= A^T(\Sigma^s)^{-1}G + S^b \;.
\end{aligned}
$$

Figure 3.5 The backward stochastic model.

Theorem 4 Backward Riccati equation

The solution of the Riccati equation (3.13) can be found from the generalized eigenvalue problem

$$
\begin{pmatrix} A - G\Lambda_0^{-1}C & 0 \\ -C^T\Lambda_0^{-1}C & I_n \end{pmatrix} \begin{pmatrix} W_1 \\ W_2 \end{pmatrix} = \begin{pmatrix} I_n & -G\Lambda_0^{-1}G^T \\ 0 & A^T - C^T\Lambda_0^{-1}G^T \end{pmatrix} \begin{pmatrix} W_1 \\ W_2 \end{pmatrix} \Lambda \;,
$$

as $N = W_2 W_1^{-1}$. Λ *contains the* n *stable (i.e. inside the unit circle) eigenvalues of the generalized eigenvalue pencil.*

Laub [Lau 79] presented a robust algorithm for solving the Riccati equation based on the Schur decomposition. The properties of the backward innovation model are summarized in Figure 3.6.

The Matlab function `solvric.m` contains a Matlab implementation of this Riccati solver. See also Appendix B.

It is well known that the stochastic model for the data y_k is not unique. Not only can we introduce an arbitrary similarity transformation for the state ($x^s_k \rightarrow T x^s_k$) as with all state space models. In addition, there is a whole set of possible state covariance

The backward innovation model:

$$\begin{aligned} z^b_{k-1} &= A^T z^b_k + K^b e^b_k \ , \\ y_k &= G^T z^b_k + e^b_k \ , \end{aligned}$$

$$\begin{aligned} \mathbf{E}[e^b_k (e^b_k)^T] &= (\Lambda_0 - G^T N G) \ , \\ \mathbf{E}[z^b_k (z^b_k)^T] &= N \ , \end{aligned}$$

$$N = A^T N A + (C^T - A^T N G)(\Lambda_0 - G^T N G)^{-1}(C^T - A^T N G)^T \ ,$$

$$K^b = (C^T - A^T N G)(\Lambda_0 - G^T N G)^{-1} \ .$$

Figure 3.6 The backward innovation stochastic model.

matrices and covariance matrices Q, R and S that give the same second order output covariance matrices Λ_i. This fact was described in detail by Faurre [Fau 76] and we provide here only a summary of some interesting results:

Theorem 5 Faurre's Theorem

The set of forward and backward stochastic models that generate the output covariance matrices Λ_i is characterized as follows.

1. *The matrices A, C and G are unique up to within a similarity transformation.*
2. *The set of all forward state covariance matrices Σ^s is a closed convex and bounded set: $P \leq \Sigma^s \leq N^{-1}$ where P and N are the solutions of the forward (3.10) respectively backward (3.13) Riccati equations*[3]*. Alternatively, the set of all corresponding backward state covariance matrices is given by the closed convex and bounded set: $N \leq (\Sigma^s)^{-1} \leq P^{-1}$*
3. *For every state covariance matrix Σ^s satisfying these bounds, the matrices R, S and Q associated to the forward stochastic model follow from the equations $Q = \Sigma^s - A\Sigma^s A^T$, $S = G - A\Sigma^s C^T$ and $R = \Lambda_0 - C\Sigma^s C^T$. The matrices R^b, S^b and Q^b associated to the backward stochastic model follow from $Q^b = (\Sigma^s)^{-1} - A^T(\Sigma^s)^{-1}A$, $S^b = C^T - A^T(\Sigma^s)^{-1}G$ and $R^b = \Lambda_0 - G^T(\Sigma^s)^{-1}G$.*

[3] Inequalities are to be interpreted in the sense of nonnegative definiteness.

Definition 6 Balanced stochastic realization

A stochastic model will be called balanced *if the solutions to the forward and backward Riccati equations are diagonal and equal.*

3.1.3 Notation

Most of the notation will be drawn from the previous Chapter (Section 2.1.2). In this Section, we only introduce the new notation used in this Chapter.

Block Hankel matrices

Since there are no external inputs for purely stochastic systems, we will not use any input block Hankel matrices. The output block Hankel matrices are defined as in the previous Chapter. Since stochastic systems require a certain amount of statistical analysis, we will in this Chapter assume that $j \to \infty$ for all theoretical derivations. For notational convenience, we define the shorthand notation[4] Y_p, Y_p^+ and Y_f, Y_f^-:

$$
Y_{0|2i-1} \stackrel{\text{def}}{=}
\begin{array}{c}
\\
\begin{array}{c} i \\ \\ \\ i \end{array}
\end{array}
\begin{array}{c}
\overleftrightarrow{\quad\quad j \quad\quad} \\
\left(\begin{array}{cccc}
y_0 & y_1 & \cdots & y_{j-1} \\
\cdots & \cdots & \cdots & \cdots \\
y_{i-2} & y_{i-1} & \cdots & y_{i+j-3} \\
y_{i-1} & y_i & \cdots & y_{i+j-2} \\
\hline
y_i & y_{i+1} & \cdots & y_{i+j-1} \\
y_{i+1} & y_{i+2} & \cdots & y_{i+j} \\
\cdots & \cdots & \cdots & \cdots \\
y_{2i-1} & y_{2i} & \cdots & y_{2i+j-2}
\end{array}\right)
\end{array}
\begin{array}{c}
\\
\begin{array}{c} \text{"past"} \\ \\ \\ \text{"future"} \end{array}
\end{array}
\stackrel{\text{def}}{=} \left(\begin{array}{c} Y_{0|i-1} \\ \hline Y_{i|2i-1} \end{array}\right) \stackrel{\text{def}}{=} \left(\begin{array}{c} Y_p \\ \hline Y_f \end{array}\right)
$$

[4]Note that, as before, the superscripts $+$ and $-$ refer to the size of the matrices. A matrix with superscript "$+$" has $l(i+1)$ block rows, while a matrix with superscript "$-$" has $l(i-1)$ block rows.

$$
\begin{array}{l} \text{i+1} \\ \stackrel{\text{def}}{=} \\ \\ \text{i-1} \end{array}
\begin{array}{c} \xleftrightarrow{\qquad \text{j} \qquad} \\
\left(\begin{array}{cccc}
y_0 & y_1 & \cdots & y_{j-1} \\
\cdots & \cdots & \cdots & \cdots \\
y_{i-2} & y_{i-1} & \cdots & y_{i+j-3} \\
y_{i-1} & y_i & \cdots & y_{i+j-2} \\
y_i & y_{i+1} & \cdots & y_{i+j-1} \\
\hline
y_{i+1} & y_{i+2} & \cdots & y_{i+j} \\
\cdots & \cdots & \cdots & \cdots \\
y_{2i-1} & y_{2i} & \cdots & y_{2i+j-2}
\end{array}\right) \end{array}
\begin{array}{l} \text{"past"} \\ \\ \text{"future"} \end{array}
\stackrel{\text{def}}{=} \left(\frac{Y_{0|i}}{Y_{i+1|2i-1}}\right) \stackrel{\text{def}}{=} \left(\frac{Y_p^+}{Y_f^-}\right)
$$

System related matrices

The extended ($i > n$) observability matrix Γ_i was already defined in Section 2.1.2. The pair $\{A, C\}$ is assumed to be observable. The reversed extended stochastic controllability matrix Δ_i^c (where the subscript i denotes the number of block columns and the superscript “c” stands for “covariance”) is defined as:

$$\Delta_i^c \stackrel{\text{def}}{=} \begin{pmatrix} A^{i-1}G & A^{i-2}G & \dots & AG & G \end{pmatrix} \quad \in \mathbb{R}^{n \times li} \ . \tag{3.14}$$

We assume the pair $\{A, Q^{1/2}\}$ to be controllable. This implies that all dynamical modes of the system are excited by the process noise. It is indeed impossible to identify un-excited modes. We also assume all modes of A to be stable. The block Toeplitz matrices C_i and L_i are constructed from the output covariance matrices as:

$$C_i \stackrel{\text{def}}{=} \begin{pmatrix} \Lambda_i & \Lambda_{i-1} & \dots & \Lambda_2 & \Lambda_1 \\ \Lambda_{i+1} & \Lambda_i & \dots & \Lambda_3 & \Lambda_2 \\ \Lambda_{i+2} & \Lambda_{i+1} & \dots & \Lambda_4 & \Lambda_3 \\ \dots & \dots & \dots & \dots & \dots \\ \Lambda_{2i-1} & \Lambda_{2i-2} & \dots & \Lambda_{i+1} & \Lambda_i \end{pmatrix} \in \mathbb{R}^{li \times li} \ , \tag{3.15}$$

$$L_i \stackrel{\text{def}}{=} \begin{pmatrix} \Lambda_0 & \Lambda_{-1} & \Lambda_{-2} & \dots & \Lambda_{1-i} \\ \Lambda_1 & \Lambda_0 & \Lambda_{-1} & \dots & \Lambda_{2-i} \\ \Lambda_2 & \Lambda_1 & \Lambda_0 & \dots & \Lambda_{3-i} \\ \dots & \dots & \dots & \dots & \dots \\ \Lambda_{i-1} & \Lambda_{i-2} & \Lambda_{i-3} & \dots & \Lambda_0 \end{pmatrix} \in \mathbb{R}^{li \times li} \ . \tag{3.16}$$

An important remark concerns the estimation of the output covariance matrices. Hereto we assume that they can be estimated as (see Subsection 1.4.4 for a definition of $\mathbf{E_j}$

and Subsection 1.4.5 for a definition of $\Phi_{[A,B]}$):

$$\begin{aligned}\Lambda_i &= \mathbf{E_j}[\sum_{k=0}^{j-1} y_{k+i} y_k^T] \\ &= \Phi_{[Y_{i|i}, Y_{0|0}]} \ .\end{aligned}$$

From this observation it easily follows that (see also [VODM 94a]):

$$C_i = \Phi_{[Y_f, Y_p]} \ , \tag{3.17}$$

$$L_i = \Phi_{[Y_p, Y_p]} \tag{3.18}$$

$$= \Phi_{[Y_f, Y_f]} \ . \tag{3.19}$$

3.1.4 Kalman filter states

In the derivation of the subspace identification algorithms for stochastic system identification, the Kalman filter plays a crucial role. In this Subsection, we introduce a closed form equation for the forward and backward non-steady state Kalman filter state estimate. We also introduce a bank of non-steady state Kalman filters generating a sequence of state estimates. In the main Theorem of this Chapter (Section 3.2.1), we then indicate how this state sequence can be recovered directly from the output data y_k.

In the following we indicate the state estimates by a hat i.e. $\hat{x}_k$ for the forward Kalman filter estimate and $\hat{z}_k$ for the backward state estimate.

Theorem 6 Forward non-steady state Kalman filter

Given:

- *The initial state estimate:* $\hat{x}_0 = 0$
- *The initial covariance of the state estimate* $P_0 = \mathbf{E}[\hat{x}_0 . \hat{x}_0^T] = 0$
- *The output measurements* $y_0, \ldots, y_{k-1}$

then the non-steady state Kalman filter state estimate $\hat{x}_k$ *defined by the following recursive formulas:*

$$\hat{x}_k = A\hat{x}_{k-1} + K_{k-1}(y_{k-1} - C\hat{x}_{k-1}) \ , \tag{3.20}$$

$$
\begin{aligned}
K_{k-1} &= (G - AP_{k-1}C^T)(\Lambda_0 - CP_{k-1}C^T)^{-1} \ , && (3.21)\\
P_k &= AP_{k-1}A^T + (G - AP_{k-1}C^T) \\
&\quad \times(\Lambda_0 - CP_{k-1}C^T)^{-1}(G - AP_{k-1}C^T)^T \ , && (3.22)
\end{aligned}
$$

can be explicitly written as:

$$\hat{x}_k = \Delta_k^c . L_k^{-1} . \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{k-1} \end{pmatrix} \ . \tag{3.23}$$

The explicit solution of the covariance matrix P_k is equal to:

$$P_k = \Delta_k^c . L_k^{-1} . (\Delta_k^c)^T \ . \tag{3.24}$$

The proof in Appendix A.2 is a proof by induction. It is a little different from the proof presented in [VODM 93a], to make it similar to the proofs presented in the next Chapter. Appendix A.3 contains some notes on the form of the Kalman filter equations (3.20)-(3.22). From this last Appendix, we find that the covariance matrix of the state error $\widetilde{P}_k$ is given by:

$$
\begin{aligned}
\widetilde{P}_k &= \mathbf{E}[\ (x_k^s - \hat{x}_k).(x_k^s - \hat{x}_k)^T\ ] \\
&= \Sigma^s - P_k \ ,
\end{aligned}
$$

from which we conclude that:

- The assumption $P_0 = 0$ is the same as the assumption $\widetilde{P}_0 = \Sigma^s$. This means that at time zero, the covariance of the state error is equal to the covariance of the state x_k^s itself.
- When we let time go to infinity ($k \to \infty$), we find that $P_k = P$, where P is the solution of the forward algebraic Riccati equation (3.10). This implies that the covariance matrix of the state error is equal to $\widetilde{P}_\infty = \Sigma^s - P$. As proven by Faure [Fau 76], this is indeed the smallest state error covariance matrix that can be obtained.
- This also implies that when the original model was in forward innovation form ($\Sigma^s = P$), then $\widetilde{P}_\infty = 0$. This means that the state will be estimated exactly when an infinite amount of output data is available.

The significance of Theorem 6 is that it indicates how the Kalman filter state estimate $\hat{x}_k$ can be written as a linear combination of the past output measurements $y_0, \ldots, y_{k-1}$. This observation allows for the definition of the forward Kalman filter state sequence (that will be recovered by the stochastic subspace identification algorithms) as:

$$\begin{aligned} \widehat{X}_i &= \begin{pmatrix} \hat{x}_i & \hat{x}_{i+1} & \ldots & \hat{x}_{i+j-1} \end{pmatrix} \\ &= \Delta_i^c . L_i^{-1} . Y_p \ . \end{aligned} \tag{3.25}$$

This state sequence is generated by a bank of non-steady state Kalman filters working in parallel on each of the columns of the block Hankel matrix of past outputs Y_p. Figure 3.7 illustrates this concept. The bank of Kalman filters runs in a *vertical* direction (over the columns). It should be noted that the Kalman filters only use *partial* output information. For instance, the $(q+1)^{\text{th}}$ column of $\widehat{X}_i$ can be written as:

$$\hat{x}_{i+q} = \Delta_i^c . L_i^{-1} . \begin{pmatrix} y_q \\ \vdots \\ y_{i+q-1} \end{pmatrix} \ ,$$

which indicates that the Kalman filter generating the estimate of $\hat{x}_{i+q}$ only uses i output measurements $y_q, \ldots, y_{i+q-1}$, instead of all the output measurements up until time $i+q-1$: $y_0, \ldots, y_{i+q-1}$ (as would be expected).

Finally note that for the backward stochastic model, a dual Theorem can be derived:

Theorem 7 Backward non-steady state Kalman filter

Given:

$$\begin{aligned} \hat{z}_0 &= 0 \ , \\ N_0 &= \mathbf{E}[\hat{z}_0 \hat{z}_0^T] = 0 \ , \end{aligned}$$

and the output measurements $y_0, \ldots, y_{-k+1}$, then the non-steady state Kalman filter state estimate $\hat{z}_{-k}$ defined by the following recursive formulas:

$$\hat{z}_{-k} = A^T \hat{z}_{-k+1} + K_{-k+1}(y_{-k+1} - C\hat{z}_{-k+1}) \ , \tag{3.26}$$

$$K_{-k+1} = (C^T - A^T N_{-k+1} G)(\Lambda_0 - G^T N_{-k+1} G)^{-1} \ , \tag{3.27}$$

$$\begin{aligned} N_{-k} &= A^T N_{-k+1} A + (C^T - A^T N_{-k+1} G) \\ &\quad \times (\Lambda_0 - G^T N_{-k+1} G)^{-1} (C^T - A^T N_{-k+1} G)^T \ . \end{aligned} \tag{3.28}$$

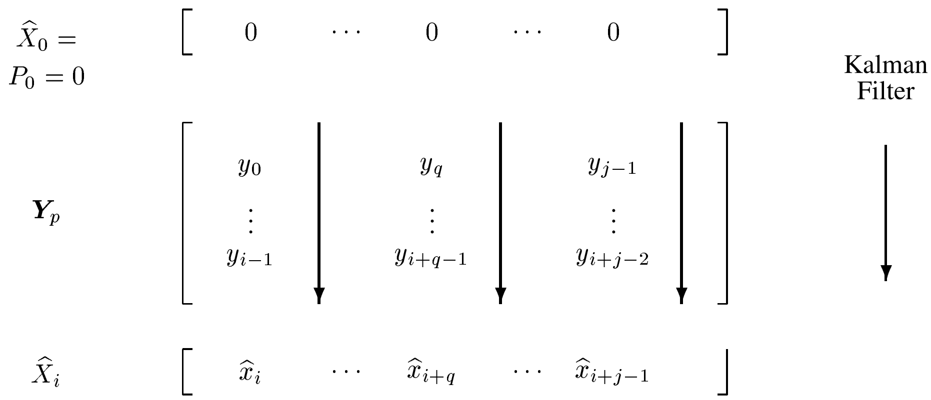

Figure 3.7 Interpretation of the sequence $\widehat{X}_i$ as a sequence of non-steady state Kalman filter state estimates based upon i measurements of y_k. When the system matrices A, C, Q, R, S would be known, the state $\widehat{x}_{i+q}$ could be determined from a non-steady state Kalman filter as follows: Start the filter at time q, with an initial state estimate 0. Now iterate the non-steady state Kalman filter over i time steps (the vertical arrow down). The Kalman filter will then return a state estimate $\widehat{x}_{i+q}$. This procedure could be repeated for each of the j columns, and thus we speak about a *bank* of non-steady state Kalman filters. The major observation in the main Theorem 8 of this Chapter will be that the system matrices A, C, Q, R, S do not have to be known to determine the state sequence $\widehat{X}_i$. It can be determined directly from output data through geometric manipulations (see Theorem 8).

can be written as:

$$\hat{z}_{-k} = \Gamma_k^T . L_k^{-1} . \begin{pmatrix} y_0 \\ y_{-1} \\ \vdots \\ y_{-k+1} \end{pmatrix} . \tag{3.29}$$

The explicit solution of the covariance matrix N_k is equal to:

$$N_k = \Gamma_k^T . L_k^{-1} . \Gamma_k \ . \tag{3.30}$$

Analogous to the definition of the forward Kalman filter state sequence $\widehat{X}_i$, we can define the backward Kalman filter state sequence $\widehat{Z}_i$ as:

$$\begin{aligned} \widehat{Z}_i &\stackrel{\text{def}}{=} \begin{pmatrix} \hat{z}_{i-1} & \hat{z}_i & \ldots & \hat{z}_{i+j-2} \end{pmatrix} \\ &= \Gamma_i^T . L_i^{-1} . Y_f \ . \end{aligned} \tag{3.31}$$

3.1.5 About positive real sequences

In this Section we introduce a last important property of stochastic systems. An arbitrary sequence Λ_i can not always be considered as a valid output covariance sequence. It has to satisfy the positive real conditions for which we introduce several equivalent tests.

The covariance sequence Λ_k should be a positive real sequence. We present the following equivalent statements to check the positive realness of a sequence. The statements are borrowed from [Fau 76].

Definition 7 Positive real sequences

The following statements are equivalent:

- *A sequence Λ_i generated by the matrices A, G, C, Λ_0 is a positive real sequence.*
- *The double infinite matrix L_∞ is positive definite:*

$$L_\infty = \begin{pmatrix} \Lambda_0 & \Lambda_{-1} & \Lambda_{-2} & \dots \\ \Lambda_1 & \Lambda_0 & \Lambda_{-1} & \dots \\ \Lambda_2 & \Lambda_1 & \Lambda_0 & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix} > 0 \ .$$

- *The $\mathcal{Z}$ transform of the sequence Λ_k (the power spectrum) is a positive definite (Hermitian) matrix for all $z = e^{j\omega}$ on the unit circle:*

$$\left[C(zI_n - A)^{-1}G + \Lambda_0 + G^T(z^{-1}I_n - A^T)^{-1}C^T\right]_{z=e^{j\omega}} > 0 \ . \tag{3.32}$$

- *The algebraic Riccati equations (3.10) and (3.13) have a positive definite solution.*
- *The covariance matrix of the process and measurement noise is positive definite:*

$$\begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} > 0 \ .$$

When the covariance sequence is a positive real sequence, it is possible to calculate a spectral factor by solving the Riccati equation (3.10) or (3.13). However, when the sequence is not positive real, the set of possible Markovian realizations is empty, and we are not able to compute a forward or backward innovation model, let alone any other model. In other words, it is highly necessary to identify a covariance sequence (determined by the matrices A, G, C, Λ_0) that is positive real. We will elaborate on this fact when presenting the algorithms (see Subsection 3.4.3).

3.2 GEOMETRIC PROPERTIES OF STOCHASTIC SYSTEMS

In this Section we present the main Theorem for the stochastic subspace identification problem. The geometric interpretation of the Theorem in the j-dimensional space is also presented.

3.2.1 Main Theorem

Just as for the deterministic identification (Section 2.2.2), we present a main Theorem for the stochastic identification problem. This Theorem allows for the computation of the row space of the state sequence $\widehat{X}_i$ and the column space of the extended observability matrix Γ_i directly from the output data, without any knowledge of the system matrices. The system matrices can then be extracted from $\widehat{X}_i$ or Γ_i. An overview of the general stochastic identification procedure is presented in Figure 3.8.

Note that just as in the deterministic main Theorem 2 we introduce two user-defined weighting matrices W_1 and W_2. The interpretation and use of these matrices will become clear in Section 3.3 and Chapter 5.

Finally note that the presentation of this general stochastic identification Theorem is different from the treatment in the literature. Actually, as will be shown in Section 3.3, this new Theorem encompasses all published algorithms.

Theorem 8 Stochastic identification

Under the assumptions that:

1. *The process noise w_k and the measurement noise v_k are not identically zero.*
2. *The number of measurements goes to infinity $j \to \infty$.*

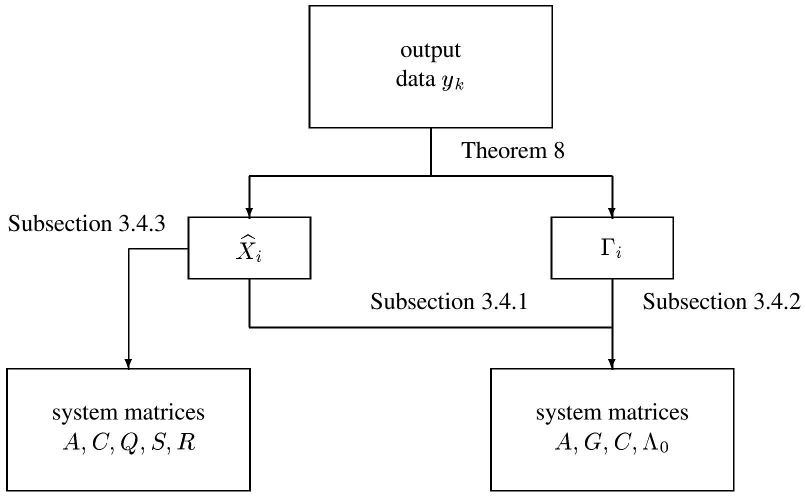

Figure 3.8 An overview of the stochastic subspace identification procedure. Through the main Theorem 8 the state sequence $\widehat{X}_i$ and the extended observability matrix Γ_i are determined. The system matrices are then extracted using any of the three algorithms described in Sections 3.4.1, 3.4.2 or 3.4.3.

3. *The user-defined weighting matrices $W_1 \in \mathbb{R}^{li \times li}$ and $W_2 \in \mathbb{R}^{j \times j}$ are such that W_1 is of full rank and W_2 obeys: rank $(Y_p) =$ rank $(Y_p.W_2)$, where Y_p the block Hankel matrix containing the past outputs.*

And with $\mathcal{O}_i$ defined as:

$$\mathcal{O}_i \stackrel{def}{=} Y_f/\boldsymbol{Y}_p \ , \tag{3.33}$$

and the singular value decomposition:

$$W_1 \mathcal{O}_i W_2 = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} S_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} = U_1 S_1 V_1^T \ , \tag{3.34}$$

we have:

1. *The matrix $\mathcal{O}_i$ is equal to the product of the extended observability matrix and the forward Kalman filter state sequence:*

$$\mathcal{O}_i = \Gamma_i . \widehat{X}_i \ . \tag{3.35}$$

2. *The order of the system (3.1)-(3.2) is equal to the number of singular values in equation (3.34) different from zero.*
3. *The extended observability matrix* Γ_i *and the associated extended controllability matrix* Δ_i^c *are equal to:*

$$\begin{aligned} \Gamma_i &= W_1^{-1} U_1 S_1^{1/2} . T \ , & (3.36) \\ \Delta_i^c &= \Gamma_i^\dagger . \Phi_{[Y_f, Y_p]} \ . & (3.37) \end{aligned}$$

4. *The part of the state sequence* $\widehat{X}_i$ *that lies in the column space of* W_2 *can be recovered from:*

$$\widehat{X}_i W_2 = T^{-1} . S_1^{1/2} V_1^T \ . \quad (3.38)$$

5. *The forward state sequence* $\widehat{X}_i$ *and the associated backward state sequence* $\widehat{Z}_i$ *are equal to:*

$$\begin{aligned} \widehat{X}_i &= \Gamma_i^\dagger . \mathcal{O}_i \ , & (3.39) \\ \widehat{Z}_i &= \Gamma_i^T . \Phi_{[Y_f, Y_f]}^{-1} Y_f \ . & (3.40) \end{aligned}$$

The proof of this Theorem can be found in Appendix A.4.

Remarks & comments

1. Theorem 8 can algebraically be summarized as follows:

$$\boxed{\begin{aligned} \text{rank } (Y_f / \boldsymbol{Y}_p) &= n \\ \text{row space } (Y_f / \boldsymbol{Y}_p) &= \text{row space } (\widehat{X}_i) \\ \text{column space } (Y_f / \boldsymbol{Y}_p) &= \text{column space } (\Gamma_i) \end{aligned}}$$

This summary is the essence of why these algorithms are called *subspace* algorithms: they retrieve system related matrices as subspaces of projected data matrices.

2. Note that a dual Theorem can be formulated when replacing $\mathcal{O}_i$ by $\mathcal{B}_i$, where $\mathcal{B}_i$ is the projection of the past outputs on the future outputs:

$$\mathcal{B}_i = Y_p / \boldsymbol{Y}_f = (\Delta_i^c)^T . \widehat{Z}_i \ . \quad (3.41)$$

One could wonder why $\mathcal{O}_i$ (3.33) *and* $\mathcal{B}_i$ (3.41) are not *both* included in Theorem 8. One could indeed calculate a singular value decomposition of $\mathcal{O}_i$ *and* $\mathcal{B}_i$. From the first **SVD**, Γ_i and $\widehat{X}_i$ could be determined as in Theorem 8. From the second **SVD**, the backward states $\widehat{Z}_i$ and the extended controllability matrix Δ_i^c could be determined through (3.41). The problem with this procedure is that the matrices $\Gamma_i, \widehat{X}_i$ and $\Delta_i, \widehat{Z}_i$ determined in this way will *not* be in the same state space basis. This leads to problems when determining the system matrices from these quantities, since the matrices determined from Γ_i and/or $\widehat{X}_i$ will *not* be in the same state space basis as the matrices determined from Δ_i^c and/or $\widehat{Z}_i$.

3. The study of the effect of the weighting matrices W_1 and W_2 is postponed to Section 3.3 and Chapter 5. Suffices to say, that both W_1 and W_2 determine the state space basis in which the identified model will be identified.

4. The same remarks on the similarity transformation T hold as for the deterministic Theorem (Remark 5 on page 43). This implies that we can put the similarity transformation T equal to I_n. In doing so, when using different weighting matrices W_1 and W_2, "different[5]" observability and controllability matrices and different state space sequences will be obtained from Theorem 8. However, each set of quantities will lead to a set of state space matrices that is equivalent (up to a similarity transformation) to the original set of matrices.

3.2.2 Geometrical interpretation

Formula (3.33) indicates that the row space of the forward states $\widehat{X}_i$ can be found by orthogonally projecting the row space of the future outputs Y_f, onto the row space of the past outputs Y_p. Alternatively, Formula (3.41) shows that the row space of the backward states $\widehat{Z}_i$ is equal to the orthogonal projection of the row space of the past outputs onto the row space of the future outputs. These concepts are illustrated in Figure 3.9.

3.3 RELATION TO OTHER ALGORITHMS

In this Section, we show how special cases of Theorem 8 (special choices of the weights W_1 and W_2) correspond to algorithms described in the literature: Principal component **PC**, *Unweighted principal component* **UPC** *and Canonical variate analysis*

[5] In a sense that the numbers in these matrices are different, because the choice of basis in the state space is different.

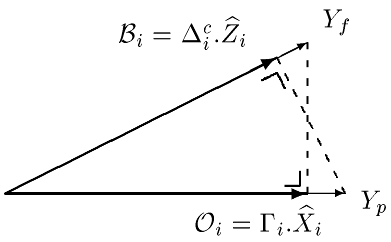

Figure 3.9 Graphical illustration of Theorem 8. The orthogonal projection of the future outputs Y_f on the past outputs Y_p determines the forward state sequence $\widehat{X}_i$. Alternatively, the orthogonal projection of the past outputs Y_p on the future outputs Y_f determines the backward state sequence $\widehat{Z}_i$.

	W_1	W_2
PC	I_{li}	$Y_p^T.\Phi^{-1/2}_{[Y_p,Y_p]}.Y_p$
UPC	I_{li}	I_j
CVA	$\Phi^{-1/2}_{[Y_f,Y_f]}$	I_j

Table 3.1 Overview of the special choices of W_1 and W_2 to obtain the published algorithms **PC**, **UPC** and **CVA**.

CVA. *The name, abbreviation and description of the algorithms presented in this Section correspond to the conventions in [AK 90].*

In this Section we show how through a specific choice of the weighting matrices W_1 and W_2, published algorithms are recovered. The results are summarized in Table 3.1.

3.3.1 The principal component algorithm (PC)

The principal component algorithm (see also [Aok 87] [AK 90]) determines the principal components of the cross-covariance Toeplitz matrix C_i, which can be estimated from the data as in (3.17):

$$C_i = \Phi_{[Y_f,Y_p]} \; .$$

From the fact that (see Appendix A.4):

$$C_i = \Gamma_i.\Delta_i^c \ ,$$

we find that a singular value decomposition of the estimated matrix C_i will reveal the order of the system, the column range of Γ_i and the row range of Δ_i^c. For more details, we refer to [Aok 87] [AK 90]. A nice side result is that the resulting realization of A, G, C will be balanced[6] in the deterministic sense when [Moo 81].

The principal component algorithm fits into the framework of Theorem 8 by simply taking the following weighting matrices:

$$\begin{aligned} W_1 &= I_{li} \ , \\ W_2 &= Y_p^T.\Phi_{[Y_p,Y_p]}^{-1/2}.Y_p \ . \end{aligned}$$

It is easy to show, that in this case the weighted projection (3.34) becomes equal to:

$$W_1\mathcal{O}_iW_2 = \Phi_{[Y_f,Y_p]}.\Phi_{[Y_p,Y_p]}^{-1/2}.Y_p \ .$$

The singular values and the left singular vectors of the weighted projection $W_1\mathcal{O}_iW_2$ are now exactly equal to the singular values and the left singular vectors of the matrix C_i, since:

$$\begin{aligned} \mathbf{E_j}\left[(W_1\mathcal{O}_iW_2).(W_1\mathcal{O}_iW_2)^T\right] &= \Phi_{[Y_f,Y_p]}.\Phi_{[Y_p,Y_p]}^{-1/2}.\Phi_{[Y_p,Y_p]}.\Phi_{[Y_p,Y_p]}^{-1/2}.\Phi_{[Y_p,Y_f]} \\ &= \Phi_{[Y_f,Y_p]}.\Phi_{[Y_p,Y_f]} \\ &= C_i.C_i^T \ . \end{aligned}$$

Thus we have shown how the principal component algorithms fits in the framework of Theorem 8.

3.3.2 The unweighted principal component algorithm (UPC)

In the unweighted principal component algorithm (**UPC**), the extended observability matrix is determined from the range of the left singular vectors of [AK 90]:

$$C_iL_i^{-1}C_i^T = U_1S_1^2U_1^T \ .$$

The extended observability matrix is then determined as:

$$\Gamma_i = U_1S_1^{1/2} \ .$$

[6] Note that the system is only exactly balanced when $i \to \infty$. For finite i the system is very near to being balanced.

Once again, this algorithm can be easily fitted in the framework of Theorem 8 by taking:

$$\begin{aligned} W_1 &= I_{li} , \\ W_2 &= I_j . \end{aligned}$$

Since in this case, the weighted projection $W_1 \mathcal{O}_i W_2 = \mathcal{O}_i$ has a covariance matrix $\mathcal{O}_i.\mathcal{O}_i^T$ equal to $C_i L_i^{-1} C_i^T$, and thus has the same left singular vectors as $C_i L_i^{-1} C_i^T$. We conclude that, once again, by the proper choice of the weights W_1 and W_2 the **UPC** algorithms can be fitted in the framework of Theorem 8.

For this case, it is shown in [AK 90] that the *forward innovation model* is balanced in the deterministic sense[7].

3.3.3 The canonical variate algorithm (CVA)

In this Section we show how a certain choice of weighting matrices W_1 and W_2 corresponds to the canonical variate algorithm of [Aka 74] [Aka 75] [AK 90] or the principal angles and direction algorithm of [VODM 93a].

The canonical variate algorithm of for instance [AK 90] [VODM 93a] computes the principal angles and directions between the row spaces of the past outputs Y_p and the future outputs Y_f. This can be brought back to Theorem 8 by choosing the following weights:

$$\begin{aligned} W_1 &= \Phi^{-1/2}_{[Y_f,Y_f]} , \\ W_2 &= I_j . \end{aligned}$$

We then find for the covariance matrix of the weighted projection:

$$\begin{aligned} (W_1\mathcal{O}_i)(W_1\mathcal{O}_i)^T &= \Phi^{-1/2}_{[Y_f,Y_f]}.\Phi_{[Y_f,Y_p]}\Phi^{-1}_{[Y_p,Y_p]}.\Phi_{[Y_p,Y_f]}.\Phi^{-1/2}_{[Y_f,Y_f]} \\ &= U_1 S_1^2 U_1^T . \end{aligned}$$

By comparing this expression with equation (1.10), we see that the diagonal of S_1 contains the cosines of the principal angles. For this choice of the weights, and with U_1, S_1, V_1 the singular value decomposition of the weighted projection of Theorem 8:

$$W_1\mathcal{O}_i = \Phi^{-1/2}_{[Y_f,Y_f]}\mathcal{O}_i$$

[7]Note that the system is only exactly balanced when $i \to \infty$. For finite i the system is very near to being balanced.

$$
\begin{aligned}
&= \Phi^{-1/2}_{[Y_f,Y_f]}.\Phi_{[Y_f,Y_p]}.\Phi^{-1}_{[Y_p,Y_p]}.Y_p \\
&= U_1 S_1 V_1^T \ ,
\end{aligned}
$$

and with (1.11)-(1.13) it can be easily shown that:

$$[\mathbf{Y}_p \sphericalangle Y_f] = V_1^T \ , \tag{3.42}$$

$$[Y_p \sphericalangle \mathbf{Y}_f] = U_1^T.\Phi^{-1/2}_{[Y_f,Y_f]} Y_f \ , \tag{3.43}$$

$$[Y_p \sphericalangle Y_f] = S_1 \ . \tag{3.44}$$

Furthermore, we find from Theorem 8 that:

$$
\begin{aligned}
\Gamma_i &= \Phi^{1/2}_{[Y_f,Y_f]}.U_1.S_1^{1/2} \ , \\
\Delta_i^c &= S_1^{1/2}.\Phi_{[V_1^T,Y_p]} \ ,
\end{aligned}
$$

and that the forward and backward state sequences can be expressed as:

$$
\begin{aligned}
\widehat{X}_i &= S_1^{1/2}.V_1^T \ , \\
\widehat{Z}_i &= S_1^{1/2}.U_1^T.\Phi^{-1/2}_{[Y_f,Y_f]} Y_f \ .
\end{aligned}
$$

By comparing these expressions with (3.42)-(3.44), we find that the state sequences $\widehat{X}_i$ and $\widehat{Z}_i$ are equal to $S_1^{1/2}$ times the principal directions in Y_p respectively Y_f. This is exactly the claim of for instance [VODM 93a].

It can be shown (see for instance [AK 90]) that the resulting model is stochastically balanced[8] for this choice of weights (see also Chapter 5). Finally, it should be noted that the idea of forward and backward states is not restricted to the principal angles and directions (as is sometimes mistakingly thought). Theorem 8 indicates indeed that for each choice of W_1 and W_2 a forward and a backward state sequence can be calculated.

3.3.4 A simulation example

In this Subsection we explain how the system order can be determined from the singular values of $W_1\mathcal{O}_iW_2$, and this for the three different choices of W_1 and W_2 presented in this Section (**PC**, **UPC** and **CVA**). We consider the system in forward innovation form:

$$
\begin{aligned}
x^f_{k+1} &= \begin{pmatrix} 0.6 & 0.6 & 0 \\ -0.6 & 0.6 & 0 \\ 0 & 0 & 0.4 \end{pmatrix} x^f_k + \begin{pmatrix} 0.1706 \\ -0.1507 \\ 0.2772 \end{pmatrix} e^f_k \ , \\
y_k &= \begin{pmatrix} 0.7831 & 0.5351 & 0.9701 \end{pmatrix} x^f_k + e^f_k \ ,
\end{aligned}
$$

[8] Note that the system is only exactly balanced when $i \to \infty$. For finite i the system is very near to being balanced.

and:

$$\mathbf{E}[e_k^f.(e_k^f)^T] = 6.3663 \; .$$

An output sequence y_k of 4000 points was generated using this model. The number of block rows i is taken equal to 10. For each j between 100 and 4000, the singular values of $W_1\mathcal{O}_iW_2$ were calculated (with W_1 and W_2 as described in this Section), and plotted. This is illustrated in Figure 3.10. On this Figure, the convergence in function of j can clearly be observed.

The Matlab file `sto_sim1.m` contains a Matlab implementation of this example.

3.4 COMPUTING THE SYSTEM MATRICES

In this Section we illustrate how the system matrices A, C and Q, S, R (or G, Λ_0) can be computed from Theorem 8 in three different ways.

A schematic overview of the three algorithms can be found in Figures 3.11 through 3.13. For the implementation of the algorithms we refer to Section 6.1 and [VODM 93a]. Note that the first steps of all three algorithms are the same and coincide with the steps described in Theorem 8.

3.4.1 Algorithm 1 using the states

From Theorem 8, we find:

- The order of the system from inspection of the singular values of equation (3.34).
- The extended observability matrix Γ_i from equation (3.36) and the extended controllability matrix Δ_i^c from (3.37).
- The state sequences $\widehat{X}_i$.

Through a similar reasoning and proof as in Theorem 8, it can be shown that the following holds:

$$\begin{aligned} \mathcal{O}_{i-1} &\stackrel{\text{def}}{=} Y_f^-/\boldsymbol{Y}_p^+ \\ &= \Gamma_{i-1}.\widehat{X}_{i+1} \; . \end{aligned}$$

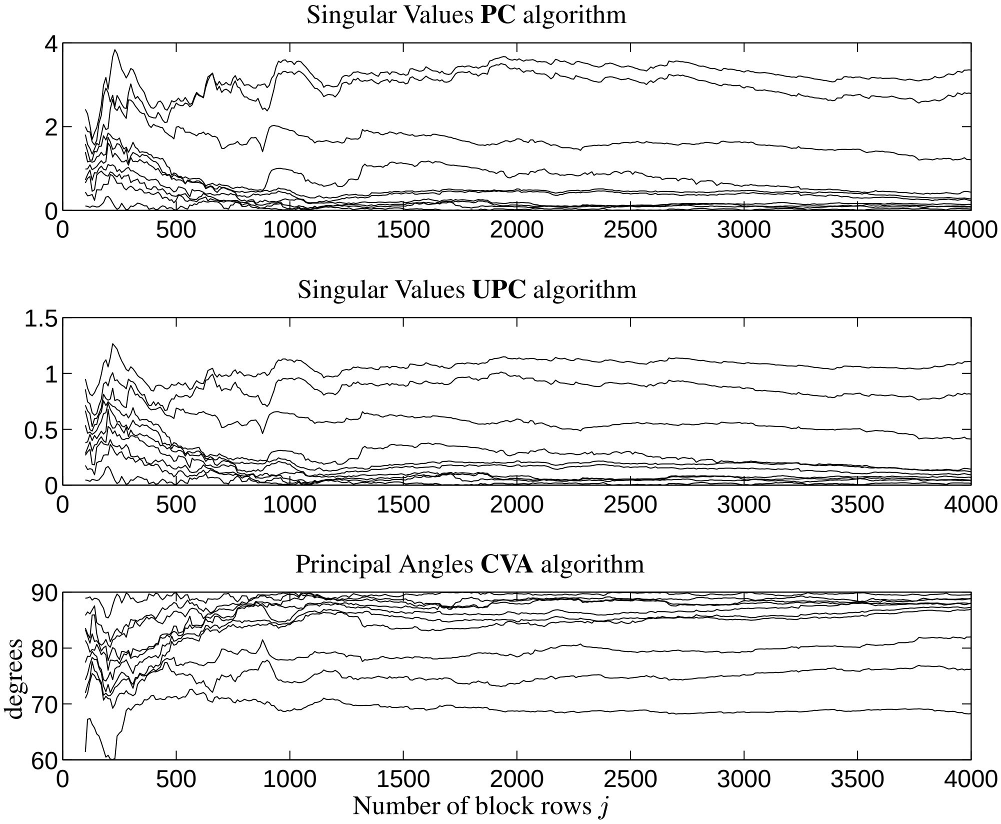

Figure 3.10 The order determining singular values (or principal angles) in function of the number of block columns j for the three algorithms presented in this Section: Principal component (**PC**), Unweighted principal component (**UPC**) and Canonical variate analysis (**CVA**). For each of the three plots, the system order (three) can be easily be determined. The convergence of the singular values in function of j can also be observed. For the first two plots all singular values go to zero, except three. For the last plot, all principal angles go to 90 degrees, except three. This Figure was generated using the Matlab file sto_sim1.m.

So, $\mathcal{O}_{i-1}$ can be easily calculated from the given output data. It is also easy to check that if we strip the last l (number of outputs) rows of Γ_i calculated from (3.36), we find Γ_{i-1}:

$$\Gamma_{i-1} = \underline{\Gamma_i} \ ,$$

where $\underline{\Gamma_i}$ denotes the matrix Γ_i without the last l rows. Now $\widehat{X}_{i+1}$ can be calculated from:

$$\widehat{X}_{i+1} = \Gamma_{i-1}^{\dagger}.\mathcal{O}_{i-1} \ .$$

At this moment, we have calculated $\widehat{X}_i$ and $\widehat{X}_{i+1}$, using only the output data. We can now form the following set of equations:

$$\underbrace{\begin{pmatrix} \widehat{X}_{i+1} \\ Y_{i|i} \end{pmatrix}}_{\text{known}} = \begin{pmatrix} A \\ C \end{pmatrix} \underbrace{\begin{pmatrix} \widehat{X}_i \end{pmatrix}}_{\text{known}} + \underbrace{\begin{pmatrix} \rho_w \\ \rho_v \end{pmatrix}}_{\text{residuals}} , \tag{3.45}$$

where $Y_{i|i}$ is a block Hankel matrix with only one row of outputs. This set of equations can be easily solved for A, C. Intuitively, since the Kalman filter residuals ρ_w, ρ_v (the innovations) are uncorrelated with $\widehat{X}_i$, it seems natural to solve this set of equations in a least squares sense (since the least squares residuals are orthogonal and thus uncorrelated with the regressors $\widehat{X}_i$). In [VODM 93a] it is shown that the least squares solution indeed computes an asymptotically unbiased estimate of A and C as:

$$\begin{pmatrix} A \\ C \end{pmatrix} = \begin{pmatrix} \widehat{X}_{i+1} \\ Y_{i|i} \end{pmatrix} .\widehat{X}_i^{\dagger} \ . \tag{3.46}$$

The matrix G can be determined as the last l columns of Δ_i^c. The matrix Λ_0 finally can be determined as a sample covariance matrix of the output. For instance:

$$\Lambda_0 = \Phi_{[Y_{i|i}, Y_{i|i}]} \ .$$

With the weights W_1 and W_2 as described for the **CVA** algorithm (see Section 3.3.3), this algorithm corresponds to the algorithm described in [VODM 93a]. More interpretations in terms of principal angles and directions can also be found in that paper. An interesting implementation of this algorithm using the quotient singular value decomposition (**QSVD**) is also described in [VODM 93a]. Figure 3.11 summarizes the steps of the algorithm.

Note that a dual algorithm could be derived using the backward projections $\mathcal{B}_i$ and states $\widehat{Z}_i$. In that case, first the states $\widehat{Z}_i$ and $\widehat{Z}_{i-1}$ are determined from the data. By solving a set of least squares equations, we find the matrices A^T and G^T. The matrix C can be recovered from the first row of Γ_i. Finally the matrix Λ_0 is determined as a sample covariance as before.

Finally, note also that the covariance sequence generated by the identified matrices A, G, C, Λ_0 is not guaranteed to be positive real, as will be illustrated in Subsection 3.4.4. The third algorithm (Subsection 3.4.3) will take care of this problem.

The Matlab function `sto_stat.m` contains a Matlab implementation of this algorithm. See also Section 6.1 and Appendix B.

3.4.2 Algorithm 2 using the extended matrices

Similar to the second deterministic identification algorithm of Chapter 2, the stochastic matrices A, G, C can also be determined from the extended observability and controllability matrices. From Theorem 8, we find:

- The order of the system from inspection of the singular values of equation (3.34).
- The extended observability matrix Γ_i from equation (3.36) and the extended controllability matrix Δ_i^c from (3.37).

The matrix C can be taken equal to the first l rows of Γ_i. The dynamical feedback matrix A is determined from Γ_i in the same way as it was determined for the deterministic case (see page 53). The matrix G can be determined as the last l columns of Δ_i^c. Finally, the matrix Λ_0 is determined as a sample covariance matrix:

$$\Lambda_0 = \Phi_{[Y_{i|i}, Y_{i|i}]} \ .$$

These steps are summarized in Figure 3.12. Note that, once again, this algorithm is not guaranteed to compute a set A, G, C, Λ_0 leading to a positive real covariance sequence. The next Subsection deals with this problem.

The Matlab function `sto_alt.m` contains a Matlab implementation of this algorithm. See also Section 6.1 and Appendix B.

3.4.3 Algorithm 3 leading to a positive real sequence

An important observation is that the identified covariance sequence determined by A, G, C, Λ_0 should be a positive real sequence. If this is not true, a spectral factor can not be computed, and the set of forward (or backward) realizations of the covariance sequence is empty. It turns out that if one starts from raw data, even when it was generated by simulation of a linear stochastic system, the positive real condition of the

Stochastic algorithm 1:

1. Calculate the projections:

$$\mathcal{O}_i \stackrel{\text{def}}{=} Y_f/\boldsymbol{Y}_p \ ,$$
$$\mathcal{O}_{i-1} \stackrel{\text{def}}{=} Y_f^-/\boldsymbol{Y}_p^+ \ .$$

2. Calculate the **SVD** of the weighted projection:

$$W_1\mathcal{O}_iW_2 = USV^T \ .$$

3. Determine the order by inspecting the singular values in S and partition the **SVD** accordingly to obtain U_1 and S_1.

4. Determine Γ_i and Γ_{i-1} as:

$$\Gamma_i = W_1^{-1}U_1S_1^{1/2} \quad , \quad \Gamma_{i-1} = \underline{\Gamma_i} \ .$$

5. Determine $\widehat{X}_i$ and $\widehat{X}_{i+1}$ as:

$$\widehat{X}_i = \Gamma_i^\dagger \mathcal{O}_i \quad , \quad \widehat{X}_{i+1} = \Gamma_{i-1}^\dagger \mathcal{O}_{i-1} \ .$$

6. Solve for A and C as:

$$\begin{pmatrix} A \\ C \end{pmatrix} = \begin{pmatrix} \widehat{X}_{i+1} \\ Y_{i|i} \end{pmatrix} .\widehat{X}_i^\dagger \ .$$

7. Determine G as the last l columns of Δ_i^c:

$$\Delta_i^c = \Gamma_i^\dagger.\Phi_{[Y_f,Y_p]} \ .$$

8. Compute Λ_0 as:

$$\Lambda_0 = \Phi_{[Y_{i|i},Y_{i|i}]} \ .$$

Figure 3.11 A schematic overview of the first stochastic identification algorithm. See Section 6.1 for implementation issues. This algorithm has been implemented in the Matlab function sto_stat.m

Stochastic algorithm 2:

1. Calculate the projection:

$$\mathcal{O}_i \stackrel{\text{def}}{=} Y_f/\boldsymbol{Y}_p \ .$$

2. Calculate the **SVD** of the weighted projection:

$$W_1 \mathcal{O}_i W_2 = USV^T \ .$$

3. Determine the order by inspecting the singular values in S and partition the **SVD** accordingly to obtain U_1, U_2 and S_1.

4. Determine Γ_i as:

$$\Gamma_i = W_1^{-1} U_1 S_1^{1/2} \ .$$

5. Determine A from Γ_i as $A = \underline{\Gamma_i}^{\dagger}\overline{\Gamma_i}$ or from any other method described on page 53. Determine C as the first l rows of Γ_i.

6. Determine G as the last l columns of Δ_i^c:

$$\Delta_i^c = \Gamma_i^{\dagger}.\Phi_{[Y_f,Y_p]} \ .$$

7. Compute Λ_0 as:

$$\Lambda_0 = \Phi_{[Y_{i|i},Y_{i|i}]} \ .$$

Figure 3.12 A schematic overview of the second stochastic identification algorithm. See Section 6.1 for implementation issues. This algorithm has been implemented in the Matlab function sto_alt.m.

identified covariance sequence is hardly ever satisfied. This is due to either the finite amount of data available or to the fact that real data is often *not* generated by a linear stochastic time invariant system.

This practical problem is rarely addressed in the literature. Aoki ([Aok 87, page 124]) mentions it once for a single-input single-output system of order one, and Vaccaro addresses the problem for a single-input single-output system in [Vac 87] and for general multiple-input multiple-output systems in [VV 93], where he scales a Riccati equation until it has a positive definite solution. Other interesting results on the covariance extension problem in the context of stochastic subspace identification can be found in [LP 93] [LP 94].

In this Subsection, we introduce an alternative way to ensure positive realness of the estimated covariance sequence. Unfortunately, this computation does not lead to an asymptotically unbiased estimate (unless the number of block rows in the Hankel matrices goes to infinity: $i \to \infty$). In practice however, the (small) bias which is a function of the convergence of the non-steady state Kalman filter, is a prize worth paying for the guaranteed positive realness of the resulting covariance sequence.

We alter the first stochastic identification algorithm so that it will always identify a positive real covariance sequence. The calculation of A and C proceeds as before through the least squares solution of equation (3.45). However, the calculation of G and Λ_0 is altered as follows. We can recover the covariance of the process and measurement noise from the residuals ρ_w and ρ_v of equation (3.45) as:

$$\mathbf{E_j}[\begin{pmatrix} \rho_w \\ \rho_v \end{pmatrix} . \begin{pmatrix} \rho_w^T & \rho_v^T \end{pmatrix}] = \begin{pmatrix} Q_i & S_i \\ S_i^T & R_i \end{pmatrix} .$$

The subscript i here indicates that the estimated covariances are not the *steady state* covariances as introduces in (3.3), but are the non-steady state covariance matrices of the *non-steady state* Kalman filter equation:

$$\begin{aligned} P_{i+1} &= AP_iA^T + Q_i \ , \\ G &= AP_iC^T + S_i \ , \\ \Lambda_0 &= CP_iC^T + R_i \ . \end{aligned}$$

When $i \to \infty$, which is upon convergence of the Kalman filters, we have that $Q_i \to Q, S_i \to S, R_i \to R$. As a simple approximation we can however put $Q = Q_i, S = S_i$ and $R = R_i$. As stated, this introduces a bias when $i \neq \infty$. However, in return for this bias, we get a guaranteed positive real covariance sequence. This is easily seen from the fact that (by construction):

$$\begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} = \mathbf{E_j}[\begin{pmatrix} \rho_w \\ \rho_v \end{pmatrix} . \begin{pmatrix} \rho_w^T & \rho_v^T \end{pmatrix}] \ > \ 0 \ ,$$

which coincides with the fifth condition for positive realness in Section 3.1.5. The quintuplet of matrices A, C, Q, S, R found in this way thus *always* leads to a positive real covariance sequence. The matrices G and Λ_0 can be extracted from the quintuplet by first solving the Lyapunov equation for Σ^s:

$$\Sigma^s = A\Sigma^s A^T + Q \ ,$$

after which G and Λ_0 can be computed from:

$$\begin{aligned} G &= A\Sigma^s C^T + S \ , \\ \Lambda_0 &= C\Sigma^s C^T + R \ . \end{aligned}$$

Finally, the model can be converted to a forward innovation form by solving the Riccati equation (3.10) for the matrix P. The Kalman gain can then be easily computed. Figure 3.13 summarizes the algorithm.

The Matlab function `sto_pos.m` contains a Matlab implementation of this algorithm. See also Section 6.1 and Appendix B.

This algorithm can be seen as an algorithm that estimates A and C asymptotically unbiased, but introduces a small perturbation on G and Λ_0. An alternative algorithm that only modifies Λ_0 until the resulting covariance sequence is positive real can be found in [VODM 91a]. The modification of Λ_0 is based on a level set algorithm similar to the algorithm used to compute the H_∞ norm presented in [BB 90].

We conclude this Section with the following interesting observation (see also the simulation example in Subsection 3.4.4): Algorithm 1 and 2 identify the stochastic system asymptotically unbiased, but they can not guarantee positive realness of the identified covariance sequence. Algorithm 3 will always identify a positive real sequence, but it is not asymptotically unbiased (for a finite number of block rows i). An interesting question is if it is possible to find an algorithm that is asymptotically unbiased and always identifies positive real sequences. This is still an open problem.

3.4.4 A simulation example

In this Subsection we illustrate the properties of the three algorithms of Figure 3.11, 3.12 and 3.13 with a simple example. We will especially investigate the asymptotic unbiasedness in function of the number of block rows i (to illustrate the asymptotic bias of algorithm 3), and the positive realness of the computed covariance sequence.

We consider the stochastic first order system in forward innovation form:

$$x^f_{k+1} = 0.949 x^f_k + 0.732 e^f_k \ ,$$

Stochastic algorithm 3:

1. Repeat steps 1 through 5 of the first stochastic identification algorithm.

2. Solve the set of linear equations for A and C:
$$\begin{pmatrix} \widehat{X}_{i+1} \\ Y_{i|i} \end{pmatrix} = \begin{pmatrix} A \\ C \end{pmatrix} \begin{pmatrix} \widehat{X}_i \end{pmatrix} + \begin{pmatrix} \rho_w \\ \rho_v \end{pmatrix} .$$

3. Determine Q, S and R from:
$$\begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} = \mathbf{E_j}[\begin{pmatrix} \rho_w \\ \rho_v \end{pmatrix} . \begin{pmatrix} \rho_w^T & \rho_v^T \end{pmatrix}] .$$

4. Determine Σ^s, G and Λ_0 from:
$$\begin{aligned} \Sigma^s &= A\Sigma^s A^T + Q && \text{solve for } \Sigma^s \text{ (Lyapunov)} , \\ G &= A\Sigma^s C^T + S && \text{solve for } G , \\ \Lambda_0 &= C\Sigma^s C^T + R && \text{solve for } \Lambda_0 . \end{aligned}$$

5. Determine P and K^f of the forward innovation model by solving the Riccati equation:
$$\begin{aligned} P &= APA^T + (G - APC^T)(\Lambda_0 - CPC^T)^{-1}(G - APC^T)^T , \\ K^f &= (G - APC^T)(\Lambda_0 - CPC^T)^{-1} . \end{aligned}$$

6. The identified forward innovation model is given by:
$$\begin{aligned} x_{k+1}^f &= Ax_k^f + K^f e_k^f , \\ y_k &= Cx_k^f + e_k^f , \\ \mathbf{E}[e_k^f (e_k^f)^T] &= R . \end{aligned}$$

Figure 3.13 A schematic overview of the third stochastic identification algorithm. This algorithm always computes a positive real covariance sequence. See Section 6.1 for implementation issues. This algorithm has been implemented in the Matlab function sto_pos.m.

$$y_k = 0.1933 x_k^f + e_k^f \ ,$$

with:

$$\mathbf{E}[e_k^f.(e_k^f)^T] = 0.8066 \ .$$

For each i between 2 and 10, we generated 100 output sequences y_k, which were then used to identify 100 stochastic models using the algorithms described in this Section ($j = 1000$). The average eigenvalue (A) and zero of the covariance sequence ($A - G\Lambda_0^{-1}C$) are plotted in Figure 3.14. The bias on the zero $A - G\Lambda_0^{-1}C$ for the third algorithm is clearly visible. It can also be seen from the Figure that for algorithm 1 and 2 the identified covariance sequence was not always positive real for small i. The decision whether a system A, G, C, Λ_0 is positive real or not is based on the distance of the eigenvalues of the Hamiltonian (3.11) to the unit circle. When this distance is smaller than 10^{-10} for one of the eigenvalues, the system is labeled "not positive real". Another way to check this is by inspecting the positivity of expression (3.32). This returned exactly the same results. Note that when checking this last equation, the deviation from positivity was often very large as is illustrated in Figure 3.15. From Figure 3.14, one could deduce that algorithm 1 and 2 estimate positive covariance sequences for larger i. This is however misleading, as is illustrated in Figure 3.16 (same experiment, with the system: $A = 0.85, C = -0.5998, K^f = 0.2450, \mathbf{E}[e_k^f.(e_k^f)^T] = 2.7942$).

The Matlab files `sto_sim2.m` and `sto_sim3.m` contain a Matlab implementation of these examples.

3.5 CONCLUSIONS

In this Chapter we treated the subspace identification of purely stochastic systems. A survey of properties of stochastic systems was introduced, among which the positive real condition for covariance sequences. The concept of a bank of non-steady state Kalman filters lead to the new and general main stochastic identification Theorem. This indicates how the Kalman filter state sequence can be extracted directly from the output data. Existing stochastic identification techniques proved to be a special case of this Theorem. Application of the Theorem led to three algorithms of which the first two were asymptotically unbiased, while the third algorithm guaranteed positive realness of the identified covariance sequence.

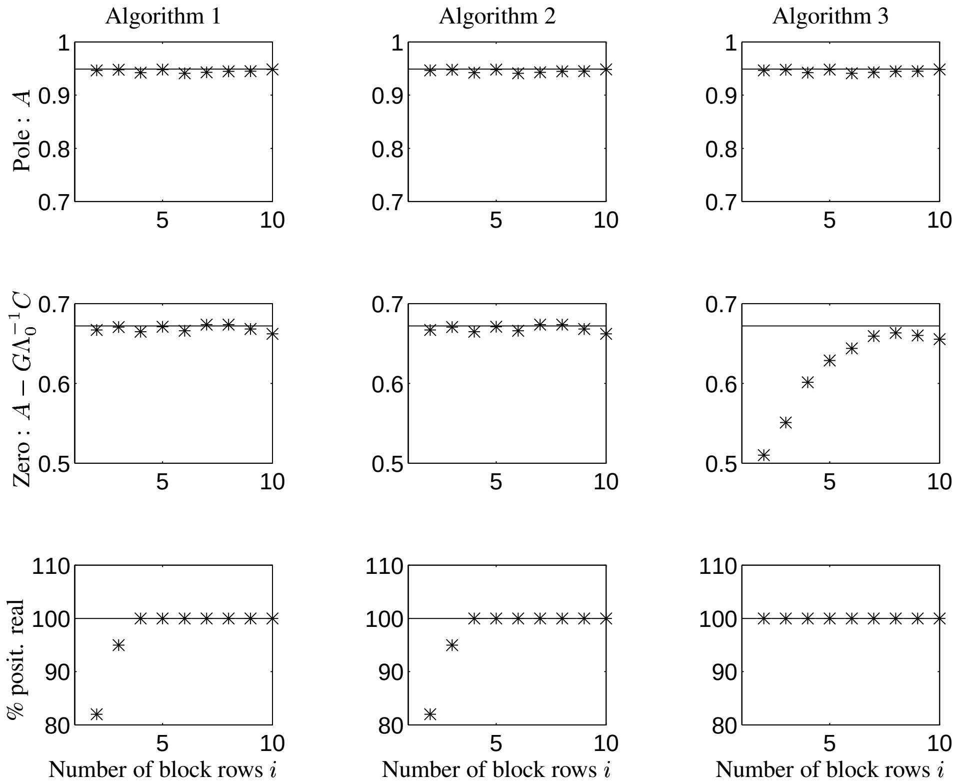

Figure 3.14 Illustration of the properties of the three algorithms presented in this Section. The expected value of the eigenvalue (row 1) and zero of $A - G\Lambda_0^{-1}C$ (row 2) are plotted. The third row contains the percentage of identified positive real covariance sequences during the 100 Monte Carlo experiments. The first row illustrates that the three algorithms estimate the dynamical system matrix A asymptotically unbiased. From the second row, we see that the third algorithm returns asymptotically biased estimates of the zero of the covariance sequence, which is due to the bias on G and Λ_0. However, the third row illustrates the advantage of the third algorithm. The covariance sequence is always positive real. One could mistakenly deduce from the first and second plot of the third row that the first and second algorithm compute positive real sequences for large i, however Figure 3.16 illustrates that this is not always the case. This Figure was generated using the Matlab file sto_sim2.m.

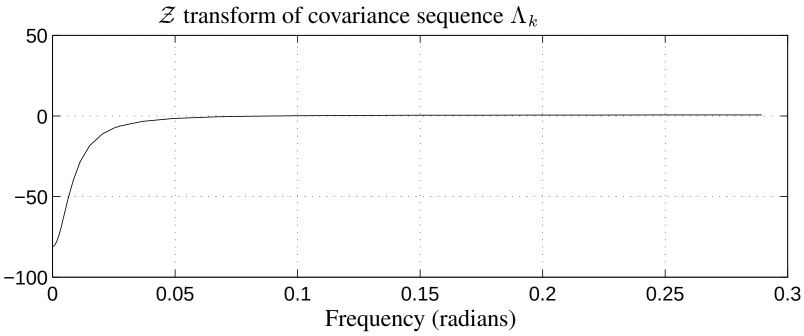

Figure 3.15 $\mathcal{Z}$-transform of one of the identified "non-positive" covariance sequences (see (3.32)). When the covariance sequence would be positive real, its $\mathcal{Z}$-transform would have been positive. This Figure indicates the severe violation of the positivity constraint for some of the identified sequences.

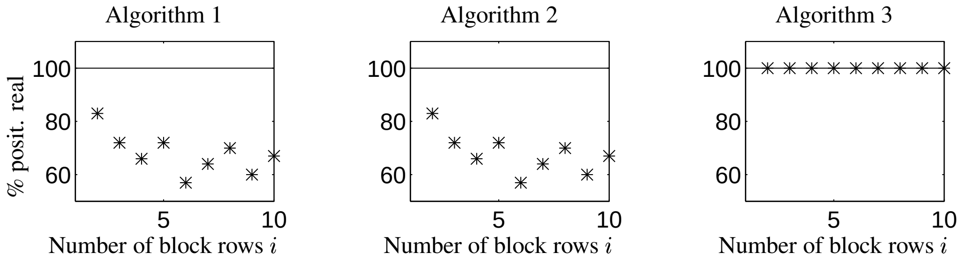

Figure 3.16 Percentage identified positive real sequences for 100 Monte Carlo experiments, for a different system as in Figure 3.14. This Figure illustrates that for algorithm 1 and 2, the identified covariance sequence is *not* always positive real, even for larger i. This Figure was generated using the Matlab file sto_sim3.m.

4

COMBINED DETERMINISTIC-STOCHASTIC IDENTIFICATION

In this Chapter we describe the subspace identification of combined deterministic-stochastic systems. For these systems, both the external input u_k *and the process and measurement noise* w_k *and* v_k *are different from zero. We use the results and ideas of the two previous Chapters to derive the main Theorem, which indicates how the combined deterministic-stochastic Kalman filter states can be extracted from the input-output data.*

In this Chapter we also derive an "industrially robust" combined deterministic-stochastic subspace identification algorithm.

At the end of the Chapter, we indicate how the two previous Chapters can be considered as special (limiting) cases of the combined algorithms presented in this Chapter. We found this to be an important observation, since it implies that most of the subspace algorithms published in the literature (deterministic, stochastic and combined), can be unified and are in essence the same.

This Chapter is organized as follows. In Section 4.1 we mathematically state the combined deterministic-stochastic (subspace) identification problem and introduce the notation. We also show how the combined non-steady state Kalman filter states can be computed from a non-iterative formula. Section 4.2 contains a projection Theorem and the main Theorem. The combination of both Theorems allows for the computation of the non-steady state Kalman filter sequence directly from the input-output data, without knowledge of the system matrices. Section 4.3 discusses the relation between the main Theorem and some published algorithms. Three algorithms are described in Section 4.4. Finally Section 4.5 indicates the similarities between the three main Theorems of Chapters 2, 3 and 4. The conclusions can be found in Section 4.6. Appendix B describes the software implementation of the algorithms in this Chapter.

Combined identification problem:

Given: s measurements of the input $u_k \in \mathbb{R}^m$ and the output $y_k \in \mathbb{R}^l$ generated by the unknown combined system of order n:

$$x_{k+1} = Ax_k + Bu_k + w_k \ , \tag{4.1}$$
$$y_k = Cx_k + Du_k + v_k \ , \tag{4.2}$$

with w_k and v_k zero mean, white vector sequences with covariance matrix:

$$\mathbf{E}[\begin{pmatrix} w_p \\ v_p \end{pmatrix} \begin{pmatrix} w_q^T & v_q^T \end{pmatrix}] = \begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} \delta_{pq} \ . \tag{4.3}$$

Determine:

- The order n of the unknown system
- The system matrices $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}, C \in \mathbb{R}^{l\times n}, D \in \mathbb{R}^{l\times m}$ up to within a similarity transformation and the matrices $Q \in \mathbb{R}^{n\times n}, S \in \mathbb{R}^{n\times l}, R \in \mathbb{R}^{l\times l}$ so that the second order statistics of the output of the stochastic subsystem *and* of the stochastic part of the given output are equal.

Figure 4.1 The combined subspace identification problem.

4.1 COMBINED SYSTEMS

4.1.1 Problem description

Combined subspace identification algorithms compute state space models from given input-output data. Figure 4.1 states the combined (subspace) identification problem. The unknown combined system is represented in Figure 4.2.

The contributions of this book to the solution of the combined deterministic-stochastic identification problem are the following (see also [VODMS 91] [VODM 92] [VODM 93b] [VODM 94a] [VODM 94b] [VODM 94c] [VODM 95a]):

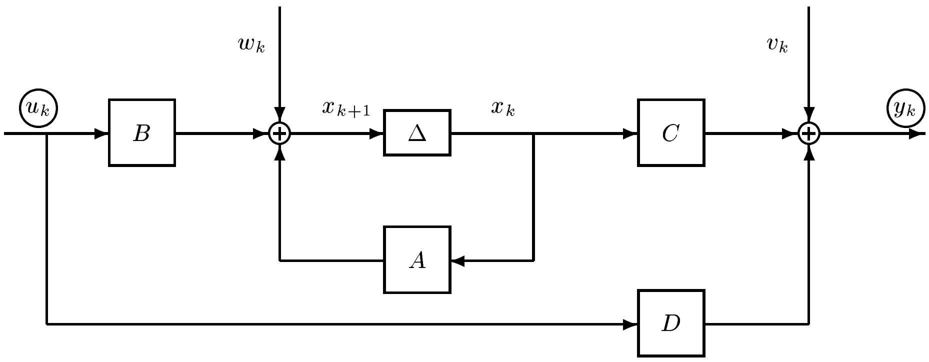

Figure 4.2 A linear time-invariant combined deterministic-stochastic system with inputs u_k, outputs y_k and states x_k, described by the matrices A, B, C, D and the covariance matrices Q, S, R. The symbol Δ represents a delay. In the combined identification problem, the input and output signals are measured. The state is unknown, but will be determined as an intermediate result of the subspace identification algorithms.

- We have derived two new algorithms that solve the combined deterministic-stochastic identification problem. These algorithms can be very nicely interpreted in the framework of the non-steady state Kalman filter sequences. It is proven that, just as for the purely stochastic case, these Kalman filter sequences can be determined directly from the input-output data, without knowing the system.

- We have studied the asymptotic behavior of the algorithms and proved through the connections with the Kalman filter theory, that the first algorithm computes asymptotically unbiased estimates of the system matrices, while the solutions of the second simpler algorithm are proven to be slightly biased. We also showed that this bias is a function of the convergence rate of the non-steady state Kalman filter to a steady state Kalman filter.

- We have shown that through the right choice of user defined weighting matrices, algorithms previously described in the literature (**N4SID** [VODM 94a], **MOESP** [Ver 94], **CVA** [Lar 90]) can be recovered easily from the general Theorems presented in this Chapter. Often the proofs and/or interpretations of these algorithms were incomplete in the literature. However, through the link between our results and the results in the literature, this problem has been solved. The possibility of choosing certain weights also leads to a whole new class of combined subspace identification algorithms.

- From our experience with practical industrial data sets, we have derived a "user-robustified" combined identification algorithm. The word "robust" here indicates that the algorithm computes good models for almost all practical cases.
- Finally, we have shown how the algorithms can be efficiently implemented using the connections between the geometric interpretations and numerical linear algebra tools.

4.1.2 Notation

Most of the notation will be drawn from the previous Chapters (Section 2.1.2 and 3.1.3). In this Section, we only introduce the new notation used in this Chapter.

We require the pair $\{A, C\}$ to be observable since only the modes that are observed can be identified. Furthermore, we require the pair $\{A, \begin{bmatrix} B & Q^{1/2} \end{bmatrix}\}$ to be controllable. This implies that all modes are excited by either the external input u_k or the process noise w_k.

The deterministic and stochastic subsystem

The system (4.1)-(4.2) is split in a deterministic and a stochastic subsystem, by splitting the state (x_k) and output (y_k) in a deterministic ($\bullet^d$) and stochastic ($\bullet^s$) component:

$$\begin{aligned} x_k &= x_k^d + x_k^s \ , \\ y_k &= y_k^d + y_k^s \ . \end{aligned}$$

The deterministic state (x_k^d) and output (y_k^d) follow from the deterministic subsystem, which describes the influence of the deterministic input (u_k) on the deterministic output:

$$x_{k+1}^d = Ax_k^d + Bu_k \ , \tag{4.4}$$

$$y_k^d = Cx_k^d + Du_k \ . \tag{4.5}$$

The controllable modes of $\{A, B\}$ can be either stable or unstable. The stochastic state (x_k^s) and output (y_k^s) follow from the stochastic subsystem, which describes the influence of the noise sequences (w_k and v_k) on the stochastic output:

$$x_{k+1}^s = Ax_k^s + w_k \ , \tag{4.6}$$

$$y_k^s = Cx_k^s + v_k \ . \tag{4.7}$$

The controllable modes of $\{A, (Q^s)^{1/2}\}$ are assumed to be stable. The deterministic inputs (u_k) and states (x_k^d) and the stochastic states (x_k^s) and outputs (y_k^s) are assumed to be quasi-stationary (as defined in [Lju 87, page 27]). Note that even though the deterministic subsystem can have unstable modes, the excitation (u_k) has to be chosen in such a way that the deterministic states and output are finite for all time. Also note that since the systems $\{A, B\}$ and $\{A, (Q^s)^{1/2}\}$ are not assumed to be controllable (only the concatenation of the deterministic and stochastic subsystem as a whole should be controllable), the deterministic and stochastic subsystem may have common as well as completely decoupled input-output dynamics.

Block Hankel matrices

The block Hankel matrices $U_{0|2i-1}$ and $Y_{0|2i-1}$ are defined as in Section 2.1.2. The block Hankel matrices $Y^d_{0|2i-1}$ and $Y^s_{0|2i-1}$ are defined in a similar way using the deterministic respectively stochastic output. For notational convenience we use (just as in Chapter 2, Section 2.1.2) the shorthand notation W_p, Y_f, U_f and W_p^+, Y_f^-, U_f^-. The state sequence is defined as:

$$X_i \stackrel{\text{def}}{=} \begin{pmatrix} x_i & x_{i+1} & \dots & x_{i+j-2} & x_{i+j-1} \end{pmatrix} \quad \in \mathbb{R}^{n \times j} \ .$$

The *deterministic* state sequence X_i^d and *stochastic* state sequence X_i^s are defined as:

$$\begin{aligned} X_i^d &\stackrel{\text{def}}{=} \begin{pmatrix} x_i^d & x_{i+1}^d & \dots & x_{i+j-2}^d & x_{i+j-1}^d \end{pmatrix} \quad \in \mathbb{R}^{n \times j} \ , \\ X_i^s &\stackrel{\text{def}}{=} \begin{pmatrix} x_i^s & x_{i+1}^s & \dots & x_{i+j-2}^s & x_{i+j-1}^s \end{pmatrix} \quad \in \mathbb{R}^{n \times j} \ . \end{aligned}$$

In a similar way as in Section 2.1.2, the past and future deterministic and stochastic state sequences are defined as:

$$\begin{aligned} X_p^d = X_0^d \quad &, \quad X_f^d = X_i^d \ , \\ X_p^s = X_0^s \quad &, \quad X_f^s = X_i^s \ . \end{aligned}$$

Note that these are not the key state sequences for combined system identification. The introduction of the more important Kalman filter state sequence is postponed until Section 4.1.3.

System related matrices

For the deterministic subsystem (4.4)-(4.5) we adhere to the notation introduced in Section 2.1.2. We will use the extended observability matrix Γ_i, the (reversed) extended controllability matrix Δ_i^d and the lower triangular block-Toeplitz matrix with Markov parameters H_i^d. We also define the following covariance and cross covariance matrices (which exist due to the quasi stationarity of u_k and x_k^d, see above):

$$\begin{aligned}
R^{uu} &\stackrel{\text{def}}{=} \Phi_{[U_{0|2i-1},U_{0|2i-1}]} \\
&= \begin{pmatrix} \Phi_{[U_p,U_p]} & \Phi_{[U_p,U_f]} \\ \Phi_{[U_f,U_p]} & \Phi_{[U_f,U_f]} \end{pmatrix} \\
&= \begin{pmatrix} R_p^{uu} & R_{pf}^{uu} \\ (R_{pf}^{uu})^T & R_f^{uu} \end{pmatrix} , \\
S^{xu} &\stackrel{\text{def}}{=} \Phi_{[X_p^d,U_{0|2i-1}]} \\
&= \begin{pmatrix} \Phi_{[X_p^d,U_p]} & \Phi_{[X_p^d,U_f]} \end{pmatrix} \\
&= \begin{pmatrix} S_p^{xu} & S_f^{xu} \end{pmatrix} , \\
\Sigma^d &\stackrel{\text{def}}{=} \Phi_{[X_p^d,X_p^d]} .
\end{aligned}$$

For the stochastic subsystem (3.1)-(3.2) we use the notation of Section 3.1.3. We will use the the (reversed) extended controllability matrix Δ_i^c (3.14), and the block-Toeplitz matrices C_i and L_i (3.15)-(3.16).

4.1.3 Kalman filter states

In the derivation of the subspace identification algorithms for combined deterministic - stochastic system identification, the Kalman filter plays a crucial role. In this Subsection, we introduce a closed form equation for the non-steady state Kalman filter state estimate for the combined system. We also introduce a bank of non-steady state Kalman filters generating a sequence of state estimates. In the main Theorem of this Chapter (Section 4.2.3), we then indicate how this state sequence can be recovered directly from the input-output data.

Just as in Chapter 3 we indicate the state estimate by a hat: $\hat{x}_k$.

Theorem 9 Combined non-steady state Kalman filter

Given:

- *The initial state estimate:* $\hat{x}_0$
- *The initial estimate of the matrix:* P_0
- *The input and output measurements* $u_0, y_0, \ldots, u_{k-1}, y_{k-1}$

then the non-steady state Kalman filter state estimate $\hat{x}_k$ *defined by the following recursive formulas:*

$$\begin{aligned} \hat{x}_k &= A\hat{x}_{k-1} + Bu_k + K_{k-1}(y_{k-1} - C\hat{x}_{k-1} - Du_{k-1}) \;, && (4.8)\\ K_{k-1} &= (G - AP_{k-1}C^T)(\Lambda_0 - CP_{k-1}C^T)^{-1} \;, && (4.9)\\ P_k &= AP_{k-1}A^T + (G - AP_{k-1}C^T) \\ &\quad \times(\Lambda_0 - CP_{k-1}C^T)^{-1}(G - AP_{k-1}C^T)^T \;, && (4.10) \end{aligned}$$

can be explicitly written as:

$$\hat{x}_k = \left(\begin{array}{c|c|c} A^k - \Omega_k\Gamma_k & \Delta_k^d - \Omega_k H_k^d & \Omega_k \end{array} \right) \left(\begin{array}{c} \hat{x}_0 \\ \hline u_0 \\ \ldots \\ u_{k-1} \\ \hline y_0 \\ \ldots \\ y_{k-1} \end{array} \right) \;, \tag{4.11}$$

where:

$$\Omega_k \stackrel{def}{=} (\Delta_k^c - A^k P_0 \Gamma_k^T)(L_k - \Gamma_k P_0 \Gamma_k^T)^{-1} \;. \tag{4.12}$$

The explicit solution of the matrix P_k *is equal to:*

$$P_k = A^k P_0 (A^T)^k + (\Delta_k^c - A^k P_0 \Gamma_k^T)(L_k - \Gamma_k P_0 \Gamma_k^T)^{-1}(\Delta_k^c - A^k P_0 \Gamma_k^T)^T \;. \tag{4.13}$$

The proof in Appendix A.5 is one by induction. Just as for purely stochastic systems, it can be proven (see for instance Appendix A.3) that the covariance matrix of the state error $\widetilde{P}_k$ is given by:

$$\begin{aligned} \widetilde{P}_k &= \mathbf{E}[\, (x_k - \hat{x}_k).(x_k - \hat{x}_k)^T \,] \\ &= \Sigma^s - P_k \;. \end{aligned}$$

From this we conclude that the covariance of the initial error on the state estimate is given by $\widetilde{P}_0 = \Sigma^s - P_0$. This indirectly implies that P_0 should be negative definite (or smaller than Σ^s at least[1]). The other two remarks that can be drawn from the above formula are the same as for the stochastic Kalman filter on page 70.

The significance of Theorem 9 is that it indicates how the Kalman filter state estimate $\hat{x}_k$ can be written as a linear combination of the past inputs and output measurements $u_0, y_0, \ldots, u_{k-1}, y_{k-1}$ and of the initial state estimate $\hat{x}_0$.

This observation allows the definition of the state sequence that will be recovered by the combined deterministic-stochastic subspace identification algorithms as (with $\widehat{X}_0$ the sequence of initial states):

$$\begin{aligned}
\widehat{X}_i &= \begin{pmatrix} \hat{x}_i & \hat{x}_{i+1} & \ldots & \hat{x}_{i+j-1} \end{pmatrix} \\
&= \begin{pmatrix} A^i - \Omega_i \Gamma_i \mid \Delta_i^d - \Omega_i H_i^d \mid \Omega_i \end{pmatrix} \begin{pmatrix} \hat{X}_0 \\ \hline U_p \\ \hline Y_p \end{pmatrix} \\
&= \begin{pmatrix} A^i - \Omega_i \Gamma_i \mid \begin{pmatrix} \Delta_i^d - \Omega_i H_i^d & \Omega_i \end{pmatrix} \end{pmatrix} \begin{pmatrix} \hat{X}_0 \\ W_p \end{pmatrix} .
\end{aligned} \tag{4.14}$$

This state sequence is generated by a bank of non-steady state Kalman filters, working in parallel on each of the columns of the block Hankel matrix of past inputs and outputs W_p. Figure 4.3 illustrates this concept. The bank of Kalman filters run in a *vertical* direction (over the columns). Just as for the stochastic Kalman filter sequence, this state sequence only uses *partial* input-output information.

Contrary to the Kalman filter state sequence as defined for the purely stochastic case in the previous Chapter, the Kalman filter state sequence as defined by (4.14) is not unique. Indeed, the sequence depends on the choice of the initial state sequence $\widehat{X}_0$ (which was taken to be zero in the stochastic case) and the initial matrix P_0 (which was also taken equal to zero in the stochastic case). The reason why for the combined case these quantities are not taken equal to zero is that the sequence recovered through the combined subspace identification algorithms is a Kalman filter sequence with $\widehat{X}_0 \neq 0$ and $P_0 \neq 0$. Note also that (as would be expected) the results of the stochastic Kalman filter Theorem 6 can be recovered from the results of Theorem 9 by putting $\hat{x}_0 = 0, P_0 = 0$ and $u_k \equiv 0$.

[1] It would just as well have been possible to substitute P_k by $-P_k$, which would have led to $\widetilde{P}_k = \Sigma^s + P_k$, which seems intuitively more logical. The reason why this is not done is partially historical: In the stochastic framework, the Riccati equations have always been presented as in the book; and partially practical: Changing the sign of P_k in the Riccati equations would also change the sign of the limiting solution when $k \to \infty$. This would imply a negative definite limiting solution $-P$ at infinite k.

$$P_0 \;,\; \widehat{X}_0 = \begin{bmatrix} \widehat{x}_0^i & \cdots & \widehat{x}_q^i & \cdots & \widehat{x}_{j-1}^i \end{bmatrix}$$

$$\boldsymbol{W}_p = \begin{pmatrix} \boldsymbol{U}_p \\ \boldsymbol{Y}_p \end{pmatrix} \qquad \begin{bmatrix} u_0 & u_q & u_{j-1} \\ \vdots & \vdots & \vdots \\ u_{i-1} & u_{i+q-1} & u_{i+j-2} \\ y_0 & y_q & y_{j-1} \\ \vdots & \vdots & \vdots \\ y_{i-1} & y_{i+q-1} & y_{i+j-2} \end{bmatrix} \qquad \begin{array}{c} \text{Kalman} \\ \text{Filter} \\ \downarrow \end{array}$$

$$\widehat{X}_i \qquad \begin{bmatrix} \widehat{x}_i & \cdots & \widehat{x}_{i+q} & \cdots & \widehat{x}_{i+j-1} \end{bmatrix}$$

Figure 4.3 Interpretation of the sequence $\widehat{X}_i$ as a sequence of non-steady state Kalman filter state estimates based upon i measurements of u_k and y_k. When the system matrices A, B, C, D, Q, R, S would be known, the state $\widehat{x}_{i+q}$ could be determined from a non-steady state Kalman filter as follows: Start the filter at time q, with an initial state estimate $\widehat{x}_q^i$ (the superscript "i" is introduced to distinguish the estimated $\widehat{x}_q$ from the initial $\widehat{x}_q^i$) and initial error covariance matrix $\Sigma^s - P_0$. Now iterate the non-steady state Kalman filter over i time steps (the vertical arrow down). The Kalman filter will then return a state estimate $\widehat{x}_{i+q}$. This procedure could be repeated for each of the j columns, and thus we speak about a *bank* of non-steady state Kalman filters. The major observation in subspace algorithms is that the system matrices A, B, C, D, Q, R, S do not have to be known to determine the state sequence $\widehat{X}_i$. It can be determined directly from input-output data (see Theorem 12).

In what follows we will encounter different Kalman filter sequences (in the sense of different initial states $\widehat{X}_0$ and initial matrices P_0). Therefore we will denote the Kalman filter state sequence with initial state $\widehat{X}_0$ and initial matrix P_0 as:

$$\widehat{X}_{i_{[\widehat{X}_0, P_0]}} \; .$$

4.2 GEOMETRIC PROPERTIES OF COMBINED SYSTEMS

4.2.1 Matrix input-output equations

The *matrix input-output equations* for the combined system (similar to the matrix input output equations of Section 2.2.1) are defined in the following Theorem:

Theorem 10 Combined matrix input-output equations

$$Y_p = \Gamma_i X_p^d + H_i^d U_p + Y_p^s \ , \tag{4.15}$$

$$Y_f = \Gamma_i X_f^d + H_i^d U_f + Y_f^s \ , \tag{4.16}$$

$$X_f^d = A^i X_p^d + \Delta_i^d U_p \ . \tag{4.17}$$

The Theorem is easy to prove by recursive substitution into the state space equations.

4.2.2 A Projection Theorem

In this section, we introduce the projection of the future outputs onto the past and future inputs and the past outputs. Through this projection, the Kalman filter states can be related to the data. Contrary to the two previous Chapters, for the combined case there are two main Theorems. The first one is presented in this Section while the second Theorem (which is very similar to the main Theorems of the previous Chapters) will be presented in Subsection 4.2.3.

It is tedious though straightforward to prove the following Theorem which relates the Kalman filter state sequences to the input-output data.

Theorem 11 Orthogonal projection

Under the assumptions that:

1. *The deterministic input u_k is uncorrelated with the process noise w_k and the measurement noise v_k (see also Subsection 1.4.4).*
2. *The input u_k is persistently exciting of order $2i$ (Definition 5).*
3. *The number of measurements goes to infinity $j \to \infty$.*

4. *The process noise w_k and the measurement noise v_k are not identically zero.*

Then:

$$\begin{aligned} \mathcal{Z}_i &\stackrel{def}{=} Y_f / \begin{pmatrix} \boldsymbol{W}_p \\ \boldsymbol{U}_f \end{pmatrix} \\ &= \Gamma_i \widehat{X}_i + H_i^d U_f \ , \end{aligned} \tag{4.18}$$

with:

$$\widehat{X}_i \stackrel{def}{=} \widehat{X}_{i[\widehat{X}_0, P_0]} \ , \tag{4.19}$$

$$\widehat{X}_0 = S^{xu}.(R^{uu})^{-1}. \begin{pmatrix} U_p \\ U_f \end{pmatrix} \ , \tag{4.20}$$

$$P_0 = -[\Sigma^d - S^{xu}.(R^{uu})^{-1}.(S^{xu})^T] \ . \tag{4.21}$$

A proof can be found in Appendix A.6. The importance of Theorem 11 is that it reveals one way in which the Kalman filter state sequence $\widehat{X}_i$ relates to given input-output data. The projected matrix $\mathcal{Z}_i$ can indeed be computed from the given data, without knowing the system matrices. It may come as no surprise that the state sequence $\widehat{X}_i$ shows up in this projection. Indeed, the projection $\mathcal{Z}_i$ could be considered as an optimal prediction of the future output data Y_f, given the past input and output data W_p *and* the future input data U_f (see also Section 4.2.4).

Examining the formulas for the initial state sequence $\widehat{X}_0$ (4.20) and the initial matrix P_0 (4.21) a little closer, leads to the following interesting formulas:

$$\widehat{X}_0 = X_p^d / \begin{pmatrix} \boldsymbol{U}_p \\ \boldsymbol{U}_f \end{pmatrix} \ , \tag{4.22}$$

$$P_0 = -\Phi_{\left[X_p^d / \begin{pmatrix} \boldsymbol{U}_p \\ \boldsymbol{U}_f \end{pmatrix}^{\perp} , X_p^d / \begin{pmatrix} \boldsymbol{U}_p \\ \boldsymbol{U}_f \end{pmatrix}^{\perp} \right]} \ . \tag{4.23}$$

First note that the "real" initial state would be $X_p^d + X_p^s$. Just as for the stochastic case, the statistical part of the initial state X_p^s is impossible to estimate and is thus set to zero (in the previous Chapter $\widehat{X}_0$ was indeed taken equal to zero). This explains why X_0^s does not appear in (4.22). The deterministic part X_p^d of the *real* initial state however enters (4.22). Note that the row space of $\widehat{X}_0$ has to lie in the combined row spaces of W_p and U_f (since $\mathcal{Z}_i$ is constructed by a projection on the combined row space). From (4.22) we can see that $\widehat{X}_0$ is the best estimate of X_p^d lying in the row space of past and future *inputs*.

From formula (4.23) we find that the covariance $\widetilde{P}_0$ of the *error* on the initial state estimate is given by:

$$\begin{aligned}
\widetilde{P}_0 &= \Sigma^s - P_0 \\
&= \Sigma^s + \Phi_{\left[X_p^d / \begin{pmatrix} \boldsymbol{U}_p \\ \boldsymbol{U}_f \end{pmatrix}^{\perp}, X_p^d / \begin{pmatrix} \boldsymbol{U}_p \\ \boldsymbol{U}_f \end{pmatrix}^{\perp}\right]} \\
&= \Phi_{[(X_p^s + X_p^d) - \widehat{X}_0, (X_p^s + X_p^d) - \widehat{X}_0]} ,
\end{aligned}$$

which is positive definite. The last equation indicates that the error on the initial state is given by the variance of the part of the *real* initial state $X_p^d + X_p^s$ that does not lie in the combined row space of U_p and U_f.

Both equations (4.20) and (4.21) can thus be explained intuitively. The initial state estimate is the best estimate of the *real* initial state, lying in the combined row space of U_p and U_f. The initial state error covariance is the covariance of the difference between the *real* initial state and the estimated initial state $\widehat{X}_0$. Finally note that when the inputs u_k are white noise the initial state $\widehat{X}_0 = 0$. Since in this case, there is no correlation between the *real* initial state $X_p^d + X_p^s$ and the inputs U_p and U_f.

4.2.3 Main Theorem

Just as for the deterministic and stochastic identification (Section 2.2.2 and 3.2.1), we present a main Theorem for the combined identification problem. This Theorem allows for the calculation of the row space of a Kalman filter state sequence and of the column space of the extended observability matrix Γ_i directly from the input-output data, without any knowledge of the system matrices. The system matrices can then afterwards be extracted from the state sequence $\widehat{X}_i$ or from Γ_i. An overview of the general combined identification procedure is presented in Figure 4.4.

Note that just as for the deterministic main Theorem 2 and the stochastic main Theorem 8 we introduce two weighting matrices W_1 and W_2. The interpretation and use of these matrices will become clear in Section 4.3 and Chapter 5.

Theorem 12 Combined identification

Under the assumptions that:

1. *The deterministic input u_k is uncorrelated with the process noise w_k and measurement noise v_k (see also Section 1.4.4).*

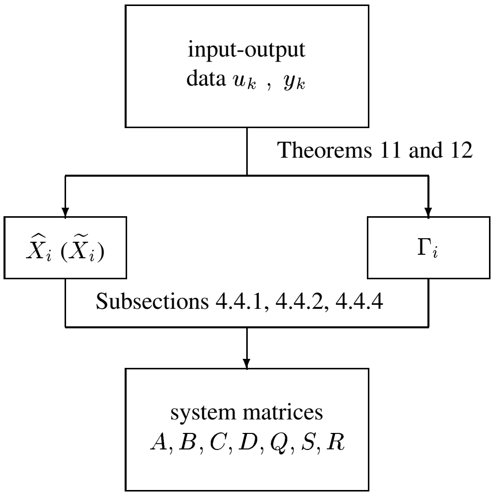

Figure 4.4 An overview of the combined deterministic-stochastic subspace identification procedure. Through the Theorems 11 and 12 the state sequence $\widehat{X}_i$ ($\widetilde{X}_i$) and the extended observability matrix Γ_i are determined. The system matrices are then extracted using any of the three algorithms described in Sections 4.4.1, 4.4.2 and 4.4.4.

2. *The input u_k is persistently exciting of order $2i$ (Definition 5).*
3. *The number of measurements goes to infinity $j \to \infty$.*
4. *The process noise w_k and the measurement noise v_k are not identically zero.*
5. *The user-defined weighting matrices $W_1 \in \mathbb{R}^{li \times li}$ and $W_2 \in \mathbb{R}^{j \times j}$ are such that W_1 is of full rank and W_2 obeys: rank $(W_p) =$ rank $(W_p.W_2)$, where W_p is the block Hankel matrix containing the past inputs and outputs.*

And with $\mathcal{O}_i$ defined as the oblique projection:

$$\mathcal{O}_i \overset{def}{=} Y_f /_{U_f} \boldsymbol{W}_p \; , \tag{4.24}$$

and the singular value decomposition:

$$W_1 \mathcal{O}_i W_2 = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} S_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} = U_1 S_1 V_1^T \; , \tag{4.25}$$

we have:

1. *The matrix $\mathcal{O}_i$ is equal to the product of the extended observability matrix and the Kalman filter state sequence $\widetilde{X}_i$:*
$$\mathcal{O}_i = \Gamma_i.\widetilde{X}_i \ , \tag{4.26}$$
with:
$$\begin{aligned} \widetilde{X}_i &\stackrel{def}{=} \widehat{X}_{i[\widehat{X}_0,P_0]} \ , \\ \widehat{X}_0 &= X_p^d /_{U_f} \boldsymbol{U}_p \ , && (4.27) \\ P_0 &= -[\Sigma^d - S^{xu}.(R^{uu})^{-1}.(S^{xu})^T] \ . && (4.28) \end{aligned}$$
2. *The order of the system (4.1)-(4.2) is equal to the number of singular values in equation (4.25) different from zero.*
3. *The extended observability matrix Γ_i is equal to:*
$$\Gamma_i = W_1^{-1} U_1 S_1^{1/2}.T \ . \tag{4.29}$$
4. *The part of the state sequence $\widetilde{X}_i$ that lies in the column space of W_2 can be recovered from:*
$$\widetilde{X}_i W_2 = T^{-1}.S_1^{1/2} V_1^T \ . \tag{4.30}$$
5. *The state sequence $\widetilde{X}_i$ is equal to:*
$$\widetilde{X}_i = \Gamma_i^\dagger.\mathcal{O}_i \ . \tag{4.31}$$

The proof of this Theorem can be found in Appendix A.7.

Remarks & comments

1. Theorem 12 can algebraically be summarized as follows:

$$\boxed{\begin{aligned} \text{rank } (Y_f /_{U_f} \boldsymbol{W}_p) &= n \\ \text{row space } (Y_f /_{U_f} \boldsymbol{W}_p) &= \text{row space } (\widetilde{X}_i) \\ \text{column space } (Y_f /_{U_f} \boldsymbol{W}_p) &= \text{column space } (\Gamma_i) \end{aligned}}$$

This summary is the essence of why these algorithms are called *subspace* algorithms: they retrieve system related matrices as subspaces of projected data matrices.

2. It should be noted that the state sequence $\widetilde{X}_i$ recovered through this Theorem differs from the state sequence $\widehat{X}_i$ that was introduced in Theorem 11. The two sequences are different due to their different initial state $\widehat{X}_0$:

   - For Theorem 11:
   $$\widehat{X}_0 = X_p^d / \begin{pmatrix} \boldsymbol{U}_p \\ \boldsymbol{U}_f \end{pmatrix} . \tag{4.32}$$

   - For Theorem 12:
   $$\widehat{X}_0 = X_p^d /_{U_f} \boldsymbol{U}_p . \tag{4.33}$$

   This difference in initial states will play a crucial role in the derivation of the algorithms in Section 4.4. Finally note that even though their initial state sequence is different, both sequences $\widehat{X}_i$ and $\widetilde{X}_i$ are generated by a bank of non-steady state Kalman filter sequences.

3. The study of the effect of the weighting matrices W_1 and W_2 is postponed to Section 4.3 and Chapter 5. Suffices to say, that both W_1 and W_2 determine the state space basis in which the identified model will be identified.

4. The same remarks on the similarity transformation T hold as for the deterministic and stochastic main Theorem (Remark 5 on page 43). This implies that we can put the similarity transformation T equal to I_n. In doing so, when using different weighting matrices W_1 and W_2, "different[2]" observability matrices and different state space sequences will be obtained from Theorem 12. However, each set of quantities will lead to a set of state space matrices that is equivalent (up to a similarity transformation) to the original set of matrices.

4.2.4 Intuition behind the Theorems

In this Subsection we present some intuition behind the different projections of Theorem 11 and 12. More intuitive explanations can be found in [VODM 95a] [VODM 94b] [VODM 94c].

[2] In a sense that the numbers in these matrices are different, because the choice of basis in the state space is different.

The goal of an identification procedure is to find a model of which the input-output behavior approximates that of the process under consideration. This goal is classically solved by minimizing a "prediction error criterion" which expresses the "prediction performance" of the model on the given data set. The minimizing solution is designated as the optimal model (see for instance [Lju 87]). In the framework of subspace identification, the identification goal is attained by solving two subsequent problems:

Optimal Prediction: As stated above we want to find a model that will predict the behavior of the process sufficiently accurate. This can be formulated as: predict the future outputs (Y_f) as accurately as possible, using all the information that can be obtained from the past (W_p), and using the knowledge of the inputs that will be presented to the system in the future (U_f).

Inspired by the linearity of the system, we propose to combine the past (W_p) and the future inputs (U_f) linearly to predict the future outputs (Y_f). We denote the linear combinations respectively with L_p and L_u. The quality of the prediction is measured in the Frobenius norm. Mathematically, the first part of the identification goal thus becomes:

$$\min_{\substack{L_p \in \mathbb{R}^{li \times (m+l)i} \\ L_u \in \mathbb{R}^{li \times mi}}} \|Y_f - \begin{pmatrix} L_p & L_u \end{pmatrix} \begin{pmatrix} W_p \\ U_f \end{pmatrix} \|_F^2 \ . \tag{4.34}$$

The projection $\mathcal{Z}_i$ of Theorem 11 is exactly equal to this linear combination of W_p and U_f:

$$\mathcal{Z}_i = \begin{pmatrix} L_p & L_u \end{pmatrix} \begin{pmatrix} W_p \\ U_f \end{pmatrix} \ .$$

The optimal combination of the past (W_p) to predict the future is $L_p.W_p$, which is exactly equal to the oblique projection of Theorem 12:

$$\mathcal{O}_i = L_p.W_p \ .$$

Complexity Reduction: Apart from the fact that we want to find a model that can predict the future, we also want the order of this model to be as low as possible. Or equivalently, we want to reduce the complexity of the amount of "information" of the past that we need to keep track of to predict the future. We thus need to reduce the complexity of $\mathcal{O}_i$. Since the rows of $\mathcal{O}_i$ span an li dimensional subspace in the j dimensional ambient space, we can introduce a complexity reduction by reducing the subspace dimension to n (the order of the system). Intuitively, this implies that we only have to remember n different directions of the past to predict the future. Mathematically, the second step can be formulated:

$$\min_{\mathcal{R} \in \mathbb{R}^{li \times j}} \|W_1 \, [\mathcal{O}_i - \mathcal{R}] \, W_2\|_F^2 \, , \tag{4.35}$$

$$\text{subject to: rank } (\mathcal{R}) = n \, .$$

The user defined weighting matrices W_1 and W_2 determine which part of the "information" of $\mathcal{O}_i$ is important to retain. A more rigorous interpretation of the weighting matrices is presented in Chapter 5. It is easy to show [VODM 95a] that $\mathcal{R}$ is determined through the singular value decomposition (4.25) as:

$$\mathcal{R} = W_1^{-1} U_1 S_1 V_1^T W_2^{\dagger} \, .$$

Note that $\mathcal{R} = \mathcal{O}_i$ when all the assumptions of Theorem 12 are satisfied exactly (which however is never the case with real data). In that case, $\mathcal{R}$ is merely a basis for the row space of $\mathcal{O}_i$. However, when $j \neq \infty$ or when the data generating system is not linear, the singular values of $W_1 \mathcal{O}_i W_2$ (4.25) are all different from zero. In that case, the row space of $\mathcal{O}_i$ is of dimension li, and the order has to be chosen equal to the number of "dominant" singular values. The complexity reduction step is then truly a reduction of the dimension of the row space of $\mathcal{O}_i$, and the weights W_1 and W_2 play an important role in determining which part of the original row space of $\mathcal{O}_i$ is retained.

4.3 RELATION TO OTHER ALGORITHMS

*In this Section we indicate how three subspace algorithms of the literature (***N4SID***,* **MOESP** *and* **CVA***) are special cases of Theorem 12 with for each of the algorithms a specific choice of the weighting matrices W_1 and W_2. These results were drawn from [VODM 94b], to which we refer for the detailed proofs.*

In this Section we show how through a specific choice of the weighting matrices W_1 and W_2, published algorithms are recovered. The results are summarized in Table 4.1.

	W_1	W_2
N4SID	I_{li}	I_j
MOESP	I_{li}	$\Pi_{U_f^\perp}$
CVA	$\Phi^{-1/2}_{[Y_f/U_f^\perp, Y_f/U_f^\perp]}$	$\Pi_{U_f^\perp}$

Table 4.1 Overview of the special choices of W_1 and W_2 to obtain the published algorithms **N4SID**, **MOESP** and **CVA**.

4.3.1 N4SID

The acronym **N4SID** stands for "**N**umerical algorithms for **S**ubspace **S**tate **S**pace **S**ystem **ID**entification" and is pronounced as a Californian license plate: *enforce it*. The algorithm of [VODM 94a] determines the order of the system and Γ_i directly from the singular value decomposition of the oblique projection (the superscript n stands for "n4sid"):

$$\mathcal{O}_i = \begin{pmatrix} U_1^n & U_2^n \end{pmatrix} \begin{pmatrix} S_1^n & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} (V_1^n)^T \\ (V_2^n)^T \end{pmatrix} . \tag{4.36}$$

The order is equal to the number of non-zero singular values in S_1^n. The observability matrix is taken equal to:

$$\Gamma_i = U_1^n . (S_1^n)^{1/2} . \tag{4.37}$$

The algorithm of [VOWL 93] calculates the singular value decomposition of $\hat{G}$ (adjusted to our notation):

$$\hat{G} = \Phi_{[Y_f, \begin{pmatrix} W_p \\ U_f \end{pmatrix}]} . [\Phi_{[\begin{pmatrix} W_p \\ U_f \end{pmatrix}, \begin{pmatrix} W_p \\ U_f \end{pmatrix}]}]^{-1} . \begin{pmatrix} I_{(m+l)i} & 0 \\ 0 & 0 \end{pmatrix} . [\Phi_{[\begin{pmatrix} W_p \\ U_f \end{pmatrix}, \begin{pmatrix} W_p \\ U_f \end{pmatrix}]}]^{1/2} . \tag{4.38}$$

It is easy to show that (see [VODM 94b]):

$$\begin{aligned} \hat{G}.\hat{G}^T &= \Phi_{[Y_f/U_f^\perp, W_p/U_f^\perp]} . \left[\Phi_{[W_p/U_f^\perp, W_p/U_f^\perp]}\right]^{-1} . \Phi_{[W_p, W_p]} \\ &\quad . \left[\Phi_{[W_p/U_f^\perp, W_p/U_f^\perp]}\right]^{-1} . \Phi_{[W_p/U_f^\perp, Y_f/U_f^\perp]} , \end{aligned}$$

which is exactly equal to $\mathcal{O}_i \mathcal{O}_i^T$ (use formula (1.7)). This implies that both algorithms [VODM 94a] and [VOWL 93] calculate the same singular values S_1^n and the same left singular space U_1^n, and thus determine the order n and Γ_i in exactly the same way.

Note that in [VOWL 93] an interpretation of the algorithm is given in an instrumental variable framework.

It is now easy to see that the algorithms of [VODM 94a] [VOWL 93] correspond to Theorem 12 with the weights:

$$\begin{aligned} W_1 &= I_{li} \ , \\ W_2 &= I_j \ . \end{aligned}$$

A consequence of $W_2 = I_j$ is that we do not need formula (4.31) to determine the states $\widetilde{X}_i$, but that they can be determined simply from the singular value decomposition (4.36) (using (4.30)) as:

$$\widetilde{X}_i^n = (S_1^n)^{1/2}(V_1^n)^T \ . \tag{4.39}$$

In the next subsections, it will become clear that for the other two algorithms, the determination of the state sequence $\widetilde{X}_i$ requires the use of formula (4.31).

4.3.2 MOESP

The acronym **MOESP** stands for "**M**ultivariable **O**utput-**E**rror **S**tate s**P**ace". Verhaegen considers in [Ver 94] the following LQ decomposition [Ver 94, page 12] (adapted to our notation):

$$\begin{pmatrix} U_f \\ W_p \\ Y_f \end{pmatrix} = \begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} Q_1^T \\ Q_2^T \\ Q_3^T \end{pmatrix} \ . \tag{4.40}$$

With the singular value decomposition (the superscript m stands for "moesp"):

$$L_{32} = \begin{pmatrix} U_1^m & U_2^m \end{pmatrix} \begin{pmatrix} S_1^m & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} (V_1^m)^T \\ (V_2^m)^T \end{pmatrix} \ . \tag{4.41}$$

Verhaegen proves that the system order can be retrieved from the number of singular values of S_1^m different from zero. He also proves that a possible choice for Γ_i is:

$$\Gamma_i = U_1^m.(S_1^m)^{1/2} \ . \tag{4.42}$$

It is proven in [VODM 95a] that the algorithm of [Ver 94] corresponds to Theorem 12 with the following weights:

$$\begin{aligned} W_1 &= I_{li} \ , \\ W_2 &= \Pi_{U_f^\perp} \ . \end{aligned}$$

Note that, since W_2 is not of full rank, we do not recover the full state from the singular value decomposition of $L_{32}Q_2$. According to formula (4.30), we only recover the projection of the state:

$$\widetilde{X}_i^m.W_2 = \widetilde{X}_i^m.\Pi_{U_f^\perp} \ .$$

The state could be determined through formula (4.31). However, since **MOESP** does not use state sequences, we will not elaborate on this any further.

4.3.3 CVA

Larimore considers in [Lar 90] the canonical correlations between, on the one hand the past W_p conditional to the future inputs U_f, and on the other hand the future outputs Y_f conditional to the future inputs U_f. In formulas, this means that he considers the principal angles and directions between $W_p/\boldsymbol{U}_f^\perp$ and $Y_f/\boldsymbol{U}_f^\perp$. He denotes his class of algorithms by **CVA** which stands for "**C**anonical **V**ariate **A**nalysis".

In [VODM 95a] it is shown that the number of principal angles in $[W_p/U_f^\perp \sphericalangle Y_f/U_f^\perp]$ different from $\pi/2$ is equal to the model order. The way we understand it, Larimore defines a "memory" $\mathcal{M}$ by making linear combinations of the rows of the past inputs and outputs W_p: With

$$[\boldsymbol{W}_p/\boldsymbol{U}_f^\perp \sphericalangle Y_f/U_f^\perp] = L_c \ . \ W_p/\boldsymbol{U}_f^\perp \ ,$$

the "memory" $\mathcal{M}$ in [Lar 90] is defined as:

$$\mathcal{M} = L_c \ . \ W_p \ .$$

Note that this "memory" is different from the principal directions in the projected past $[\boldsymbol{W}_p/\boldsymbol{U}_f^\perp \sphericalangle Y_f/U_f^\perp]$. In [VODM 95a] it is shown that the algorithm of [Lar 90] corresponds to Theorem 12 with the following weights:

$$\begin{aligned} W_1 &= \Phi^{-1/2}_{[Y_f/U_f^\perp, Y_f/U_f^\perp]} \ , \\ W_2 &= \Pi_{U_f^\perp} \ . \end{aligned}$$

From this we conclude that applying the above weighting matrices to the results of Theorem 12 leads to a principal direction analysis between the past inputs and outputs orthogonalized to the future inputs $W_p/\boldsymbol{U}_f^\perp$ and the future outputs orthogonalized to the future inputs $Y_f/\boldsymbol{U}_f^\perp$. The singular values of (4.25) correspond to the cosines of the principal angles. Furthermore, it is shown in [VODM 95a] that the "memory" as defined by Larimore corresponds to the states as defined in equation (4.31). The partial states recovered from (4.30) are equal to the principal directions $[\boldsymbol{W}_p/\boldsymbol{U}_f^{\perp} \sphericalangle Y_f/U_f^{\perp}]$. This proves formally that the principal directions are no states, but projected states:

$$[\boldsymbol{W}_p/\boldsymbol{U}_f^{\perp} \sphericalangle Y_f/U_f^{\perp}] = \widetilde{X}_i^c/\boldsymbol{U}_f^{\perp} \ .$$

An additional point concerning the **CVA** method is that Larimore claims that the weighting used in this method is optimal for state order determination from finite sample sizes. This has been shown by example in [Lar 94] but has never been proven. A last remark concerns the sensitivity to scaling. While the two algorithms **N4SID** and **MOESP** are sensitive to scaling of the inputs and/or outputs, the **CVA** algorithm is insensitive. This is because only angles and normalized directions are considered in the **CVA** algorithm.

Note that the general framework proposed in [Lar 90] corresponds to a generalization of principal directions and angles. In this framework, the matrix

$$W_1 = \Phi^{-1/2}_{[Y_f/U_f^{\perp}, Y_f/U_f^{\perp}]} \ ,$$

is replaced by another weighting matrix

$$W_1 = \Lambda^{-1/2} \ ,$$

which still falls into the unifying approach of Theorem 12.

4.3.4 A simulation example

In this Subsection, we illustrate the behavior of the three algorithms of this Section (**N4SID**, **MOESP** and **CVA**) with a simple example. Consider the following combined system in forward innovation form:

$$\begin{aligned} x_{k+1} &= \begin{pmatrix} 0.6 & 0.6 & 0 \\ -0.6 & 0.6 & 0 \\ 0 & 0 & 0.7 \end{pmatrix} x_k + \begin{pmatrix} 1.6161 \\ -0.3481 \\ 2.6319 \end{pmatrix} u_k + \begin{pmatrix} -1.1472 \\ -1.5204 \\ -3.1993 \end{pmatrix} e_k^f \ , \\ y_k &= \begin{pmatrix} -0.4373 & -0.5046 & 0.0936 \end{pmatrix} x_k - 0.7759 u_k + e_k^f \ , \end{aligned}$$

and:

$$\mathbf{E}[e_k^f.(e_k^f)^T] = 0.0432 \ .$$

An output sequence y_k of 4000 data points was generated using this model. The input sequence consisted of the sum of a zero mean, Gaussian, white noise signal (variance 1) filtered with a second order Butterworth filter (cutoff at 0.3 times the Nyquist frequency, $T = 1$), and a zero mean, Gaussian, white noise signal with variance 0.01.

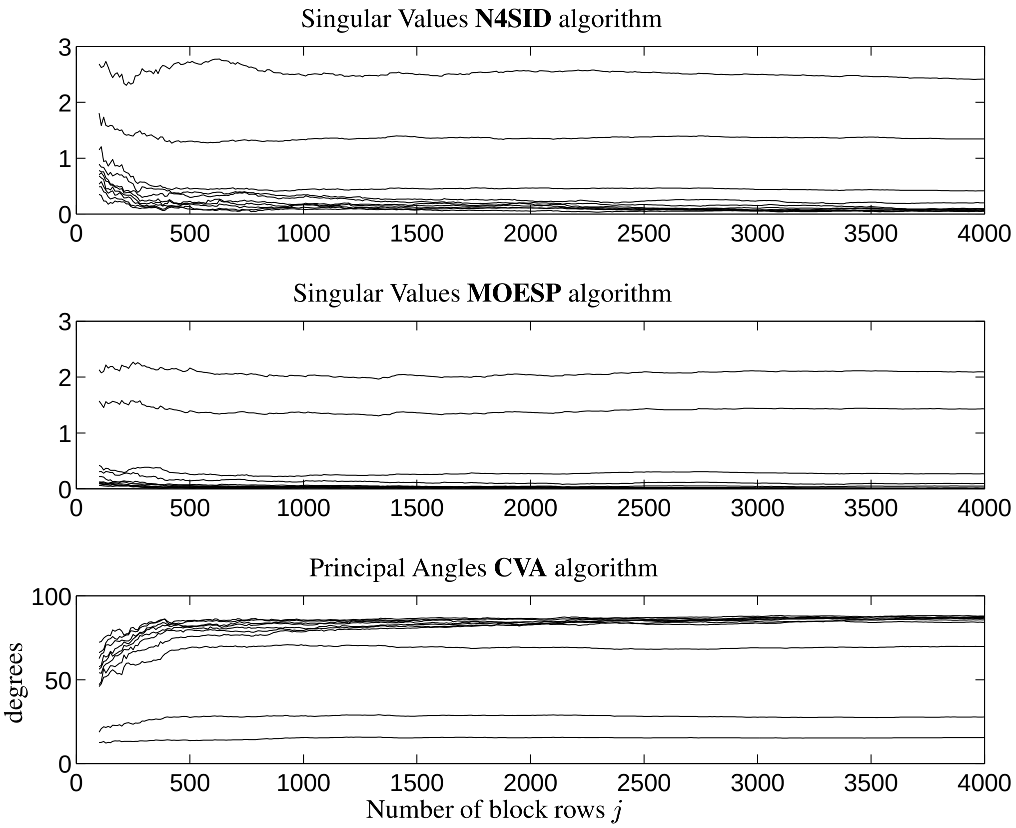

Figure 4.5 The order determining singular values (or principal angles) in function of the number of block columns j for the three algorithms presented in this Section: **N4SID**, **MOESP** and **CVA**. For each of the three plots, the system order three can be be determined (even though it is the most obvious for the **CVA** algorithm). The convergence of the singular values in function of j can also be observed. For the first two plots all singular values go to zero, except three. For the last plot, all principal angles go to 90 degrees, except three. This Figure was generated using the Matlab file com_sim1.m.

The number of block rows i is taken equal to 10. For each j between 100 and 4000, the singular values of $W_1 \mathcal{O}_i W_2$ were plotted in Figure 4.5. The convergence in function of j can be clearly observed.

The Matlab file `com_sim1.m` contains a Matlab implementation of this example.

4.4 COMPUTING THE SYSTEM MATRICES

In this Section we explain how the system matrices A, B, C, D and Q, S, R can be computed from Theorem 11 and/or 12 in two different ways. A schematic overview of the two algorithms can be found in Figures 4.6 and 4.7. Note that the first part of the two algorithms are the same and coincide with the steps described in Theorem 12. At the end of the Section, we present variations and optimizations of the first algorithm, which are motivated from a practical rather than from a theoretical point of view, and lead to the "robust" algorithm of Figure 4.8.

4.4.1 Algorithm 1: unbiased, using the states

From Theorem 12, we find:

- The order of the system from inspection of the singular values of equation (4.25).
- The extended observability matrix Γ_i from equation (4.29) and the matrix Γ_{i-1} as $\underline{\Gamma_i}$.

The following side result of Theorem 11 can easily be proven. By shifting the border between "past" and "future" one down, we obtain the matrices W_p^+, U_f^- and Y_f^-. The same projection as in Theorem 11 can now be performed with these matrices. This leads to the sequence $\mathcal{Z}_{i+1}$ and the Kalman filter states $\hat{X}_{i+1}$:

$$\begin{aligned} \mathcal{Z}_{i+1} &= Y_f^- / \begin{pmatrix} \boldsymbol{W}_p^+ \\ \boldsymbol{U}_f^- \end{pmatrix} &\quad (4.43) \\ &= \Gamma_{i-1}\hat{X}_{i+1} + H_{i-1}^d U_f^- \ , &\quad (4.44) \\ \hat{X}_{i+1} &= \hat{X}_{i+1}{}_{[\hat{X}_0, P_0]} \ , &\quad (4.45) \end{aligned}$$

with $\hat{X}_0$ and P_0 given by equations (4.20)-(4.21). Since the corresponding columns of $\hat{X}_i$ (equation (4.19)) and of $\hat{X}_{i+1}$ (equation (4.45)) are state estimates of the same non-steady state Kalman filters at two consecutive time instants[3], we can write with (4.8):

$$\hat{X}_{i+1} = A\hat{X}_i + BU_{i|i} + K_i(Y_{i|i} - C\hat{X}_i - DU_{i|i}) \ . \quad (4.46)$$

It is also trivial that:

$$Y_{i|i} = C\hat{X}_i + DU_{i|i} + (Y_{i|i} - C\hat{X}_i - DU_{i|i}) \ . \quad (4.47)$$

[3]The Kalman filters are the same in the sense that they have the same initial state $\hat{X}_0$ and the same initial error covariance $\tilde{P}_0$.

It is now easy to prove (see [VODM 94a]) that the row space of $Y_{i|i} - C\widehat{X}_i - DU_{i|i}$ is orthogonal to the row spaces of W_p, U_f and $\widehat{X}_i$. This result can also be intuitively proven by noticing that the innovations $(Y_{i|i} - C\widehat{X}_i - DU_{i|i})$ of a non-steady state Kalman filter are uncorrelated with the states $\widehat{X}_i$, the past inputs and outputs W_p and the future inputs U_f. We thus have:

$$\mathbf{E_j}\left[(Y_{i|i} - C\widehat{X}_i - DU_{i|i}).\begin{pmatrix} W_p \\ U_f \\ \widehat{X}_i \end{pmatrix}^T\right] = 0 \ .$$

This implies that (4.46)-(4.47):

$$\begin{pmatrix} \widehat{X}_{i+1} \\ Y_{i|i} \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}\begin{pmatrix} \widehat{X}_i \\ U_{i|i} \end{pmatrix} + \begin{pmatrix} \rho_w \\ \rho_v \end{pmatrix} \ , \tag{4.48}$$

where the row spaces of ρ_w and ρ_v are orthogonal to the row space of W_p, U_f and $\widehat{X}_i$. If we would now be able to compute the state sequences $\widehat{X}_i$ and $\widehat{X}_{i+1}$ from the input-output data, we would (as in the previous Chapters) solve equation (4.48) in a least squares sense for the system matrices A, B, C, D. Unfortunately, it is not possible to determine the state sequences $\widehat{X}_i$ and $\widehat{X}_{i+1}$ directly from the input-output data[4]. However, from (4.18) and (4.44) we can determine $\widehat{X}_i$ and $\widehat{X}_{i+1}$ as:

$$\widehat{X}_i = \Gamma_i^\dagger \ . \ [\mathcal{Z}_i - H_i^d.U_f] \ , \tag{4.49}$$

$$\widehat{X}_{i+1} = \Gamma_{i-1}^\dagger \ . \ [\mathcal{Z}_{i+1} - H_{i-1}^d.U_f^-] \ . \tag{4.50}$$

In these formulas the only unknowns on the right hand side are H_i^d and H_{i-1}^d, since $\mathcal{Z}_i$ and $\mathcal{Z}_{i+1}$ can be determined as a projection of the input-output block Hankel matrices and Γ_i and Γ_{i-1} are determined through Theorem 12. Substitution of (4.49) and (4.50) into (4.48) leads to:

$$\left(\frac{\Gamma_{i-1}^\dagger.\mathcal{Z}_{i+1}}{Y_{i|i}}\right) = \underbrace{\left(\frac{A}{C}\right)}_{\text{term 1}}.\Gamma_i^\dagger.\mathcal{Z}_i + \underbrace{\mathcal{K}}_{\text{term 2}}.U_f + \underbrace{\left(\frac{\rho_w}{\rho_v}\right)}_{\text{term 3}} \ , \tag{4.51}$$

where we defined:

$$\mathcal{K} \stackrel{\text{def}}{=} \begin{pmatrix} \left(\ B \ \middle| \ \Gamma_{i-1}^\dagger.H_{i-1}^d \ \right) - A.\Gamma_i^\dagger.H_i^d \\ \left(\ D \ \middle| \ 0 \ \right) - C.\Gamma_i^\dagger.H_i^d \end{pmatrix} \ . \tag{4.52}$$

[4] For all clarity, Theorem 12 determines the state sequence $\widetilde{X}_i$ directly from the input-output data. This sequence is different from $\widehat{X}_i$, in a sense that the initial conditions of the Kalman filter are different (see (4.32) and (4.33)). We will use the sequence $\widetilde{X}_i$ in the second algorithm.

Observe that the matrices B and D appear linearly in the matrix $\mathcal{K}$. We can now solve equation (4.51) in a least squares sense for A,C and $\mathcal{K}$. Since the row spaces of ρ_w and ρ_v have been shown to be orthogonal to the row spaces of $\mathcal{Z}_i$ and U_f and since the least squares solution computes residuals that are orthogonal to the regressors, the least squares solution will compute asymptotically unbiased estimates of the system matrices (see [VODM 94a]). From (4.51) we find in this way (term by term):

term 1. The system matrices A and C.

term 2. The matrix $\mathcal{K}$ from which B and D can be computed. This is done in a similar way as described in Section 2.4.2. Define (and compute, since all quantities at the right hand side are known at this moment):

$$\begin{aligned}
\mathcal{L} &\stackrel{\text{def}}{=} \begin{pmatrix} A \\ C \end{pmatrix} .\Gamma_i^\dagger \quad \in \mathbb{R}^{(n+l)\times li} \\
&= \begin{pmatrix} \mathcal{L}_{1|1} & \mathcal{L}_{1|2} & \dots & \mathcal{L}_{1|i} \\ \mathcal{L}_{2|1} & \mathcal{L}_{2|2} & \dots & \mathcal{L}_{2|i} \end{pmatrix} , \\
\mathcal{M} &\stackrel{\text{def}}{=} \Gamma_{i-1}^\dagger \quad \in \mathbb{R}^{n\times l(i-1)} \\
&= \begin{pmatrix} \mathcal{M}_1 & \mathcal{M}_2 & \dots & \mathcal{M}_{i-1} \end{pmatrix} , \\
\mathcal{K} &= \begin{pmatrix} \mathcal{K}_{1|1} & \mathcal{K}_{1|2} & \dots & \mathcal{K}_{1|i} \\ \mathcal{K}_{2|1} & \mathcal{K}_{2|2} & \dots & \mathcal{K}_{2|i} \end{pmatrix} ,
\end{aligned}$$

where $\mathcal{L}_{1|k}, \mathcal{M}_k \in \mathbb{R}^{n\times l}$, $\mathcal{K}_{1|k} \in \mathbb{R}^{n\times m}$, $\mathcal{L}_{2|k} \in \mathbb{R}^{l\times l}$ and $\mathcal{K}_{2|k} \in \mathbb{R}^{l\times m}$. Through similar manipulations as in Section (2.4.2), Equation (4.52) can then be rewritten as:

$$\begin{pmatrix} \mathcal{K}_{1|1} \\ \mathcal{K}_{1|2} \\ \mathcal{K}_{1|3} \\ \vdots \\ \mathcal{K}_{1|i} \\ \hline \mathcal{K}_{2|1} \\ \mathcal{K}_{2|2} \\ \mathcal{K}_{2|3} \\ \vdots \\ \mathcal{K}_{2|i} \end{pmatrix} = \mathcal{N} \begin{pmatrix} D \\ B \end{pmatrix} , \tag{4.53}$$

with:

$$\mathcal{N} = \left(\begin{array}{ccccc} -\mathcal{L}_{1|1} & \mathcal{M}_1 - \mathcal{L}_{1|2} & \dots & \mathcal{M}_{i-2} - \mathcal{L}_{1|i-1} & \mathcal{M}_{i-1} - \mathcal{L}_{1|i} \\ \mathcal{M}_1 - \mathcal{L}_{1|2} & \mathcal{M}_2 - \mathcal{L}_{1|3} & \dots & \mathcal{M}_{i-1} - \mathcal{L}_{1|i} & 0 \\ \mathcal{M}_2 - \mathcal{L}_{1|3} & \mathcal{M}_3 - \mathcal{L}_{1|4} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \mathcal{M}_{i-1} - \mathcal{L}_{1|i} & 0 & \dots & 0 & 0 \\ \hline I_l - \mathcal{L}_{2|1} & -\mathcal{L}_{2|2} & \dots & -\mathcal{L}_{2|i-1} & -\mathcal{L}_{2|i} \\ -\mathcal{L}_{2|2} & -\mathcal{L}_{2|3} & \dots & -\mathcal{L}_{2|i} & 0 \\ -\mathcal{L}_{2|3} & -\mathcal{L}_{2|4} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ -\mathcal{L}_{2|i} & 0 & \dots & 0 & 0 \end{array}\right) \times \begin{pmatrix} I_l & 0 \\ 0 & \Gamma_{i-1} \end{pmatrix} \in \mathbb{R}^{i(n+l)\times(n+l)} . \qquad (4.54)$$

Formula (4.53) is an overdetermined set of linear equations in the unknowns B and D, which could for instance be solved using least squares.

term 3. The covariances Q, S and R can be approximated from the residuals ρ_w and ρ_v in a similar way as in Section 3.4.3 (stochastic identification algorithm 3):

$$\begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} \simeq \mathbf{E_j}[\begin{pmatrix} \rho_w \\ \rho_v \end{pmatrix} . \begin{pmatrix} \rho_w^T & \rho_v^T \end{pmatrix}] .$$

As indicated in Section 3.4.3, the approximation is due to the fact that the bank of Kalman filters is not in steady state for finite i. As i grows larger, the approximation error grows smaller. For infinite i and j the stochastic system is determined asymptotically unbiased. The stochastic subsystem however always leads to a positive real covariance sequence (just as in Section 3.4.3), since the covariance matrix is always positive definite.

The Matlab function `com_alt.m` contains a Matlab implementation of this algorithm. See also Section 6.1 and Appendix B.

4.4.2 Algorithm 2: biased, using the states

Algorithm 1 computes unbiased estimates of the system matrices A, B, C and D. However, compared to the algorithms based on the state sequences of the previous Chapters, Algorithm 1 is quite complicated (especially the extraction of B and D). One could wonder if there does not exist an algorithm that has the same simplicity for the combined identification case. The answer is yes, but this algorithm calculates asymptotically biased solutions (see also [VODM 94a]).

Combined algorithm 1:

1. Calculate the oblique and orthogonal projections:

$$\begin{aligned} \mathcal{O}_i &= Y_f /_{U_f} \boldsymbol{W}_p \ , \\ \mathcal{Z}_i &= Y_f / \begin{pmatrix} \boldsymbol{W}_p \\ \boldsymbol{U}_f \end{pmatrix} \ , \\ \mathcal{Z}_{i+1} &= Y_f^- / \begin{pmatrix} \boldsymbol{W}_p^+ \\ \boldsymbol{U}_f^- \end{pmatrix} \ . \end{aligned}$$

2. Calculate the **SVD** of the weighted oblique projection:

$$W_1 \mathcal{O}_i W_2 = U S V^T \ .$$

3. Determine the order by inspecting the singular values in S and partition the **SVD** accordingly to obtain U_1 and S_1.

4. Determine Γ_i and Γ_{i-1} as:

$$\Gamma_i = W_1^{-1} U_1 S_1^{1/2} \quad , \quad \Gamma_{i-1} = \underline{\Gamma_i} \ .$$

5. Solve the set of linear equations for A, C and $\mathcal{K}$:

$$\left(\frac{\Gamma_{i-1}^{\dagger} . \mathcal{Z}_{i+1}}{Y_{i|i}} \right) = \left(\frac{A}{C} \right) . \Gamma_i^{\dagger} . \mathcal{Z}_i + \mathcal{K} . U_f + \left(\frac{\rho_w}{\rho_v} \right) \ .$$

6. Determine B and D from $\mathcal{K}, A, C, \Gamma_i, \Gamma_{i-1}$ through equation (4.53).

7. Determine Q, S and R from the residuals as:

$$\begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} = \mathbf{E_j}[\begin{pmatrix} \rho_w \\ \rho_v \end{pmatrix} . \begin{pmatrix} \rho_w^T & \rho_v^T \end{pmatrix}] \ .$$

Figure 4.6 A schematic overview of the first combined deterministic-stochastic identification algorithm. See Section 6.1 for implementation issues. This algorithm has been implemented in the Matlab function com_alt.m.

As stated before, the state sequence $\widehat{X}_i$ can not be calculated directly from the data. However, from Theorem 12 we find that the states $\widetilde{X}_i$ can be determined from the data. From a similar reasoning as in Theorem 12, we find that:

$$\begin{aligned}\mathcal{O}_{i+1} &= Y_f^- /_{U_f^-} \boldsymbol{W}_p^+ \\ &= \Gamma_{i-1}.\widetilde{X}_{i+1} \ .\end{aligned}$$

And we thus find $\widetilde{X}_i$ and $\widetilde{X}_{i+1}$. The problem however is that this new Kalman filter sequence $\widetilde{X}_{i+1}$ has a different initial state as the sequence $\widetilde{X}_i$. Indeed, we have:

- Initial state for $\widetilde{X}_i$: $X_p^d /_{U_f} \boldsymbol{U}_p$.
- Initial state for $\widetilde{X}_{i+1}$: $X_p^d /_{U_f^-} \boldsymbol{U}_p^+$.

So, we can not write a formula similar to (4.48) with $\widehat{X}_i, \widehat{X}_{i+1}$ replaced by $\widetilde{X}_i, \widetilde{X}_{i+1}$. It can be proven however (see [VODM 94a]) that the difference between $\widehat{X}_i$ and $\widetilde{X}_i$ is zero when at least one of the following conditions is satisfied:

- $i \to \infty$. The difference between $\widehat{X}_i$ and $\widetilde{X}_i$ goes to zero at the same rate the non-steady state Kalman filter converges to a steady state Kalman filter. This is intuitively clear, since by the time the Kalman filter is in steady state, the effect of the initial conditions has died out. In [VODM 94a] a rigorous proof is given.
- The system is purely deterministic. This is the case treated in Chapter 2.
- The deterministic input u_k is white noise. In this case the deterministic state X_p^d is uncorrelated with U_p and U_f. This implies that the initial state sequences of $\widehat{X}_i, \widehat{X}_{i+1}, \widetilde{X}_i$ and $\widetilde{X}_{i+1}$ are all equal to zero (and are thus equal to each other).

When either one of these conditions is satisfied, we can replace $\widehat{X}_i$ and $\widehat{X}_{i+1}$ in (4.48) with $\widetilde{X}_i$ and $\widetilde{X}_{i+1}$, and solve for A, B, C and D in a least squares sense. The details of this simple algorithm are summarized in Figure 4.7. If none of the above conditions is satisfied, this second algorithm will return biased estimates of the system matrices (since the states are replaced by approximations). In [VODM 94a] an expression for this bias is derived. When one of the three above condition is satisfied however, the algorithm is asymptotically unbiased.

Even though the algorithm is simpler than algorithm 1, it turns out that it should be used with care in practice. For many practical measurements the input signal u_k is anything but white noise (steps, impulses, ...). This implies that the algorithm is biased, and this bias can be significant. See also the practical examples in Section 6.4.

The Matlab function `com_stat.m` contains a Matlab implementation of this algorithm. See also Section 6.1 and Appendix B.

4.4.3 Variations and optimizations of Algorithm 1

In this Section we introduce some alterations to the first combined identification algorithm. Using algorithm 1 on practical (finite sample length) data indicated that it does not always perform well in practice. Especially for badly conditioned input Hankel matrices $U_{0|2i-1}$, the performance was not satisfactory (see Section 6.4). Since in practice many input signals are for instance steps (which lead to badly conditioned input Hankel matrices), this is a significant problem. Before introducing the alterations, we should note that:

- The alterations lead to an increased computational load. However, in most cases we believe this is a prize worth paying for the increased accuracy.
- Many new algorithms can be devised by including one or more alterations in algorithm 1. Unfortunately, at the current status of the research, it is not clear which combination of alterations leads to the "best" algorithm. However, at the end of this Section we will suggest a robust[5] algorithm that performed well on all industrial data sets at our disposition (see Section 6.4). It should be taken into account that when more research results become available, the algorithm is likely to be the subject of more fine-tuning.
- The motivation behind the alterations is not always based on theoretical grounds. Some of them are motivated heuristically. The heuristics however have their basis in our practical experience (see Section 6.4).

1. A first alteration comes from the observation that once A and C are determined from the first term of (4.51), the matrices Γ_i and Γ_{i-1} can be recomputed. Indeed, observe that the original Γ_i and Γ_{i-1} determined from (4.25) are only approximations[6] of the exact Γ_i and Γ_{i-1}. Since these matrices play an important

[5] Robust in the sense that it will perform reasonably well for most practical and industrial conditions.

[6] As mentioned before, the matrices will be exact when there is an infinite amount of data generated by a linear time-invariant system available. In practice this is never the case.

Combined algorithm 2:

1. Calculate the oblique projections:

$$\begin{aligned}\mathcal{O}_i &= Y_f /_{U_f} \boldsymbol{W}_p \ , \\ \mathcal{O}_{i+1} &= Y_f^- /_{U_f^-} \boldsymbol{W}_p^+ \ .\end{aligned}$$

2. Calculate the **SVD** of the weighted oblique projection:

$$W_1 \mathcal{O}_i W_2 = U S V^T \ .$$

3. Determine the order by inspecting the singular values in S and partition the **SVD** accordingly to obtain U_1 and S_1.

4. Determine Γ_i and Γ_{i-1} as:

$$\Gamma_i = W_1^{-1} U_1 S_1^{1/2} \quad , \quad \Gamma_{i-1} = \underline{\Gamma_i} \ .$$

5. Determine the state sequences:

$$\begin{aligned}\widetilde{X}_i &= \Gamma_i^\dagger . \mathcal{O}_i \ , \\ \widetilde{X}_{i+1} &= \Gamma_{i-1}^\dagger . \mathcal{O}_{i+1} \ .\end{aligned}$$

6. Solve the set of linear equations for A, B, C and D:

$$\begin{pmatrix} \widetilde{X}_{i+1} \\ Y_{i|i} \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} \widetilde{X}_i \\ U_{i|i} \end{pmatrix} + \begin{pmatrix} \rho_w \\ \rho_v \end{pmatrix} \ .$$

7. Determine Q, S and R from the residuals as:

$$\begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} = \mathbf{E_j}[\begin{pmatrix} \rho_w \\ \rho_v \end{pmatrix} . \begin{pmatrix} \rho_w^T & \rho_v^T \end{pmatrix}] \ .$$

Figure 4.7 A schematic overview of the second combined deterministic-stochastic identification algorithm. This algorithm computes asymptotically biased solutions. See Section 6.1 for implementation issues. This algorithm has been implemented in the Matlab function com_stat.m.

role in the determination of B and D, it is better to recompute them. In this way, the matrices B and D that are determined afterwards will be more "compatible" with A and C.

2. A second alteration of Algorithm 1 consists of a different way of calculating A and C. Since Γ_i can be easily determined from Theorem 12, the matrices A and C can be extracted from Γ_i in any of the ways described on page 53. In this way one could for instance guarantee the stability of A [Mac 94]. In practice, it is often known in advance that the system is stable, and unstable models are thus very undesirable, especially when the system identification is running autonomous (for instance in an on-line application). Unfortunately (until now), it is still an open problem how stability of A can be guaranteed when it is calculated as described in algorithm 1. Once A and C are determined, a new Γ_i and Γ_{i-1} can be computed.

3. The accuracy of B and D tends to be bad in case of badly condition input Hankel matrices (see Section 6.4). We believe this is due to the introduction of a large correlation in the sample error when multiplying with $U_f^\dagger$ to determine $\mathcal{K}$. It is clearly better to avoid this step, and thus to compute B and D directly from (4.51) once the matrices A, C, Γ_i and Γ_{i-1} are determined. This can be done as follows (from (4.51)):

$$B, D = \arg \min_{B,D} \| \underbrace{\left(\frac{\Gamma_{i-1}^\dagger . \mathcal{Z}_{i+1}}{Y_{i|i}} \right) - \left(\frac{A}{C} \right) . \Gamma_i^\dagger . \mathcal{Z}_i}_{\text{known}} - \mathcal{K}(B, D) \, . \underbrace{U_f}_{\text{known}} \|_F^2 \, , \tag{4.55}$$

where $\mathcal{K}(B, D)$ refers to the linear matrix function defined by formula (4.52). Since $\mathcal{K}(B, D)$ is linear in B and D, the overall optimization problem is convex, and thus has a unique minimum. The problem can be solved by either using a non-linear optimization algorithm (which will always converge, due to the convexity), or by rewriting the function between the norm signs in (4.55) as an explicit linear combination of B and D and solving this set of equations in a least squares sense. We will pursue the second way. For simplicity of notation, we define the following matrices:

$$\begin{aligned} \mathcal{P} &\stackrel{\text{def}}{=} \left(\frac{\Gamma_{i-1}^\dagger . \mathcal{Z}_{i+1}}{Y_{i|i}} \right) - \left(\frac{A}{C} \right) . \Gamma_i^\dagger . \mathcal{Z}_i \quad \in \mathbb{R}^{(l+n) \times j} \, , \\ \mathcal{Q} &\stackrel{\text{def}}{=} U_f \quad \in \mathbb{R}^{mi \times j} \end{aligned} \tag{4.56}$$

$$= \begin{pmatrix} \mathcal{Q}_1 \\ \mathcal{Q}_2 \\ \vdots \\ \mathcal{Q}_i \end{pmatrix} ,$$

$$\mathcal{N}_1 \stackrel{\text{def}}{=} \underbrace{\begin{pmatrix} -\mathcal{L}_{1|1} & \dots & \mathcal{M}_{i-1} - \mathcal{L}_{1|i} \\ I_l - \mathcal{L}_{2|1} & \dots & -\mathcal{L}_{2|i} \end{pmatrix} \begin{pmatrix} I_l & 0 \\ 0 & \Gamma_{i-1} \end{pmatrix}}_{\in \mathbb{R}^{(l+n)\times(l+n)}} , \tag{4.57}$$

$$\mathcal{N}_2 \stackrel{\text{def}}{=} \underbrace{\begin{pmatrix} \mathcal{M}_1 - \mathcal{L}_{1|2} & \dots & \mathcal{M}_{i-1} - \mathcal{L}_{1|i} & 0 \\ -\mathcal{L}_{2|2} & \dots & -\mathcal{L}_{2|i} & 0 \end{pmatrix} \begin{pmatrix} I_l & 0 \\ 0 & \Gamma_{i-1} \end{pmatrix}}_{\in \mathbb{R}^{(l+n)\times(l+n)}} \tag{4.58}$$

$$\vdots$$

$$\mathcal{N}_i \stackrel{\text{def}}{=} \underbrace{\begin{pmatrix} \mathcal{M}_{i-1} - \mathcal{L}_{1|i} & 0 & \dots & 0 & 0 \\ -\mathcal{L}_{2|i} & 0 & \dots & 0 & 0 \end{pmatrix} \begin{pmatrix} I_l & 0 \\ 0 & \Gamma_{i-1} \end{pmatrix}}_{\in \mathbb{R}^{(l+n)\times(l+n)}} , \tag{4.59}$$

where $\mathcal{L}_{.|.}, \mathcal{M}_{.}$ were defined in the previous Subsection and $\mathcal{Q}_k \in \mathbb{R}^{m\times j}$. Note that the rows of the matrices $\mathcal{N}_k$ are rows of the matrix $\mathcal{N}$, which implies that the matrices $\mathcal{N}_k$ satisfy (from (4.53)):

$$\begin{pmatrix} \mathcal{K}_{1|k} \\ \mathcal{K}_{2|k} \end{pmatrix} = \mathcal{N}_k \, . \begin{pmatrix} D \\ B \end{pmatrix} .$$

Equation (4.55) can now be rewritten as:

$$B, D = \arg \min_{B,D} \| \mathcal{P} - \sum_{k=1}^{i} \mathcal{N}_k \begin{pmatrix} D \\ B \end{pmatrix} \mathcal{Q}_k \|_F^2 . \tag{4.60}$$

In the following, we will use the equality (see for instance [Bre 78]):

$$\text{vec}\,(AXB) = (B^T \otimes A).\text{vec}\, X ,$$

where $\otimes$ denotes the Kronecker product and vec A denotes the vector operation i.e. stacking the columns of A on top of each other in a vector. This last equality allows us to rewrite (4.60) as (applying the vector operation on the matrices inside the norm signs):

$$B, D = \arg \min_{B,D} \| \text{ vec}\, \mathcal{P} - \left[\sum_{k=1}^{i} \mathcal{Q}_k^T \otimes \mathcal{N}_k \right] . \text{ vec} \begin{pmatrix} D \\ B \end{pmatrix} \|_F^2 ,$$

which can now be solved using classical least squares regression as:

$$\text{vec}\begin{pmatrix} D \\ B \end{pmatrix} = \left[\sum_{k=1}^{i} \mathcal{Q}_k^T \otimes \mathcal{N}_k\right]^{\dagger} . \text{vec}\,\mathcal{P}\ . \tag{4.61}$$

The problem with this last equation is its size. The matrix that has to be (pseudo-) inverted is of size $j(l+n) \times m(l+n)$. It should be noted however that the dimension j can be reduced to $2i(l+m)$ by making use of the RQ decomposition as will be explained in Section 6.1. This reduces the size to $2i(l+n)(l+m) \times m(l+n)$. In the previous Subsection, we only had to invert a matrix (4.53) of size $i(l+n) \times (l+n)$. The difference between the two can become significant when there are a lot of inputs and/or outputs, since the increase in number of rows is equal to $2(l+m)$ and the increase in number of columns equals m. The improved accuracy is (in our opinion) more important though, and we will thus include this alteration in the robust algorithm at the end of this Section (see also Section 6.4 for some practical examples).

4. An alternative way (similar to the way mentioned in [McK 94a]) to determine B and D is by "going back to the data". Which means that the subspace ideas are only applied to determined A and C. The matrices B and D are then determined by classical least squares regression. Indeed, once A and C are determined, the overall problem becomes linear in the unknowns B and D. A major drawback of this solution will become clear in the Section 6.1: It is not possible any more to compute everything in function of the R factor of the overall RQ factorization, which thus implies an increased computational complexity.

Simulation error: See also [McK 94a]. The simulated output $\hat{y}_k$ can be determined as:

$$\begin{aligned} \hat{y}_k &= Du_k + \sum_{r=0}^{k-1} CA^{k-r-1}Bu_r \\ &= [u_k^T \otimes I_l]\ .\ \text{vec}\,D + (\sum_{r=0}^{k-1} u_r^T \otimes CA^{k-r-1})\ .\ \text{vec}\,B\ . \end{aligned}$$

With this last equation, we can find B and D as the solution of the following minimization criterion:

$$B, D = \arg\min_{B,D} \sum_{k=0}^{s} \left[y_k - [u_k^T \otimes I_l]\ .\ \text{vec}\,D - (\sum_{r=0}^{k-1} u_r^T \otimes CA^{k-r-1})\ .\ \text{vec}\,B \right]^2 ,$$

which is a linear regression problem in vec D and vec B. An initial state x_0 can be easily included.

Prediction Error: Alternatively, we can bring the Kalman gain K into account when estimating B and D as follows. First note that in Algorithm 1, B and D are not required to determine the Kalman gain K. Indeed, K can simply be determined from Q, S and R which are found from term 3 of (4.48). Once K is known, the prediction error minimization problem becomes linear in B and D (and x_0). With the steady state Kalman filter we can write the optimal *prediction* of the output y_k as (with B,D and x_0 unknown):

$$\begin{aligned}
\hat{y}_k &= C(A-KC)^k x_0 + Du_k + \sum_{r=0}^{k-1} C(A-KC)^{k-r-1}(B-KD)u_r \\
&\quad + \sum_{r=0}^{k-1} C(A-KC)^{k-r-1}Ky_r \\
&= y_k^K + M_k \begin{pmatrix} x_0 \\ \text{vec } D \\ \text{vec } B \end{pmatrix} ,
\end{aligned}$$

with:

$$\begin{aligned}
y_k^K &\stackrel{\text{def}}{=} \sum_{r=0}^{k-1} C(A-KC)^{k-r-1}Ky_r , \\
M_k &\stackrel{\text{def}}{=} \Big(\; C(A-KC)^k \;\Big|\; [u_k^T \otimes I_l \; - \; \textstyle\sum_{r=0}^{k-1} u_r^T \otimes C(A-KC)^{k-r-1}K] \;\Big| \\
&\qquad [\textstyle\sum_{r=0}^{k-1} u_r^T \otimes C(A-KC)^{k-r-1}] \; \Big) \; .
\end{aligned}$$

With this last equation, we can find B, D and x_0 as the solution of the following minimization criterion:

$$B, D, x_0 = \text{arg} \min_{B,D,x_0} \sum_{k=0}^{s} \left[y_k - y_k^K - M_k \begin{pmatrix} x_0 \\ \text{vec } D \\ \text{vec } B \end{pmatrix} \right]^2 , \qquad (4.62)$$

which is a linear regression problem in vec D, vec B and x_0.

4.4.4 Algorithm 3: a robust identification algorithm

Figure 4.8 contains a robust subspace algorithm for combined deterministic-stochastic subspace identification. This is the final algorithm we would like to present in this book. It has (to our experience) proven to work well on practical data (see Section 6.4), and contains several of the alterations mentioned in the previous Subsection.

Note that:

- The weights W_1 and W_2 have been chosen equal to respectively I_{li} and $\Pi_{U_f^\perp}$. Finding the optimal weights is still a topic of ongoing research.
- B and D could be determined through minimization of the prediction error (4.62). This would lead to a major increase in computational complexity. Furthermore, the algorithm as it is described in Figure 4.8 can be totally implemented by only making use of the R factor of the RQ decomposition. This will be illustrated in Section 6.1. Including the prediction criterion of (4.62) would imply that the not only the R factor is needed but also the original data.
- Finally, it should be noted that unfortunately much of the symmetry and simplicity of the deterministic and stochastic algorithms is lost. On the other hand the algorithm of Figure 4.8 *works* in all practical situations we considered (see Section 6.4).
- As indicated, a stable matrix A could be determined from Γ_i. Even though guaranteed stability [Mac 94] is a desirable feature in some cases, algorithms enforcing stability should be used with care. By guaranteeing the stability, even marginally stable systems (for instance systems containing an integrator or systems with lightly damped poles) and unstable systems will be identified as stable systems. Our experience is that for low order models, obtained from "linear" data, the identified systems are always stable (even without the guarantee), and the extra zeros introduced in Γ_i^0 to ensure stability only degenerate the result. On the other hand, for higher order systems, or for "non-linear" data, it is sometimes desirable to force the stability, since subspace algorithms tend to compute unstable systems in these cases. To ensure stability, steps 4 and 5 in Figure 4.8 should be changed to:

> **4.** Determine Γ_i and Γ_i^0 as:
>
> $$\Gamma_i = U_1 S_1^{1/2} \quad , \quad \Gamma_i^0 = \begin{pmatrix} \overline{\Gamma_i} \\ 0 \end{pmatrix} .$$
>
> **5.** Determine C as the first l rows of Γ_i and a stable A as $A = \Gamma_i^\dagger . \Gamma_i^0$. Recompute Γ_i and Γ_{i-1} from A and C.

We suggest to compute the A matrix without the guaranteed stability. When the system of the desired order turns out to be unstable, stability could be enforced as described above.

The Matlab function `subid.m` contains a Matlab implementation of this robust algorithm. See also Section 6.1 and Appendix B.

4.4.5 A simulation example

In this Subsection we investigate the three algorithms presented in this Section. In a simulation example, we investigate the asymptotic bias in function of the number of block rows i. For a comparison of the three algorithms on practical data we refer to Section 6.4. Consider the following first order system in forward innovation form:

$$\begin{aligned} x_{k+1} &= 0.949x_k + 1.8805u_k - 0.0580e^f_k \ , \\ y_k &= 0.8725x_k - 2.0895u_k + e^f_k \ , \end{aligned}$$

and:

$$\mathbf{E}[e^f_k.(e^f_k)^T] = 6.705 \ .$$

The input u_k is the same for all experiments and is equal to the sum of a filtered (cut off frequency equal to 0.05 times the Nyquist frequency, $T = 1$), zero mean, Gaussian, white noise sequence with variance 1 and a zero mean, Gaussian, white noise sequence with variance 0.01. The number of data used for the identification is fixed at $j = 5000$. One hundred different disturbances e^f_k were generated. For each of these sequences and for each i between 2 and 10, three models were identified using the three algorithms of Figures 4.6, 4.7 and 4.8. For algorithm 1 and 2, the weights were chosen as $W_1 = I_{li}, W_2 = I_j$. The sample mean of the eigenvalue (A) and the deterministic zero $(A - BD^{-1}C)$ was computed over the hundred experiments. The results are displayed in Figure 4.9.

The Matlab file `com_sim2.m` contains a Matlab implementation of this example.

4.5 CONNECTIONS TO THE PREVIOUS CHAPTERS

Since purely deterministic and purely stochastic identification are both special cases of the general combined identification problem, there must be some connections between the three Chapters. In this Section we explore these similarities.

Most of the similarities can be found in the main Theorems of the three Chapters. Theorem 2 for the deterministic case, Theorem 8 for the stochastic case and Theorem 12 for the combined case all look very similar. The differences are:

Robust combined algorithm:

1. Calculate the oblique and orthogonal projections:

$$\mathcal{O}_i = Y_f /_{U_f} \boldsymbol{W}_p \quad , \quad \mathcal{Z}_i = Y_f / \begin{pmatrix} \boldsymbol{W}_p \\ \boldsymbol{U}_f \end{pmatrix} \quad , \quad \mathcal{Z}_{i+1} = Y_f^- / \begin{pmatrix} \boldsymbol{W}_p^+ \\ \boldsymbol{U}_f^- \end{pmatrix} .$$

2. Calculate the **SVD** of the weighted oblique projection:

$$\mathcal{O}_i \Pi_{U_f^\perp} = U S V^T .$$

3. Determine the order by inspecting the singular values in S and partition the **SVD** accordingly to obtain U_1 and S_1.

4. Determine Γ_i and Γ_{i-1} as:

$$\Gamma_i = U_1 S_1^{1/2} \quad , \quad \Gamma_{i-1} = \underline{\Gamma_i} .$$

5. Solve the set of linear equations for A and C:

$$\left(\frac{\Gamma_{i-1}^\dagger . \mathcal{Z}_{i+1}}{Y_{i|i}} \right) = \left(\frac{A}{C} \right) . \Gamma_i^\dagger . \mathcal{Z}_i + \mathcal{K} . U_f + \left(\frac{\rho_w}{\rho_v} \right) .$$

   Recompute Γ_i and Γ_{i-1} from A and C.

6. Solve B and D from:

$$B, D = \arg \min_{B,D} \| \begin{pmatrix} \Gamma_{i-1}^\dagger . \mathcal{Z}_{i+1} \\ Y_{i|i} \end{pmatrix} - \begin{pmatrix} A \\ C \end{pmatrix} . \Gamma_i^\dagger . \mathcal{Z}_i \; - \; \mathcal{K}(B, D) \, . \, U_f \, \|_F^2 .$$

7. Finally, determine the covariance matrices Q, S and R as:

$$\begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} = \mathbf{E_j} [\begin{pmatrix} \rho_w \\ \rho_v \end{pmatrix} . \begin{pmatrix} \rho_w^T & \rho_v^T \end{pmatrix}] .$$

Figure 4.8 A schematic overview of a robust deterministic-stochastic identification algorithm. In Subsection 6.1.3 the implementation of this robust algorithm is discussed. This algorithm has been implemented in the Matlab function subid.m.

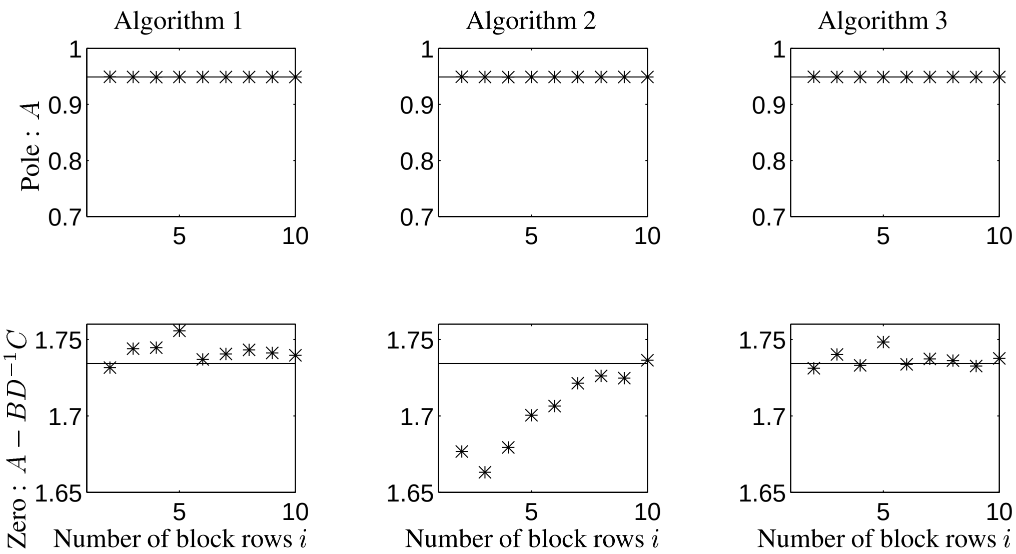

Figure 4.9 The stars show the expected value of the pole (row 1) and deterministic zero (row 2) in function of the number of block rows i. The fill lines show the exact value of pole and zero. Hundred experiments were performed, of which the sample average is plotted. Clearly, algorithm 1 and 3 are asymptotically unbiased, while algorithm 2 displays a clear bias on the zero of the system. This is because the input is a colored noise sequence, and the process and measurement noise is different from zero. Clearly, as the number of block rows i increases, the bias grows smaller. This Figure was generated using the Matlab file com_sim2.m.

- For the deterministic case, the number of measurements does not have to go to infinity. This is plausible, since when there is no noise, the system can be determined exactly from a finite number of data.
- For the stochastic case, W_p is replaced by Y_p. When we introduce zeros for the inputs (conceptually), and form the matrices W_p, Y_f and U_f as in the combined (or deterministic) case[7], we see that the combined Theorem 12 reduces to the stochastic Theorem 8.

[7] The projection $\Pi_{U_f^\perp}$ is equal to the identity and the oblique projection along U_f boils down to an orthogonal projection when $U_f \equiv 0$ (see Section 1.4).

As a conclusion, we can state that for all three cases (with zeros introduced for the stochastic case), we have:

$$\begin{aligned} \mathcal{O}_i &= Y_f /_{U_f} \boldsymbol{W}_p \\ &= \Gamma_i \widetilde{X}_i \ , \end{aligned}$$

with:

$$\begin{aligned} \widetilde{X}_i &= \widehat{X}_{i[\widehat{X}_0, P_0]} \ , \\ \widehat{X}_0 &= X_p^d /_{U_f} \boldsymbol{U}_p \ , \\ P_0 &= -[\Sigma^d - S^{xu}.(R^{uu})^\dagger.(S^{xu})^T] \ . \end{aligned}$$

For the deterministic case, these states $\widetilde{X}_i$ are equal to the exact deterministic states X_f^d. For the stochastic case, $\widehat{X}_0$ is equal to zero, just as P_0, which corresponds to Theorem 8 indeed. And finally for the combined identification problem, the above statements correspond exactly to Theorem 12. As a conclusion, we can state that the combined Theorem contains the deterministic and stochastic Theorem as special cases. In any subspace algorithm, it should thus be possible to recognize the oblique projection $\mathcal{O}_i$ as playing an important role.

Following from this, it can also be seen that the first deterministic algorithm the third stochastic algorithm and the second combined algorithm are virtually the same. All three algorithms compute two oblique projections, determine Γ_i from a singular value decomposition and solve the system matrices from a simple set of equations using the states. Unfortunately, as indicated in Section 4.4.2, the algorithm is asymptotically biased for the combined case for a finite number of block rows i.

The unbiased combined algorithm on the other hand (and also the robust combined algorithm), is similar to the second deterministic identification algorithm. It can also be shown to coincide with the third stochastic identification algorithm (when replacing zeros for the inputs). As a conclusion, we can state that the unbiased and robust combined algorithm will work well for all three cases, and boil down to the third stochastic algorithm for purely stochastic systems and to an algorithm similar to the second deterministic algorithm for the purely deterministic case.

Finally we note that it is also possible to model combined systems with the stochastic identification theory of Chapter 3. Therefore, the inputs *and* outputs of the combined process are considered as outputs of a stochastic process. After the identification, the stochastic model is converted to a combined deterministic-stochastic model. See for instance [DG 76] [NA 81] for more details.

4.6 CONCLUSIONS

In this Chapter we have treated the subspace identification of combined deterministic-stochastic systems. The concept of a bank of non-steady state Kalman filters has been extended from the stochastic to the combined case. As in the previous Chapters, it has been shown in two Theorems how the Kalman filter state sequence and the extended observability matrix can be extracted directly from input-output data. Published combined deterministic-stochastic identification algorithms have been shown to be special cases of the main Theorem. An asymptotically unbiased and a (simpler) asymptotically biased algorithm have been presented. Modifications and optimizations of the first algorithm have lead to a robust, practical algorithm. Finally, the connections between the three main Theorems of this book have been indicated.

5

STATE SPACE BASES AND MODEL REDUCTION

In this Chapter we describe how the state space basis of models identified with subspace identification algorithms can be determined. It is shown that this basis is determined by the input spectrum and by user defined input and output weights (the weights introduced in the three main Theorems of the previous Chapters).

The connections between subspace identification and frequency weighted balancing are explored in two main Theorems. The state space basis of the subspace identified models is shown to coincide with a frequency weighted balanced basis. The effects for reduced order model identification are elaborated on.

In the literature, the choice of the state space bases has been treated for the deterministic case in [MR 93] in which it is described how for purely deterministic cases a balanced state space basis can be obtained for $i \to \infty$. In [AK 90] the problem is treated for purely stochastic systems. In this Chapter, we show how the state space basis of combined deterministic-stochastic systems can be determined (see also [VODM 95b]). The results of [AK 90] and [MR 93] are just special cases of this general treatment.

We show that by a proper choice of the weighting matrices W_1 and W_2 in Theorem 12, the state space basis of the resulting model can be determined beforehand. The choice of the weighting matrices is illustrated with an example.

This Chapter is organized as follows. In Section 5.2 we introduce some notation. Section 5.3 introduces the concepts of frequency weighted balancing. In Section 5.4 the two main Theorems are presented. These Theorems connect the frequency weighted balancing result to the subspace identification result of Theorem 12. Section 5.5 addresses the problem of reduced order identification. Finally Section 5.6 contains a simulation example and Section 5.7 the conclusions.

5.1 INTRODUCTION

We consider the combined deterministic-stochastic identification as described in Figure 4.1. As was stated before, the state space matrices A, B, C, D, Q, S, R are only recovered up to within a similarity transformation. Alternatively, one could state that Γ_i and $\widetilde{X}_i$ are only determined up to within a non-singular similarity transformation $T \in \mathbb{R}^{n \times n}$:

$$\begin{aligned}\Gamma_i &\leftarrow \Gamma_i T \ , \\ \widetilde{X}_i &\leftarrow T^{-1}\widetilde{X}_i \ .\end{aligned}$$

Inspired by this ambiguity, the following question is raised: *In which state space basis are Γ_i and $\widetilde{X}_i$ determined when a subspace method is used to estimate them ?* We have already seen that this basis is a function of the weights W_1 and W_2 (Remark 5 on page 43 for Theorem 2 and similar remarks for Theorems 8 and 12), and that by a proper choice of these weights, the basis can be altered in a user controlled manner, the elements of which will be explored in this Chapter.

We show that in the framework of Enns [Enn 84], the state space basis corresponds to an input and output frequency weighted balanced basis. The weights are function of the input (u_k) applied to the system and of the weighting matrices W_1 and W_2 introduced in Theorem 12. By proper choice of these weighting matrices, the state space basis can be influenced in a frequency specific way. Furthermore, it will be shown that the singular values S_1 used to determine the system order have a clear interpretation from a linear system theoretical point of view.

We present two Theorems that link the state space basis of the identified model to a frequency weighted balanced basis. These Theorems introduce specific choices for the weighting matrices, such that the system is identified in a predefined state space basis. As special cases, we will investigate the algorithms of the literature [Lar 90] [VODM 93b] [VD 92] [VOWL 93] and we will show which state space basis is used by these algorithms.

A nice side result is the lower order identification problem. Since the basis in which the state space matrices are identified is well defined and is frequency weighted balanced, it is easy to truncate the model after identification to obtain a lower order model. This corresponds exactly to the technique of frequency weighted model reduction of Enns [Enn 84]. This observation allows for an easy and straightforward identification of lower order models directly from input-output data. The weighted H-infinity norm of the difference between the original model and the reduced order model can be approximated by two times the sum of the neglected weighted Hankel singular values (two times the sum of the "tail"). The usefulness of the Theorems will be illustrated by a simulation example.

5.2 NOTATION

Most of the notation will be drawn from the previous Chapter (Section 4.1.2). In this Section we only introduce the new notation.

The model

Throughout the Chapter, we consider the model (4.1)-(4.3) in its forward innovations form (see also Section 3.3) as:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Ee_k \ , \quad (5.1) \\ y_k &= Cx_k + Du_k + Fe_k \ , \quad (5.2) \end{aligned}$$

$$\mathbf{E}[e_p e_q^T] = I_l \ . \ \delta_{pq} \ ,$$

with $E \in \mathbb{R}^{n \times l}$, $F \in \mathbb{R}^{l \times l}$, and the innovations $e_k \in \mathbb{R}^l$. The relation between this model and the forward innovation model of Figure 3.4 is as follows:

$$\begin{aligned} F &= (\Lambda_0 - CPC^T)^{1/2} \ , \\ E &= K^f F \ . \end{aligned}$$

On the other hand, to go from this model to the model of Figure 3.4, the following transformations can be applied:

$$\begin{aligned} K^f &= EF^{-1} \ , \\ \mathbf{E}[e_p e_q^T] &= (FF^T) \ . \ \delta_{pq} \ . \end{aligned}$$

There is no real difference between the two representations. For the intuitive understanding of the concepts in this Chapter it is however easier to use (5.1)-(5.2).

Block Hankel matrices

For shorthand notation, we define (as before, see Subsection 2.1.2):

$$\begin{array}{rclcrcl} U_p & \stackrel{\text{def}}{=} & U_{0|i-1} & , & U_f & \stackrel{\text{def}}{=} & U_{i|2i-1} \ , \\ Y_p & \stackrel{\text{def}}{=} & Y_{0|i-1} & , & Y_f & \stackrel{\text{def}}{=} & Y_{i|2i-1} \ , \\ E_p & \stackrel{\text{def}}{=} & E_{0|i-1} & , & E_f & \stackrel{\text{def}}{=} & E_{i|2i-1} \ . \end{array}$$

As before, and somewhat loosely we denote U_p as the *past* inputs and U_f as the *future* inputs. Similarly we denote Y_p and Y_f as the past respectively the future outputs and E_p and E_f as the past and the future innovations. We assume that $j \to \infty$ throughout this Chapter. This ensures that we are in the *asymptotic* regime (in the sense that covariances can be replaced by sample covariances). As in the previous Chapters, the user-defined index i should be *large enough* (larger than the order n of the system). Even though i is not required to go to infinity for asymptotic consistency reasons as was explained in the previous Chapters, in this Chapter we will assume that it goes to infinity for simplicity.

The covariance matrix of the past inputs $R_p^{uu} \in \mathbb{R}^{mi \times mi}$ (as defined in Section 4.1.2) will play an important role in several derivations:

$$R_p^{uu} = \Phi_{[U_p,U_p]} \stackrel{\text{def}}{=} L_p^{uu}.(L_p^{uu})^T \ ,$$

where $L_p^{uu} \in \mathbb{R}^{mi \times mi}$ is a lower triangular square root of R_p^{uu} obtained e.g. via a Cholesky decomposition of R_p^{uu} or via a QR decomposition of U_p.

The covariance matrix of the past inputs projected on the future inputs will also play an important role:

$$\begin{aligned} R_{p^\perp}^{uu} &\stackrel{\text{def}}{=} \Phi_{[U_p/U_f^\perp, U_p/U_f^\perp]} \\ &\stackrel{\text{def}}{=} L_{p^\perp}^{uu}.(L_{p^\perp}^{uu})^T \ . \end{aligned}$$

System related matrices

The extended ($i > n$) observability matrix Γ_i and the reversed extended controllability matrix Δ_i^d were already defined in Section 2.1.2. The (reversed) stochastic extended controllability matrix is defined as:

$$\Delta_i^s \stackrel{\text{def}}{=} \begin{pmatrix} A^{i-1}E & A^{i-2}E & \dots & AE & E \end{pmatrix} \in \mathbb{R}^{n \times li} \ .$$

The block Toeplitz matrix H_i^d containing the deterministic Markov parameters has been defined in Section 2.1.2. We also define the block Toeplitz matrices containing the Markov parameters of the stochastic subsystem as:

$$H_i^s \stackrel{\text{def}}{=} \begin{pmatrix} F & 0 & \dots & 0 \\ CE & F & \dots & 0 \\ \dots & \dots & \dots & \dots \\ CA^{i-2}E & CA^{i-3}E & \dots & F \end{pmatrix} \in \mathbb{R}^{li \times li} .$$

The (non-steady state) Kalman filter state sequence $\widetilde{X}_i$ was defined in Theorem 12. The system (5.1)-(5.2) can be considered as a system with $m + l$ inputs (u_k and e_k) and l outputs (y_k). Just as for the purely deterministic case (see Chapter 2, equations (2.5)-(2.7)), we can write the Input-Output equations for (5.1)-(5.2) as:

$$\begin{aligned} Y_p &= \Gamma_i X_p + H_i^d U_p + H_i^s E_p , & (5.3)\\ Y_f &= \Gamma_i X_f + H_i^d U_f + H_i^s E_f , & (5.4)\\ X_f &= A^i X_p + \Delta_i^d U_p + \Delta_i^s E_p , & (5.5) \end{aligned}$$

with:

$$\begin{aligned} X_p &= \begin{pmatrix} x_0 & x_1 & \dots & x_{j-1} \end{pmatrix} , \\ X_f &= \begin{pmatrix} x_i & x_{i+1} & \dots & x_{i+j-1} \end{pmatrix} , \qquad (5.6) \end{aligned}$$

where the x_k's are the states of the forward innovations model (5.1)-(5.2). We will consider the asymptotic behavior of the subspace algorithms not only when the number of measurements goes to infinity ($j \to \infty$), but also when the number of block rows i goes to infinity. In that case, the bank of Kalman filters becomes a bank of steady-state Kalman filters and the following Lemma is applicable:

Lemma 1

For $i \to \infty$ and A asymptotically stable, the Kalman filter state sequence $\widetilde{X}_i$ can be rewritten as:

$$\widetilde{X}_i = \Delta_i^d U_p + \Delta_i^s E_p . \qquad (5.7)$$

Moreover, the sequence $\widetilde{X}_i$ becomes a steady state Kalman filter sequence and thus $\widetilde{X}_i \to X_f$, where X_f is the forward innovation state sequence (5.6).

The proof of the Lemma can be found in [VODM 95b].

Transfer and weighting functions

The $\mathcal{Z}$-transforms of u_k and y_k are denoted by respectively $U(z) \in \mathbb{C}^m$ and $Y(z) \in \mathbb{C}^l$, while the spectral factor of e_k is denoted by $E(z) \in \mathbb{C}^l$. From (5.1)-(5.2) we then find:

$$Y(z) = G(z)U(z) + H(z)E(z) \ ,$$

with

$$\begin{aligned} G(z) &= D + C(zI_n - A)^{-1}B \quad \in \mathbb{C}^{l \times m} \ , \\ H(z) &= F + C(zI_n - A)^{-1}E \quad \in \mathbb{C}^{l \times l} \ . \end{aligned}$$

The spectral factor of u_k is denoted by $S_u(z) \in \mathbb{C}^{m \times m}$: $U(z)U^T(z^{-1}) = S_u(z)S_u^T(z^{-1})$ with all poles of $S_u(z)$ and $S_u^{-1}(z)$ inside the unit circle. For instance when u_k is generated by sending white noise through a stable and inversely stable filter, then the spectral factor $S_u(z)$ is equal to the transfer matrix of this filter. The spectral factor $S_u(z)$ contains information about the energy distribution of u_k in the frequency domain. We define the input and output weighting matrix functions as:

$$\begin{aligned} W_u(z) &\stackrel{\text{def}}{=} D_u + C_u(zI_{n_u} - A_u)^{-1}B_u \quad \in \mathbb{C}^{m \times m} \ , \\ W_y(z) &\stackrel{\text{def}}{=} D_y + C_y(zI_{n_y} - A_y)^{-1}B_y \quad \in \mathbb{C}^{l \times l} \ . \end{aligned}$$

From the Markov parameters of these weighting transfer matrices, the weighting matrices W_u and W_y can be formed:

$$W_u \stackrel{\text{def}}{=} \begin{pmatrix} D_u & 0 & 0 & \dots & 0 \\ C_uB_u & D_u & 0 & \dots & 0 \\ C_uA_uB_u & C_uB_u & D_u & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ C_uA_u^{i-2}B_u & C_uA_u^{i-3}B_u & C_uA_u^{i-4}B_u & \dots & D_u \end{pmatrix} \in \mathbb{R}^{mi \times mi} \ ,$$

$$W_y \stackrel{\text{def}}{=} \begin{pmatrix} D_y & 0 & 0 & \dots & 0 \\ C_yB_y & D_y & 0 & \dots & 0 \\ C_yA_yB_y & C_yB_y & D_y & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ C_yA_y^{i-2}B_y & C_yA_y^{i-3}B_y & C_yA_y^{i-4}B_y & \dots & D_y \end{pmatrix} \in \mathbb{R}^{li \times li} \ .$$

The distinction between the matrix W_u and the transfer matrix $W_u(z)$ should be clear from the complex argument z and the context.

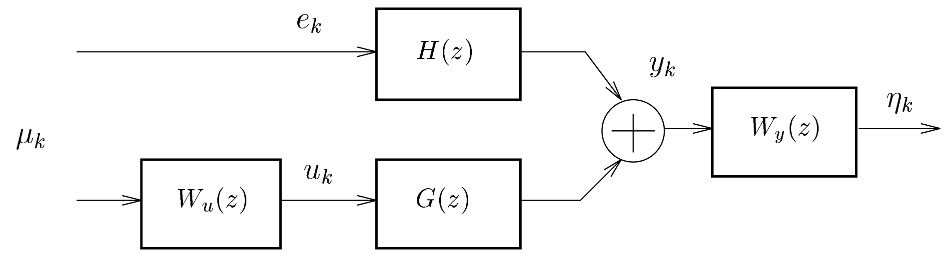

Figure 5.1 Cascade system used for the interpretation of frequency weighted balancing. The weights $W_u(z)$ and $W_y(z)$ are user defined. Note that the noise input e_k has no extra weight, since from an input-output point of view, this weight is indistinguishable from $H(z)$.

5.3 FREQUENCY WEIGHTED BALANCING

In this Section we recall the results of Enns [Enn 84] for frequency weighted balancing. We also show how the frequency weighted Grammians introduced by Enns can be calculated from the extended observability and controllability matrices and from the weighting matrices.

Well known in system theory is the notion of *balanced realization* [Moo 81]. Enns [Enn 84] developed a frequency weighted extension of this result. The idea is that input and output frequency weights can be introduced as to enhance certain frequency bands in the balancing procedure.

The reasoning of Enns goes as follows. Consider the input weighting function $W_u(z)$ and the output weighting function $W_y(z)$, and put these functions in series with the transfer function of the original system (5.1)-(5.2) (see also Figure 5.1). We call the combined input of this weighted system μ_k (the input of $W_u(z)$ combined with e_k) and the output of the weighted system η_k. Enns now considers the two following (dual) questions:

1. What set of states could be part of the state-response to some input μ_k (with $\|\mu_k\|_2 \leq 1$); with zero initial conditions for the states of the system and of the input weighting $W_u(z)$. We call this set $\mathcal{X}[W_u(z)]$.
2. What set of initial states could produce an output η_k (with $\|\eta_k\|_2 \leq 1$); with zero input and zero initial conditions for the states of the output weighting $W_y(z)$. This set will be denoted by $\mathcal{X}[W_y(z)]$.

The solution to these two questions is given by means of the frequency weighted controllability and observability Grammians.

Definition 8 Frequency Weighted Controllability Grammian

The solution P_{11} of the Lyapunov equation:

$$\begin{aligned} & \begin{pmatrix} A & BC_u \\ 0 & A_u \end{pmatrix} \begin{pmatrix} P_{11} & P_{21}^T \\ P_{21} & P_{22} \end{pmatrix} \begin{pmatrix} A & BC_u \\ 0 & A_u \end{pmatrix}^T \\ + & \begin{pmatrix} BD_u & E \\ B_u & 0 \end{pmatrix} \begin{pmatrix} BD_u & E \\ B_u & 0 \end{pmatrix}^T = \begin{pmatrix} P_{11} & P_{21}^T \\ P_{21} & P_{22} \end{pmatrix} \end{aligned} \tag{5.8}$$

is called the $W_u(z)$ weighted controllability Grammian and is denoted by:

$$P[W_u(z)] \stackrel{def}{=} P_{11} \ .$$

Note that the weight of the innovations e_k is always taken equal to I_l (see also Figure 5.1). We thus should write $P[W_u(z), I_l]$ but this would make the notation more complicated.

Definition 9 Frequency Weighted Observability Grammian

The solution Q_{11} of the Lyapunov equation:

$$\begin{aligned} & \begin{pmatrix} A & 0 \\ B_y C & A_y \end{pmatrix}^T \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{pmatrix} \begin{pmatrix} A & 0 \\ B_y C & A_y \end{pmatrix} \\ + & \begin{pmatrix} D_y C & C_y \end{pmatrix}^T \begin{pmatrix} D_y C & C_y \end{pmatrix} = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{pmatrix} \end{aligned} \tag{5.9}$$

is called the $W_y(z)$ weighted observability Grammian and is denoted by:

$$Q[W_y(z)] \stackrel{def}{=} Q_{11} \ .$$

Enns provided the answer to the questions raised above using these weighted Grammians. The sets $\mathcal{X}[W_u(z)]$ and $\mathcal{X}[W_y(z)]$ can be described by two ellipsoids as:

$$\begin{aligned} \mathcal{X}[W_u(z)] &= \{ \quad x \in \mathbb{R}^{n\times 1} \quad \text{such that} \quad x^T (P[W_u(z)])^{-1} x \ \leq 1 \quad \} \ , \\ \mathcal{X}[W_y(z)] &= \{ \quad x \in \mathbb{R}^{n\times 1} \quad \text{such that} \quad x^T Q[W_y(z)] x \ \leq 1 \quad \} \ . \end{aligned}$$

The two weighted Grammians $P[W_u(z)]$ and $Q[W_y(z)]$ determine the state space basis uniquely. Just as for the classical balancing procedure, a similarity transformation can be found that makes both weighted Grammians diagonal and equal to each other. In that case the system is said to be *Frequency Weighted Balanced*.

Definition 10 Frequency Weighted Balancing

The system (5.1)-(5.2) is called $[W_u(z), W_y(z)]$ *frequency weighted balanced when:*

$$P[W_u(z)] = Q[W_y(z)] = \Sigma \ ,$$

where $\Sigma = \text{diagonal}[\sigma_1, \sigma_2, \dots, \sigma_n]$*. The diagonal elements* σ_k *are called the frequency weighted Hankel singular values, and will generally be denoted by:*

$$\sigma_k[W_u(z), W_y(z)] \ .$$

Even though (5.8) and (5.9) are easily solvable for $P[W_u(z)]$ and $Q[W_y(z)]$, we present a different way to compute these weighted Grammians. These expressions will enable us to make the connection between subspace identification and frequency weighted balancing.

Lemma 2

With A *asymptotically stable and* $i \to \infty$*, we have:*

$$\begin{aligned} P[W_u(z)] &= \Delta_i^d.[W_u W_u^T].(\Delta_i^d)^T + \Delta_i^s(\Delta_i^s)^T \ , &(5.10)\\ Q[W_y(z)] &= \Gamma_i^T.[W_y^T W_y].\Gamma_i \ . &(5.11)\end{aligned}$$

A proof can be found in Appendix A.8. Fixing $P[W_u(z)]$ and $Q[W_y(z)]$ is equivalent to fixing the state space basis of the model. In the next Section we show how the weighted controllability and observability Grammians corresponding to the state space basis of Γ_i and $\widetilde{X}_i$ can be determined from the weights W_1 and W_2.

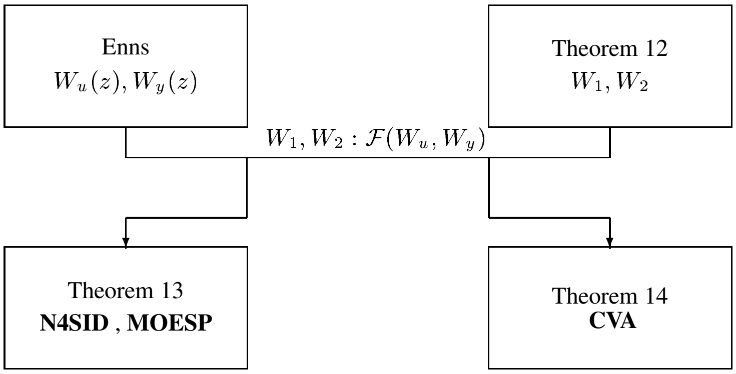

Figure 5.2 Interplay between different Theorems. In Chapter 4 we introduced the main combined deterministic-stochastic Theorem 12. This Theorem involved two weighting matrices W_1 and W_2. The theory of Enns on the other hand involves two weighting matrices $W_u(z)$ and $W_y(z)$, with which we can relate two matrices of Markov parameters W_u and W_y. In the Theorems of this Chapter (Theorem 13, 14), we described how to choose W_1 and W_2 as a function of the matrices W_u and W_y so that the basis of the identified system matrices can be interpreted in Enns's framework. We present two Theorems, since the first Theorem 13 encompasses the special cases of **N4SID** and **MOESP** (and balanced), while the second Theorem 14 has as a special case the canonical variate algorithm **CVA**.

5.4 SUBSPACE IDENTIFICATION AND FREQUENCY WEIGHTED BALANCING

In this Section we consider two Theorems that connect the results of frequency weighted balancing to the results of subspace identification. We show in these two Theorems how the weights W_1 and W_2 influence the Grammians $P[W_u(z)]$ and $Q[W_y(z)]$ corresponding to the state space basis of Γ_i and $\widetilde{X}_i$.

The two Theorems presented in this Section introduce certain choices of the weighting matrices W_1 and W_2 (see Theorem 12) that lead to a Γ_i and $\widetilde{X}_i$ corresponding to a frequency weighted balanced basis. The reason why there are two Theorems and not just one is because the **N4SID** [VODM 93b] and **MOESP** [VD 92] algorithm fit in the framework of the first Theorem while the Canonical Variate Analysis algorithm (**CVA**) [Lar 90] fits into the framework of the second Theorem. In order to cover the class of *all* published combined deterministic-stochastic subspace identification algorithms, we present both Theorems. See also Figure 5.2 and Table 5.1.

	Th.	Section	W_1	W_2	W_u	W_y	$W_u(z)$	$W_y(z)$
N4SID	13	5.4.2	(5.12)	(5.13)	L_p^{uu}	I_{li}	$S_u(z)$	$I_l(z)$
MOESP	13	5.4.2	(5.12)	(5.14)	$L_{p^\perp}^{uu}$	I_{li}	$S_u(z)$ (wn)	$I_l(z)$
Balanced	13	5.4.2	(5.12)	(5.13)	I_{mi}	I_{li}	$I_m(z)$	$I_l(z)$
CVA	14	5.4.4	(5.17)	(5.18)	$L_{p^\perp}^{uu}$	$-$	$S_u(z)$ (wn)	$H^{-1}(z)$

Table 5.1 Overview of the interplay between W_1, W_2 and W_u, W_y and $W_u(z), W_y(z)$. The abbreviation (wn) means that that result is only valid for white noise input signals.

5.4.1 Main Theorem 1

Theorem 13 - Main Theorem 1

Under the conditions of Theorem 12 and with A asymptotically stable and $i \to \infty$, we have with:

$$W_1 = W_y \; , \tag{5.12}$$

$$W_2 = U_p^T.(R_p^{uu})^{-1}.W_u.(L_p^{uu})^{-1}.U_p + \Pi_{U_p^\perp} \; , \tag{5.13}$$

or

$$W_2 = (U_p/\boldsymbol{U}_f^\perp)^T.(R_{p^\perp}^{uu})^{-1}.W_u.(L_{p^\perp}^{uu})^{-1}.U_p/\boldsymbol{U}_f^\perp + \Pi_{U_p^\perp, U_f^\perp} \; , \tag{5.14}$$

that the $W_u(z)$ weighted controllability Grammian and $W_y(z)$ weighted observability Grammian of the state space basis corresponding to Γ_i and $\widetilde{X}_i$ are given by (with S_1 from equation (4.25)):

$$P[W_u(z)] = S_1 \; , \tag{5.15}$$

$$Q[W_y(z)] = S_1 \; . \tag{5.16}$$

A proof of the Theorem can be found in Appendix A.9.

The Theorem implies that the state space basis of Γ_i and $\widetilde{X}_i$ for the choice of W_1 and W_2 given by (5.12)-(5.13) or (5.12)-(5.14) is the $[W_u(z), W_y(z)]$ frequency weighted balanced basis. It also implies that the singular values S_1, which are determined directly from the data, are the $[W_u(z), W_y(z)]$ frequency weighted Hankel singular values. The consequences of this Theorem for reduced order identification will be further elaborated on in Section 5.5.

There are two possible choices for the weighting matrix W_2 ((5.13) and (5.14)) since the **N4SID** algorithm is a special case of the first choice (5.13), while the **MOESP** algorithm is a special case of the second choice (5.14), as will be discussed in the next Subsection.

5.4.2 Special cases of the first main Theorem

Even though the weighting matrices W_u and W_y in the first main Theorem can be chosen arbitrarily, there are some special cases that lead to algorithms published in the literature.

N4SID

Corollary 2 - N4SID

The **N4SID** *algorithm described in Subsection 4.3.1 corresponds to the following choice of weighting matrices in Theorem 13 (use formula (5.12) and (5.13)):*

$$\begin{aligned} W_u &= L_p^{uu} \ , \\ W_y &= I_{li} \ . \end{aligned}$$

A proof of the Corollary can be found in Appendix A.10. The proof shows that for this special choice of weights W_u and W_y, the weighting matrices $W_1 = I_{li}$ and $W_2 = I_j$ are recovered.

It is easy to verify that (for $i \to \infty$) the lower triangular matrix L_p^{uu} corresponds to the Toeplitz matrix generated by the Markov parameters of the spectral factor $S_u(z)$ of u_k. This implies that the input weight $W_u(z)$ in the balancing procedure corresponds to the spectral factor $S_u(z)$.

MOESP

Corollary 3 - MOESP

The **MOESP** *algorithm described in Subsection 4.3.2 corresponds to the following choice of weighting matrices in Theorem 13 (use formula (5.12) and (5.14)):*

$$\begin{aligned} W_u &= L_{p^\perp}^{uu} \ , \\ W_y &= I_{li} \ . \end{aligned}$$

A proof of the Corollary is straightforward (Appendix A.10). The proof shows that for this special choice of weights W_u and W_y, the weighting matrices $W_1 = I_{li}$ and $W_2 = \Pi_{U_f^\perp}$ are recovered.

Unfortunately, the lower triangular matrix $L_{p^\perp}^{uu}$ does not correspond to the Toeplitz matrix generated by the Markov parameters of the spectral factor $S_u(z)$ of u_k ($L_{p^\perp}^{uu}$ is not equal to any Toeplitz matrix in the general case). Only for a white noise input (where $L_p^{uu} = L_{p^\perp}^{uu} = I_{li}$) can we state that the matrix $L_{p^\perp}^{uu}$ contains the Markov parameters of $S_u(z)$. The interpretation of the state space basis for the **MOESP** algorithm with colored noise inputs is still an open problem.

Balanced Realization

With the weighting matrices $W_u = I_{mi}$, $W_y = I_{li}$ we find (use formula (5.12) and (5.13)):

$$\begin{aligned} P[I_m(z)] &= S_1 \ , \\ Q[I_l(z)] &= S_1 \ . \end{aligned}$$

Now it is easy to verify that $P[I_m(z)]$ and $Q[I_l(z)]$ are equal to the unweighted controllability respectively observability Grammian. This implies that the basis of Γ_i and $\widetilde{X}_i$ is the classical balanced basis as described in [Moo 81]. A similar result for purely deterministic systems had been obtained in [MR 93]. It should be noted that the result above also holds for purely deterministic systems, and in that sense this is an extension of the result of [MR 93].

5.4.3 Main Theorem 2

Theorem 14 - Main Theorem 2

Under the conditions of Theorem 12 and with A asymptotically stable and $i \to \infty$ we have with:

$$\begin{aligned} W_1^T W_1 &= \Phi^{-1}_{[(Y_f W_2),(Y_f W_2)]} \ , && (5.17) \\ W_2 &= (U_p/\boldsymbol{U}_f^\perp)^T.(R_{p^\perp}^{uu})^{-1}.W_u.(L_{p^\perp}^{uu})^{-1}.U_p/\boldsymbol{U}_f^\perp + \Pi_{U_p^\perp, U_f^\perp} \ , && (5.18) \end{aligned}$$

that the $W_u(z)$ weighted controllability Grammian and $H^{-1}(z)$ weighted observability Grammian of the state space basis corresponding to Γ_i and $\widetilde{X}_i$ are given by:

$$\begin{aligned} P[W_u(z)] &= S_1 \ , && (5.19) \\ Q[H^{-1}(z)] &= S_1(I_n - S_1^2)^{-1} \ . && (5.20) \end{aligned}$$

A proof of the Theorem can be found in Appendix A.11. The Theorem states that both $P[W_u(z)]$ and $Q[H^{-1}(z)]$ are diagonal (not equal). It thus implies that the state space basis of Γ_i and $\widetilde{X}_i$ is (within a diagonal scaling of the states) the $[W_u(z), H^{-1}(z)]$ frequency weighted balanced basis. It also implies that the $[W_u(z), H^{-1}(z)]$ frequency weighted Hankel singular values can be expressed as (with σ_k the elements of S_1):

$$\sigma_k[W_u(z), H^{-1}(z)] = \frac{\sigma_k}{\sqrt{1-\sigma_k^2}} \; .$$

The last formula is very suggestive since it only makes sense when $\sigma_k < 1$. It can indeed be proven (see [VODM 95b]) that the diagonal elements of S_1 are the cosines of the principal angles between the row spaces of $W_p.W_2$ and $Y_f.W_2$ (where W_2 is given as in Theorem 14):

$$S_1 = [W_p.W_2 \sphericalangle Y_f.W_2] \; .$$

From the fact that these cosines are always smaller than one, we can conclude that for the specific choice of weights of Theorem 14, the singular values in S_1 are always smaller than one.

The consequences of this Theorem for reduced order identification will be further elaborated on in Section 5.5.

5.4.4 Special cases of the second main Theorem

Corollary 4 - CVA

The canonical variate analysis of Larimore [Lar 90] (see Subsection 4.3.3) corresponds to a the choice of weighting matrix $W_u = L^{uu}_{p^\perp}$ in Theorem 14.

The proof of this Corollary can also be found in [VODM 95b]. Just as for the **MOESP** algorithm, we can state that when the input is white, the basis in which the **CVA** method determines its state space model is the $[H^{-1}(z), S_u(z)]$ weighted balanced basis. The interpretation of the state space basis for the **CVA** algorithm with colored noise inputs is still an open problem.

5.4.5 Connections between the main Theorems

Even though the two Theorems are basically different, it is possible to mimic the results of the second main Theorem by using the appropriate weights in the first main Theorem.

From (5.4) and Lemma 1 it is easy to show that for $i \to \infty$ and A asymptotically stable, we have:

$$\mathcal{R} \stackrel{\text{def}}{=} Y_f / \begin{pmatrix} \boldsymbol{W}_p \\ \boldsymbol{U}_f \end{pmatrix}^{\perp} = H_i^s E_f \ .$$

Knowing this, we can take the weighting matrices of the first main Theorem equal to (use formula (5.12) and (5.13) or (5.14)):

$$\begin{aligned} W_u &= W_u \ , \\ W_y^T W_y &= [\mathcal{R}\mathcal{R}^T]^{-1} \ . \end{aligned}$$

The first equation (which is trivial) states that W_u can be chosen arbitrarily. We then find straightforwardly that:

$$P[W_u(z)] = S_1 \ , \tag{5.21}$$

$$Q[H^{-1}(z)] = S_1 \ . \tag{5.22}$$

If we compare (5.19)-(5.20) with (5.21)-(5.22) we find that both results are similar. Thus the application of these specific weights to the first main Theorem leads to a model in (almost) the same basis as when we had used the weights of the second main Theorem. The only difference is a diagonal scaling of the basis and the different singular values S_1 that are calculated. It is easy to verify that for this specific choice of weights, and with $S_1[1]$ and $S_1[2]$ the singular values calculated by respectively the first and the second main Theorem, we have for this specific choice of weights:

$$(S_1[2])^2 = (S_1[1])^2 (I_n - (S_1[1])^2)^{-1} \ .$$

5.5 CONSEQUENCES FOR REDUCED ORDER IDENTIFICATION

In this Section we apply the results of the two main Theorems to the identification of lower order systems. The connections with frequency weighted model reduction are exploited.

5.5.1 Error bounds for truncated models

An important consequence of the two main Theorems is the analysis of reduced order identification. As has been proven in this Chapter, subspace identification of a model of order n (the exact state space order) leads to a state space system that is $[W_u(z), W_y(z)]$ frequency weighted balanced for the first main Theorem and $[W_u(z), H^{-1}(z)]$ frequency weighted balanced for the second main Theorem. This nth order model can then be easily reduced to a model of lower order r by truncating it as follows:

$$A = \begin{array}{c c} & \begin{array}{cc} r & n-r \end{array} \\ \begin{array}{c} r \\ n-r \end{array} & \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \end{array} \quad , \quad B = \begin{array}{c c} & m \\ \begin{array}{c} r \\ n-r \end{array} & \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} \end{array} ,$$

$$C = \begin{array}{c c} & \begin{array}{cc} r & n-r \end{array} \\ l & \begin{pmatrix} C_1 & C_2 \end{pmatrix} \end{array} \quad , \quad E = \begin{array}{c c} & l \\ \begin{array}{c} r \\ n-r \end{array} & \begin{pmatrix} E_1 \\ E_2 \end{pmatrix} \end{array} .$$

The reduced order model is described by the matrices: $A_{11}, B_1, C_1, D, E_1, F$. The reduced transfer functions are denoted by:

$$\begin{aligned} \widehat{G}(z) &= D + C_1(zI_r - A_{11})^{-1}B_1 \ , \\ \widehat{H}(z) &= F + C_1(zI_r - A_{11})^{-1}E_1 \ . \end{aligned}$$

Enns [Enn 84] now suggested the following conjecture in his thesis:

Conjecture 1 Enns's conjecture

When truncating a $[W_u(z), W_y(z)]$ frequency balanced system, the infinity norm of the weighted difference between the original and the reduced system can be upperbounded by the neglected weighted Hankel singular values. In the framework of this Chapter (see also Figure 5.1), this conjecture becomes:

$$\begin{aligned} &\| \ W_y(z) \ [\ G(z) - \widehat{G}(z) \ ] \ W_u(z) \quad | \quad W_y(z) \ [\ H(z) - \widehat{H}(z) \ ] \ \|_\infty \\ &\leq 2 \sum_{k=r+1}^{n} \sigma_k[W_u(z), W_y(z)].(1+\alpha) \quad , \quad \alpha > 0 \ , \end{aligned} \tag{5.23}$$

where α is small.

Let us pounder a bit about this conjecture. We have tried to find a simple expression for an upper bound on α, but didn't succeed (as didn't anyone else as far as we know, even though the problem has been open since 1984). Even though the result is ambiguous (α has never been proven to be bounded, let alone to be small), the *heuristic* model reduction technique seems to work very well in practice (see for instance [AM 89] [WB 92]). In practice it turns out that, even though not a real upper bound, two times the sum of the neglected singular values gives a good indication about the size of the error. In what follows, we will *loosely* state result (5.23) as:

$$\| \; W_y(z) \; [\; G(z) - \widehat{G}(z) \;] \; W_u(z) \quad | \quad W_y(z) \; [\; H(z) - \widehat{H}(z) \;] \; \|_\infty$$
$$\simeq 2 \sum_{k=r+1}^{n} \sigma_k [W_u(z), W_y(z)] \; . \tag{5.24}$$

This result very much resembles the result of [AF 87] were the truncation error is given by two times the sum of the neglected Hankel singular values (two times the "tail"). We will now apply this to the two main Theorems of this Chapter. We denote the diagonal elements of S_1 by σ_k.

For Theorem 13:

$$\| \; W_y(z) \; [\; G(z) - \widehat{G}(z) \;] \; W_u(z) \quad | \quad W_y(z) \; [\; H(z) - \widehat{H}(z) \;] \; \|_\infty \simeq 2 \sum_{k=r+1}^{n} \sigma_k \; .$$

For Theorem 14:

$$\| \; H^{-1}(z) \; [\; G(z) - \widehat{G}(z) \;] \; W_u(z) \quad | \quad H^{-1}(z) \; [\; H(z) - \widehat{H}(z) \;] \; \|_\infty$$
$$\simeq 2 \sum_{k=r+1}^{n} \frac{\sigma_k}{\sqrt{1-\sigma_k^2}} = 2 \sum_{k=r+1}^{n} \text{cotan}(\theta_k) \; ,$$

where cotan(θ_k) is the co-tangent of the principal angle θ_k. As indicated before, these values do not give "hard" bounds, but give an indication of the weighted reduction error induced by reducing the nth order model to a model of order r. More importantly, these equations state that the fit of the truncated lower order model will be good where $W_u(z)$ and $W_y(z)$ (or $H^{-1}(z)$) are large. This implies that by a proper choice of $W_u(z)$ and $W_y(z)$ the distribution of the error in the frequency domain can be shaped. We find for the special cases:

N4SID

$$\| \; [\; G(z) - \widehat{G}(z) \;] \; S_u(z) \quad | \quad [\; H(z) - \widehat{H}(z) \;] \; \|_\infty \simeq 2 \sum_{k=r+1}^{n} \sigma_k \; .$$

We can conclude that the error of the model will be small where the frequency content of the input is large. This is a very intuitive result: much input energy in a certain frequency band leads to an accurate model in that band. Also note that the error on the noise model can not be shaped by the user.

MOESP (white noise inputs)

$$\| \, [\, G(z) - \widehat{G}(z) \,] \quad | \quad [\, H(z) - \widehat{H}(z) \,] \, \|_\infty \simeq 2 \sum_{k=r+1}^{n} \sigma_k \; .$$

This result is restricted to white noise inputs (as indicated in Subsection 5.4.2).

Balanced basis

$$\| \, [\, G(z) - \widehat{G}(z) \,] \quad | \quad [\, H(z) - \widehat{H}(z) \,] \, \|_\infty \leq 2 \sum_{k=r+1}^{n} \sigma_k \; .$$

Note that for this case the less than or equal sign is justified, since for (unweighted) balanced model reduction two times the sum of the Hankel singular values is really an upper bound for the truncation error (see for instance [AF 87]).

CVA (white noise inputs)

$$\| \, H^{-1}(z) \, [\, G(z) - \widehat{G}(z) \,] \quad | \quad H^{-1}(z) \, [\, H(z) - \widehat{H}(z) \,] \, \|_\infty$$
$$\simeq 2 \sum_{k=r+1}^{n} \frac{\sigma_k}{\sqrt{1 - \sigma_k^2}} \; .$$

We can conclude that the error will be small in frequency bands where the input to noise ratio is large (the white noise input has a flat spectrum equal to 1). Also note that the error on the noise model is a relative error. Again, this result is restricted to white inputs (see Subsection 5.4.4).

Remarks & comments

- For the purely stochastic case ($G(z) \equiv 0$), we see that the result for **N4SID** reduces to:

  $$\|H(z) - \widehat{H}(z)\|_\infty \simeq 2 \sum_{k=r+1}^{n} \sigma_k \; .$$

  For $G(z) \equiv 0$ we find for **N4SID** that $P[I(z)] = Q[I(z)] = S_1$, where $P[I(z)]$ indicates the controllability Grammian of the forward innovation model (since the deterministic part is equal to zero). This means that the forward innovation model is balanced (in the regular sense [Moo 81]). Indeed, **N4SID** reduces for the stochastic case to the **UPC** algorithm of Subsection 3.3.2 (cfr. the weightings $W_1 = I_{li}, W_2 = I_j$), and as was already mentioned in that Subsection, Arun & Kung [AK 90] showed that the forward innovation model is balanced in the deterministic sense. This also implies that we can write (exactly, see the results of [AF 87]):

$$\|H(z) - \widehat{H}(z)\|_\infty \leq 2 \sum_{k=r+1}^{n} \sigma_k \ .$$

- For the purely stochastic case ($G(z) \equiv 0$), we see that the result for **CVA** reduces to:

$$\|H^{-1}(z)[H(z) - \widehat{H}(z)]\|_\infty \simeq 2 \sum_{k=r+1}^{n} \frac{\sigma_k}{\sqrt{1-\sigma_k^2}} \ . \tag{5.25}$$

In this case, the weights of Theorem 14 reduce to $W_1 = \Phi^{-1/2}_{[Y_f,Y_f]}, W_2 = I_j$, which are indeed the weights for the stochastic **CVA** as described in Subsection 3.3.3. This implies that the combined **CVA** of Subsection 4.3.3 reduces to the stochastic **CVA** of Subsection 3.3.3 in this case. Now, we know also that for the stochastic **CVA**, the stochastic model is "stochastically balanced" (see Definition 6). In [WS 91] the following result is derived for stochastically balanced models:

$$\|H^{-1}(z)[H(z) - \widehat{H}(z)]\|_\infty \leq 2 \sum_{k=r+1}^{n} \frac{\sigma_k}{1-\sigma_k} \ . \tag{5.26}$$

Even though there is a resembles between (5.25) and (5.26), they are not equal. First, it should be noted that the expression in (5.25) is smaller than the expression in (5.26) (for $\sigma_k < 1$, which is the case). This means that the bound given by Enns is too small (recall the factor α). However, the difference between the two expressions is smaller than 10% for $\sigma_k < 0.3$. Or in other words, when the angles that were discarded are larger than 70 degrees, the unknown alpha factor accounts for about 10%. More results on the connections between stochastic balancing and Enns's theory can be found in [SDM 94].

5.5.2 Reduced order identification

The H-infinity norm bounds stated above were derived for the case where first the nth order model was identified, after which it was truncated to a model of order r (we will call this the *truncated* lower order model, see Figure 5.3). It is very tempting to extrapolate these results to the case where the rth order model is identified directly by the subspace algorithm, without first identifying the nth order model. This can easily be done by truncating U_1, S_1 and V_1 (in Theorem 12) to an $\mathbb{R}^{li\times r}$, $\mathbb{R}^{r\times r}$ and $\mathbb{R}^{j\times r}$ matrix respectively. The resulting rth order model depends on the exact details of how the system matrices are extracted from Γ_i and $\widetilde{X}_i$ (we will call this the *identified* lower order model, see Figure 5.3). The identified lower order model and the truncated lower

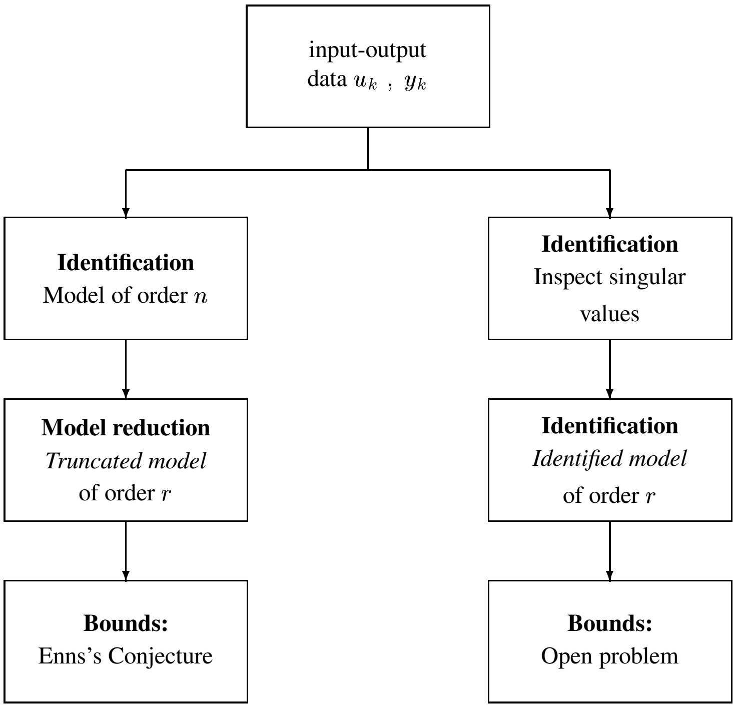

Figure 5.3 The left hand side of the Figure illustrates how first an nth order model is obtained through identification, after which it is truncated to obtain the *truncated* lower order model. The right hand side illustrates how the reduction can be done in the identification step. This leads to the *identified* lower order model.

order model will be different (in general). However, we experienced from simulations that both models are "close". The H-infinity norm results do not hold any more now (note that in practical situation they would never hold, since $i, j \neq \infty$ in practice), but the norms can be used as guidelines for the choice of $W_u(z)$ and $W_y(z)$ and as a crude estimate of the size of the errors. This will be further illustrated by an example in Section 5.6.

We conclude this Section with the remark that the implementation of the algorithms with the weights W_1 and W_2 can easily and computational efficiently be done by making use of the RQ decomposition as established in Section 6.1.

5.6 EXAMPLE

In this section a simulation example is presented to illustrate the effect of different weightings W_1 and W_2 for the two main Theorems.

The system under consideration is (in forward innovation form):

$$A = \begin{pmatrix} 0.603 & 0.603 & 0 & 0 \\ -0.603 & 0.603 & 0 & 0 \\ 0 & 0 & -0.603 & -0.603 \\ 0 & 0 & 0.603 & -0.603 \end{pmatrix} , \quad B = \begin{pmatrix} 0.9238 \\ 2.7577 \\ 4.3171 \\ -2.6436 \end{pmatrix} ,$$

$$C = \begin{pmatrix} -0.5749 \\ 1.0751 \\ -0.5225 \\ 0.1830 \end{pmatrix}^T , \quad E = \begin{pmatrix} -0.0205 \\ 0.1100 \\ 0.0709 \\ -0.1027 \end{pmatrix} ,$$

and $D = -0.7139$, $F = 0.4853$. The innovation sequence e_k is a zero mean, unit variance, Gaussian white noise sequence of 1000 points. The number of block rows in the block Hankel matrices is chosen equal to 30. 100 different input and innovation sequences are generated according to the scheme outlined below. With these inputs the output y_k is simulated using Matlab. For each input output pair, a state space model of order 4 and of order 2 (identified lower order model) is identified using subspace identification. The model of order 4 is also truncated to order 2 (truncated lower order model). We define $W_{\text{low}}(z)$ as the fourth order Butterworth low pass filter with cut-off frequency equal to 0.5 times the Nyquist frequency (with $T_s = 1$, the Nyquist frequency is equal to 0.5). Similarly, $W_{\text{high}}(z)$ is defined as the fourth order Butterworth high pass filter with the same cut-off frequency. We consider six different cases (different inputs u_k and weighting matrices W_1 and W_2):

Case 1: u_k a zero mean, unit variance white noise sequence and W_1 and W_2 are computed from the first main Theorem (5.13) with $W_u = L_p^{uu}$ and $W_y = I_{30}$. This corresponds to **N4SID** as explained in Section 5.4.2.

Case 2: u_k is the sum of a zero mean, unit variance white noise sequence filtered with $W_{\text{low}}(z)$ and superposed on that a zero mean, 0.0025 variance white noise sequence. W_1 and W_2 are computed from the first main Theorem (5.13) with $W_u(z) = I_{30}$ and $W_y(z) = I_{30}$. This corresponds to a balanced basis as explained in Section 5.4.2.

Case 3: u_k a zero mean, unit variance white noise sequence and W_1 and W_2 are computed from the first main Theorem (5.13) with $W_u = L_p^{uu}$ and $W_y(z) = W_{\text{low}}(z)$.

Case 4: u_k a zero mean, unit variance white noise sequence and W_1 and W_2 are computed from the first main Theorem (5.13) with $W_u = L_p^{uu}$ and $W_y(z) = W_{\text{high}}(z)$.

Case 5: u_k a zero mean, unit variance white noise sequence and W_1 and W_2 are computed from the second main Theorem with $W_u = L_p^{uu}$. This corresponds to **CVA** as explained in Section 5.4.4.

Case 6: u_k a zero mean, unit variance white noise sequence and W_1 and W_2 are computed from the second main Theorem with $W_u(z) = W_{\text{low}}(z)$.

The extraction of the system matrices A, B, C, D, E, F is done as described in Figure 4.8. The average[1] fourth order identified deterministic and stochastic transfer matrix was for all cases almost indistinguishable from the original fourth order transfer matrix. Only for case 2 there was a difference at high frequencies due to the small excitation in that frequency band (colored input u_k). It was also verified that the average weighted Grammians P and Q of the identified fourth order systems were diagonal. On top of that, for Case 1-4 the average weighted controllability Grammian was equal to the average weighted observability Grammian (as predicted by Theorem 13). For Case 5-6 the two Grammians were connected through the relationship (as predicted by Theorem 14):

$$Q[H^{-1}(z)] = P[W_u(z)](I_4 - P[W_u(z)]^2)^{-1} \ .$$

Figure 5.4 shows the average singular values (Case 1-4) and angles (Case 5-6) for the 6 different cases. Clearly these singular values and angles are altered by using different weighting matrices.

Figure 5.5 shows the exact fourth order model, together with the average identified second order model and the average truncated second order model (see Section 5.5 for more details). The effect of the weights is clearly visible.

Table 5.2 shows the errors between the second order identified transfer function and the original fourth order transfer function. These errors were experimentally computed from the average over the 100 experiments. An upper bound for the errors was also predicted from the average weighted Hankel singular values (see also Section 5.5). Clearly the upper bounds hold for this example.

[1] All average properties of this example were calculated as the sample mean of the properties of the 100 estimated models. For instance, the average transfer matrix is calculated as the sample mean of the 100 transfer matrices.

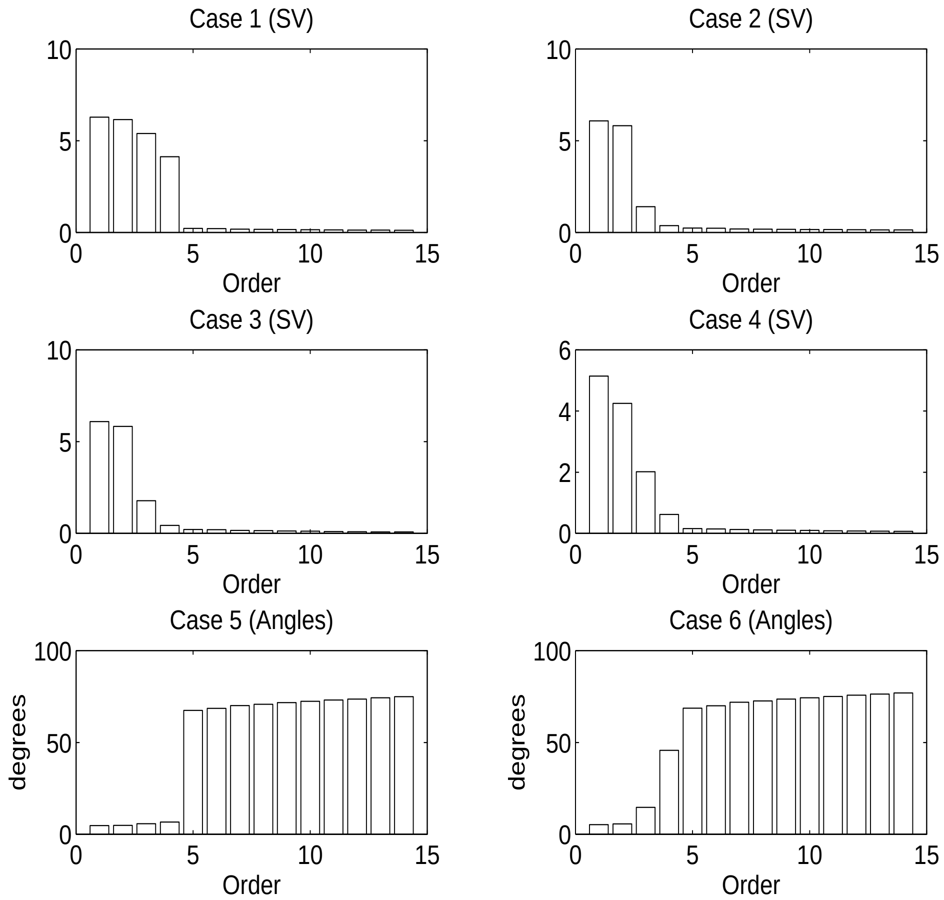

Figure 5.4 The average singular values (Case 1-4) and angles (Case 5-6) that indicate the model order. For Case 1 and 5, the order is clearly equal to 4 (this is expected, since the data was generated by a fourth order system). For Case 2,3,4 and 6, the order is less clear. This is not unexpected, since one of the dynamic modes is filtered out by introducing either $W_{\text{low}}(z)$ or $W_{\text{high}}(z)$.

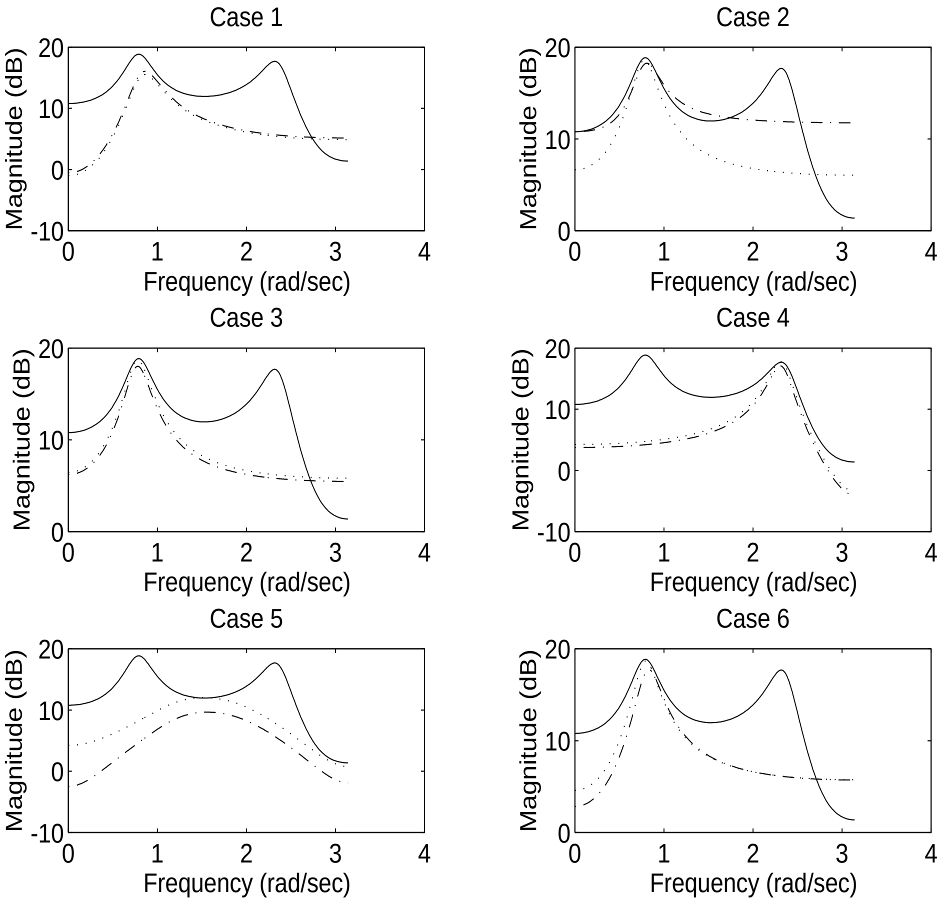

Figure 5.5 Original transfer function (full line), average second order identified transfer function (dashed line) and average second order truncated transfer function (dotted line). The choice of weights determines the shape of the second order identified transfer function. When the weight or input has much energy at low frequencies, the lower resonance is retained (Case 2,3,6). On the other hand when the weight has more energy at high frequencies (Case 4), the high frequency resonance is retained. The weight used in Case 5 ($H^{-1}(z)$) gives poor results for this example. For case 1,3,4 and 6, the lower order truncated and identified transfer function are almost equal. For case 2 and 5 there is a difference. For case 2, this is due to the bad excitation at higher frequencies, and for case 5 it is due to the bad second order approximation of the original fourth order system.

Case	Norm	Exper.	Predicted
1	$\|(G(z) - \hat{G}(z))\|_\infty$	9.2	19.0
2	$\|(G(z) - \hat{G}(z))W_{\text{low}}(z)\|_\infty$	1.4	3.6
3	$\|W_{\text{low}}(z)(G(z) - \hat{G}(z))\|_\infty$	2.7	4.4
4	$\|W_{\text{high}}(z)(G(z) - \hat{G}(z))\|_\infty$	2.1	5.3
5	$\|H^{-1}(z)(G(z) - \hat{G}(z))\|_\infty$	13.9	36.7
6	$\|H^{-1}(z)(G(z) - \hat{G}(z))W_{\text{low}}(z)\|_\infty$	5.1	9.6

Table 5.2 Predicted upper bound from Formula (5.23) with $\alpha = 0$ and experimentally computed H-infinity norms of the difference between the original fourth order transfer function and the identified second order transfer function.

5.7 CONCLUSIONS

In this Chapter we have shown that the state space basis of the subspace identified models corresponds to a frequency weighted balanced basis. The frequency weights are determined by the input spectrum and by user defined input and output weighting functions. The effect of the weights on reduced order system identification has been treated. Finally an example was presented to illustrate the main Theorems.

6

IMPLEMENTATION AND APPLICATIONS

"Less is more . . .
and my architecture is very little, indeed"
Le Corbusier, 1930
cited by Prof. Stephen Boyd, Stanford University, August 1992
(while referring to **GUI**'s).

In this Chapter we take the leap from theory to practice, which involves two major parts:

Implementation: *In a first Section 6.1 we describe how standard numerical tools like the QR and Singular Value decomposition can be used to translate the geometric operations of Section 1.4 and thus the subspace identification algorithms of this book.*

To use (and test) the algorithms of the preceding Chapters in practice, they should be implemented. The implementation in Xmath resulted in a commercially available system identification toolbox **ISID**[1]*. It contains, apart from the subspace algorithms described in this book, a whole scale of processing, classical identification and validation utilities. The toolbox has, as one of the first, a graphical user interface* **GUI***, which reduces the user-threshold for novice users significantly. This is the topic of Section 6.2.*

This book also contains the implementation of the subspace algorithms as a collection of Matlab functions. Since these files do not contain the innovative features of the **GUI** *of* **ISID***, we postpone the description to Appendix B.*

[1] **ISID** stands for **I**nteractive **S**ystem **Id**entification. Xmath is a product of Integrated Systems Inc., CA, USA.

Application: *In Section 6.3 we describe in detail how the* **GUI** *driven toolbox* **ISID** *is used to identify a discrete time linear model of a glass tube manufacturing process, based on which an optimal controller is designed. In Section 6.4 we present the results of the application of the Matlab implementation of the robust subspace algorithm to ten practical examples.*

6.1 NUMERICAL IMPLEMENTATION

In this Section we describe how the algorithms presented in this book can be implemented in a numerically stable and efficient way. We make use of the RQ and the singular value decomposition. It would lead us too far to go into the implementation details of all algorithms presented in this book. Only the robust combined deterministic-stochastic identification algorithm (algorithm 3, Figure 4.8) will be worked out totally. The implementation of other combined algorithms can be easily derived from it. Implementing the specific deterministic and stochastic algorithms can also be done in a similar way. In [VODM 93a] more specific details are given on how to implement the canonical variate analysis through a quotient singular value decomposition **QSVD**. *Other specific implementations can be found in for instance [Aok 87] [DMo 88] [MDMVV 89] [VODM 94a] [VD 92] [VD 91]. We refer the reader to those papers for further details.*

All Matlab files of Appendix B have been implemented using the techniques of this Section. For technical details we refer to Appendix B and the M-files.

6.1.1 An RQ decomposition

The common factor in the implementation of all the algorithms is the RQ decomposition of the block Hankel matrix formed of the input and output measurements[2]:

$$\begin{aligned}\mathcal{H} &\stackrel{\text{def}}{=} \frac{1}{\sqrt{j}}\underbrace{\begin{pmatrix} U_{0|2i-1} \\ Y_{0|2i-1} \end{pmatrix}}_{\in\mathbb{R}^{2(m+l)i\times j}} \\ &= R.Q^T \ ,\end{aligned}$$

[2]The scalar $1/\sqrt{j}$ is used to be conform with the definition of $\mathbf{E_j}$.

with $Q^T \in \mathbb{R}^{2(m+l)i \times j}$ orthonormal ($Q^T Q = I_{2(m+l)i}$) and $R \in \mathbb{R}^{2(m+l)i \times 2(m+l)i}$ lower triangular[3]. This decomposition has several advantages:

- The main advantage will only become clear at the end of this Section where it will be noted that in the final calculations of the system matrices, only the R factor of this decomposition is needed. This implies that in this first step the Q matrix should not be calculated[4]. Since typically $j \gg 2(m+l)i$, this implies that the computational complexity and memory requirements are reduced significantly.
- As will be indicated in the next Subsection, all geometric operations can be easily expressed in terms of this RQ decomposition.
- The amount of computation needed to obtain the R factor of the factorization is of the order of magnitude $i^2.j$. However, the speed of this factorization can be drastically improved by making use of the *Hankel structure*. Indeed, as described in [CXK 94a] [CXK 94b] [CK 95] [Cho 93] the displacement theory can be used to obtain a fast decomposition of the block Hankel matrix. We will not go into the technical details of the procedure, however we would like to note that:
  - The computational load reduces from order of magnitude $i^2.j$ to $i.j$. This can be significant.
  - A straightforward implementation of the Schur algorithm leads to problems with rank deficient Hankel matrices $\mathcal{H}$. The matrix $\mathcal{H}$ becomes rank deficient when the data was generated by a purely deterministic system. This is not often the case in practice. However systems with many outputs can generate Hankel matrices $\mathcal{H}$ that are nearly rank deficient. This is because the outputs become almost co-linear. Also, heavily colored input signals u_k can lead to nearly rank deficient matrices $\mathcal{H}$. A numerically stable implementation of block Hankel matrices should thus be pursued. Until then, we suggest to use any classical RQ decomposition algorithm. This is also why we didn't include this fast decomposition in the software accompanying the book, even though it is illustrated in the examples of Section 6.4.

---

[3] Due to historical reasons we use the symbols R and Q. They should not be confused with the covariance matrices R and Q. This should pose no problem since the R and Q of the RQ decomposition generally have a subscript.

[4] The matrix Q can be viewed as an auxilary matrix needed for the derivations.

For convenience of notation, the decomposition is partitioned as follows:

$$\frac{1}{\sqrt{j}}\begin{pmatrix} U_{0|2i-1} \\ Y_{0|2i-1} \end{pmatrix} = \frac{1}{\sqrt{j}}\begin{pmatrix} U_p \\ U_f \\ Y_p \\ Y_f \end{pmatrix} = \frac{1}{\sqrt{j}}\begin{pmatrix} U_p^+ \\ U_f^- \\ Y_p^+ \\ Y_f^- \end{pmatrix}$$

$$= R.Q^T = \frac{1}{\sqrt{j}} \begin{array}{l} \\ mi \\ m \\ m(i-1) \\ li \\ l \\ l(i-1) \end{array} \begin{array}{c} j \\ \left(\begin{array}{c} U_{0|i-1} \\ U_{i|i} \\ U_{i+1|2i-1} \\ Y_{0|i-1} \\ Y_{i|i} \\ Y_{i+1|2i-1} \end{array}\right) \end{array}$$

$$= \begin{array}{l} \\ mi \\ m \\ m(i-1) \\ li \\ l \\ l(i-1) \end{array} \begin{array}{c} \begin{array}{cccccc} mi & m & m(i-1) & li & l & l(i-1) \end{array} \\ \left(\begin{array}{cccccc} R_{11} & 0 & 0 & 0 & 0 & 0 \\ R_{21} & R_{22} & 0 & 0 & 0 & 0 \\ R_{31} & R_{32} & R_{33} & 0 & 0 & 0 \\ R_{41} & R_{42} & R_{43} & R_{44} & 0 & 0 \\ R_{51} & R_{52} & R_{53} & R_{54} & R_{55} & 0 \\ R_{61} & R_{62} & R_{63} & R_{64} & R_{65} & R_{66} \end{array}\right) \end{array} \begin{array}{c} j \\ \left(\begin{array}{c} Q_1^T \\ Q_2^T \\ Q_3^T \\ Q_4^T \\ Q_5^T \\ Q_6^T \end{array}\right) \end{array} .$$

We will use the shorthand Matlab notation $R_{[4:6],[1:3]}$ for the submatrix of R consisting of *block* rows 4 to 6 and *block* columns 1 to 3, and the shorthand $Q_{3:4}^T$ in a similar way. For instance:

$$R_{[4:5],[4:6]} = \begin{pmatrix} R_{44} & 0 & 0 \\ R_{54} & R_{55} & R_{56} \end{pmatrix} ,$$

and also:

$$R_{[1,4],[1,3]} = \begin{pmatrix} R_{11} & 0 \\ R_{41} & R_{43} \end{pmatrix} .$$

For transposed matrices, the subscript has priority over the transposition:

$$\begin{aligned} R_{[1,4],[1,3]}^T &= \left[R_{[1,4],[1,3]}\right]^T , \\ Q_{3:4}^T &= \left[Q_{3:4}\right]^T . \end{aligned}$$

6.1.2 Expressions for the geometric operations

In this Subsection we give expressions for the geometric operations introduced in Section 1.4 in terms of the RQ decomposition of the previous Subsection.

Orthogonal projections

Orthogonal projections can be easily expressed in function of the RQ decomposition. We first treat the general case $A/\boldsymbol{B}$, where A and B consist of any number of rows of $\mathcal{H}$, which implies that they can be expressed as linear combinations of the matrix Q^T as:

$$\begin{aligned} A &= R_A Q^T \ , \\ B &= R_B Q^T \ , \end{aligned}$$

we thus get:

$$\begin{aligned} A/\boldsymbol{B} &= \Phi_{[A,B]}.\Phi^{\dagger}_{[B,B]}.B \\ &= [R_A Q^T Q R_B^T].[R_B Q^T Q R_B^T]^{\dagger}.R_B Q^T \\ &= R_A R_B^T.[R_B R_B^T]^{\dagger}.R_B Q^T \ . \end{aligned}$$

In many cases, this can be even further simplified. Consider for instance $\mathcal{Z}_i$ (4.18):

$$\mathcal{Z}_i = Y_{i|2i-1}/\begin{pmatrix} \boldsymbol{U}_{0|2i-1} \\ \boldsymbol{Y}_{0|i-1} \end{pmatrix} \ .$$

We have:

$$\begin{aligned} R_A &= R_{[5:6],[1:6]} \ , \\ R_B &= R_{[1:4],[1:6]} \ . \end{aligned}$$

Assume (for simplicity) that R_B is of full row rank. We then get (note that the last two block columns of R_B are zero):

$$\begin{aligned} \mathcal{Z}_i &= R_{[5:6],[1:6]} R^T_{[1:4],[1:6]} [R_{[1:4],[1:6]} R^T_{[1:4],[1:6]}]^{-1} R_{[1:4],[1:6]} Q^T \\ &= R_{[5:6],[1:4]} R^T_{[1:4],[1:4]} [R_{[1:4],[1:4]} R^T_{[1:4],[1:4]}]^{-1} R_{[1:4],[1:4]} Q^T_{1:4} \\ &= R_{[5:6],[1:4]}.\underbrace{R^T_{[1:4],[1:4]} R^{-T}_{[1:4],[1:4]}}_{=I}.\underbrace{R^{-1}_{[1:4],[1:4]} R_{[1:4],[1:4]}}_{=I}.Q^T_{1:4} \\ &= R_{[5:6],[1:4]} Q^T_{1:4} \ , \end{aligned}$$

which is a very simple expression for $\mathcal{Z}_i$. In a similar way, the projection on the orthogonal complement of the row space of a given matrix can be computed. For instance:

$$\begin{aligned} \Pi_{A^\perp} &= I_j - A^T [A A^T]^{\dagger} A \\ &= I_j - Q R_A^T [R_A R_A^T]^{\dagger} R_A Q^T \\ &= Q.[I_{2(m+l)i} - R_A^T [R_A R_A^T]^{\dagger} R_A].Q^T \ , \end{aligned}$$

where the middle matrix (between the square brackets) has "small" dimensions. Once again this expression can often be simplified. For instance the projection on the orthogonal complement of $U_{0|i-1}$ can be written as:

$$\begin{aligned}
\Pi_{U^{\perp}_{0|i-1}} &= Q.[I_{2(m+l)i} - \begin{pmatrix} R_{11}^T \\ 0 \end{pmatrix} [R_{11}R_{11}^T]^{-1} \begin{pmatrix} R_{11} & 0 \end{pmatrix}].Q^T \\
&= Q.[I_{2(m+l)i} - \begin{pmatrix} I_{mi} & 0 \\ 0 & 0 \end{pmatrix}].Q^T \\
&= Q_{2:6}Q_{2:6}^T \ .
\end{aligned}$$

Oblique projections

The oblique projection can also be written in function of the RQ decomposition. For instance with:

$$\begin{aligned}
A &= R_A Q^T \ , \\
B &= R_B Q^T \ , \\
C &= R_C Q^T \ ,
\end{aligned}$$

we find that:

$$\begin{aligned}
A/\boldsymbol{B}^{\perp} &= R_A[I_{2(m+l)i} - R_B^T[R_B R_B^T]^{\dagger} R_B].Q^T \ , \\
C/\boldsymbol{B}^{\perp} &= R_C[I_{2(m+l)i} - R_B^T[R_B R_B^T]^{\dagger} R_B].Q^T \ ,
\end{aligned}$$

and with the orthogonal projection operator being idempotent:

$$\Pi_{B^{\perp}}.\Pi_{B^{\perp}} = \Pi_{B^{\perp}} \ ,$$

we find through formula (1.7) for the oblique projection:

$$\begin{aligned}
A/_{B}\boldsymbol{C} &= A/\boldsymbol{B}^{\perp}.\left[C/\boldsymbol{B}^{\perp}\right]^{\dagger}.C \\
&= R_A[I_{2(m+l)i} - R_B^T[R_B R_B^T]^{\dagger} R_B] \\
&\quad \times \left[R_C[I_{2(m+l)i} - R_B^T[R_B R_B^T]^{\dagger} R_B]\right]^{\dagger} \ . R_C Q^T \ . \qquad (6.1)
\end{aligned}$$

For instance with:

$$\begin{aligned}
A &= Y_f = R_{[5:6],[1:6]}Q^T \ , \\
B &= U_f = R_{[2:3],[1:6]}Q^T \ , \\
C &= W_p = R_{[1,4],[1:6]}Q^T \ ,
\end{aligned}$$

we could compute the oblique projection $Y_f/_{U_f}\,\boldsymbol{W}_p$. Even though this would lead to a valid expression for the oblique projection, there is a better way to calculate this quantity by first projecting Y_f on the row space of the past outputs and the past and future inputs, and then separating the effect of the future inputs U_f out of this projection (see also Section 1.4). This can be done as follows: With L_{U_p}, L_{U_f} and L_{Y_p} defined as:

$$\left(\underbrace{L_{U_p}}_{\in\mathbb{R}^{li\times mi}} \quad \underbrace{L_{U_f}}_{\in\mathbb{R}^{li\times mi}} \quad \underbrace{L_{Y_p}}_{\in\mathbb{R}^{li\times li}} \right) \stackrel{\text{def}}{=} R_{[5:6],[1:4]} R^{\dagger}_{[1:4],[1:4]} \; , \tag{6.2}$$

we have:

$$\mathcal{Z}_i = L_{u_p}.U_p + L_{U_f}.U_f + L_{Y_p}.Y_p \; .$$

We thus get for the oblique projection:

$$\begin{aligned} Y_f/_{U_f}\,\boldsymbol{W}_p &= L_{U_p}.U_p + L_{Y_p}.Y_p \\ &= \left[L_{U_p}.R_{[1:1],[1:4]} + L_{Y_p}.R_{[4:4],[1:4]} \right] Q^T_{1:4} \; . \end{aligned} \tag{6.3}$$

This computation is significantly faster than the previous one, since (when $R_{[1:4],[1:4]}$ is of full rank) the matrices L_{U_p}, L_{U_f} and L_{Y_p} can be computed using back-substitution (since $R_{[1:4],[1:4]}$ is a lower triangular matrix).

Principal angles and directions

In [VODM 93a] it is described how the principal angles and directions can be computed from the **QSVD** of certain R factors. Since we will not be using these results in this book, we refer to [VODM 93a] for further details.

Finally we want to note that the "best" numerical implementation is still an open problem. The implementation presented in this Section is one possible implementation, that allows to calculate all quantities presented throughout the book. When only one specific matrix is needed, there often exist more optimal implementations. For instance, when only:

$$Y_f/_{U_f}\,\boldsymbol{W}_p.\Pi_{U_f^\perp}$$

were needed (for instance in **MOESP** and in the third combined identification algorithm of Figure 4.8), it would be better to use the RQ decomposition of formula (4.40):

$$\begin{pmatrix} U_f \\ W_p \\ Y_f \end{pmatrix} = \begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} Q_1^T \\ Q_2^T \\ Q_3^T \end{pmatrix} \; ,$$

since it can be verified that in this case:

$$Y_f /_{U_f} \boldsymbol{W}_p.\Pi_{U_f^\perp} = L_{32}.Q_2^T \ .$$

The advantage of the different ordering of the matrices U_f, W_p and Y_f in the RQ decomposition is however lost when the matrix:

$$Y_f^- /_{U_f^-} \boldsymbol{W}_p^+$$

has to be calculated. That is why we stick to the RQ decomposition presented at the beginning of this Section.

6.1.3 An implementation of the robust identification algorithm

To illustrate the ease of use of the RQ decomposition when implementing subspace identification algorithms, we give the details of the implementation of the third combined algorithm (Figure 4.8). Further details on the first and second combined algorithms can be found in [VODM 94a]. The final implementation is summarized in Figure 6.1. The projected oblique projection can be computed as:

$$\mathcal{O}_i \Pi_{U_f^\perp} = \left[L_{U_p}.R_{[1:1],[1:4]} + L_{Y_p}.R_{[4:4],[1:4]} \right] Q_{1:4}^T$$

$$\times Q \left[I_{2(m+l)i} - \begin{pmatrix} R_{[2:3],[1:3]}^T \\ 0 \end{pmatrix} [R_{[2:3],[1:3]} R_{[2:3],[1:3]}^T]^\dagger \begin{pmatrix} R_{[2:3],[1:3]} & 0 \end{pmatrix} \right] Q^T \ .$$

With:

$$\Pi = I_{2mi} - R_{[2:3],[1:3]}^T [R_{[2:3],[1:3]} R_{[2:3],[1:3]}^T]^{-1} R_{[2:3],[1:3]} \ ,$$

this leads to:

$$\mathcal{O}_i \Pi_{U_f^\perp} = \begin{pmatrix} (L_{U_p} R_{[1:1],[1:3]} + L_{Y_p} R_{[4:4],[1:3]}).\Pi & \Big| & L_{Y_p} R_{44} \end{pmatrix} Q_{1:4}^T \ . \qquad (6.4)$$

The other steps of the implementation are straightforward (except maybe for the step where B and D are determined). The overall implementation is illustrated in Figure 6.1.

The Matlab function `subid.m` contains a Matlab implementation of this robust algorithm. See also Appendix B.

Implementation of algorithm 3:

1. Compute $L_{U_p} \in \mathbb{R}^{li \times mi}$ and $L_{Y_p} \in \mathbb{R}^{li \times li}$ from:

$$\begin{pmatrix} L_{U_p} & L_{U_f} & L_{Y_p} \end{pmatrix} = R_{[5:6],[1:4]} R^{\dagger}_{[1:4],[1:4]} \,,$$

where the pseudo inverse can be substituted by a backsubstitution when $R_{[1:4],[1:4]}$ is of full rank.

2. With:

$$\Pi = I_{2mi} - R^T_{[2:3],[1:3]} [R_{[2:3],[1:3]} R^T_{[2:3],[1:3]}]^{-1} R_{[2:3],[1:3]} \,.$$

Compute the **SVD** of:

$$\begin{pmatrix} (L_{U_p} R_{[1:1],[1:3]} + L_{Y_p} R_{[4:4],[1:3]}).\Pi \,\big|\, L_{Y_p} R_{44} \end{pmatrix} \,.$$

3. Determine the order by inspection of the singular values and partition the **SVD** accordingly to obtain U_1 and S_1.

4. Determine $\Gamma_i = U_1 S_1^{1/2}$ and $\Gamma_{i-1} = \underline{\Gamma_i}$. Define $\mathcal{T}_l$ and $\mathcal{T}_r$ as:

$$\mathcal{T}_l \stackrel{\text{def}}{=} \begin{pmatrix} \Gamma^{\dagger}_{i-1} R_{[6:6],[1:5]} \\ R_{[5:5],[1:5]} \end{pmatrix} \quad , \quad \mathcal{T}_r \stackrel{\text{def}}{=} \begin{pmatrix} \Gamma^{\dagger}_{i} R_{[5:6],[1:5]} \\ R_{[2:3],[1:5]} \end{pmatrix} \,.$$

Compute A and C as the first n columns of: $\mathcal{S} = \mathcal{T}_l . \mathcal{T}_r^{\dagger}$.

5. In order to solve B and D, the matrices $\mathcal{P}$ and $\mathcal{Q}$ are set equal to:

$$\begin{aligned} \mathcal{P} &= \mathcal{T}_l - \begin{pmatrix} A \\ C \end{pmatrix} \Gamma^{\dagger}_i R_{[5:6],[1:5]} \,, \\ \mathcal{Q} &= R_{[2:3],[1:5]} \,. \end{aligned}$$

Also determine the matrices $\mathcal{N}_k$ (4.57)-(4.59). Solve (4.61) for B and D.

6. Determine the covariance matrices Q, S and R as:

$$\begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} = (\mathcal{T}_l - \mathcal{S}.\mathcal{T}_r)(\mathcal{T}_l - \mathcal{S}.\mathcal{T}_r)^T \,.$$

Figure 6.1 Implementation of a robust deterministic-stochastic identification algorithm. Note that the Q matrix of the RQ decomposition is never needed. This algorithm has been implemented in the Matlab function subid.m.

6.2 INTERACTIVE SYSTEM IDENTIFICATION

*In this Section, we first explain in Subsection 6.2.1 the trend in present days software towards graphical user interfaces (***GUI***s). In Subsection 6.2.2, we describe our* **GUI ISID**[5]*, which was developed to do interactive system identification in an ultimately user friendly manner. We describe the use of* **ISID** *in Subsection 6.2.3 while in Subsection 6.2.4 we give an overview of the algorithms that are implemented in* **ISID** *(preprocessing, identification, validation and display). Since it is impossible to summarize all the features of* **ISID***, let alone that we could visualize all its graphical functionalities, we would like to refer the interested reader to the* **ISID** *manual [VODMAKB 94].*

6.2.1 Why a graphical user interface ?

*In this Subsection, we motivate the use of a graphical user interface (***GUI***) for system identification, by giving a short historical overview of the different types of user-interfaces.*

The overview is confined to the history of user interfaces for identification and control of processes, which can be split up in three stages: Program User Interfaces, Command-line User Interfaces and Graphical User Interfaces. An overview of the discussion is given in Table 6.1.

Program User Interface: This is the era of lower level programming languages (Fortran, Pascal and C). Typically, the user-input consisted of programs with low level commands. To solve a problem, one had to write a library of applicable functions, which were then combined to a program. The programming had to be done on a very low level, and it was only at the end, when all the bits and pieces were put together, that the results were obtained. If the results were not satisfactory, parts of the programs had to be rewritten. Due to the inflexibility of the programs, most of the time was spent programming. Investigation of the influence of different parameters (and different methods) on the result was hard and time consuming. Especially the intermediate bookkeeping tasks were very tedious.

Command-line User Interface: The second type of user interfaces is the command-line interface. This interface allows the user to enter commands at the command-line. Contrary to the previous generation, the effect of the commands could

[5] **ISID** (Part 2) was developed in Xmath, a trademark from Integrated Systems Inc., Santa Clara, California, USA, and is the successor of **ISID** Part 1 (which is a command-line user interface package for system identification, see [AMKMVO 93] [AKVODMB 93] for more details about **ISID** Part 1).

immediately be inspected. The commands also became more powerful (higher level), which made writing programs and experimenting with different methods less time consuming. Another feature was the bundling of programs into so called Toolboxes. These bundles contained all the necessary "tools" to solve a whole class of similar problems. There was for instance a Toolbox for control problems and one for identification problems. However, due to the complexity of the commands it was not easy for a novice user to start using these Toolboxes. First, he had to become familiar with all methods implemented in the Toolbox. Then, he also had to understand and study the sometimes complicated syntax of the commands. Finally, he had to understand how to *interconnect* the commands into a program that would solve his problem.

Typically the user had to read thick manuals with extended syntax conventions before he could start. On top of that, after he had read the manuals, it was not always clear how to solve his problem. This is because there was hardly any user guidance available (apart from the examples in the manuals). A thorough understanding of the implemented methods was thus still needed to use the Toolbox.

Even though a lot easier to use and more flexible than the Program User Interface programs, this new generation still had significant drawbacks i.e. the extended syntax, the complicated interconnections of commands and the lack of user-guidance.

Graphical User Interface: The most recent interface is the graphical user interface (**GUI**). A **GUI** is a user interface made up of graphical objects such as menus, buttons and plots. Using it is straightforward since it only requires manipulation of the three mouse buttons and at rare occasions, typing in the name of an object or data file.

A first feature of a **GUI** is that it makes thick manuals virtually obsolete. Most graphical objects are clearly labeled so that their function is immediately clear. There is no need to study complicated syntax. An overview of the functionality can be found by browsing through the menus of the interface.

Another **GUI** feature is that the effect of changes in parameters (or methods) is depicted graphically. In the previous generation, the results were obtained as variables. These variables had to be transformed to figures to be interpreted. This made it necessary to add extra visualization commands. A **GUI** presents all results graphically, which excludes this last step, and which turns it into an elegant tool to perform varying parameter experiments.

On top of that, a **GUI** for identification and control system design provides the user with guidelines to solve her problem. By equipping the **GUI** with a certain intelligence (highlighting certain menus, buttons and plot handles), the user is guided through the interconnection of complicated functions. This

	Program User Interface	Command-line User Interface	Graphical User Interface
Date	±1960-1982	1982-1990	1990-?
Input	programs	command-line	graphical
Output	text	variables	graphical
Elementary Block	low level command	high level command	window
Start Threshold	very high	high	low
Flexibility	low	high	high
User guidance	none	limited	high
Syntax	complex	complex	simple
Expertise required	high	high	limited
Computer required	simple	simple	fast

Table 6.1 Comparison of the different stages in the history of user interfaces for identification and control software.

interconnection is also graphically depicted, which enables the user to retain a clear overview (see for instance Figure 6.5).

A **GUI** for system identification thus enables a novice user to get acquainted with the intuitive software without the need for thick manuals or extensive (identification) expertise. In the next Subsections these advantages will become even more apparent.

6.2.2 ISID: Where system identification and GUI meet

The combination of powerful numerical algorithms for system identification (such as subspace identification algorithms) and a graphical user interface leads to intelligent and user-friendly identification software. The **I***nteractive* **S***ystem* **Id***entifiction software* **ISID***, contains 3 major concepts: the data objects, the building blocks and the chain. These concepts are briefly reviewed in this Subsection.*

Data Objects

System identification typically requires a fair amount of data-handling. This handling is organized in **ISID** using so-called **data objects**. The basic idea behind it is to bundle data and related information into one object. The five different data objects in **ISID** are:

- **Input-Output Records:** contain input-output data in the time domain. In most identification problems the data is measured from the plant. Alternatively, the data could be gathered by simulating a model of the plant in any of your favorite numerical simulator.
- **Frequency Responses:** contain complex data in the frequency domain representing a frequency response or a spectral density function.
- **Impulse Responses:** contain data in the time domain representing an impulse response or a covariance sequence.
- **Models:** contain models of systems (one or more). The end result of an identification session is typically a model data object.
- **Square Roots:** contain intermediate identification results. A square root contains the same information as the input-output data sequence it was derived from, but in a condensed form. Basically, it contains the R factor of the RQ decomposition of Section 6.1.

Apart from the actual data (the matrix containing the numerical values), data objects also contain the following information: sampling time and unit, channel names and units, data object history[6] and a data object name.

Data objects are typically used for three purposes: At the beginning of an identification session, a variable is loaded in **ISID** from the Xmath workspace. It is internally converted to a data object. Data objects are also used to transfer information between different algorithms. As will be explained below, algorithms have as inputs and outputs one (or more) data objects. Finally, at the end of an identification session the data objects of interest are converted to Xmath variables and saved in the Xmath workspace.

[6] The history consists of a textual description of how the object came about. This releaves the user of the bookkeeping details of the identification session.

Building Blocks

The second major concept of **ISID** is that of **building blocks**. On an abstract level, a building block is an operator with inputs and outputs. The building block operates on the inputs to calculate its outputs. Within **ISID**, the building blocks have the following structure (see also Figure 6.2):

- **Inputs:** one or more data objects
- **Operation:** determined by a set of parameters
- **Outputs:** one or more data objects

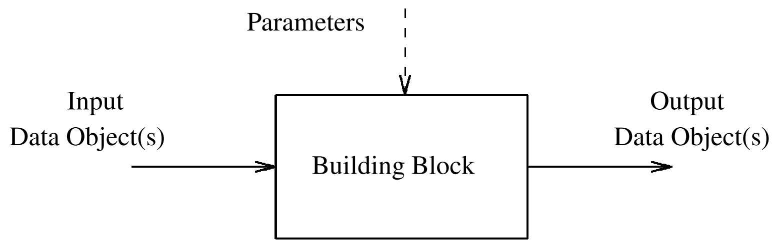

Figure 6.2 An **ISID** building block: the inputs and outputs are data objects, the operation of the block is determined by a set of parameters.

Each building block has an algorithm window associated with it. This window displays information about the building block and allows for graphical interaction to set some of the parameters.

In **ISID** there are two different types of building blocks: Algorithms building blocks and data boxes, which differ slightly from the first type since they have an Xmath variable as input (and not a data object).

There are 4 classes of algorithm building blocks: processing, identification, validation and display algorithms. Each of the algorithm blocks operates on an input data object to create the output object.

Figure 6.3 shows an example of an identification algorithm building block. The block itself is represented by an icon (the rectangular box containing the text “Algorithm Building Block”). The algorithm window associated with the building block is also displayed.

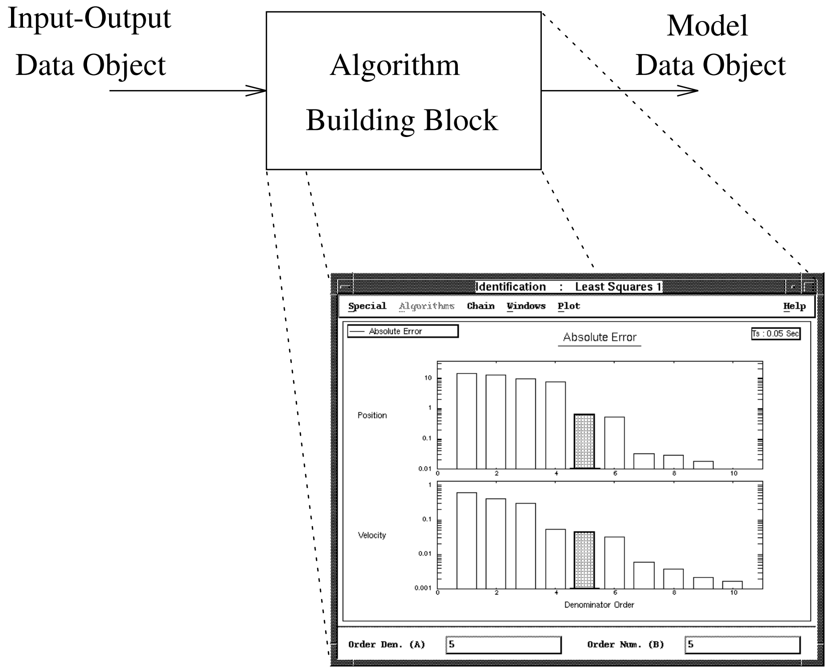

Figure 6.3 An example of an algorithm building block. The block itself is represented by an icon (the rectangular box). The input and output data objects are indicated by the arrows. The algorithm window associated with the building block is displayed in the right bottom corner. This window allows for graphical interaction with the algorithm parameters. Furthermore, it is possible to zoom in on one subplot or on a specific part of the data. The axes of the plot can be set interactively and the data values can be viewed by a single click of the mouse.

Data boxes are very similar to algorithms, with the only difference that their input is not a data object but an Xmath variable. Every time an Xmath variable is loaded into **ISID**, a data box is generated. Data boxes are always found at the beginning of the chain (see below). Figure 6.4 shows an example of a data box containing an input-output data object.

A complete list of building blocks (algorithms and data boxes) and their parameters can be found in Subsection 6.2.3.

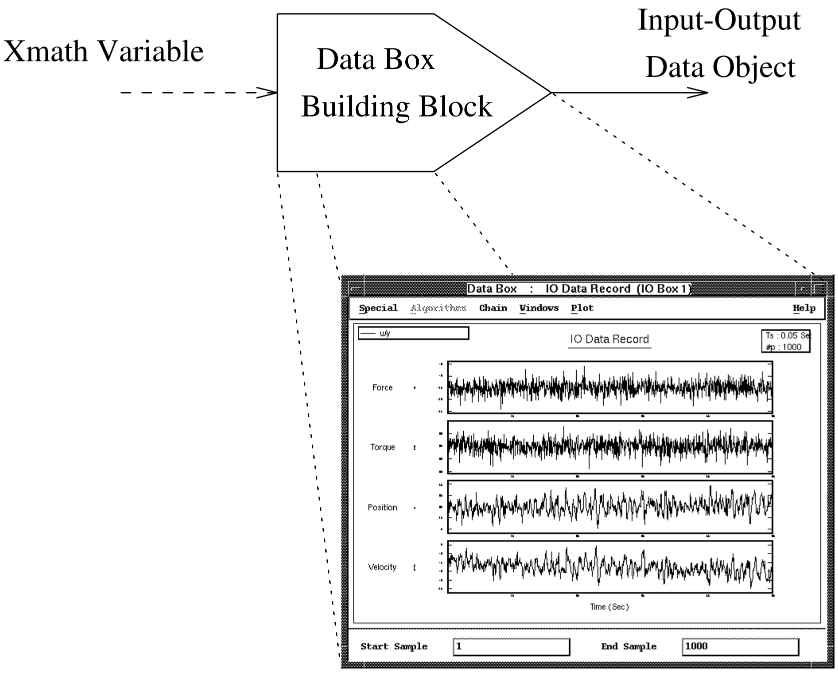

Figure 6.4 An example of a data box. The input of the data box is an Xmath variable, the output is an **ISID** data object. Just as algorithms, data boxes have an algorithm window associated with it. This window allows for graphical interaction with the parameters.

Chain

The main idea of **ISID** is to connect building blocks one to another. The output of one building block is used as the input to another building block. In this way, different building blocks can be put in series and in parallel. The result is called a **chain** (of building blocks).

Figure 6.5 shows an example of a chain. The different building blocks are represented by an icon in the Chain Plot. One can distinguish the data box icons at the beginning of the chain and the algorithm icons that are connected back to front.

The chain thus consists of a "causal" connection of algorithms in series or parallel (no feedback). Whenever either the input object or the parameters of any of the algorithms in the chain change, these changes are propagated causally (from left to right) through the chain.

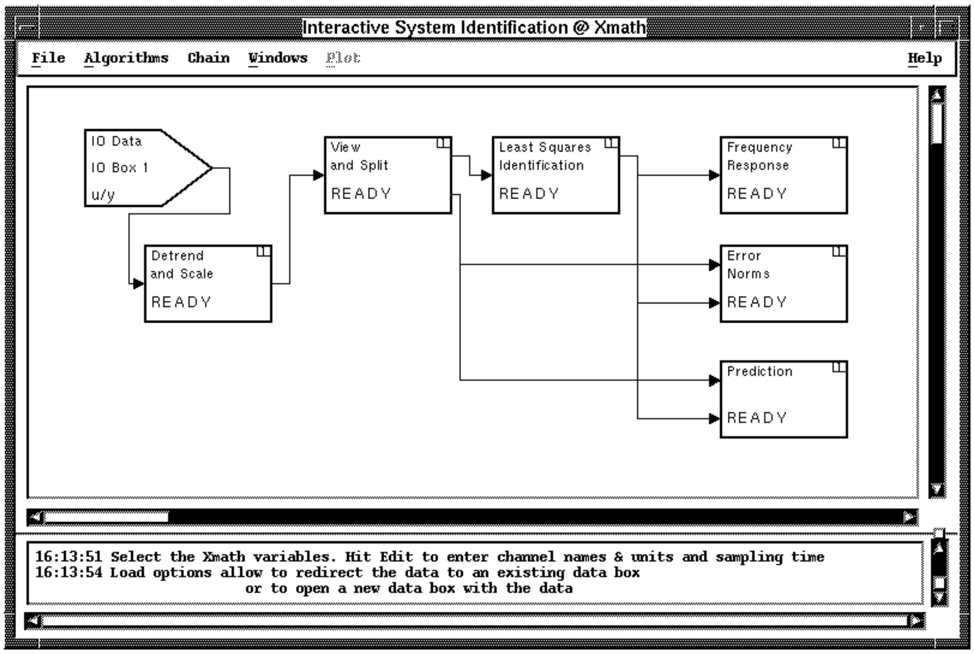

Figure 6.5 An example of a chain. Each icon represents a building block. The data box (arrow shaped) is located at the beginning of the chain. The algorithms are connected back to front.

For instance, when the output of an algorithm (say A) is used as input for another algorithm (say B), then these algorithms are coupled and thus form a part of the chain. Every time a parameter of algorithm A changes, a new output of A is calculated which in turn triggers the recomputation of algorithm B. This procedure is not restricted to two algorithms, and it is thus possible to couple many algorithms back to front, to form a chain.

Typically, the front end of a chain consists of data boxes, containing variables loaded from Xmath. The back end consists of validation or display algorithms. The intermediate part of the chain consists of processing algorithms in series and one or more identification algorithms in parallel.

The graphical interaction with the chain and with the parameters of the building blocks allows for a fast inspection of the influence of certain parameters on the quality of the identified model.

6.2.3 Using ISID

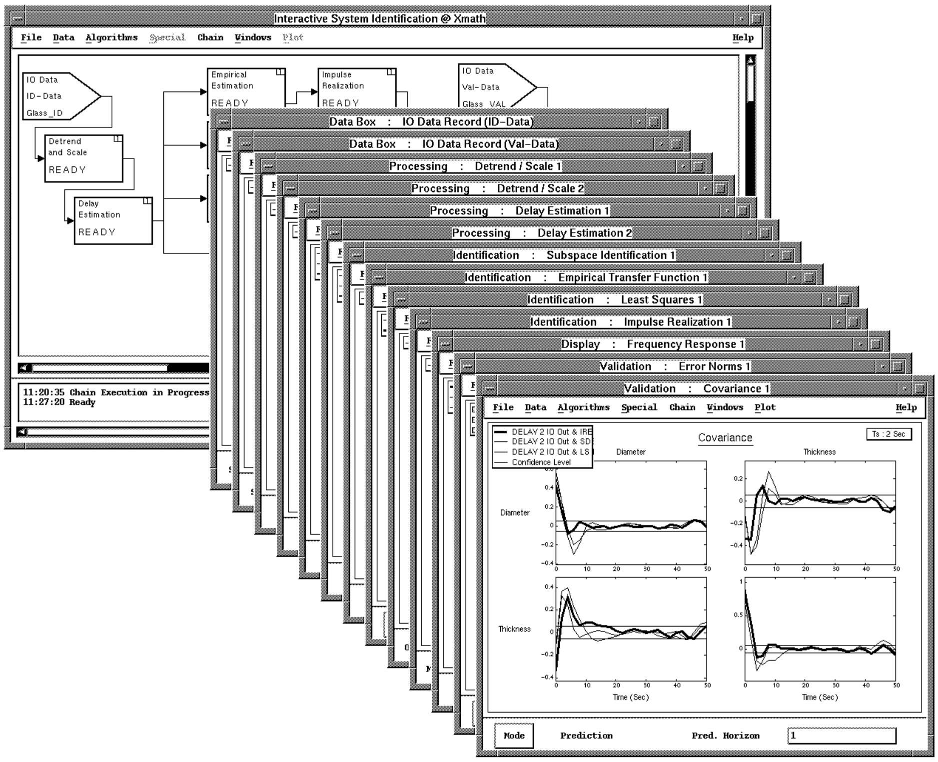

Figure 6.6 An overview of an **ISID** session. Shown in the background is a particular chain window. In the foreground, we see the algorithm windows which are automatically ordered as a "card box". There are several functionalities to manipulate these windows, details of which can be found in [VODMAKB 94].

An overview of a typical **ISID** session is depicted in Figure 6.6. **ISID** can be used in many different ways. As an example you might:

- interactively identify a model from input-output data, frequency response data or impulse response data
- compare the results of different identification techniques (the more these results are alike, the more you can trust your models)
- pre-process data measured from an industrial plant to a form suitable for linear system identification

- cross-validate identified models on validation data
- display several properties of the identified models
- interactively analyze the influence of parameter changes on the resulting model quality
- save the identified model to Xmath and use it (after conversion to a continuous time model) for control design

6.2.4 An overview of ISID algorithms

Each of the **ISID** algorithm building blocks takes a data object as input and returns another data object at its output. The four data object classes (and their abbreviations) are: Input-Output Data (IO), Frequency Response (Freq), Impulse Response (Imp) and Models (Model)[7]. There are 4 different types of algorithm building blocks, namely pre-processing, identification, validation and display algorithms. Since obviously we can not describe all of these functionalities in full detail (for which we would like to refer the interested reader to the manual [VODMAKB 94]), we will restrict ourselves here to a mere enumeration of all the possibilities of algorithm building blocks. In each of the following tables, the first column contains the name of the functionality, the second column is the data object that acts as an input, the third column the data object that is delivered as an output while the fourth column is a one-line description of the functionality.

(Pre-)Processing

Industrially measured data sets often need to be *pre-conditioned* to make them suitable for linear time-invariant identification. The processing algorithms are:

Name	Input	Output	Functionality
Detrend and Scale	IO	IO	Scaling and trend removal (drift ...)
Peak Shaving	IO	IO	Removal of outliers (sensor failure ...)
Delay Estimation	IO	IO	Estimation and extraction of delays.
Filtering	IO	IO	Low, high or band pass filtering.
Post Sampling	IO	IO	Subsampling of a data set.
View and Split	IO	IO	Split in Identif. and Validation set.

[7] Since the square root object is only used by expert users, it is dropped from this discussion.

Identification

ISID contains a whole range of identification algorithms. Apart from the "classical" identification algorithms there are also two subspace identification algorithms described in this book implemented: The stochastic identification algorithm[8] of Figure 3.13 and the combined deterministic stochastic identification algorithm of Figure 4.8. The collection of identification algorithms is the following:

Name	Input	Output	Functionality
Least Squares	IO	Model	Least Squares in time domain.
Subspace Identification	IO	Model	**This book.**
Instrumental Variables	IO	Model	Uses inputs as instruments.
Prediction Error Method	IO	Model	Classical cost function minimization.
Empirical Estimation	IO	Freq & Imp	Non-parametric identification.
Frequency Least Squares	Freq	Model	Least squares in frequency domain.
Impulse Realization	Imp	Model	Realization of impulse responses.

Validation

The validation algorithms of **ISID** allow to asses the "quality" of the identified models. This is done by inspection of different properties of the prediction errors. The validation algorithms are:

Name	Input	Output	Functionality
Error Norms	IO & Model	-	Displays prediction error norms.
Prediction	IO & Model	IO	Predicted signals.
Prediction Errors	IO & Model	IO	Prediction errors.
Covariance	IO & Model	Imp	Covariance of prediction errors.
Spectral Density	IO & Model	Freq	Spectral density of prediction errors.
Cross Correlation	IO & Model	Imp	Correlation inputs $\leftrightarrow$ prediction errors.

[8]Note that the stochastic identification algorithm is only included in the command line interface **ISID** Part 1 and not in the graphical toolbox **ISID** Part 2, because (for now) the **GUI** only allows identification of *input-output* data.

Display

These algorithms are intended to display properties of the identified model which can then be interpreted by the engineer. The display algorithms are:

Name	Input	Output	Functionality
Frequency Response	Model	Freq	Displays frequency responses.
Impulse Response	Model	Imp	Displays impulse responses.
Spectral Density	Model	Freq	Displays spectral densities.
Covariance	Model	Imp	Displays covariances.
Poles & Zeros	Model	-	Displays poles & zeros.

6.2.5 Concluding remarks

GUI *driven toolboxes belong to a third generation of user-friendly software packages. Users can tackle more serious problems than previously possible because of the fact they don't need to bother about programming subtleties. In addition, bookkeeping of tasks to do, of interconnections of models and data sets becomes straightforward. Finally, there is lots of user guidance and as a matter of fact, most of the time default settings are provided (and they represent the most commonly performed actions anyway).*

We have been developing a **GUI**, *called* **ISID**, *which provides all of these functionalities for a* system identification *environment, including the subspace identification algorithms described in this book. Obtaining mathematical models from experimental data from industrial processes now comes within reach of* every *control system design engineer.*

6.3 AN APPLICATION OF ISID

To illustrate the possibilities of the subspace identification algorithms developed in this book and of the software implementation **ISID**, *we describe one industrial case study in detail. We will build a state space model, from input-output records of a glass tube manufacturing process*[9]. *At the same time, we will show in some more detail the different* **GUI** *functionalities that we have been describing in the previous Section 6.2.*

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

Other identification methods are applied to the glass tube manufacturing process in [Bac 87] [DBC 81] [Fal 94] [Hak 94].

In Subsection 6.3.1, we give a short description of the process. In Subsection 6.3.2, we build the Chain that allows the identification of the process and we discuss some intermediate options and results. Although it is not a formal part of this book, we show the results of a control design based on the derived model in Subsection 6.3.3.

6.3.1 Problem description

Figure 6.7 shows a schematic description of the glass tube production process. Quartz sand is fed to the machine at the top. The sand is melted to glass inside the furnace. The glass tubes are drawn at the bottom of the machine.

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

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

Two input-output data sequences were measured. The diameter is measured in two orthogonal directions and averaged. The wall thickness is measured in four directions in a plane, and once again, these measurements are averaged. The input and output signals are scaled so that the original signals can not be retrieved (confidentiality of industrial information), after which they are oversampled with a factor 10. These processing steps can also be found in [Bac 87] [Fal 94] [VO 94]. The first input-output sequence (1355 points, see Figure 6.8) is used for the identification of the process. The second sequence (893 points) is used for the validation of the results. For both input signals a pseudo random binary noise sequence was used as input.

6.3.2 Chain description and results

One of the main features of **ISID** described in Section 6.2, is the graphical representation of the chain of algorithms to be executed to find a mathematical model of the process under study. Figure 6.9 shows the chain that was used to identify the glass tube manufacturing process.

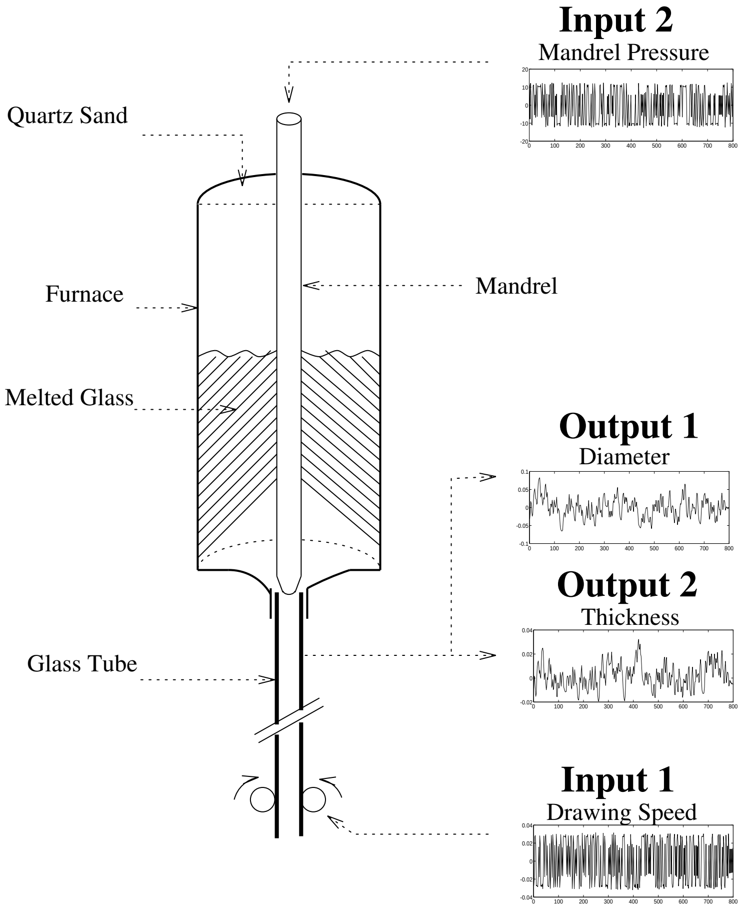

Figure 6.7 The glass tube manufacturing process. Inputs are drawing speed and mandrel pressure. Outputs are tube diameter and thickness. The input signals were pseudo-binary noise sequences. These inputs are sufficiently "wild" to excite all the dynamic modes of the system (which is necessary since we want to control these modes afterwards). The tubes that are produced from these inputs are worthless since they are too irregular due to the wild inputs. This situation is typical for industrial system identification: There is a basic trade-off between the production loss that goes together with experimenting and the production quality enhancement as a result of the identification/model-based control design.

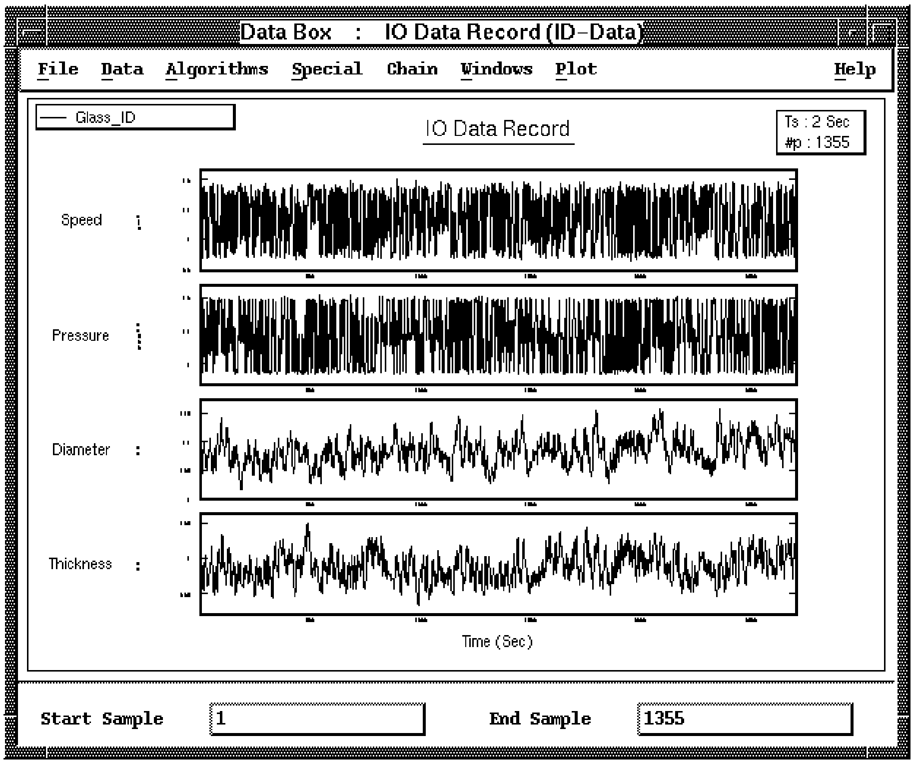

Figure 6.8 Data set used for the identification of the glass tube production process. The inputs (top two signals) are drawing speed and mandrel pressure. They are excited using pseudo-binary noise sequences. The outputs (bottom two signals) are tube diameter and thickness.

Processing: Both input-output data records are first detrended to remove the mean and the linear trends in the data. Since the measurements of the outputs can only be done when the tubes are sufficiently cooled down, there are significant time delays. The two delay estimation blocks estimate these delays and compensate for them by shifting the input and output signals in the appropriate direction. Figure 6.10 shows the algorithm window behind the delay estimation algorithm block.

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

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

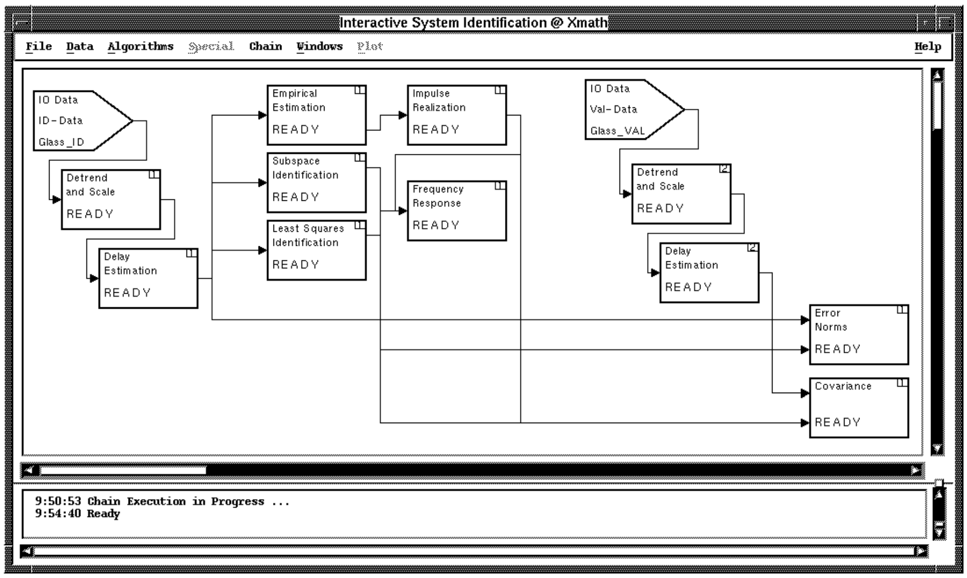

Figure 6.9 **ISID** chain representing the identification of the industrial glass-tube manufacturing process. Each block represents an algorithm. The blocks to the left are the input-output data and preprocessing blocks, followed by the identification blocks. To the right, we also see some validation blocks. Behind each block there is an algorithm window that visualizes the algorithm specific data and allows for adjustment of the parameters.

- A second model is obtained from subspace identification. The algorithm window corresponding to this identification block is shown in Figure 6.11.
- A last model is obtained by a least squares identification.

Validation: The models are validated by comparing the measured and simulated outputs (validation and identification data). For this example, the subspace model has the smallest error and is thus used for control design. Through computation of the covariance of the prediction errors, the whiteness of the errors can be checked.

Display: The transfer functions of the obtained models are displayed on the same plot to compare the models in the frequency domain. Other model properties can also be easily displayed.

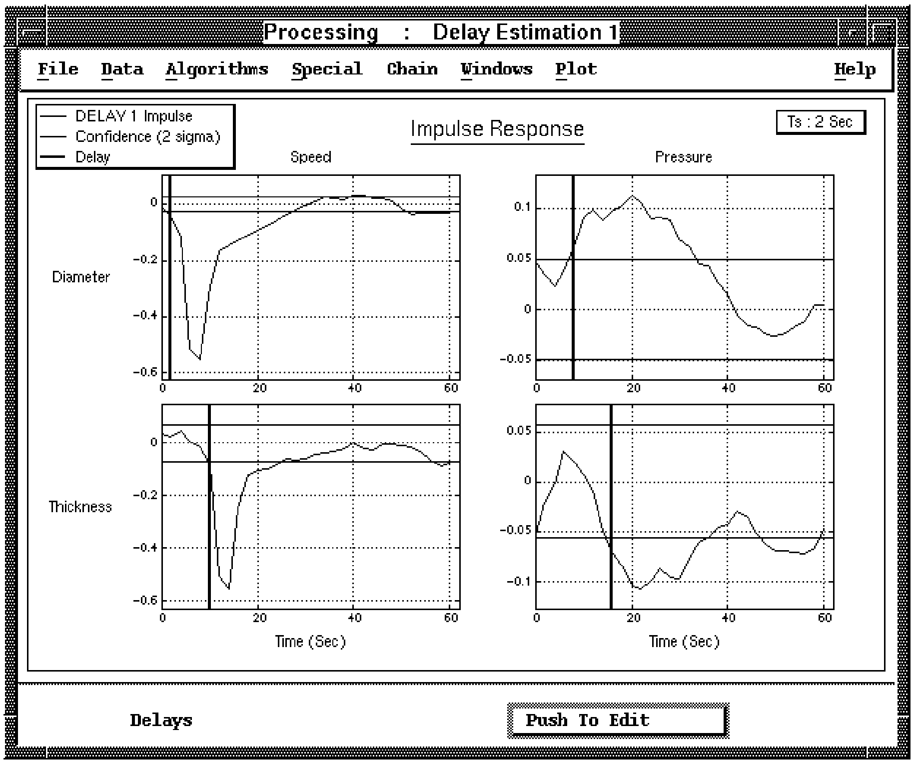

Figure 6.10 The **ISID** algorithm window for the delay estimation algorithm. The delays are indicated by the vertical lines. They can be changed by clicking on the vertical lines and dragging them to the desired value i.e. the intersection of the impulse response and the confidence bounds (horizontal lines). This contrasts with the Command-line User Interface where the user had to read the intersecting point from the scales.

6.3.3 PIID control of the process

Even though this is not a part of the **ISID** software, we describe the result of a controller design to illustrate the use of the subspace algorithm based model. The control design technique, based on multi-objective optimization of the free parameters of the controller is described in [VDM 93]. Figure 6.12 illustrates the noise reduction and thus the quality enhancement that can be obtained with this controller[10]

[10]Unfortunately, at the printing of the book, the industrial closed-loop experiments were still to be performed. Figure 6.12 thus shows the simulated product enhancement using the measured open-loop disturbances.

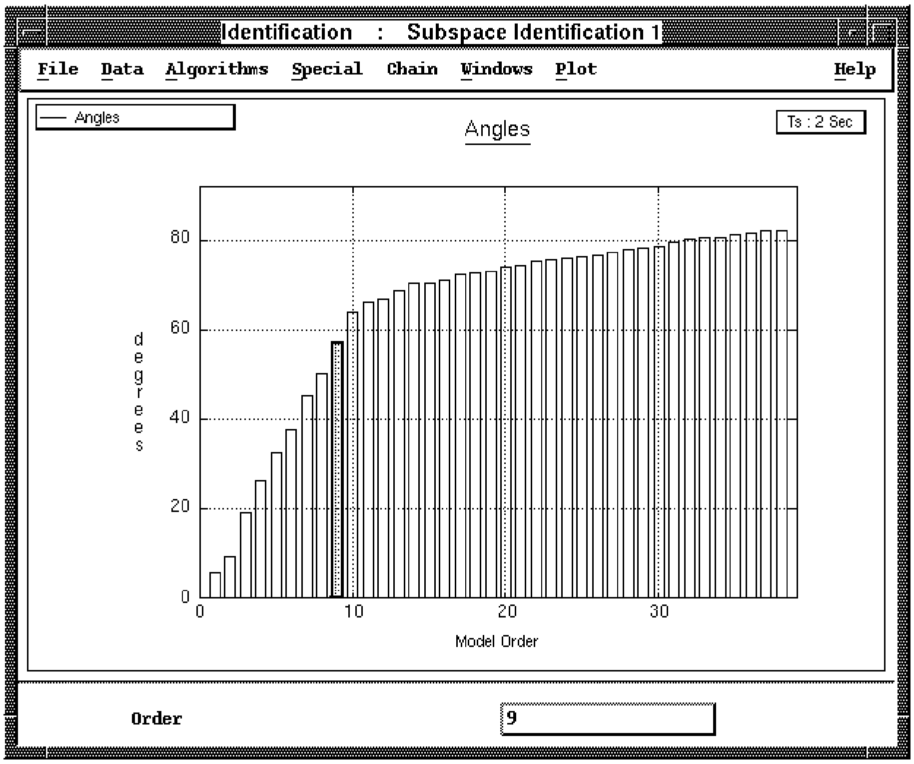

Figure 6.11 The algorithm window behind the subspace identification algorithm. The plot displays the principal angles (algorithm of Figure 4.8 with $W_1 = \Phi^{-1/2}_{[Y_f/U_f^\perp, Y_f/U_f^\perp]}$), which allow the user to make a decision on the order of the system. The order (9 in this case) can be selected by clicking and dragging with the left mouse over the desired orders.

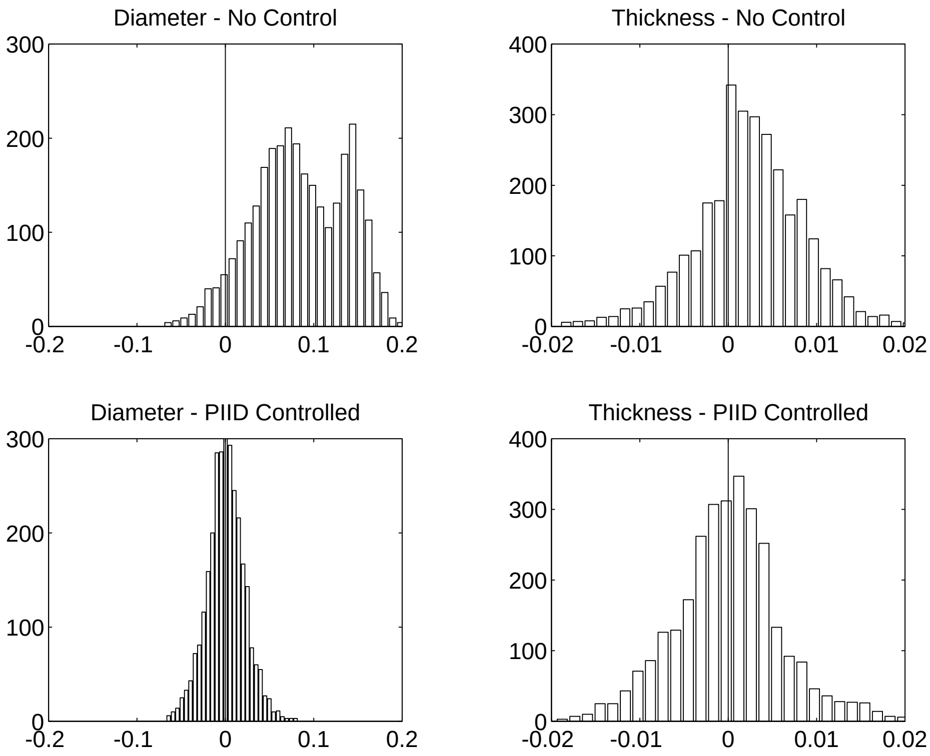

Figure 6.12 Illustration of the quality improvement. The top two figures show a histogram of the measured diameter and thickness without the optimal controller installed. The reference setpoint for production is at zero (the vertical line). Clearly, both diameter and thickness are too large (on average). Especially the diameter does not satisfy the production specifications. The bottom two figures show the histograms of the controlled system. The variance on the diameter is a factor two smaller. The mean diameter is also exactly at its reference. The variance of the thickness is not reduced (not that important in the specifications). However the mean value is right at the specification now. This figure clearly illustrates the benefits of subspace identification and of model-based control system design.

6.4 PRACTICAL EXAMPLES IN MATLAB

In this Section, we summarize identification results for ten different practical data sets. They indicate that the robust algorithm of Figure 4.8 fastly computes accurate models. Refinement of these models through a prediction error approach is possible, but at a higher computational expense.

The results were obtained using the Matlab implementations of the subspace algorithms described in this book. For a full overview of all the implemented algorithms, see Appendix B. To ensure maximal reproducibility, we included most of the data files on the diskette accompanying the book. This diskette also contains M-files which allow for the recomputation of the examples.

Note that some of these (and also other) applications are described in [Abd 94] [AML 94] [Cho 93] [CK 93] [Chu 94] [DMVO 94] [FVOMHL 94] [Gyu 93] [LS 91] [LSS 92] [VVDVOV 94] [VODM 93c] [VODMAKB 94] [ZVODML 94].

First we shortly describe the ten different practical processes. Some relevant numbers are displayed in Table 6.2.

1. **Glass Tubes.** This is the same process as in Section 6.2, however a different input-output data set is used. The input for this experiment is a filtered pseudo random binary noise sequence. The relevant Matlab files are `appl1.m` and `appl1.mat`.
2. **Dryer.** Laboratory setup acting like a hair dryer. Air is fanned through a tube and heated at the inlet. The air temperature is measured by a thermocouple at the output. The input is the voltage over the heating device (a mesh of resistor wires). The data was adopted from [Lju 87] [Lju 91c]. The relevant Matlab files are `appl2.m` and `appl2.mat`.
3. **Glass Oven.** The glass oven has 3 inputs (2 burners and 1 ventilator) and 6 outputs (temperature measured in a plane). The data has been pre-processed : detrending, peak shaving, delay estimation and normalization. For more information about the data itself and the pre-processing steps we refer to [Bac 87]. The relevant Matlab files are `appl3.m` and `appl3.mat`.
4. **Flutter.** Wing flutter data. Due to industrial secrecy agreements we are not allowed to reveal more details[11]. Important to know is that the input is highly colored. The relevant Matlab files are `appl4.m` and `appl4.mat`.

[11] This constraint may be un-allowable for the pure scientist, whose scientific beliefs may be shocked by such a non-disclosure agreement and who may argue that the essential requirements of scientific research -namely full experimental detail to guarantee complete reproducibility by independent third parties- are

5. **Robot.** Data from a flexible robot arm[12]. The arm is installed on an electrical motor. We have modeled the transfer function from the measured reaction torque of the structure on the ground to the acceleration of the flexible arm. The applied input is a periodic sine sweep. The relevant Matlab files are `appl5.m` and `appl5.mat`.

6. **Evaporator.** A four-stage evaporator to reduce the water content of a product, for example milk. The 3 inputs are feed flow, vapor flow to the first evaporator stage and cooling water flow. The three outputs are the dry matter content, the flow and the temperature of the outcoming product. The process is described in more detail in [ZVODML 94]. The relevant Matlab files are `appl6.m` and `appl6.mat`.

7. **Chemical.** A chemical process with 6 inputs and 7 outputs. Due to reasons of industrial secrecy, no more details can be revealed[11]. We have however included this process because it contains measurements of a quite complicated multivariable process. Unfortunately we were not allowed to include this data set on the diskette.

8. **CD Player.** Data from the mechanical construction of a CD player arm. The inputs are the forces of the mechanical actuators while the outputs are related to the tracking accuracy of the arm. The data was measured in closed loop, and then through a two-step procedure (as described in [VDHS 93]) converted to open loop equivalent data[13]. The inputs are highly colored. The relevant Matlab files are `appl8.m` and `appl8.mat`.

9. **Ball & Beam.** The ball and beam system consists of a horizontal beam with a ball placed on top of it[14]. The angle of the beam (with the horizontal axis) can be controlled, making the ball roll back and forth on the beam. The system is considered to be linear around its horizontal working point. The input of the process is the angle (in radians) of the beam. The output of the process is the velocity of the ball. In fact we are interested in the position of the ball, but that means we have to identify an integrator (one integrator from beam position to ball position). We have thus differentiated the position signal to obtain the ball velocity. The relevant Matlab files are `appl9.m` and `appl9.mat`.

---

not respected. This scientist is however free to disregard this data-set and concentrate on the eight other publically available sets.

[12]We are grateful to Hendrik Van Brussel and Jan Swevers of the laboratory of Production Engineering, Machine Design and Automation of the Katholieke Universiteit Leuven, who provided us with these data, which were obtained in the framework of the Belgian Programme on Interuniversity Attraction Poles (IUAP-nr.50) initiated by the Belgian State - Prime Minister's Office - Science Policy Programming.

[13]We are grateful to R. de Callafon of the Mechanical Engineering Systems and Control group of Delft and to the Philips Research Laboratories, who provided us with these data.

[14]We are greatful to Bart Motmans of the Department of Electrical Engineering of the Katholieke Universiteit Leuven, who provided us with these data.

	m	l	s_{id}	s_{val}	n	i
Glass Tubes	2	2	900	426	8	20
Dryer	1	1	500	500	4	15
Glass Oven	3	6	900	347	5	10
Flutter	1	1	1024	—	6	20
Robot	1	1	800	224	5	20
Evaporator	3	3	3300	3005	5	15
Chemical	6	7	1000	501	4	10
CD Player	2	2	1024	1024	7	20
Ball & Beam	1	1	1000	—	2	20
Wall Temp.	2	1	1200	480	3	20

Table 6.2 Overview of the ten practical examples. The first and second column contain the number of inputs (m) and outputs (l). The third and fourth column display the number of data points s used for identification s_{id} and validation s_{val}. Note that for the Flutter and Ball & Beam example no validation data was used. The reason for this is that the data sets did not contain enough information to be split in two (identification on a shorter data set gave unacceptable results). The fifth column indicates the system order n (the order of the state space model). Finally, the sixth column indicates the user defined index i, being the number of block rows used in the input-output block Hankel matrices in algorithm A1, A2 and A3 (see further).

10. **Wall Temp.** Heat flow density through a two layer wall (brick and insulation layer). The inputs are the internal and external temperature of the wall. The output is the heat flow density through the wall. The data was adopted from [Sys 1994]. The relevant Matlab files are `appl10.m` and `appl10.mat`.

On each data set we have applied 7 different identification algorithms :

A1: The subspace algorithm `com_alt.m` of Figure 4.6.

A2: The subspace algorithm `com_stat.m` of Figure 4.7.

A3: The robust subspace algorithm `subid.m` of Figure 4.8.

P1: The prediction error algorithm `pem.m` of the Matlab identification toolbox [Lju 91c]. As an initial guess we took the result of the command `canstart.m` in the same toolbox, which implements an instrumental variable method. See [Lju 91c] for more information.

O1: The output error algorithm `oe.m` of the Matlab identification toolbox [Lju 91c]. As an initial guess we took the (output error) result of the command `canstart.m` in the same toolbox.

P2: The prediction error algorithm described in [McK 94a], which uses a full parametrization of the state space model combined with regularization. The implementation in Matlab of the algorithm was obtained from McKelvey [McK 94c]. We did not include this implementation on the diskette, however it can be easily obtained through World Wide Web from control.isy.liu.se (directory /pub/Software/SSID). As an initial starting value we took the result of the robust subspace identification algorithm A3.

O2: The output error algorithm described in [McK 94a], which uses a full parametrization of the state space model combined with regularization. The implementation in Matlab of the algorithm was obtained from McKelvey [McK 94c]. Similarly, this software can be obtained from the same World Wide Web site. As an initial starting value we took the input-output part of the robust identification algorithm A3.

Table 6.3 shows the *simulation* errors (in percentage) for identification and validation data and for all cases and examples. The simulation errors are computed as (with $(y_k^s)_c$ channel "c" of the simulated output) :

$$\epsilon = 100 \cdot \frac{1}{l} \cdot \sum_{c=1}^{l} \left[\sqrt{\frac{\sum_{k=1}^{s}((y_k)_c - (y_k^s)_c)^2}{\sum_{k=1}^{s}((y_k)_c)^2}} \right] \% \, . \tag{6.5}$$

A Matlab function to compute these simulation errors has been implemented in `simul.m`. See also Appendix B..

Table 6.4 shows the *prediction* errors (in percentage) for identification and validation data and for all cases and all examples. These prediction errors were computed as in (6.5) but with $(y_k^s)_c$ replaced by $(y_k^p)_c$ the one step ahead predicted output.

A Matlab function to compute these prediction errors has been implemented in `predic.m`. See also Appendix B..

An entry "∞" in the tables indicates that the identified system was unstable. An entry "b" indicates that the optimization procedure did not start up due to a bug in `canstart()` [15]. An entry "$\sim$" indicates that the input-output part of the initial

[15] In the version of Matlab we used (Version 4.2), the bug in `canstart()` shows up when the number of states is smaller than the number of outputs (this is the case for the Glass Oven and the Chemical example). One of the observability indices has to be chosen equal to zero, which crashes `canstart()`. Through private conversation with Prof. Lennart Ljung, we have learned about this bug, which will be fixed in the next release.

model was unstable (for O1). Finally "—" indicates that there was no validation data at hand.

Table 6.5 shows the computational complexity of the algorithms. This is the number of floating point operations as computed by Matlab.

These examples and Tables justify the heuristics in the derivation of the third combined algorithm of Figure 4.8. We conclude that this third algorithm is a fast, robust and accurate algorithm that works well on practical and industrial examples. We also conclude that (if wanted) the computed model is an excellent starting value for optimization based identification algorithms.

6.5 CONCLUSIONS

In this final Chapter, we have shown how the subspace algorithms can be easily implemented using tools from the numerical linear algebra: the QR and Singular Value decomposition.

We have also demonstrated in some detail the efficiency of using subspace identification algorithms and the **GUI**-*based software tool* **ISID** *on an industrial glass tube manufacturing process.*

We have illustrated the power of the robust subspace identification algorithm on ten practical data sets. In the mean time, we (and others) have built up additional equally successful experiences with other industrial production processes [Abd 94] [AML 94] [Cho 93] [CK 93] [Chu 94] [DMVO 94] [FVOMHL 94] [Gyu 93] [LS 91] [LSS 92] [VVDVOV 94] [VODM 93c] [VODMAKB 94] [ZVODML 94].

Our faith in the applicability of our new subspace identification algorithms is confirmed by the implementation of our robust algorithm in different commercial computer aided control design packages such as Matlab's System Identification Toolbox [Lju 91c, Version 4], and Xmath's **ISID** *[AKVODMB 93] [VODMAKB 94].*

	A1	A2	A3	P1	O1	P2	O2
Glass Tubes	57.1	35.8	**35.5**	133	$\sim$	37.4	49.5
Dryer	8.22	8.2	**8.2**	8.66	8.6	**7.09**	8.34
Glass Oven	39.8	39.9	**39.7**	b	b	39.5	**34.8**
Flutter	∞	∞	**25.6**	106	29.6	25	**15.5**
Robot	∞	∞	**4.62**	∞	$\sim$	8.23	6.58
Evaporator	34.3	**33.7**	34.3	40.7	593	34.9	**30.5**
Chemical	468	∞	**64.2**	b	b	65.8	**60.6**
CD Player	∞	∞	**18.4**	∞	$\sim$	**17.3**	20.5
Ball & Beam	132	∞	**53.9**	72.6	$\sim$	71.8	**46.9**
Wall Temp.	40.5	12.1	**11.8**	∞	$\sim$	11.8	**11.7**

	A1	A2	A3	P1	O1	P2	O2
Glass Tubes	36.1	17	**15.8**	91.7	$\sim$	18.4	23
Dryer	7.45	**7.39**	7.49	**7.21**	7.33	7.4	7.47
Glass Oven	**32.7**	34.1	32.8	b	b	34.9	**31.4**
Flutter	—	—	—	—	—	—	—
Robot	∞	∞	**2.44**	∞	3.68	4.71	3.41
Evaporator	32.4	**31.2**	31.9	39.9	558	33.3	34.5
Chemical	365	∞	**60.2**	b	b	60.7	63
CD Player	∞	∞	**20.7**	∞	$\sim$	20.7	**20.5**
Ball & Beam	—	—	—	—	—	—	—
Wall Temp.	25	7.49	**7.36**	∞	$\sim$	7.41	7.42

Table 6.3 Simulation errors (in percentage (6.5)) for the identification data (top) and validation data (bottom). "∞" indicates that the input-output part of the identified model was unstable, "$\sim$","—" or "b" indicate that these entries could not be computed (see text). In each row, the entry of the most accurate model and the most accurate subspace model are highlighted. Clearly, of the subspace algorithms, the robust algorithm (A3) of Figure 4.8 is the most accurate for almost all examples. Moreover, the first and second subspace algorithm often compute really bad results, especially when the input data is colored : Glass Tubes, Flutter, Robot, CD Player, Ball & Beam. This result justifies the heuristics involved when deriving the third combined algorithm. From the last four columns, it follows that O2 (and P2) compute the most accurate models for the identification data. For the validation data however, A3 seems to perform better. The instrumental variable method used to start up P1 and O1 does not perform very well. We conclude from these tables that the robust combined identification algorithm (A3) computes accurate models, and that these models (if needed) provide excellent initial starting values for optimization algorithms.

	A1	A2	A3	P1	O1	P2	O2
Glass Tubes	11.2	**9.87**	9.89	23.7	$\sim$	**7.49**	49.5
Dryer	**3.31**	**3.31**	**3.31**	3.27	8.6	**3.26**	8.34
Glass Oven	**10.3**	**10.3**	**10.3**	b	b	**10**	34.8
Flutter	++	++	**1.49**	**0.104**	29.6	1.23	15.5
Robot	28.7	883	**1.22**	56.8	$\sim$	**0.594**	6.58
Evaporator	20.1	**19.9**	20	21.1	593	**19**	30.5
Chemical	102	75	**51.8**	b	b	**44.2**	60.6
CD Player	++	++	**12.2**	83.2	$\sim$	**5.95**	20.5
Ball & Beam	65.3	44	**36.5**	36.7	$\sim$	**36.2**	46.9
Wall Temp.	37	12	**11.8**	216	$\sim$	**11.7**	**11.7**

	A1	A2	A3	P1	O1	P2	O2
Glass Tubes	8.28	**7.21**	7.23	17.6	$\sim$	**5.61**	23
Dryer	**3.15**	**3.15**	**3.15**	**3.07**	7.33	**3.07**	7.47
Glass Oven	**7.66**	**7.66**	7.67	b	b	**7.43**	31.4
Flutter	—	—	—	—	—	—	—
Robot	15.4	362	**0.741**	26.9	3.68	**0.335**	3.41
Evaporator	**15.5**	15.7	15.8	16.4	558	**15.1**	34.5
Chemical	90.8	70.2	**59.1**	b	b	**52.6**	63
CD Player	++	++	**12.4**	83.3	$\sim$	**5.93**	20.5
Ball & Beam	—	—	—	—	—	—	—
Wall Temp.	22.6	7.44	**7.34**	136	$\sim$	7.38	7.42

Table 6.4 Prediction errors (in percentage (6.5)) for the identification data (top) and validation data (bottom). "++" indicates that the one step ahead prediction computed large errors (> 10.000), "$\sim$", "—" or "b" indicate that these entries could not be computed (see text). In each row, the entry of the most accurate model and the most accurate subspace model are highlighted. Just as for the simulation errors, among the subspace algorithms, the robust algorithm (A3) of Figure 4.8 is the most accurate for almost all examples. Again this justifies the heuristics involved when deriving the third combined algorithm. From the last four columns, it follows that P2 computes the most accurate models (for almost all cases). Again, we conclude from these tables that the robust combined identification algorithm (A3) computes accurate models, and that these models (if needed) provide excellent initial starting values for optimization algorithms.

	Mfl	A1	A2	A3	P1	O1	P2	O2	D1	D2	D3
Glass Tubes	48	1.11	**1**	1.2	5.9	$\sim$	21	12	$\star$	$\star$	$\star$
Dryer	4	1.09	**1**	1.13	2.1	1.3	3.2	3	.29	.20	.33
Glass Oven	63	1.06	**1**	1.15	b	b	35	21	.32	.26	.41
Flutter	14	1.06	**1**	1.09	4.6	3	6.5	13	$\star$	$\star$	$\star$
Robot	10	1.06	**1**	1.1	3.3	$\sim$	8.8	7.8	$\star$	$\star$	$\star$
Evaporator	200	1.03	**1**	1.06	4	2.5	5.8	3.8	$\star$	$\star$	$\star$
Chemical	140	1.12	**1**	1.33	b	b	26	16	.40	.28	.61
CD Player	54	1.09	**1**	1.16	4.6	4.6	15	7.6	$\star$	$\star$	$\star$
Ball & Beam	6	2.18	2.07	2.22	**1**	$\sim$	2.9	3.3	.33	.23	.38
Wall Temp.	30	1.4	1.26	1.46	**1**	$\sim$	2.9	2.6	.28	.14	.34

Table 6.5 Computational complexity of the algorithms i.e. the number of floating point operations (flops) computed by Matlab. All entries are relative to the basis number of mega flops in the first column. The second combined subspace algorithm is the fastest, while the third algorithm is the slowest of the three subspace algorithms (but the most accurate) especially when there is more than one input and/or output (Glass Tubes, Glass Oven, Chemical). The optimization based algorithms are a lot slower for multivariable systems (up to a factor 35 for the Chemical example). Especially the optimizations based on a fully parametrized model (O2 and P2) are slow. The last three columns (D1, D2 and D3) indicate the number of flops when computing the R factor of the RQ decomposition making use of the Hankel structure (using displacement rank). As indicated in Subsection 6.1.1, this method is not numerically stable for all cases (cfr. the "$\star$" entries). When it is stable however, another factor 3 to 5 can be gained from using these displacement algorithms (this provides a nice motivating example for accurate numerical implementation of displacement rank algorithms). A technical note is the fact that the number of floating point operations is not always directly related to the run time. The implementation of the optimization algorithms demands many `for-end` loops, which tend to be slow in Matlab. The real execution time is thus even higher for the optimization based algorithms as would be deduced from this table.

7
CONCLUSIONS AND OPEN PROBLEMS

7.1 CONCLUSIONS

In this book we have treated the theory, implementation and application of *subspace identification* algorithms for linear time-invariant systems.

- The **theory** of subspace identification algorithms has been presented in detail. Deterministic, stochastic and combined deterministic-stochastic subspace identification algorithms have each been treated separately in Chapter 2, 3 and 4. For each case, the geometric properties have been stated in a main Theorem 2, 8 and 12, which has led to new algorithms. The connections to the existing algorithms have been indicated, as the interconnections between the different cases (Section 4.5).

  The subspace identification theory has been linked to the theory of frequency weighted model reduction in Chapter 5, which has led to new interpretations and insights. The theoretical development has evolved into a practically usable and robust subspace system identification algorithm (Figure 4.8).

- **Implementation** of the subspace identification algorithms has been discussed in terms of the numerically stable and efficient RQ and singular value decompositions in Section 6.1. The algorithms have then been implemented together with a whole scale of classical identification algorithms and processing and validation tools in a commercially available graphical user interface toolbox: **ISID**. The motivation for the graphical user interface has been given in Section 6.2.

  This book also contains a set of Matlab functions which implement the subspace algorithms in this book. These Matlab functions are easy to use and enhance the understanding as well as the applicability of the algorithms.

- One **application** of the algorithms to an industrial glass tube manufacturing process has been presented in Section 6.3. This application has been used to illustrate the power and user-friendliness of the subspace identification algorithms and of their implementation in **ISID**. The identified model has allowed for an optimal control of the process, which has led to a significant enhancement of the production quality.

  The applicability of subspace identification algorithms to practical (industrial) data has been further illustrated with a short description of ten practical applications in Section 6.4. These examples can be easily reproduced by simple running the included M-files.

7.2 OPEN PROBLEMS

Subspace identification is a young but promising field of research and there are still many open problems to be solved. In this Section we summarize the open problems that have been encountered when writing this book. The solution of these problems would contribute significantly to the maturing of the field of subspace identification.

- A first and obvious problem is that of the statistical analysis. Throughout the book, the asymptotic (un)biasedness of the algorithms has been indicated. This however doesn't indicate in any way how the algorithms perform when only a finite number of samples is available, nor does it say anything about the asymptotic statistical distribution of the results. First attempts for a statistical analysis have been made in [OV 94] [VOWL 91] [VOWL 93] (and related work for subspace tracking in [Ott 89] [OVSN 93] [OVK 92] [VO 91] [Vib 89]). In [VTS 95] a first order analysis of stochastic realization algorithms is made. This technique could be extended to the algorithms described in this book.

- Closely related to the previous point is the choice of the weights W_1 and W_2. In each of the main Theorems 2, 8 and 12, two weighting matrices W_1 and W_2 have been introduced. In Chapter 5, the effect of these weighting matrices on the state space basis in which the results are obtained has been presented. However, it is not at all clear which weightings are statistically optimal (in a sense that the covariance matrix of the results is minimal). In [Lar 94] it is claimed that the **CVA** algorithm is optimal (see also Subsection 4.3.3), it is however not proved. In [DPS 94] the relative efficiency of stochastic subspace identification methods is shown by example. Expressions for the finite sample statistics (previous point) would however indicate and prove which weights are statistically optimal.

- The order decision is fairly heuristic. Indeed, when the number of measurements is infinite, a number of singular values will be exactly equal to zero. However for a finite number of samples, one has to look for a gap in the singular value spectrum. As an alternative (and more theoretically grounded) method, statistical significance tests could be developed. A first step in that direction has been described in an Appendix of [CXK 94a].

- For the further analysis of the subspace identification algorithms, it would help to investigate the connection with the classical identification techniques (see for instance [Lju 87]). In [JW 94] a connection is made to the classical ARX modeling, while in [OV 94] [VOWL 91] [VOWL 93] an instrumental variable interpretation of the algorithms of Chapter 4 is presented. Another issue addressed in these papers is the option of splitting the past and the future in unequal parts. Throughout this book we have assumed that both have an equal number of block rows i. The effect of having α block rows in the future and $\beta \neq \alpha$ block rows in the past should still be investigated.

- Frequency domain extensions of the subspace identification ideas could lead to fast, efficient and always convergent frequency domain identification techniques. Algorithms based on the deterministic projection algorithm of Section 2.3.2 have been described in [LJM 94] [McK 94b]. Further extension (to more general noise cases) and investigation of these algorithms could lead to interesting algorithms and interpretations.

- Subspace identification algorithms as described in this book are completely black-box, in a sense that no a-priori knowledge can be included. However, in practice, many facets of the process are often known (integrator, stability, DC gain, rise times, ...). The possibility to include these constraints in the subspace identification algorithms would greatly enhance their practical value. As described in this book, the stability constraint can be included [Mac 94], but how to include the other constraints is still an open problem.

- One of the main assumption of the main Theorems is that the process and measurement noise are independent of the input u_k. This assumption is violated when the system is working in closed loop. Subspace identification techniques could be extended to also work under these conditions. For the **MOESP** framework, this has been treated in [Ver 93]. Other approaches worth investigating are the two step method [VDHS 93] and the framework presented by Schrama [Sch 91]. Preliminary results on this last approach can be found in [Wan 94]. It has been shown in for instance [Gev 93] that closed loop identification is very useful (if not necessary) when doing identification for control.

- The stochastic identification problem has a nice symmetry in it: The forward and backward innovation model. This symmetry is lost in the combined deterministic-stochastic identification problem. A topic of further research could be the quest for symmetric models (as in the stochastic case) for the combined problem.

- The error bounds for the frequency weighted model reduction provided by Enns [Enn 84] unfortunately contain an unknown factor α. Finding an upper bound for this factor is still an open problem. Another related issue is that of finding error bounds for the lower order identified models (see Chapter 5), since in this case the lower order model is not obtained by a mere truncation of the balanced high order model. A last open problem is the exact interpretation of the state space basis of the **MOESP** and **CVA** algorithm, for non-white noise inputs (see Section 5.4).

A
PROOFS

A.1 PROOF OF FORMULA (2.16)

First we prove that:

$$\text{rank } W_p = \text{rank } W_p/\boldsymbol{U}_f^{\perp} \ . \tag{A.1}$$

Using (2.5), W_p can be rewritten as:

$$W_p = \begin{pmatrix} I_{mi} & 0 \\ H_i^d & \Gamma_i \end{pmatrix} \begin{pmatrix} U_p \\ X_p^d \end{pmatrix} \ ,$$

which implies that $W_p/\boldsymbol{U}_f^{\perp}$ can be written as:

$$W_p/\boldsymbol{U}_f^{\perp} = \begin{pmatrix} I_{mi} & 0 \\ H_i^d & \Gamma_i \end{pmatrix} \begin{pmatrix} U_p/\boldsymbol{U}_f^{\perp} \\ X_p^d/\boldsymbol{U}_f^{\perp} \end{pmatrix} \ .$$

Due to the first two conditions of Theorem 2, the following holds:

$$\text{rank } \begin{pmatrix} U_p \\ X_p^d \end{pmatrix} = \text{rank } \begin{pmatrix} U_p/\boldsymbol{U}_f^{\perp} \\ X_p^d/\boldsymbol{U}_f^{\perp} \end{pmatrix} \ .$$

This proves Formula (A.1). Now, denote the singular value decomposition of $W_p/\boldsymbol{U}_f^{\perp}$ as:

$$W_p/\boldsymbol{U}_f^{\perp} = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} S_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} = U_1 S_1 V_1^T \ . \tag{A.2}$$

Since $W_p/\boldsymbol{U}_f^\perp$ is a linear combination of the *columns* of W_p, and since the rank of W_p and $W_p/\boldsymbol{U}_f^\perp$ are equal, we find that the column spaces of W_p and $W_p/\boldsymbol{U}_f^\perp$ are equal. This implies that W_p can be written as:

$$W_p = U_1 R \; . \tag{A.3}$$

Finally, with the singular value decomposition (A.2) and using (A.3), we find:

$$\begin{aligned}
[W_p/\boldsymbol{U}_f^\perp].[W_p/\boldsymbol{U}_f^\perp]^\dagger W_p &= [U_1 S_1 V_1^T].[V_1 S_1^{-1} U_1^T].[U_1 R] \\
&= U_1.\underbrace{S_1 V_1^T V_1 S_1^{-1}}_{=I}.\underbrace{U_1^T U_1}_{=I}.R \\
&= U_1 R \\
&= W_p \; ,
\end{aligned}$$

which proves (2.16). □

A.2 PROOF OF THEOREM 6

We first prove that Theorem 6 is true for $k = 1$. Then we prove that if the Theorem 6 is true for $k = p$, it is also true for $k = p + 1$. This then completes the proof.

$\underline{\boldsymbol{k} = 1}$

From (3.20), we find:

$$\begin{aligned}
\hat{x}_1 &= A \underbrace{\hat{x}_0}_{=0} + K_0(y_0 - C \underbrace{\hat{x}_0}_{=0}) \\
&= K_0 y_0 \; .
\end{aligned}$$

From (3.21), we find:

$$\begin{aligned}
K_0 &= (G - A \underbrace{P_0}_{=0} C^T)(\Lambda_0 - C \underbrace{P_0}_{=0} C^T)^{-1} \\
&= G.\Lambda_0^{-1} \\
&= \Delta_1^c.L_1^{-1} \; .
\end{aligned}$$

Which implies that:

$$\hat{x}_1 = \Delta_1^c.L_1^{-1}.y_0 \ ,$$

which is exactly (3.23) for $k = 1$. As a side result, we find from (3.22) that for $k = 1$:

$$\begin{aligned} P_1 &= A \underbrace{P_0}_{=0} A^T + (G - A \underbrace{P_0}_{=0} C^T).(\Lambda_0 - C \underbrace{P_0}_{=0} C^T)^{-1}.(G - A \underbrace{P_0}_{=0} C^T)^T \\ &= G.\Lambda_0^{-1}.G^T \\ &= \Delta_1^c.L_1^{-1}.(\Delta_1^c)^T \ , \end{aligned} \tag{A.4}$$

which is (3.24) for $k = 1$.

$\underline{\boldsymbol{k = p \ \Rightarrow \ k = p + 1}}$

We first prove that if (3.24) is true for $k = p$, it will also be true for $k = p + 1$. In the following derivation, we use the matrix inversion lemma of [Kai 80].

$$\begin{aligned} P_{p+1} &= \Delta_{p+1}^c.L_{p+1}^{-1}.(\Delta_{p+1}^c)^T \\ &= \begin{pmatrix} A\Delta_p^c & G \end{pmatrix}.\begin{pmatrix} L_p & (\Delta_p^c)^T C^T \\ C\Delta_p^c & \Lambda_0 \end{pmatrix}^{-1}.\begin{pmatrix} (\Delta_p^c)^T A^T \\ G^T \end{pmatrix} \\ &= \begin{pmatrix} A\Delta_p^c & G \end{pmatrix}.\left(\begin{array}{c|c} L_p^{-1} + L_p^{-1}(\Delta_p^c)^T C^T \Delta^{-1} C\Delta_p^c L_p^{-1} & -L_p^{-1}(\Delta_p^c)^T C^T \Delta^{-1} \\ -\Delta^{-1} C\Delta_p^c L_p^{-1} & \Delta^{-1} \end{array} \right).\begin{pmatrix} (\Delta_p^c)^T A^T \\ G^T \end{pmatrix} \\ &\quad \text{with } \Delta = \Lambda_0 - C \underbrace{\Delta_p^c L_p^{-1}(\Delta_p^c)^T}_{=P_p} C^T = \Lambda_0 - CP_pC^T \\ &= A \underbrace{\Delta_p^c L_p^{-1}(\Delta_p^c)^T}_{=P_p} A^T + (G - A \underbrace{\Delta_p^c L_p^{-1}(\Delta_p^c)^T}_{=P_p} C^T)\Delta^{-1}(G - A \underbrace{\Delta_p^c L_p^{-1}(\Delta_p^c)^T}_{=P_p} C^T)^T \\ &= AP_pA^T + (G - AP_pC^T).(\Lambda_0 - CP_pC^T)^{-1}.(G - AP_pC^T)^T \ . \end{aligned}$$

The last equation clearly indicates that the matrix P_{p+1} calculated from (3.22) and from (3.24) are the same. Which thus proves (3.24). Now, we are ready to prove (3.23). We assume that (3.23) is satisfied for $k = p$:

$$\hat{x}_p = \Delta_p^c.L_p^{-1}.\begin{pmatrix} y_0 \\ \vdots \\ y_{p-1} \end{pmatrix} \ .$$

We should prove that:

$$\hat{x}_{p+1} = \Delta^c_{p+1}.L^{-1}_{p+1}.\begin{pmatrix} y_0 \\ \vdots \\ y_p \end{pmatrix}$$

is indeed true (using the recursive formulas (3.20)-(3.22)).

$$\begin{aligned}
\hat{x}_{p+1} &= \Delta^c_{p+1}.L^{-1}_{p+1}.\begin{pmatrix} y_0 \\ \vdots \\ y_p \end{pmatrix} \\
&= \begin{pmatrix} A\Delta^c_p & G \end{pmatrix}.\begin{pmatrix} L_p & (\Delta^c_p)^T C^T \\ C\Delta^c_p & \Lambda_0 \end{pmatrix}^{-1}.\begin{pmatrix} y_0 \\ \vdots \\ y_p \end{pmatrix} \\
&= \begin{pmatrix} A\Delta^c_p & G \end{pmatrix} \\
&\quad .\begin{pmatrix} L_p^{-1} + L_p^{-1}(\Delta^c_p)^T C^T \Delta^{-1} C\Delta^c_p L_p^{-1} & -L_p^{-1}(\Delta^c_p)^T C^T \Delta^{-1} \\ -\Delta^{-1} C\Delta^c_p L_p^{-1} & \Delta^{-1} \end{pmatrix}.\begin{pmatrix} y_0 \\ \vdots \\ y_p \end{pmatrix} \\
&= [A - (G - A\underbrace{\Delta^c_p L_p^{-1}(\Delta^c_p)^T}_{=P_p} C^T)\Delta^{-1}C]\underbrace{\Delta^c_p L_p^{-1}\begin{pmatrix} y_0 \\ \vdots \\ y_{p-1} \end{pmatrix}}_{=\hat{x}_p} \\
&\quad +(G - A\underbrace{\Delta^c_p L_p^{-1}(\Delta^c_p)^T}_{=P_p} C^T)\Delta^{-1} y_p \\
&= A\hat{x}_p + \underbrace{(G - AP_pC^T)(\Lambda_0 - CP_pC^T)^{-1}}_{=K_p} .\,(y_p - C\hat{x}_p) \\
&= A\hat{x}_p + K_p.(y_p - C\hat{x}_p)\ .
\end{aligned}$$

The last equation clearly indicates that the state $\hat{x}_{p+1}$ calculated from (3.20) and from (3.23) are the same. Which thus proves (3.23).

□

A.3 NOTE ON THE SPECIAL FORM OF THE KALMAN FILTER

In this Appendix, we give two different forms of the recursive Kalman filter. The first one is the classical form of for instance [AW 84]. This form is transformed into the form of (3.20)-(3.22), which is more useful for this book.

Consider the system (3.1)-(3.3), where we assume that A, C, Q, S and R are known. Given $\hat{x}_0$, $\widetilde{P}_0$ and $y_0, \ldots, y_{k-1}$, the non-steady state Kalman filter state estimate $\hat{x}_k$ is then given by the following set of recursive formulas [AW 84]:

$$\hat{x}_k = A\hat{x}_{k-1} + K_{k-1}(y_{k-1} - C\hat{x}_{k-1}) \ , \tag{A.5}$$

with:

$$\begin{aligned} K_{k-1} &= (A\widetilde{P}_{k-1}C^T + S)(C\widetilde{P}_{k-1}C^T + R)^{-1} \ , \qquad (A.6) \\ \widetilde{P}_k &= A\widetilde{P}_{k-1}A^T + Q \\ &\quad -(A\widetilde{P}_{k-1}C^T + S)(C\widetilde{P}_{k-1}C^T + R)^{-1}(A\widetilde{P}_{k-1}C^T + S)^T \ . \qquad (A.7) \end{aligned}$$

and $\widetilde{P}_k$ the error covariance matrix:

$$\widetilde{P}_k = \mathbf{E}[(x_k^s - \hat{x}_k).(x_k^s - \hat{x}_k)^T] \ .$$

For our purpose, a different form of these recursive Kalman filter equations is more useful. With A, C, Q, S, R given, the matrices Σ^s, G and Λ_0 can be computed through the formulas in Figure 3.3. Now define the transformation:

$$\widetilde{P}_k \quad \leftarrow \quad \Sigma^s - P_k \ .$$

We then get (from (A.6)):

$$\begin{aligned} K_{k-1} &= (\underbrace{(A\Sigma^s C^T + S)}_{=G} - AP_{k-1}C^T)(\underbrace{(C\Sigma^s C^T + R)}_{=\Lambda_0} - CP_{k-1}C^T)^{-1} \\ &= (G - AP_{k-1}C^T)(\Lambda_0 - CP_{k-1}C^T)^{-1} \ . \end{aligned}$$

For the Riccati equation (from (A.7)) we have:

$$\Sigma^s - P_k \quad = \quad A\Sigma^s A^T - AP_{k-1}A^T + (\Sigma^s - A\Sigma^s A^T)$$

$$
\begin{aligned}
& -(\underbrace{(A\Sigma^s C^T + S)}_{=G} - AP_{k-1}C^T)(\underbrace{(C\Sigma^s C^T + R)}_{=\Lambda_0} - CP_{k-1}C^T)^{-1} \\
& .(\underbrace{(A\Sigma^s C^T + S)}_{=G} - AP_{k-1}C^T)^T \ , \\
P_k \quad = \quad & AP_{k-1}A^T + (G - AP_{k-1}C^T)(\Lambda_0 - CP_{k-1}C^T)^{-1}(G - AP_{k-1}C^T)^T \ .
\end{aligned}
$$

This means that the Kalman filter (3.20)-(3.22) calculates the same state estimate $\hat{x}_k$ as the "classical" Kalman filter (A.5)-(A.7) with $\widetilde{P}_k = \Sigma^s - P_k$.

A.4 PROOF OF THEOREM 8

From (3.17) and (3.18) we find for (3.33) and $j \to \infty$:

$$
\begin{aligned}
\mathcal{O}_i &= Y_f / \boldsymbol{Y}_p \\
&= C_i . L_i^{-1} . Y_p \ .
\end{aligned}
$$

Now from classical stochastic realization theory (analogous to deterministic realization [Kun 78] [ZM 74] of the covariance sequence), we find:

$$
\begin{aligned}
C_i &= \begin{pmatrix} \Lambda_i & \Lambda_{i-1} & \dots & \Lambda_2 & \Lambda_1 \\ \Lambda_{i+1} & \Lambda_i & \dots & \Lambda_3 & \Lambda_2 \\ \Lambda_{i+2} & \Lambda_{i+1} & \dots & \Lambda_4 & \Lambda_3 \\ \dots & \dots & \dots & \dots & \dots \\ \Lambda_{2i-1} & \Lambda_{2i-2} & \dots & \Lambda_{i+1} & \Lambda_i \end{pmatrix} \\
&= \begin{pmatrix} CA^{i-1}G & CA^{i-2}G & \dots & CAG & CG \\ CA^iG & CA^{i-1}G & \dots & CA^2G & CAG \\ CA^{i+1}G & CA^iG & \dots & CA^3G & CA^2G \\ \dots & \dots & \dots & \dots & \dots \\ CA^{2i-2}G & CA^{2i-3}G & \dots & CA^iG & CA^{i-1}G \end{pmatrix} \\
&= \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{i-1} \end{pmatrix} . \begin{pmatrix} A^{i-1}G & A^{i-2}G & \dots & AG & G \end{pmatrix} \\
&= \Gamma_i . \Delta_i^c \ .
\end{aligned}
$$

This implies that:

$$\mathcal{O}_i = \Gamma_i.\Delta_i^c.L_i^{-1}.Y_p \ .$$

Using (3.25) this leads to:

$$\mathcal{O}_i = \Gamma_i.\widehat{X}_i \ ,$$

which proves (3.35). Since W_1 is of full rank, and since the rank of $Y_p.W_2$ is equal to the rank of Y_p, we find that the rank of $W_1\mathcal{O}_i W_2$ is equal to the rank of $\mathcal{O}_i$, which in turn is equal to n (the system order). This is due to the fact that $\mathcal{O}_i$ is equal to the product of a matrix with n columns and a matrix with n rows. The singular value decomposition of the weighted projection $\mathcal{O}_i$ (3.34) can now be split in two parts:

$$W_1\Gamma_i = U_1 S_1^{1/2}.T \ , \tag{A.8}$$

$$\widehat{X}_i W_2 = T^{-1}.S_1^{1/2} V_1^T \ . \tag{A.9}$$

Equation (A.8) proves (3.36), while equation (A.9) proves (3.38). It is also clear that from $\mathcal{O}_i = \Gamma_i.\widehat{X}_i$ it follows that:

$$\widehat{X}_i = \Gamma_i^{\dagger}.\mathcal{O}_i \ ,$$

which proves (3.39). Equation (3.37) can be easily proven from the fact that:

$$\Phi_{[Y_f,Y_p]} = C_i = \Gamma_i.\Delta_i^c \ .$$

Finally, equation (3.40) follows directly from (3.31) combined with (3.19).

□

A.5 PROOF OF THEOREM 9

We first prove that Theorem 9 is true for $k = 1$. Then we prove that if the Theorem 9 is true for $k = p$, it is also true for $k = p + 1$. This completes the proof.

$\underline{k = 1}$

With (4.9) and (4.12) we find that:

$$\begin{aligned} \Omega_1 &= (G - AP_0C).(\Lambda_0 - CP_0C^T)^{-1} \\ &= K_0 \ . \end{aligned}$$

From (4.8), we thus find:

$$\begin{aligned}
\hat{x}_1 &= A\hat{x}_0 + Bu_0 + K_0(y_0 - C\hat{x}_0 - Du_0) \\
&= \left(\begin{array}{c|c|c} A - K_0C & B - K_0D & K_0 \end{array} \right) . \left(\begin{array}{c} \hat{x}_0 \\ \hline u_0 \\ \hline y_0 \end{array} \right) \\
&= \left(\begin{array}{c|c|c} A - \Omega_1\Gamma_1 & \Delta_1^d - \Omega_1 H_1^d & \Omega_1 \end{array} \right) . \left(\begin{array}{c} \hat{x}_0 \\ \hline u_0 \\ \hline y_0 \end{array} \right) ,
\end{aligned}$$

which is exactly (4.11) for $k = 1$. As a side result, we find from (4.10) that for $k = 1$:

$$\begin{aligned}
P_1 &= AP_0A^T + (G - AP_0C^T).(\Lambda_0 - CP_0C^T)^{-1}.(G - AP_0C^T)^T \\
&= AP_0A^T + (\Delta_1^c - AP_0\Gamma_1^T).(L_1 - \Gamma_1 P_0 \Gamma_1^T).(\Delta_1^c - AP_0\Gamma_1^T)^T ,
\end{aligned}$$

which is (4.13) for $k = 1$.

$\underline{\boldsymbol{k = p \ \Rightarrow \ k = p + 1}}$

We first prove that if (4.13) is true for $k = p$, it will also be true for $k = p + 1$. In the following derivation, we use the matrix inversion lemma of [Kai 80]. We also introduce the following convenient notation:

$$\begin{aligned}
\alpha_p &= (\Delta_p^c - A^p P_0 \Gamma_p^T) , \\
\beta_p &= (L_p - \Gamma_p P_0 \Gamma_p^T) , \\
\gamma &= G - A^{p+1} P_0 (A^T)^p C^T .
\end{aligned}$$

We thus find:

$$\begin{aligned}
P_{p+1} &= A^{p+1}P_0(A^T)^{p+1} + \alpha_{p+1}\beta_{p+1}^{-1}\alpha_{p+1}^T \\
&= A^{p+1}P_0(A^T)^{p+1} \\
&\quad +(\Delta_{p+1}^c - A^{p+1}P_0\Gamma_{p+1}^T)(L_{p+1} - \Gamma_{p+1}P_0\Gamma_{p+1}^T)^{-1}(\Delta_{p+1}^c - A^{p+1}P_0\Gamma_{p+1}^T)^T \\
&= A^{p+1}P_0(A^T)^{p+1} \\
&\quad + \left(\begin{array}{c|c} A[\Delta_p^c - A^pP_0\Gamma_p^T] & G - A^{p+1}P_0(A^T)^pC^T \end{array} \right) \\
&\quad . \left(\begin{array}{c|c} L_p - \Gamma_pP_0\Gamma_p^T & [(\Delta_p^c)^T - \Gamma_pP_0(A^T)^p]C^T \\ \hline C[\Delta_p^c - A^pP_0\Gamma_p^T] & \Lambda_0 - CA^pP_0(A^T)^pC^T \end{array} \right)^{-1} \\
&\quad . \left(\begin{array}{c} [(\Delta_p^c)^T - \Gamma_pP_0(A^T)^p]A^T \\ \hline G^T - CA^pP_0(A^T)^{p+1} \end{array} \right)
\end{aligned}$$

$$
\begin{aligned}
&= A^{p+1}P_0(A^T)^{p+1} \\
&\quad + \left(\begin{array}{c|c} A\alpha_p & \gamma \end{array} \right) \left(\begin{array}{c|c} \beta_p & \alpha_p^T C^T \\ \hline C\alpha_p & \Lambda_0 - CA^pP_0(A^T)^pC^T \end{array} \right)^{-1} \left(\begin{array}{c} \alpha_p^T A^T \\ \hline \gamma^T \end{array} \right) \\
&= A^{p+1}P_0(A^T)^{p+1} + \left(\begin{array}{c|c} A\alpha_p & \gamma \end{array} \right) \\
&\quad . \left(\begin{array}{c|c} \beta_p^{-1} + \beta_p^{-1}\alpha_p^T C^T \Delta^{-1} C\alpha_p\beta_p^{-1} & -\beta_p^{-1}\alpha_p^T C^T \Delta^{-1} \\ -\Delta^{-1}C\alpha_p\beta_p^{-1} & \Delta^{-1} \end{array} \right) \left(\begin{array}{c} \alpha_p^T A^T \\ \hline \gamma^T \end{array} \right) \\
&\quad \text{with } \Delta = \Lambda_0 - C\underbrace{[A^pP_0(A^T)^p + \alpha_p\beta_p^{-1}\alpha_p^T]}_{=P_p}C^T = \Lambda_0 - CP_pC^T \\
&= A\underbrace{[A^pP_0(A^T)^p + \alpha_p\beta_p^{-1}\alpha_p^T]}_{=P_p}A^T \\
&\quad + (\gamma - A\alpha_p\beta_p^{-1}\alpha_p^T C^T)\Delta^{-1}(\gamma - A\alpha_p\beta_p^{-1}\alpha_p^T C^T)^T \\
&= AP_pA^T + (G - A\underbrace{[A^pP_0(A^T)^p + \alpha_p\beta_p^{-1}\alpha_p^T]}_{=P_p}C^T) \\
&\quad .\Delta^{-1}(G - A\underbrace{[A^pP_0(A^T)^p + \alpha_p\beta_p^{-1}\alpha_p^T]}_{=P_p}C^T)^T \\
&= AP_pA^T + (G - AP_pC^T)(\Lambda_0 - CP_pC^T)^{-1}(G - AP_pC^T)^T \ .
\end{aligned}
$$

The last equation clearly indicates that the matrix P_{p+1} calculated from (4.10) and from (4.13) are the same. Which thus proves (4.13). Now, we are ready to prove (4.11). We assume that (4.11) is satisfied for $k = p$:

$$
\hat{x}_p = (A^p - \Omega_p\Gamma_p)\hat{x}_0 + (\Delta_p^d - \Omega_pH_p^d)\begin{pmatrix} u_0 \\ \vdots \\ u_{p-1} \end{pmatrix} + \Omega_p \begin{pmatrix} y_0 \\ \vdots \\ y_{p-1} \end{pmatrix} ,
$$

with $\Omega_p = \alpha_p\beta_p^{-1}$. We first prove that:

$$
\Omega_{p+1} = \begin{pmatrix} (A - K_pC)\Omega_p & K_p \end{pmatrix} . \tag{A.10}
$$

From the same formulas as before, we find:

$$
\begin{aligned}
\Omega_{p+1} &= \alpha_{p+1}\beta_{p+1}^{-1} \\
&= \left(\begin{array}{c|} A\alpha_p\beta_p^{-1} + A\alpha_p\beta_p^{-1}\alpha_p^T C^T\Delta^{-1}C\alpha_p\beta_p^{-1} - \gamma\Delta^{-1}C\alpha_p\beta_p^{-1} \end{array} \right. \\
&\qquad \left. -A\alpha_p\beta_p^{-1}\alpha_p^T C^T\Delta^{-1} + \gamma\Delta^{-1} \right)
\end{aligned}
$$

$$
\begin{aligned}
&= \left(\begin{array}{c|c} A\Omega_p - (\gamma - A\Omega_p\alpha_p^T C^T)\Delta^{-1} C\Omega_p & (\gamma - A\Omega_p\alpha_p^T C^T)\Delta^{-1} \end{array} \right) \\
&= \left(\begin{array}{c|c} A\Omega_p - (G - AP_pC^T)(\Lambda_0 - CP_pC^T)^{-1} C\Omega_p & (G - AP_pC^T)(\Lambda_0 - CP_pC^T)^{-1} \end{array} \right) \\
&= \left(\begin{array}{c|c} (A - K_pC)\Omega_p & K_p \end{array} \right) .
\end{aligned}
$$

We can now rewrite the estimate of $\hat{x}_{p+1}$:

$$
\begin{aligned}
\hat{x}_{p+1} &= (A^{p+1} - \Omega_{p+1}\Gamma_{p+1})\hat{x}_0 + (\Delta_{p+1}^d - \Omega_{p+1}H_{p+1}^d) \begin{pmatrix} u_0 \\ \vdots \\ u_p \end{pmatrix} + \Omega_{p+1} \begin{pmatrix} y_0 \\ \vdots \\ y_p \end{pmatrix} \\
&= (A^{p+1} - (A - K_pC)\Omega_p\Gamma_p - K_pCA^p)\hat{x}_0 \\
&\quad + (A\Delta_p^d - (A - K_pC)\Omega_p H_p^d - K_pC\Delta_p^d) \begin{pmatrix} u_0 \\ \vdots \\ u_{p-1} \end{pmatrix} + (B - K_pD)u_p \\
&\quad + (A - K_pC)\Omega_p \begin{pmatrix} y_0 \\ \vdots \\ y_{p-1} \end{pmatrix} + K_py_p \\
&= (A - K_pC). \left[(A^p - \Omega_p\Gamma_p)\hat{x}_0 + (\Delta_p^d - \Omega_pH_p^d) \begin{pmatrix} u_0 \\ \vdots \\ u_{p-1} \end{pmatrix} + \Omega_p \begin{pmatrix} y_0 \\ \vdots \\ y_{p-1} \end{pmatrix} \right] \\
&\quad + (B - K_pD)u_p + K_py_p \\
&= (A - K_pC).\hat{x}_p + (B - K_pD).u_p + K_p.y_p \; .
\end{aligned}
$$

The last formula clearly indicates that $\hat{x}_{p+1}$ is the propagation of the state $\hat{x}_p$ through formula (4.8). This proves the induction and thus (4.11).

□

A.6 PROOF OF THEOREM 11

From the first assumption of Theorem 11, we easily find that (see also Subsection 1.4.4):

$$
\begin{aligned}
\mathbf{E_j}[Y_p^s U_\cdot^T] &= 0 \qquad , \qquad \mathbf{E_j}[Y_p^s (X_\cdot^d)^T] = 0 \; , \\
\mathbf{E_j}[Y_f^s U_\cdot^T] &= 0 \qquad , \qquad \mathbf{E_j}[Y_f^s (X_\cdot^d)^T] = 0 \; ,
\end{aligned}
$$

where the subscript . denotes past or future. In the following we make use of (4.15) through (4.17) without explicitly mentioning it.

Proof of formula (4.18)

Define

$$\mathcal{A} \stackrel{\text{def}}{=} \mathbf{E_j}\,[\,Y_f.\left(\begin{array}{c|c|c} U_p^T & U_f^T & Y_p^T \end{array}\right)] = \left(\begin{array}{c|c|c} \mathcal{A}_1 & \mathcal{A}_2 & \mathcal{A}_3 \end{array}\right) \ .$$

We have:

$$\begin{aligned}
\mathcal{A}_1 &= \mathbf{E_j}[Y_f U_p^T] \\
&= \mathbf{E_j}\left[(\Gamma_i A^i X_p^d + \Gamma_i \Delta_i^d U_p + H_i^d U_f + Y_f^s).U_p^T\right] \\
&= \Gamma_i A^i S_p^{xu} + \Gamma_i \Delta_i^d R_p^{uu} + H_i^d (R_{pf}^{uu})^T \ ,
\end{aligned}$$

$$\begin{aligned}
\mathcal{A}_2 &= \mathbf{E_j}[Y_f U_f^T] \\
&= \mathbf{E_j}\left[(\Gamma_i A^i X_p^d + \Gamma_i \Delta_i^d U_p + H_i^d U_f + Y_f^s).U_f^T\right] \\
&= \Gamma_i A^i S_f^{xu} + \Gamma_i \Delta_i^d R_{pf}^{uu} + H_i^d R_f^{uu} \ ,
\end{aligned}$$

$$\begin{aligned}
\mathcal{A}_3 &= \mathbf{E_j}[Y_f Y_p^T] \\
&= \mathbf{E_j}\left[(\Gamma_i A^i X_p^d + \Gamma_i \Delta_i^d U_p + H_i^d U_f + Y_f^s).((X_p^d)^T \Gamma_i^T + U_p^T (H_i^d)^T + (Y_p^s)^T)\right] \\
&= \Gamma_i A^i \Sigma^d \Gamma_i^T + \Gamma_i A^i S_p^{xu} (H_i^d)^T + \Gamma_i \Delta_i^d (S_p^{xu})^T \Gamma_i^T + \Gamma_i \Delta_i^d R_p^{uu} (H_i^d)^T \\
&\quad + H_i^d (S_f^{xu})^T \Gamma_i^T + H_i^d (R_{pf}^{uu})^T (H_i^d)^T + C_i \\
&= \Gamma_i A^i \Sigma^d \Gamma_i^T + \Gamma_i \Delta_i^d R_p^{uu} (H_i^d)^T + \Gamma_i A^i S_p^{xu} (H_i^d)^T + \Gamma_i \Delta_i^d (S_p^{xu})^T \Gamma_i^T \\
&\quad + H_i^d (R_{pf}^{uu})^T (H_i^d)^T + H_i^d (S_f^{xu})^T \Gamma_i^T + \Gamma_i \Delta_i^c \ .
\end{aligned}$$

Thus, we find:

$$\begin{aligned}
\mathcal{A} &= \left(\begin{array}{cc|c} \underbrace{\mathcal{A}_1}_{li\times mi} & \underbrace{\mathcal{A}_2}_{li\times mi} & \underbrace{\mathcal{A}_3}_{li\times li} \end{array}\right) \\
&= \left[\begin{array}{c|c} \Gamma_i A^i S^{xu} & \Gamma_i A^i \Sigma^d \Gamma_i^T + \Gamma_i \Delta_i^c + \Gamma_i \Delta_i^d R_p^{uu} (H_i^d)^T \\ +\Gamma_i \Delta_i^d (\ R_p^{uu}\ R_{pf}^{uu}\ ) & +\Gamma_i A^i S_p^{xu} (H_i^d)^T + \Gamma_i \Delta_i^d (S_p^{xu})^T \Gamma_i^T \\ +H_i^d (\ (R_{pf}^{uu})^T\ R_f^{uu}\ ) & +H_i^d (R_{pf}^{uu})^T (H_i^d)^T + H_i^d (S_f^{xu})^T \Gamma_i^T \end{array}\right]
\end{aligned} \tag{A.11}$$

The second part we need to express in terms of the system matrices is:

$$\mathcal{B} \stackrel{\text{def}}{=} \mathbf{E_j}\left[\left(\begin{array}{c} U_p \\ U_f \\ \hline Y_p \end{array}\right).\left(\begin{array}{cc|c} U_p^T & U_f^T & Y_p^T \end{array}\right)\right] = \left(\begin{array}{c|c} \mathcal{B}_{11} & \mathcal{B}_{21}^T \\ \hline \mathcal{B}_{21} & \mathcal{B}_{22} \end{array}\right) \ ,$$

with:

$$
\begin{aligned}
\mathcal{B}_{21} &= \mathbf{E_j}\left[\, Y_p.[\, U_p^T \ U_f^T \,]\,\right] \\
&= \mathbf{E_j}\left[\, (\Gamma_i X_p^d + H_i^d U_p + Y_p^s).[\, U_p^T \ U_f^T \,]\,\right] \\
&= \Gamma_i S^{xu} + H_i^d[\, R_p^{uu} \ R_{pf}^{uu} \,]\ ,
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{B}_{22} &= \mathbf{E_j}[Y_p Y_p^T] \\
&= \mathbf{E_j}\left[(\Gamma_i X_p^d + H_i^d U_p + Y_p^s).((X_p^d)^T \Gamma_i^T + U_p^T (H_i^d)^T + (Y_p^s)^T)\right] \\
&= \Gamma_i \Sigma^d \Gamma_i^T + \Gamma_i S_p^{xu} (H_i^d)^T + H_i^d (S_p^{xu})^T \Gamma_i^T + H_i^d R_p^{uu} (H_i^d)^T + L_i\ ,
\end{aligned}
$$

and $\mathcal{B}_{11} = R^{uu}$. Note that $\mathcal{B}$ is guaranteed to be of full rank $((2m+l)i)$ due to the persistently exciting input and the non-zero stochastic subsystem. To compute $\mathcal{B}^{-1}$, we use the formula for the inverse of a block matrix [Kai 80]:

$$
\left(\begin{array}{c|c} \mathcal{B}_{11} & \mathcal{B}_{21}^T \\ \hline \mathcal{B}_{21} & \mathcal{B}_{22} \end{array}\right)^{-1} = \left(\begin{array}{c|c} \mathcal{B}_{11}^{-1} + \mathcal{B}_{11}^{-1}\mathcal{B}_{21}^T \psi^{-1} \mathcal{B}_{21} \mathcal{B}_{11}^{-1} & -\mathcal{B}_{11}^{-1}\mathcal{B}_{21}^T \psi^{-1} \\ \hline -\psi^{-1}\mathcal{B}_{21}\mathcal{B}_{11}^{-1} & \psi^{-1} \end{array}\right)\ , \tag{A.12}
$$

with $\psi = \mathcal{B}_{22} - \mathcal{B}_{21}\mathcal{B}_{11}^{-1}\mathcal{B}_{21}^T$. So, in our case, this becomes:

$$
\mathcal{B}^{-1} = \left(\begin{array}{c|c} (R^{uu})^{-1} + (\begin{pmatrix} I \\ 0 \end{pmatrix}(H_i^d)^T + (R^{uu})^{-1}(S^{xu})^T\Gamma_i^T)\psi^{-1}(H_i^d \begin{pmatrix} I & 0 \end{pmatrix} + \Gamma_i S^{xu}(R^{uu})^{-1}) & -(\begin{pmatrix} I \\ 0 \end{pmatrix}(H_i^d)^T + (R^{uu})^{-1}(S^{xu})^T\Gamma_i^T)\psi^{-1} \\ \hline -\psi^{-1}(H_i^d \begin{pmatrix} I & 0 \end{pmatrix} + \Gamma_i S^{xu}(R^{uu})^{-1}) & \psi^{-1} \end{array}\right)\ , \tag{A.13}
$$

with:

$$
\begin{aligned}
\psi &= \Gamma_i \Sigma^d \Gamma_i^T + L_i + \Gamma_i S_p^{xu}(H_i^d)^T + H_i^d (S_p^{xu})^T \Gamma_i^T + H_i^d R_p^{uu} (H_i^d)^T \\
&\quad -(\Gamma_i S^{xu} + H_i^d[\, R_p^{uu} \ R_{pf}^{uu} \,])(R^{uu})^{-1}((S^{xu})^T \Gamma_i^T + \begin{bmatrix} R_p^{uu} \\ (R_{pf}^{uu})^T \end{bmatrix}(H_i^d)^T) \\
&= L_i - \Gamma_i \underbrace{(S^{xu}(R^{uu})^{-1}(S^{xu})^T - \Sigma_d)}_{=P_0} \Gamma_i^T\ .
\end{aligned}
$$

For convenience, we alter the definition of $\mathcal{Z}_i$ slightly from (4.18) to:

$$
\mathcal{Z}_i = Y_f / \begin{pmatrix} \boldsymbol{U}_p \\ \boldsymbol{U}_f \\ \boldsymbol{Y}_p \end{pmatrix}
$$

$$
\begin{aligned}
&= \mathcal{A}.\mathcal{B}^{-1} \begin{pmatrix} U_p \\ U_f \\ Y_p \end{pmatrix} \\
&\stackrel{\text{def}}{=} \begin{pmatrix} L_{U_p} & L_{U_f} & L_{Y_p} \end{pmatrix} \begin{pmatrix} U_p \\ U_f \\ Y_p \end{pmatrix} .
\end{aligned}
$$

Note that this change of definition only consists of switching some block rows in the matrices we are projecting on. It does not change any properties. Using (A.11) and (A.13), we thus find:

$$
\begin{aligned}
\begin{pmatrix} L_{U_p} & L_{U_f} \end{pmatrix} &= \Gamma_i A^i S^{xu} (R^{uu})^{-1} + \Gamma_i \Delta_i^d \begin{pmatrix} I & 0 \end{pmatrix} + H_i^d \begin{pmatrix} 0 & I \end{pmatrix} \\
&\quad + [\Gamma_i A^i S_p^{xu} (H_i^d)^T + \Gamma_i A^i S^{xu} (R^{uu})^{-1} (S^{xu})^T \Gamma_i^T + \Gamma_i \Delta_i^d R_p^{uu} (H_i^d)^T \\
&\quad + \Gamma_i \Delta_i^d (S_p^{xu})^T \Gamma_i^T + H_i^d (R_{pf}^{uu})^T (H_i^d)^T + H_i^d (S_f^{xu})^T \Gamma_i^T - \Gamma_i A^i \Sigma^d \Gamma_i^T - \Gamma_i \Delta_i^c \\
&\quad - \Gamma_i \Delta_i^d R_p^{uu} (H_i^d)^T - \Gamma_i A^i S_p^{xu} (H_i^d)^T - \Gamma_i \Delta_i^d (S_p^{xu})^T \Gamma_i^T - H_i^d (R_{pf}^{uu})^T (H_i^d)^T \\
&\quad - H_i^d (S_f^{xu})^T \Gamma_i^T] \psi^{-1} (H_i^d \begin{pmatrix} I & 0 \end{pmatrix} + \Gamma_i S^{xu} (R^{uu})^{-1}) \\
&= \begin{pmatrix} \Gamma_i \Delta_i^d & H_i^d \end{pmatrix} + \Gamma_i A^i S^{xu} (R^{uu})^{-1} \\
&\quad - \Gamma_i (\Delta_i^c - A^i \underbrace{(S^{xu} (R^{uu})^{-1} (S^{xu})^T - \Sigma^d)}_{=P_0} \Gamma_i^T) \psi^{-1} \\
&\quad \times (H_i^d \begin{pmatrix} I & 0 \end{pmatrix} + \Gamma_i S^{xu} (R^{uu})^{-1}) \\
&= \begin{pmatrix} \Gamma_i [\Delta_i^d - \Omega_i H_i^d] & H_i^d \end{pmatrix} + \Gamma_i [A^i - \Omega_i \Gamma_i] S^{xu} (R^{uu})^{-1} .
\end{aligned}
$$

Once again, using (A.11) and (A.13), we find for L_{Y_p}:

$$
\begin{aligned}
L_{Y_p} &= [-\Gamma_i A^i S_p^{xu} (H_i^d)^T - \Gamma_i \Delta_i^d R_p^{uu} (H_i^d)^T - H_i^d (R_{pf}^{uu})^T (H_i^d)^T \\
&\quad - \Gamma_i A^i S^{xu} (R^{uu})^{-1} (S^{xu})^T \Gamma_i^T - \Gamma_i \Delta_i^d \begin{pmatrix} I & 0 \end{pmatrix} (S^{xu})^T \Gamma_i^T \\
&\quad - H_i^d \begin{pmatrix} 0 & I \end{pmatrix} (S^{xu})^T \Gamma_i^T + \Gamma_i A^i \Sigma^d \Gamma_i^T + \Gamma_i \Delta_i^c + \Gamma_i \Delta_i^d R_p^{uu} (H_i^d)^T \\
&\quad + \Gamma_i A^i S_p^{xu} (H_i^d)^T + \Gamma_i \Delta_i^d (S_p^{xu})^T \Gamma_i^T + H_i^d (R_{pf}^{uu})^T (H_i^d)^T + H_i^d (S_f^{xu})^T \Gamma_i^T] \psi^{-1} \\
&= \Gamma_i (\Delta_i^c - A^i \underbrace{(S^{xu} (R^{uu})^{-1} (S^{xu})^T - \Sigma^d)}_{=P_0} \Gamma_i^T) \psi^{-1} \\
&= \Gamma_i \Omega_i .
\end{aligned}
$$

We thus conclude that:

$$
\mathcal{Z}_i = \Gamma_i \left[(A^i - \Omega_i \Gamma_i) \underbrace{S^{xu} (R^{uu})^{-1} \begin{pmatrix} U_p \\ U_f \end{pmatrix}}_{=\widehat{X}_0} + (\Delta_i^d - \Omega_i H_i^d) U_p + \Omega_i Y_p \right]
$$

$$+H_i^d U_f \ .$$

Using (4.14) we get:

$$\mathcal{Z}_i = \Gamma_i \widehat{X}_i + H_i^d U_f \ ,$$

which proves (4.18).

$\square$

A.7 PROOF OF THEOREM 12

We prove equations (4.26)-(4.28). The rest of the Theorem follows from these equations through a similar reasoning as in the proof of Theorem 2 or 8. $\mathcal{O}_i$ is the oblique projection of the row space of Y_f along the row space of U_f on the row space of W_p. This oblique projection can also be computed by decomposing the orthogonal projection of the row space of Y_f on the combined row spaces of W_p and U_f (see Section 1.4). Dropping the part along U_f results in the oblique projection $\mathcal{O}_i$. We know now that (4.18):

$$\begin{aligned} \mathcal{Z}_i &= Y_f / \begin{pmatrix} \boldsymbol{W}_p \\ \boldsymbol{U}_f \end{pmatrix} \\ &= \Gamma_i \widehat{X}_i + H_i^d U_f \ . \end{aligned}$$

Using the expressions for $\widehat{X}_i$ (4.14) and $\widehat{X}_0$ (4.22) and the fact that:

$$X_p^d / \begin{pmatrix} \boldsymbol{U}_p \\ \boldsymbol{U}_f \end{pmatrix} = X_p^d /_{U_p} \boldsymbol{U}_f + X_p^d /_{U_f} \boldsymbol{U}_p \ ,$$

this can be rewritten as:

$$\begin{aligned} \mathcal{Z}_i &= \Gamma_i \begin{pmatrix} \Delta_i^d - \Omega_i H_i^d & \Omega_i \end{pmatrix} W_p + H_i^d U_f \\ &\quad + \Gamma_i (A^i - \Omega_i \Gamma_i) \left[X_p^d /_{U_p} \boldsymbol{U}_f + X_p^d /_{U_f} \boldsymbol{U}_p \right] \ . \end{aligned}$$

The effect of U_f is clearly visible in this last equation. When we drop the linear combinations of U_f we get the oblique projection:

$$\mathcal{O}_i = \Gamma_i \begin{pmatrix} (A^i - \Omega_i \Gamma_i) \mid \begin{pmatrix} \Delta_i^d - \Omega_i H_i^d & \Omega_i \end{pmatrix} \end{pmatrix} \left(\frac{X_p^d /_{U_f} \boldsymbol{U}_p}{W_p} \right) \ .$$

Defining $\widetilde{X}_i$ as:

$$\widetilde{X}_i = \left(\begin{array}{c} (A^i - \Omega_i \Gamma_i) \end{array} \middle| \begin{array}{cc} \left(\Delta_i^d - \Omega_i H_i^d \right. & \left. \Omega_i \right) \end{array} \right) \left(\frac{X_p^d /_{U_f} \boldsymbol{U}_p}{W_p} \right) ,$$

and comparing this last equation with the expression for the Kalman filter states (4.14) shows that $\widetilde{X}_i$ is indeed a Kalman filter sequence with initial state and covariance matrix as defined in (4.27)-(4.28).

A.8 PROOF OF LEMMA 2

We first prove equation (5.10). Hereto we consider the different sub-blocks of the weighted controllability Lyapunov equation (5.8):

$$\begin{aligned} P_{11} &= AP_{11}A^T + EE^T + AP_{21}^T C_u^T B^T + BC_u P_{21} A^T \\ &\quad + B[C_u P_{22} C_u^T + D_u D_u^T] B^T , && \text{(A.14)} \\ P_{21} &= A_u P_{21} A^T + [A_u P_{22} C_u^T + B_u D_u^T] B^T , && \text{(A.15)} \\ P_{22} &= [A_u P_{22} A_u^T + B_u B_u^T] . && \text{(A.16)} \end{aligned}$$

For convenience of notation and since $i \to \infty$, we drop all the subscripts i in the following. All superscripts are also replaced by subscripts (for instance $\Delta_i^s \leftarrow \Delta_s$). We first prove that with:

$$\Delta_u \stackrel{\text{def}}{=} \begin{pmatrix} \dots & A_u^2 B_u & A_u B_u & B_u \end{pmatrix} ,$$

we have:

$$\begin{aligned} P_{22} &= \Delta_u \Delta_u^T , && \text{(A.17)} \\ P_{21} &= \Delta_u W_u^T \Delta_d^T . && \text{(A.18)} \end{aligned}$$

Proof of (A.17):

$$\begin{aligned} \Delta_u \Delta_u^T &= \begin{pmatrix} A_u \Delta_u & B_u \end{pmatrix} \begin{pmatrix} \Delta_u^T A_u^T \\ B_u^T \end{pmatrix} \\ &= A_u [\Delta_u \Delta_u^T] A_u^T + B_u B_u^T , \end{aligned}$$

which proves that $\Delta_u \Delta_u^T$ is the solution of the same Lyapunov equation (A.16) as P_{22} and thus proves (A.17).

Proof of (A.18):

$$\begin{aligned}
\Delta_u W_u^T \Delta_d^T &= \begin{pmatrix} A_u\Delta_u & B_u \end{pmatrix} \begin{pmatrix} W_u^T & \Delta_u^T C_u^T \\ 0 & D_u^T \end{pmatrix} \begin{pmatrix} \Delta_d^T A^T \\ B^T \end{pmatrix} \\
&= A_u[\Delta_u W_u^T \Delta_d^T]A^T + [A_u[\Delta_u \Delta_u^T]C_u^T + B_u D_u^T]B^T \\
&= A_u[\Delta_u W_u^T \Delta_d^T]A^T + [A_u P_{22} C_u^T + B_u D_u^T]B^T \ ,
\end{aligned}$$

which proves that $\Delta_u W_u^T \Delta_d^T$ is the solution of the same equation as P_{21} (A.15) and thus proves (A.18).

Finally, we prove (5.10):

$$\begin{aligned}
\Delta_d W_u W_u^T \Delta_d^T + \Delta_s \Delta_s^T &= \begin{pmatrix} A\Delta_d & B \end{pmatrix} \begin{pmatrix} W_u & 0 \\ C_u\Delta_u & D_u \end{pmatrix} \begin{pmatrix} W_u^T & \Delta_u^T C_u^T \\ 0 & D_u^T \end{pmatrix} \begin{pmatrix} \Delta_d^T A^T \\ B^T \end{pmatrix} \\
&+ \begin{pmatrix} A\Delta_s & E \end{pmatrix} \begin{pmatrix} \Delta_s^T A^T \\ E^T \end{pmatrix} \\
&= A[\Delta_d W_u W_u^T \Delta_d^T + \Delta_s \Delta_s^T]A^T + EE^T \\
&\quad + A[\Delta_d W_u \Delta_u^T]C_u^T B^T + BC_u[\Delta_u W_u^T \Delta_d^T]A^T \\
&\quad + B[C_u[\Delta_u \Delta_u^T]C_u^T + D_u D_u^T]B^T \\
&= A[\Delta_d W_u W_u^T \Delta_d^T + \Delta_s \Delta_s^T]A^T + EE^T \\
&\quad + AP_{21}^T C_u^T B^T + BC_u P_{21} A^T + B[C_u P_{22} C_u^T + D_u D_u^T]B^T \ ,
\end{aligned}$$

which proves that $\Delta_d W_u W_u^T \Delta_d^T + \Delta_s \Delta_s^T$ is the solution of the same equation as P_{11} (A.14) and thus proves (5.10).

Now we prove equation (5.11). Once again, we consider the different sub-blocks of the weighted observability Lyapunov equation (5.9):

$$\begin{aligned}
Q_{11} &= A^T Q_{11} A + A^T Q_{12} B_y C + C^T B_y^T Q_{12}^T A \\
&\quad + C^T [B_y^T Q_{22} B_y + D_y^T D_y] C \ , && \text{(A.19)} \\
Q_{12} &= A^T Q_{12} A_y^T + C^T [B_y^T Q_{22} A_y + D_y^T C_y] \ , && \text{(A.20)} \\
Q_{22} &= [A_y^T Q_{22} A_y + C_y^T C_y] \ . && \text{(A.21)}
\end{aligned}$$

We again drop the subscript i and transform all superscripts to subscripts. With:

$$\Gamma_y \stackrel{\text{def}}{=} \begin{pmatrix} C_y \\ C_y A_y \\ \cdots \end{pmatrix} \ ,$$

we have:

$$\begin{aligned} Q_{22} &= \Gamma_y^T \Gamma_y \ , && \text{(A.22)} \\ Q_{12} &= \Gamma^T W_y^T \Gamma_y \ . && \text{(A.23)} \end{aligned}$$

Proof of (A.22):

$$\begin{aligned} \Gamma_y^T \Gamma_y &= \begin{pmatrix} C_y^T & A_y^T \Gamma_y^T \end{pmatrix} \begin{pmatrix} C_y \\ \Gamma_y A_y \end{pmatrix} \\ &= A_y^T [\Gamma_y^T \Gamma_y] A_y + C_y^T C_y \ , \end{aligned}$$

which proves that $\Gamma_y^T \Gamma_y$ is the solution of the same Lyapunov equation as Q_{22} (A.21) and thus proves (A.22).

Proof of (A.23):

$$\begin{aligned} \Gamma^T W_y^T \Gamma_y &= \begin{pmatrix} C^T & A^T \Gamma^T \end{pmatrix} \begin{pmatrix} D_y^T & B_y^T \Gamma_y^T \\ 0 & W_y^T \end{pmatrix} \begin{pmatrix} C_y \\ \Gamma_y A_y \end{pmatrix} \\ &= A^T [\Gamma^T W_y^T \Gamma_y] A_y + C^T [B_y^T Q_{22} A_y + D_y^T C_y] \ , \end{aligned}$$

which proves that $\Gamma^T W_y^T \Gamma_y$ is the solution of the same equation as Q_{12} (A.20) and thus proves (A.23).

Finally, we prove (5.11):

$$\begin{aligned} \Gamma^T W_y^T W_y \Gamma &= \begin{pmatrix} C^T & A^T \Gamma^T \end{pmatrix} \begin{pmatrix} D_y^T & B_y^T \Gamma_y^T \\ 0 & W_y^T \end{pmatrix} \begin{pmatrix} D_y & 0 \\ \Gamma_y B_y & W_y \end{pmatrix} \begin{pmatrix} C \\ \Gamma A \end{pmatrix} \\ &= A^T [\Gamma^T W_y^T W_y \Gamma] A + C^T B_y^T [\Gamma_y^T W_y \Gamma] A + A^T [\Gamma^T W_y^T \Gamma_y] B_y C \\ &\quad + C^T [B_y^T [\Gamma_y^T \Gamma_y] B_y + D_y^T D_y] C \\ &= A^T [\Gamma^T W_y^T W_y \Gamma] A + C^T B_y^T Q_{12}^T A + A^T Q_{12} B_y C \\ &\quad + C^T [B_y^T Q_{22} B_y + D_y^T D_y] C \ , \end{aligned}$$

which proves that $\Gamma^T W_y^T W_y \Gamma$ is the solution of the same equation as Q_{11} (A.19) and thus proves (5.11).

A.9 PROOF OF THEOREM 13

W_1 (5.12) and W_2 (5.13) are easily seen to be of full rank. Note that since $j \to \infty$ we dropped all the time average operators $\mathbf{E_j}$ in this proof (and in the proofs to follow).

Proof of (5.15):

From condition 1 of Theorem 12 we know that u_k and e_k are uncorrelated. From this it follows that (see Subsection 1.4.4 and taking into account that we dropped the symbol $\mathbf{E_j}$):

$$\begin{aligned} E_p U_p^T &= 0 \;, \\ E_p \Pi_{U_p^\perp} &= E_p \;. \end{aligned}$$

It is also trivial that $U_p \Pi_{U_p^\perp} = 0$. From these equations, it is easy to verify that (with W_2 given by equation (5.13)):

$$\begin{aligned} U_p W_2 &= \underbrace{U_p U_p^T (R_p^{uu})^{-1}}_{=I} W_u (L_p^{uu})^{-1} U_p + \underbrace{U_p \Pi_{U_p^\perp}}_{=0} \\ &= W_u (L_p^{uu})^{-1} U_p \;, \\ E_p W_2 &= \underbrace{E_p U_p^T}_{\to 0} (R_p^{uu})^{-1} W_u (L_p^{uu})^{-1} U_p + \underbrace{E_p \Pi_{U_p^\perp}}_{\to E_p} \;. \\ &= E_p \;. \end{aligned}$$

Combining these results with (5.7) we find that:

$$\begin{aligned} \widetilde{X}_i W_2 W_2^T \widetilde{X}_i^T &= \Delta_i^d U_p W_2 W_2^T U_p^T (\Delta_i^d)^T + \Delta_i^s E_p W_2 W_2^T E_p^T (\Delta_i^s)^T \\ &= \Delta_i^d W_u \underbrace{(L_p^{uu})^{-1} U_p U_p^T (L_p^{uu})^{-T}}_{=I} W_u^T (\Delta_i^d)^T + \Delta_i^s \underbrace{E_p E_p^T}_{\to I} (\Delta_i^s)^T \\ &= \Delta_i^d W_u W_u^T (\Delta_i^d)^T + \Delta_i^s (\Delta_i^s)^T \;. \end{aligned} \tag{A.24}$$

It is easy to verify that W_2 given by formula (5.14) leads to the same result (A.24). From Theorem 12 we also know that:

$$\widetilde{X}_i W_2 = S_1^{1/2} V_1^T \;.$$

Since $V_1^T V_1 = I$, this leads to:

$$\begin{aligned} \widetilde{X}_i W_2 W_2^T \widetilde{X}_i^T &= S_1^{1/2} V_1^T V_1 S_1^{1/2} \\ &= S_1 \;. \end{aligned} \tag{A.25}$$

Finally, from (A.24) and (A.25) we find:

$$\Delta_i^d W_u W_u^T (\Delta_i^d)^T + \Delta_i^s (\Delta_i^s)^T = S_1 \ .$$

From Lemma 2 we know that the left part of this expression is equal to $P[W_u(z)]$. This leads to:

$$P[W_u(z)] = S_1 \ ,$$

which is exactly (5.15).

Proof of (5.16):

From Theorem 12, we know:

$$\begin{aligned} \Gamma_i^T W_1^T W_1 \Gamma_i &= S_1^{1/2} U_1^T U_1 S_1^{1/2} \\ &= S_1 \ . \end{aligned} \tag{A.26}$$

Combining this with Lemma 2 (5.11) and with $W_1 = W_y$ (5.12), we find:

$$\begin{aligned} Q[W_y(z)] &= \Gamma_i^T W_y^T W_y \Gamma_i \\ &= \Gamma_i^T W_1^T W_1 \Gamma_i \\ &= S_1 \ , \end{aligned}$$

which is (5.16).

A.10 PROOF OF COROLLARY 2 AND 3

Proof of Corollary 2:

With $W_u = L_p^{uu}$ and $W_y = I_{li}$, we find for the weighting matrices (5.12) and (5.13) in Theorem 13:

$$\begin{aligned} W_1 &= I_{li} \ , \\ W_2 &= U_p^T (R_p^{uu})^{-1} L_p^{uu} . (L_p^{uu})^{-1} U_p + \Pi_{U_p^\perp} \\ &= U_p^T (R_p^{uu})^{-1} U_p + \Pi_{U_p^\perp} \\ &= \Pi_{U_p} + \Pi_{U_p^\perp} \\ &= I_j \ . \end{aligned}$$

In Section 4.3.1 it was shown that this choice of weights (W_1 and W_2) exactly corresponds to the **N4SID** algorithm.

Proof of Corollary 3:

With $W_u = L^{uu}_{p^\perp}$ and $W_y = I_{li}$, we find for the weighting matrices (5.12)-(5.14) in Theorem 13:

$$\begin{aligned}
W_1 &= I_{li} \ , \\
W_2 &= (U_p/\boldsymbol{U}_f^\perp)^T.(R^{uu}_{p^\perp})^{-1}.L^{uu}_{p^\perp}.(L^{uu}_{p^\perp})^{-1}.U_p/\boldsymbol{U}_f^\perp + \Pi_{U_p^\perp, U_f^\perp} \\
&= (U_p/\boldsymbol{U}_f^\perp)^T.(R^{uu}_{p^\perp})^{-1} U_p/\boldsymbol{U}_f^\perp + \Pi_{U_p^\perp, U_f^\perp} \\
&= \Pi_{U_p/U_f^\perp} + \Pi_{U_p^\perp, U_f^\perp} \\
&= \Pi_{U_f^\perp} \ .
\end{aligned}$$

In Section 4.3.2 it was shown that the **MOESP** algorithm corresponds to this choice of weights $W_1 = I_{li}$ and $W_2 = \Pi_{U_f^\perp}$.

A.11 PROOF OF THEOREM 14

It is easy to verify that for the choices (5.17)-(5.18) of W_1 and W_2 are full rank matrices.

Proof of (5.19):

From Theorem 12 we know that u_k and e_k are uncorrelated. From this it follows that (see Subsection 1.4.4 and taking into account that we dropped the symbol $\mathbf{E_j}$):

$$\begin{aligned}
E_p U_p^T &= 0 \ , \\
E_p (U_p/\boldsymbol{U}_f^\perp)^T &= 0 \ , \\
E_p \Pi_{U_p^\perp, U_f^\perp} &= E_p \ .
\end{aligned}$$

We also have:

$$\begin{aligned}
U_p (U_p/\boldsymbol{U}_f^\perp)^T (R^{uu}_{p^\perp})^{-1} &= I_{mi} \ , \\
U_f (U_p/\boldsymbol{U}_f^\perp)^T (R^{uu}_{p^\perp})^{-1} &= 0 \ , \\
U_f \Pi_{U_p^\perp, U_f^\perp} = U_p \Pi_{U_p^\perp, U_f^\perp} &= 0 \ .
\end{aligned}$$

From these equations it is easy to verify that:

$$\begin{aligned}
U_pW_2 &= \underbrace{U_p(U_p/\boldsymbol{U}_f^\perp)^T(R_{p^\perp}^{uu})^{-1}}_{=I}W_u(L_{p^\perp}^{uu})^{-1}.U_p/\boldsymbol{U}_f^\perp + \underbrace{U_p\Pi_{U_p^\perp,U_f^\perp}}_{=0}\\
&= W_u(L_{p^\perp}^{uu})^{-1}.U_p/\boldsymbol{U}_f^\perp\\
U_fW_2 &= \underbrace{U_f(U_p/\boldsymbol{U}_f^\perp)^T(R_{p^\perp}^{uu})^{-1}}_{=0}W_u(L_{p^\perp}^{uu})^{-1}.U_p/\boldsymbol{U}_f^\perp + \underbrace{U_f\Pi_{U_p^\perp,U_f^\perp}}_{=0}\\
&= 0\ ,\\
E_pW_2 &= \underbrace{E_p(U_p/\boldsymbol{U}_f^\perp)^T}_{\to 0}\Phi^{-1}_{[U_p/U_f^\perp,U_p/U_f^\perp]}W_u(L_{p^\perp}^{uu})^{-1}.U_p/\boldsymbol{U}_f^\perp + \underbrace{E_p\Pi_{U_p^\perp,U_f^\perp}}_{\to E_p}\\
&= E_p\ ,\\
E_fW_2 &= E_f\ .
\end{aligned}$$

Combining these results with (5.7), we find:

$$\begin{aligned}
\widetilde{X}_iW_2W_2^T\widetilde{X}_i^T &= \Delta_i^dU_pW_2W_2^TU_p^T(\Delta_i^d)^T + \Delta_i^sE_pW_2W_2^TE_p^T(\Delta_i^s)^T\\
&= \Delta_i^dW_uW_u^T(\Delta_i^d)^T + \Delta_i^s(\Delta_i^s)^T\ .
\end{aligned}$$

The last equation is exactly the same as (A.24) in the proof of the first main Theorem. The rest of the proof of (5.19) is the same as the proof of (5.15).

Proof of (5.20):

To get more insight in (5.17), we investigate Y_fW_2 (using (5.3)-(5.5)):

$$\begin{aligned}
Y_fW_2 &= [\Gamma_iX_f + H_i^dU_f + H_i^sE_f]W_2\\
&= [\Gamma_i[\underbrace{A^iX_p}_{\to 0} + \Delta_i^dU_p + \Delta_i^sE_p] + H_i^dU_f + H_i^sE_f]W_2\\
&= \Gamma_i[\Delta_i^dU_pW_2 + \Delta_i^s\underbrace{E_pW_2}_{\to E_p}] + H_i^d\underbrace{U_fW_2}_{=0} + H_i^s\underbrace{E_fW_2}_{\to E_f}\\
&= \Gamma_i[\Delta_i^dU_pW_2 + \Delta_i^sE_p] + H_i^sE_f\\
&= \Gamma_i[\Delta_i^dW_u(L_{p^\perp}^{uu})^{-1}U_p/\boldsymbol{U}_f^\perp + \Delta_i^sE_p] + H_i^sE_f\ .
\end{aligned}$$

It can now easily be verified that:

$$(Y_fW_2).(Y_fW_2)^T = \Gamma_i[\Delta_i^dW_u\underbrace{(L_{p^\perp}^{uu})^{-1}R_{p^\perp}^{uu}(L_{p^\perp}^{uu})^{-T}}_{=I}W_u^T(\Delta_i^d)^T$$

$$
\begin{aligned}
& +\Delta_i^s(\Delta_i^s)^T]\Gamma_i^T + H_i^s(H_i^s)^T \\
= \; & \Gamma_i[\Delta_i^d W_u W_u^T (\Delta_i^d)^T + \Delta_i^s(\Delta_i^s)^T]\Gamma_i^T + H_i^s(H_i^s)^T \\
= \; & \Gamma_i P[W_u(z)]\Gamma_i^T + H_i^s(H_i^s)^T \ .
\end{aligned}
$$

With (5.17) and the matrix inversion Lemma of [Kai 80] this leads to:

$$
\begin{aligned}
\Gamma_i^T W_1^T W_1 \Gamma_i &= \Gamma_i^T[\Gamma_i P[W_u(z)]\Gamma_i^T + H_i^s(H_i^s)^T]^{-1}\Gamma_i \\
&= \{\Gamma_i^T[H_i^s(H_i^s)^T]^{-1}\Gamma_i\} - \{\Gamma_i^T[H_i^s(H_i^s)^T]^{-1}\Gamma_i\} \\
&\quad \times[\{\Gamma_i^T[H_i^s(H_i^s)^T]^{-1}\Gamma_i\} - P[W_u(z)]^{-1}]^{-1}\{\Gamma_i^T[H_i^s(H_i^s)^T]^{-1}\Gamma_i\} \ .
\end{aligned}
$$

With $Q_n = \{\Gamma_i^T[H_i^s(H_i^s)^T]^{-1}\Gamma_i\}$, we find:

$$
\begin{aligned}
\Gamma_i^T W_1^T W_1 \Gamma_i &= Q_n - Q_n(Q_n - P[W_u(z)]^{-1})^{-1}Q_n \\
&= Q_n(I_n - P[W_u(z)]Q_n)^{-1} \ .
\end{aligned}
$$

Since (from Theorem 12) $\Gamma_i^T W_1^T W_1 \Gamma_i = S_1$ and (from (5.19)) $P[W_u(z)] = S_1$, we find the following expression for Q_n:

$$
Q_n = S_1(I_n - S_1^2)^{-1} \ .
$$

The final part of the proof consists of proving that $Q_n = Q[H^{-1}(z)]$. It is easy to verify that the block Toeplitz matrix containing the Markov parameters of $H^{-1}(z)$ is given by $(H_i^s)^{-1}$. From this and from Lemma 2 we find:

$$
\begin{aligned}
Q[H^{-1}(z)] &= \Gamma_i^T(H_i^s)^{-T}(H_i^s)^{-1}\Gamma_i \\
&= \Gamma_i^T[H_i^s(H_i^s)^T]^{-1}\Gamma_i \\
&= Q_n \ .
\end{aligned}
$$

This proves (5.20).

B

MATLAB FUNCTIONS

In this Appendix, we describe the set of Matlab files that implement the algorithms developed in this book. The idea of these files is to make subspace identification algorithms more accessible for researchers as well as for industry.

B.1 GETTING STARTED

All algorithms in this book have separately been implemented in M-files. The research however culminated in one overall algorithm called `subid.m` which implements deterministic (Figure 2.8), stochastic (Figure 3.13) and combined (Figure 4.8) subspace identification. This function is a good place to start with.

After you have copied the files from the diskette to your computer (for instance in the directory `c:\subspace`), you should include the directory `subfun` in your path:

```
> path(path,'c:\subspace\subfun');
```

Now change your directory to the `examples` directory and run the demo file `sta_demo.m`:

```
> cd c:\subspace\examples
> sta_demo
```

You will get a clear impression of what subspace identification algorithms can do. You might also want to run the stochastic identification demo `sto_demo.m`.

Now you are ready to rerun some of the applications in Section 6.4. To run for instance the application of the flexible robot arm, do:

```
> cd c:\subspace\applic
> appl5
```

This will guide you through the demo and reproduce the results of Section 6.4. You could now also run any other applications in this directory.

You are now ready to try out your own problem. You could just use one of the subspace identification functions (for instance `subid.m`). Alternatively, you could also copy and adapt one of the application M-files.

B.2 MATLAB REFERENCE

This Section contains a short description of the files on the diskette. For more information, see the Matlab on line help and the M-files themselves.

B.2.1 Directory: 'subfun'

This directory contains the Matlab subspace functions organized into main subspace identification functions and auxilary ones.

All subspace identification functions have approximately the same syntax:

Inputs:

- **`y` and `u`:**
  The output and input data arranged in a matrix. The matrices have as many rows as the number of data samples and as many columns as there are outputs respectively inputs. The stochastic identification algorithms have no input `u`.
- **`i`:**
  The number of block rows used in the block Hankel matrices. The maximal order that can be estimated is `i` times the number of outputs. Do not choose `i` too

large, since the computational time is proportional to $\mathtt{i}^2$. Typically `i` is equal to:

$$\mathtt{i} = 2.\frac{\text{Maximal Order}}{\text{Number of Outputs}}$$

- `n`:
  The order of the system. This parameter is optional. When it is not given, the singular values are plotted and the user is prompted for the system order.

- `AUX`:
  An optional auxilary variable which speeds up the identification process. For instance, when one wants to identify systems of different order, this variable is used to avoid the recomputation of the RQ decomposition every time. See also the M-function `allord.m` and the examples in `sta_demo.m`.

- `W`:
  An optional weighting flag which indicates in which basis the subspaces are computed. See also Chapter 5.

- `sil`:
  When this flag is set to one, the algorithms run silently, without any output to the screen.

Note that when an input is optional, supplying an empty matrix `[]` is the same as not supplying that specific input. For example `subid(y,u,i,[],AUX)` allows you to use the auxilary variable `AUX` without specifying the order.

Outputs:

- `A,B,C,D`:
  The state space system matrices of Formulas (1.1)-(1.3).

- `K,R`:
  The Kalman gain and covariance of the innovations. The general identified model is thus:

$$\begin{aligned} x_{k+1} &= A.x_k + B.u_k + K.e_k \\ y_k &= C.x_k + D.u_k + e_k \\ \mathbf{E}[e_p.e_q^T] &= R.\delta_{pq} \end{aligned}$$

- `G,L0`:
  The state space matrices determining the stochastic model (3.6) and (3.7). These matrices can be converted to a Kalman gain and innovations covariance through the function `gl2kr.m`.

- `AUX`:
  This output can be used as an input the next time the function is called.

- `ss`:
  A column vector containing the singular values from which the order could be determined.

Subspace functions

Function	Fig.	Page	Inputs	Options	Outputs
`intersec`	-	45	`y,u,i`	`n,AUX`	`A,B,C,D,ss`
`project`	-	46	`y,u,i`	`n,AUX`	`A,B,C,D,ss`
`det_stat`	2.7	52	`y,u,i`	`n,AUX`	`A,B,C,D,AUX,ss`
`det_alt`	2.8	56	`y,u,i`	`n,AUX`	`A,B,C,D,AUX,ss`
`sto_stat`	3.11	86	`y,i`	`n,AUX,W`	`A,K,C,R,G,L0,AUX,ss`
`sto_alt`	3.12	87	`y,i`	`n,AUX,W`	`A,K,C,R,G,L0,AUX,ss`
`sto_pos`	3.13	90	`y,i`	`n,AUX,W`	`A,K,C,R,G,L0,AUX,ss`
`com_alt`	4.6	121	`y,u,i`	`n,AUX,W`	`A,B,C,D,K,R,AUX,ss`
`com_stat`	4.7	124	`y,u,i`	`n,AUX,W`	`A,B,C,D,K,R,AUX,ss`
`subid`	4.8	131	`y,[u],i`	`n,AUX,W`	`A,B,C,D,K,R,AUX,ss`
`allord`	-	-	`y,[u],i`	`n,AUX,W`	`ersa,erpa,AUX`

Auxilary functions

Function	Page	Syntax	Description
`blkhank`	33	`H=blkhank(y,i,j)`	Make Hankel
`solvric`	62	`[P,flag]=solvric(A,G,C,L0)`	Solve Riccati
`gl2kr`	137	`[K,R]=gl2kr(A,G,C,L0)`	Transform
`kr2gl`	137	`[G,L0]=kr2gl(A,K,C,R)`	Transform
`simul`	192	`[ys,ers]=simul(y,u,A,B,C,D,ax)`	Simulation
`predic`	192	`[yp,erp]=predic(y,u,A,B,C,D,K,ax)`	Prediction
`myss2th`	-	`th=myss2th(A,B,C,D,K,flag)`	Conversion

B.2.2 Directory: 'applic'

This directory contains the applications described in Section 6.4. Both data files and M-files are provided to ensure total reproducibility. See also Table 6.2 for more information.

M-File	Data	Description
appl1	appl1.mat	Glass Tubes
appl2	appl2.mat	Dryer
appl3	appl3.mat	Glass Oven
appl4	appl4.mat	Flutter
appl5	appl5.mat	Robot
appl6	appl6.mat	Evaporator
appl8	appl8.mat	CD Player
appl9	appl9.mat	Ball & Beam
appl10	appl10.mat	Wall Temp.

B.2.3 Directory: 'examples'

This directory contains two demo files, which illustrate the possibilities and applications of the subspace functions.

M-File	Description
sta_demo	Starting up demo
sto_demo	Stochastic systems demo

B.2.4 Directory: 'figures'

This directory contains the M-files that generate the matlab Figures in this book.

M-File	Figure	Page
det_sim1	2.6	49
sto_sim1	3.10	83
sto_sim2	3.14	92
sto_sim3	3.16	93
com_sim1	4.5	116
com_sim2	4.9	132

C

NOTATION

Notation	Description	Page
A, B, C, D	Dynamical system matrices.	7
A_u, B_u, C_u, D_u	Input weight dynamical system matrices.	140
A_y, B_y, C_y, D_y	Output weight dynamical system matrices.	140
C_i	Block Toeplitz matrix of output covariances.	68
e_k^f	Forward innovation at time instant k.	61
e_k^b	Backward innovation at time instant k.	64
E, F	System matrices of forward innovation model.	137
E_p, E_f	Block Hankel matrices of past ($E_{0\|i-1}$) respectively future ($E_{i\|2i-1}$) forward innovations.	138
$E(z)$	$\mathcal{Z}$-transform of the forward innovations e_k^f.	140
G	Auxilary matrix for the description of stochastic systems.	61
$G(z)$	Deterministic transfer matrix $D + C(zI - A)^{-1}B$.	140
$\widehat{G}(z)$	Reduced order deterministic transfer matrix.	150
H_i^d	Toeplitz matrix containing the deterministic Markov parameters $D, CB, CAB, \ldots$.	36
H_i^s	Toeplitz matrix containing the stochastic Markov parameters $F, CE, CAE, \ldots$.	139
$H(z)$	Stochastic transfer matrix $F + C(zI - A)^{-1}E$.	140
$\widehat{H}(z)$	Reduced order stochastic transfer matrix.	150
i	Number of block rows in block Hankel matrices.	34

I_n	Identity matrix $(n \times n)$.	
j	Number of columns in block Hankel matrices.	34
k	Time instant.	6
K^f	Forward Kalman gain.	61
K^b	Backward Kalman gain.	64
l	Number of outputs.	6
L_i	Block Toeplitz matrix of output covariances.	68
L_p^{uu}	Square root of the matrix R_p^{uu}.	138
$L_{p^\perp}^{uu}$	Square root of the matrix $R_{p^\perp}^{uu}$.	138
m	Number of inputs.	6
n	Number of states.	6
N	Solution of the backward algebraic Riccati equation.	64
N_k	Solution of the backward difference Riccati equation.	72
$\mathcal{O}_i$	Oblique projection of the row space of $Y_{i\|2i-1}$ along the row space of $U_{i\|2i-1}$ on the row space of $W_{0\|i-1}$: $\mathcal{O}_i = Y_{i\|2i-1}/_{U_{i\|2i-1}} \boldsymbol{W}_{0\|i-1}$.	40
P	Solution of the forward algebraic Riccati equation.	62
P_k	Solution of the forward difference Riccati equation.	70
$P[W_u(z)]$	$W_u(z)$ weighted controllability Grammian.	142
Q, R, S	Covariance and cross-covariance matrices of the forward measurement and process noise.	7
Q^b, R^b, S^b	Covariance and cross-covariance matrices of the backward measurement and process noise.	65
Q_p	Part of the orthogonal matrix in the RQ decomposition.	164
$Q[W_y(z)]$	$W_y(z)$ weighted observability Grammian.	142
$R_{[p:q,r:s]}$	Part of the triangular factor in the RQ decomposition.	164
R^{uu}	Covariance of the inputs : $\Phi_{[U_{0\|2i-1},U_{0\|2i-1}]}$.	100
R_p^{uu}	Covariance of past inputs $\Phi_{[U_p,U_p]}$.	100
$R_{p^\perp}^{uu}$	Covariance of past projected inputs $\Phi_{[U_p/U_f^\perp,U_p/U_f^\perp]}$.	138
R_f^{uu}	Covariance of future inputs $\Phi_{[U_f,U_f]}$.	100
R_{pf}^{uu}	Cross-covariance of past and future inputs $\Phi_{[U_p,U_f]}$.	100
S^{xu}	Cross-covariance of the past deterministic state and inputs : $\Phi_{[X_p^d,U_{0\|2i-1}]}$.	100

S_p^{xu}	Cross-covariance of the past deterministic state and past inputs $\Phi_{[X_p^d,U_p]}$.	100
S_f^{xu}	Cross-covariance of the past deterministic state and future inputs $\Phi_{[X_p^d,U_f]}$.	100
$S_u(z)$	Spectral factor of $U(z)$.	140
s	Number of available measurements.	9
T	Non-singular $n \times n$ similarity transformation.	43
u_k	Input at time instant k.	6
U, S, V	Matrices of a singular value decomposition.	
$U_{0\|i-1}$	Input block Hankel matrix. The subscript indicates the indices of the first column of the matrix.	33
U_p, U_p^+	Past inputs $U_{0\|i-1}$ respectively $U_{0\|i}$.	33
U_f, U_f^-	Future inputs $U_{i\|2i-1}$ respectively $U_{i+1\|2i-1}$.	33
$U(z)$	$\mathcal{Z}$-transform of the input u_k.	140
v_k	(Forward) measurement noise at time instant k.	6
v_k^b	Backward measurement noise at time instant k.	64
w_k	(Forward) process noise at time instant k.	6
w_k^b	Backward process noise at time instant k.	64
W_u	Weighting matrix with input weight Markov parameters $D_u, C_uB_u, C_uA_uB_u, \ldots$.	140
W_y	Weighting matrix with output weight Markov parameters $D_y, C_yB_y, C_yA_yB_y, \ldots$.	140
$W_u(z)$	Input weighting transfer matrix $D_u + C_u(zI - A_u)^{-1}B_u$.	140
$W_y(z)$	Output weighting transfer matrix $D_y + C_y(zI - A_y)^{-1}B_y$.	140
$W_{0\|i-1}$	Past inputs ($U_{0\|i-1}$) and outputs ($Y_{0\|i-1}$). For stochastic system identification $W_{0\|i-1}$ only contains outputs.	35
W_p, W_p^+	Past inputs (U_p respectively U_p^+) and outputs (Y_f respectively Y_f^+). For stochastic system identification W_p only contains outputs.	35
x_k	State at time instant k.	6
x_k^d	Deterministic state at time instant k.	32
x_k^f	Forward innovation stochastic state at time instant k.	61
x_k^s	Forward stochastic state at time instant k.	58

$\hat{x}_k$	Forward Kalman filter state estimate at time instant k.	69
X_i	State sequence. The subscript indicates the index of the first element of the matrix.	99
X_p	Past state sequence.	139
X_f	Future state sequence.	139
X_i^d	Deterministic state sequence.	35
X_p^d	Past deterministic state sequence.	35
X_f^d	Future deterministic state sequence.	35
X_i^s	Stochastic state sequence.	74
X_p^s	Past stochastic state sequence.	99
X_f^s	Future stochastic state sequence.	99
$\widehat{X}_i$	Forward Kalman filter state sequence.	102
$\widetilde{X}_i$	Forward Kalman filter state sequence.	108
y_k	Output at time instant k.	6
y_k^d	Deterministic output at time instant k.	98
y_k^s	Stochastic output at time instant k.	98
$Y_{0\|i-1}$	Output block Hankel matrix. The subscript indicates the indices of the first column of the matrix.	35
Y_p, Y_p^+	Past outputs $Y_{0\|i-1}$ respectively $Y_{0\|i}$.	35
Y_f, Y_f^-	Future outputs $Y_{i\|2i-1}$ respectively $Y_{i+1\|2i-1}$.	35
$Y(z)$	$\mathcal{Z}$-transform of the output y_k.	140
z_k^s	Backward stochastic state at time instant k.	64
z_k^b	Backward innovation stochastic state at time instant k.	64
$\hat{z}_k$	Backward Kalman filter state estimate at time instant k.	69
$\mathcal{Z}_i$	Orthogonal projection of the row space of $Y_{i\|2i-1}$ on the sum of the row spaces of $U_{0\|2i-1}$ and $Y_{0\|i-1}$.	105
$\widehat{Z}_i$	Backward Kalman filter state sequence.	72

Γ_i	Extended observability matrix.	36
$\underline{\Gamma_i}$	Extended observability matrix Γ_i, without the last l rows.	50
$\overline{\Gamma_i}$	Extended observability matrix Γ_i, without the first l rows.	51

δ_{pq}	Kronecker delta.	6
Δ	Delay operator.	
Δ_i^d	Reversed extended controllability matrix of $\{A, B\}$.	36
Δ_i^s	Reversed extended controllability matrix of $\{A, E\}$.	138
Δ_i^c	Reversed extended controllability matrix of $\{A, G\}$.	68
Λ_0	Covariance of the stochastic output.	61
Λ_i	Stochastic output covariance sequence.	61
$\Pi(a\|b)$	Minimum variance estimate of a given b.	63
Π_A	Operator projecting the row space of a matrix onto the row space of A.	19
$\Pi_{A^\perp}$	Operator projecting the row space of a matrix onto the orthogonal complement of the row space of A.	20
$\sigma_k[W_u(z), W_y(z)]$	Frequency weighted Hankel singular values.	143
Σ^s	Covariance of the stochastic state sequence : $\Phi_{[X_0^s, X_0^s]}$.	60
Σ^d	covariance of the deterministic state sequence : $\Phi_{[X_0^d, X_0^d]}$.	100
$\Phi_{[A,B]}$	Covariance matrix of A and B : $\mathbf{E_j}[AB^T]$.	27
Ω_k	Auxilary matrix in the linear combinations of the combined Kalman filter state estimate.	101

$\mathbb{R}^l$	Vector space of l-dimensional real vectors.	
$\mathbb{R}^{l\times m}$	Vector space of $l \times m$-dimensional real matrices.	
$\mathbf{E}[\bullet]$	Expected value operator.	25
$\mathbf{E_j}[\bullet]$	Time average : $\lim_{j\to\infty} \frac{1}{j}[\bullet]$.	26
$A/\boldsymbol{B}$	Projection of the row space of A on the row space of B.	19
$A/_B\,\boldsymbol{C}$	Oblique projection of the row space of A along the row space of B on the row space of C.	21
$[\boldsymbol{A} \sphericalangle B]$	Principal directions in the row space of A.	24
$[A \sphericalangle \boldsymbol{B}]$	Principal directions in the row space of B.	24
$[A \sphericalangle B]$	Principal angles between the row spaces of A and B.	24
$A^\dagger$	Moore-Penrose pseudo-inverse of A.	
$A \otimes B$	Kronecker product of A and B.	126
vec A	Column-wise vectorization of a matrix A.	126

$\|A\|_F$	Frobenius norm of a matrix A.
$\|A(z)\|_\infty$	H-Infinity norm of a transfer matrix $A(z)$.

CACSD	**C**omputer **A**ided **C**ontrol **S**ystem **D**esign.	7
CVA	**C**anonical **V**ariate **A**nalysis.	114
GUI	**G**raphical **U**ser **I**nterface.	11
MOESP	**M**ultivariable **O**utput-**E**rror **S**tate s**p**ace.	113
N4SID	**N**umerical algorithms for **S**ubspace **S**tate **S**pace **S**ystem **Id**entification.	112
PC	**P**rincipal **C**omponent.	78
SVD	**S**ingular **V**alue **D**ecomposition.	
QSVD	**Q**uotient **S**ingular **V**alue **D**ecomposition.	60
UPC	**U**nweighted **P**rincipal **C**omponent.	79

REFERENCES

[Abd 94] Abdelghani M., *Les Algorithms Sous-Espaces pour l'Analyse Modale et l'Identification des Structures Mechaniques.* Rapport Interne LMGC-CS RI94-02, Laboratoire de Mécanique et Génie Civil, Université Montpelier, France, 1994.

[AML 94] Abrahamsson T., McKelvey T., Ljung L. *A study of some approaches to vibration data analysis.* Proc. of SYSID '94, Vol. **3**, 4-6 July, Copenhagen, Denmark, pp.289-294, 1994.

[Aka 74] Akaike H. *Stochastic theory of minimal realization.* IEEE Transactions on Automatic Control, **19**, pp. 667-674, 1974.

[Aka 75] Akaike H. *Markovian representation of stochastic processes by canonical variables.* Siam J. Control, Vol. **13**, no.1, pp. 162-173, 1975.

[AF 87] Al-Saggaf U.M., Franklin G.F. *An error bound for a discrete reduced order model of a linear multivariable system.* IEEE Trans. on Aut. Control, Vol **32**, pp. 815-819, 1987.

[AMKMVO 93] Aling H., Milletti U., Kosut R.L., Mesaros M.P., Van Overschee P., De Moor B. *An interactive system identification module for Xmath.* Proc. of the American Control Conference, June 2-4, San Francisco, USA, pp. 3071-3075, 1993.

[AKVODMB 93] Aling H., Kosut R., Van Overschee P., De Moor B., Boyd S. *Xmath Interactive System Identification Module, Part 1.* Integrated Systems Inc., Santa Clara, USA, 1993.

[AM 89] Anderson B.D.O., Moore J.B. *Optimal Control - Linear Quadratic Methods.* Prentice Hall, 1989.

[Aok 87] Aoki M. *State space modeling of time series.* Springer Verlag, Berlin, 1987.

[AK 90] Arun K.S., Kung S.Y. *Balanced approximation of stochastic systems.* SIAM J. Matrix Analysis and Applications, **11**, pp. 42-68, 1990.

[AE 71] Åström K., Eykhoff P. *System identification - A survey.* Automatica, Vol. **7**, pp. 123-167, 1971.

[AW 84] Åström K., Wittenmark B. *Computer Controlled Systems: Theory and Design.* Prentice Hall, 1984.

[Aut 13] Autonne L. *Sur les matrices hypohermitiennes et les unitaires.* Comptes Rendus de l'Académie des Sciences, Paris, Vol. **156**, pp. 858-860, 1913.

[Bac 87] Backx T. *Identification of an Industrial Process : A Markov Parameter Approach.* Doctoral Dissertation, Technical University Eindhoven, The Netherlands, 1987.

[BK 79] Baksalary J.K., Kala R. *Two relations Between Oblique and Λ-Orthogonal Projections.* Linear Algebra and its Applications, Vol. **24**, pp. 99-103, 1979.

[Bel 1873] Beltrami E. *Sulle Funzioni Bilineari*, Giornale di Mathematiche, Battagline G., Fergola E. (editors), Vol. **11**, pp. 98-106, 1873.

[BJ 76] Box G.E., Jenkins G.M. *Time series analysis, forecasting and control.* Revised edition, Holden-Day series in time series analysis and digital processing, Holden-Day, Oackland, 1976.

[BB 90] Boyd S., Balakrishnan V. *A Regularity Result for the Singular Values of a Transfer Matrix and a Quadratically Convergent Algorithm for Computing its L_∞-norm.* Systems and Control Letters, Vol. **15**, pp. 1-7, 1990.

[BB 91] Boyd S., Barrat C. *Linear controller design: Limits of performance.* Prentice Hall Information and System Sciences Series, Englewood Cliffs, NJ, 1991.

[Bre 78] Brewer J.W. *Kronecker Products and Matrix Calculus in System Theory.* IEEE Trans. on Circuits and Systems, Vol CAS-**25**, no. 9, pp. 772-780, 1978.

[Bud 71] Budin M. *Minimal realization of discrete linear systems from input-output observations.* IEEE Transactions on Automatic Control, Vol. AC-**16**, no. 5, pp. 395-401, 1971.

[Cai 88] Caines P. *Linear Stochastic Systems.* Wiley Series in Probability and Mathematical Statistics, 1988.

[Cho 93] Cho Y.M. *Fast subspace based system identification: Theory and practice.* Ph.D. Thesis, Information Systems Lab, Stanford University, CA, USA, 1993.

[CK 93] Cho. Y.M., Kailath T. *Model Identification in Rapid Thermal Processing Systems.* IEEE Trans. on Semicond. Manuf., Vol. **6**, no. 3, 1993.

[CXK 94a] Cho Y.M., Xu G., Kailath T. *Fast Recursive Identification of State Space Models via Exploitation of Displacement Structure.* Automatica, Special Issue on Statistical Signal Processing and Control, Vol. **30**, no. 1, pp. 45-59, 1994.

[CXK 94b] Cho Y.M., Xu G., Kailath T. *Fast Identification of State Space Models via Exploitation of Displacement Structure.* IEEE Transactions on Automatic Control, Vol. AC-**39**, no. 10, 1994.

[CK 95] Cho Y.M., Kailath T. *Fast Subspace-Based System Identification : An Instrumental Variable Approach.* To appear in Automatica.

[Chu 94] Chun T.C. *Geometry of linear systems and identification.* PhD Thesis, Cambridge University, England, March 1994.

[DG 76] Dablemont S.P., Gevers M.R. *Identification and feedback of an industrial glass furnace.* IFAC Symp. on Identification, 1976.

[DBC 81] De Bock H., Claeys F. *ASOLVE: een efficient adaptief impulsresponsiealgorithme voor systemen met meerdere ingangen en uitgangen.* Master's thesis, Department of Electrical Engineering, Katholieke Universiteit Leuven, Belgium, 1981.

[DPS 94] Deistler M., Peternell K., Scherrer W. *Consistency and Relative Efficiency of Subspace Methods.* Proc. of SYSID '94, Vol. **2**, 4-6 July, Copenhagen, Denmark, pp.157-163, 1994.

[DMV 87] De Moor B., Vandewalle J. *A geometrical strategy for the identification of state space models of linear multivariable systems with singular value decomposition.* Proc. of the 3rd International Symposium on Applications of Multivariable System Techniques, April 13-15, Plymouth, UK, pp. 59-69, 1987.

[DMo 88] De Moor B. *Mathematical concepts and techniques for modeling of static and dynamic systems.* PhD thesis, Department of Electrical Engineering, Katholieke Universiteit Leuven, Belgium, 1988.

[DMVMVVM 88] De Moor B., Vandewalle J., Moonen M., Vandenberghe L., Van Mieghem P. *A geometrical approach for the identification of state space models with singular value decomposition.* Symposium on Identification and System Parameter Estimation, 27-31 August, Beijing China, pp. 700-704, 1988.

[DMMVV 88a] De Moor B., Moonen M., Vandenberghe L. and Vandewalle J. *Identification of linear state space models with singular value decomposition using canonical correlation concepts.* SVD and Signal Processing: Algorithms, Applications and Architectures, E. Deprettere (Ed.), Elsevier Science Publishers B.V. (North-Holland), pp. 161-169, 1988.

[DMMVV 88b] De Moor B., Moonen M., Vandenberghe L. and Vandewalle J. *The application of the canonical correlation concept to the identification of linear state space models.* Analysis and Optimization of Systems, A. Bensousan, J.L. Lions, (Eds.), Springer Verlag, Heidelberg, pp. 1103-1114, 1988.

[DMMVV 88c] De Moor B., Moonen M., Vandenberghe L., Vandewalle J. *A geometrical approach for the identification of state space models with singular value decomposition.* Proceedings of the International Conference on Acoustics, Speech and Signal Processing, New York, pp. 2244-2247, 1988.

[DMo 93] De Moor B. *The singular value decomposition and long and short spaces of noisy matrices.* IEEE Transactions on Signal Processing, Vol. **41**, no. 9, pp. 2826-2838, 1993.

[DMo 94] De Moor B. *Numerical algorithms for state space subspace system identification.* Academia Analecta, Klasse der Wetenschappen, Koninklijke Akademie voor Wetenschappen, jaargang 55, no. 5, België, 1994.

[DMVO 94] De Moor B., Van Overschee P., *Graphical User Interface Software for System Identification*, Award winning paper of the *Siemens Award 1994*. ESAT-SISTA Report 94-06I, Department of Electrical Engineering, Katholieke Universiteit Leuven, 1994.

[DMVO 95] De Moor B., Van Overschee P., *Numerical Algorithms for Subspace State Space System Identification.* Trends in Control. A European Perspective. Ed. A. Isidori, European Control Conference, Italy, pp. 385-422, 1995.

[DP 84] Desai U.B., Pal D. *A transformation approach to stochastic model reduction.* IEEE Transactions on Automatic Control, Vol. AC-**29**, no. 12, 1984.

[DKP 85] Desai U.B., Kirkpatrick R.D., Pal D. *A realization approach to stochastic model reduction.* International Journal of Control, Vol. **42**, no. 4, pp. 821-838, 1985.

[DMK 74a] Dickinson B., Morf M., Kailath T. *A minimal realization algorithm for matrix sequences.* IEEE Transactions on Automatic Control, Vol. AC-**19**, no. 1, pp. 31-38, 1974.

[DKM 74b] Dickinson B., Kailath T., Morf M. *Canonical Matrix fraction and state space descriptions for deterministic and stochastic linear systems.* IEEE Transactions on Automatic Control, Vol. AC-**19**, pp. 656-667, 1974.

[EY 36] Eckart C., Young G. *The approximation of one matrix by another of lower rank.* Psychometrika, **1**, pp. 211-218, 1936.

[Enn 84] Enns D. *Model reduction for control system design.* Ph.D. dissertation, Dep. Aeronaut. Astronaut., Stanford University, Stanford CA, 1984.

[Eyk 74] Eykhoff P. *System identification.* Wiley, London, 1974.

[Fal 94] Falkus H. *Parametric Uncertainty in System Identification.* Ph.D. Thesis, Vakgroep Meten en Regelen, EE Department, TU Eindhoven, The Netherlands, 1994.

[FVOMHL 94] Falkus H., Van Overschee P., Murad G., Hakvoort H., Ludlage J. *Advanced Identification and Robust Control of a Glass Tube Production Process.* Third Philips Conference on Applications of Systems and Control Theory (PACT), November 9-10, Doorwerth, The Netherlands, 1994.

[Fau 76] Faure P. *Stochastic realization algorithms.* System Identification: Advances and case studies (Eds.) Mehra R., Lainiotis D., Academic Press, 1976.

[FPW 90] Franklin G.F., Powell J.D., Workman M.L. *Digital Control of Dynamic Systems.* Second edition, Addison-Wesley Publishing Company, New York, 1990.

[Gau 1857] Gauss C. F. *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium.* F.Perthes and J.H. Besser, Hamburg, 1809 (reprinted in: Carl Friedrich Gauss Werke, Vol. VII, Königlichen Gesellschaft der Wissenschaften zu Göttingen, translation: Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections, Little, Brown and Co., Boston, 1857)(reprinted: Dover, New York, 1963).

[Gev 93] Gevers M. *Towards a joint design of identification and control?* In: H.L. Trentelman and J.C. Willems (Eds.), *Essays on Control: Perspectives in the Theory and its Applications.* Birkhäuser, Boston, pp. 111-151, 1993.

[GW 74] Glover K., Willems J. *Parametrizations of linear dynamical systems: canonical forms and identifiability.* IEEE Transactions on Automatic Control, Vol. AC-**19**, pp. 640-645, 1974.

[Glo 84] Glover K. *All optimal Hankel norm approximations of linear multivariable systems and their L^∞ error bounds.* International Journal of Control, **39**, pp. 1115-1193, 1984.

[GVL 89] Golub G., Van Loan C. *Matrix computations.*, second edition, Johns Hopkins University Press, Baltimore, Maryland, 1989.

[Gop 69] Gopinath B. *On the identification of linear time-invariant systems from input-output data.* The Bell System Technical Journal, Vol. **48**, no. 5, pp. 1101-1113, 1969.

[Gra 1883] Gram J.P. *Über die Entwickelung reeler Functionen in Reihen mittelst der Methode der kleinsten Quadrate.* Journal für die reine und angewandte Mathematik, **94**, pp. 41-73, 1883.

[Gui 75] Guidorzi R. *Canonical structures in the identification of multivariable systems.* Automatica, **11**, pp. 361-374, 1975.

[Gui 81] Guidorzi R. *Invariants and canonical forms for systems structural and parametric identification.* Automatica, **17**, pp. 177-133, 1981.

[Gyu 93] Gyugyi Paul. *Model-Based Control Applied to Rapid Thermal Processing*. Ph.D. Thesis Stanford University, CA, USA, 1993.

[Hak 94] Hakvoort R. *System Identification for robust control.* PhD Thesis, Delft University of Technology, The Netherlands, November 1994.

[HD 88] Hannan E.J., Deistler M. *The statistical theory of linear systems.* Wiley Series in Probability and Mathematical Statistics, John Wiley and Sons, New York, 1988.

[HK 66] Ho B.L., Kalman R.E. *Efficient construction of linear state variable models from input/output functions.* Regelungstechnik, **14**, pp. 545-548,1966.

[Hot 36] Hotelling H. *Relation between two sets of variates.* Biometrika, Vol. **28**, pp. 321-372, 1936.

[JW 94] Jansson M., Wahlberg B. *4SID Linear Regression.* Proc. 33rd IEEE Conference on Decision and Control, Lake Buena Vista, Florida, USA, 14-16 Dec., pp. 2858-2863, 1994.

[Jor 1874] Jordan C. *Mémoire sur les formes bilinéaires.* J. Math. Pures Appl. II, Vol. **19**, pp. 35-54, 1874.

[Jor 1875] Jordan C. *Essai sur la géometrie à n dimensions.* Bull. Soc. Math. France, **3**, pp. 103-174, 1875.

[Kai 80] Kailath T. *Linear Systems.* Prentice Hall, Englewood Cliffs, New Jersey, 1980.

[Kal 60] Kalman R. *A new approach to linear filtering and prediction problems.* J. Basic Eng. **82**, pp. 35-50, March 1960.

[Kun 78] Kung S.Y. *A new identification method and model reduction algorithm via singular value decomposition.* 12th Asilomar Conf. on Circuits, Systems and Comp., pp. 705-714, Asilomar, CA, 1978.

[Lar 83] Larimore W.E. *System identification, reduced order filtering and modeling via canonical variate analysis.* Proc. of the American Control Conference, San Francisco, USA, 1983.

[Lar 90] Larimore W.E. *Canonical variate analysis in identification, filtering and adaptive control.* Proc. 29th Conference on Decision and Control, Hawai, USA, pp. 596-604, 1990.

[Lar 94] Larimore W.E. *The Optimality of Canonical Variate Identification by Example*, Proc. of SYSID '94, Vol. **2**, 4-6 July, Copenhagen, Denmark, pp.151-156, 1994.

[Lau 79] Laub A. *A Schur method for solving algebraic Riccati equations.* IEEE Transactions on Automatic Control, AC-**24**, pp. 913-921, 1979.

[LP 93] Lindquist A., Picci G. *On "Subspace Methods" Identification.* Proc. of MTNS, Regensburg, Germany, August 2-6, Vol. **2**, pp. 315-320, 1993.

[LP 94] Lindquist A., Picci G. *On "Subspace Methods" Identification and Stochastic Model reduction.* Proc. of SYSID '94, Vol. **2**, 4-6 July, Copenhagen, Denmark, pp. 397-404, 1994.

[LS 91] Liu K., Skelton R.E., *Identification and Control of NASA's ACES Structure.* Proceedings American Control Conference, Boston, MA, USA, pp. 3000-3006, 1991.

[Liu 92] Liu K., *Identification of Multi-Input and Multi-Output Systems by Observability Range Space Extraction.* Proc. 31st Conference on Decision and Control, Tucson, Arizona, USA, pp. 915-920, 1992.

[LSS 92] Liu K., Skelton R.E., Sharkey J.P., *Modeling Hubble Space Telescope Flight Data by Q-Markov Cover Identification.* Proc. of the American Control Conference, pp. 1961-1965, 1992.

[LJM 94] Liu K., Jacques R.N., Miller D.W. *Frequency Domain Structural System Identification by Observability Range Space Extraction.* Proc. of the American Control Conference, Baltimore, Maryland, Vol. **1**, pp. 107-111, 1994.

[LS 77] Liu R., Suen L.C. *Minimal dimension realization and identifiability of input-output sequences.* IEEE Transactions on Automatic Control, Vol. AC-**22**, pp. 227-232, 1977.

[Lju 87] Ljung L. *System identification - Theory for the User.* Prentice Hall, Englewood Cliffs, NJ, 1987.

[Lju 91a] Ljung L. *Issues in System Identification.* IEEE Control Systems, Vol. **11**, no. 1, pp. 25-29, 1991.

[Lju 91b] Ljung. L. *A Simple Start-Up Procedure for Canonical Form State Space Identification, Based on Subspace Approximation.* 30th IEEE Conference on Decision and Control, Brighton, UK, pp. 1333-1336, 1991.

[Lju 91c] Ljung. L. *System Identification Toolbox For Use with Matlab.* The Mathworks Inc., Mass., U.S.A., 1991.

[Lue 67] Luenberger D.G. *Canonical forms for linear multivariable systems.* IEEE Transactions on Automatic Control, Vol. AC-**12:290**, 1967.

[Mac 94] Maciejowski J.M. *Guaranteed Stability with Subspace Methods.* Submitted for publication.

[McK 94a] McKelvey T., *On State-Space Models in System Identification*, Thesis no. 447, Department of Electrical Engineering, Linköping university, Sweden, 1994.

[McK 94b] McKelvey T. *An Efficient Frequency Domain State-Space Identification Algorithm.* Proc. 33rd IEEE Conference on Decision and Control, Lake Buena Vista, Florida, USA, 14-16 Dec., pp. 3359-3364, 1994.

[McK 94c] McKelvey T. *SSID - A MATLAB Toolbox for Multivariable State-Space Model Identification.* Dept. of EE, Linköping University, Linköping Sweden, 1994.

[MDMVV 89] Moonen M., De Moor B., Vandenberghe L., Vandewalle J. *On and off-line identification of linear state space models.* International Journal of Control, Vol. **49**, no. 1, pp. 219-232, 1989.

[MV 90] Moonen M., Vandewalle J. *A QSVD approach to on- and off-line state space identification.* International Journal of Control, Vol. **51**, no. 5, pp. 1133-1146, 1990.

[MDMV 91] Moonen M., De Moor B., Vandewalle J. *SVD-based subspace methods for multivariable continuous time system identification.* Identification of continuous-time systems, G.P. Rao, N.K. Sinha (Eds.), Kluwer Academic Publications, pp. 473-488, 1991.

[MDM 92] Moonen M., De Moor B. *Comments on 'State-space model identification with data correlation'.* International Journal of Control, Vol. **55**, no. 1, pp. 257-259, 1992.

[MDMRT 92] Moonen M., De Moor B., Ramos J., Tan S. *A subspace identification algorithm for descriptor systems.* Systems & Control Letters, Vol. **19**, pp. 47-52, 1992.

[MVODM 92] Moonen M., Van Overschee P., De Moor B., *A subspace algorithm for combined stochastic-deterministic state space identification.* ESAT-SISTA Report 92-10I, Department of Electrical Engineering, Katholieke Universiteit Leuven, 1992.

[MR 93] Moonen M, Ramos J. *A subspace algorithm for balanced state space system identification.* IEEE Transactions on Automatic Control, Vol. **38**, pp. 1727-1729, 1993

[Moo 81] Moore B.C. *Principal component analysis in linear systems: Controllability, Observability and Model Reduction.* IEEE Transactions on Automatic Control, Vol. AC-**26**, no. 1, pp 17-32, 1981.

[MR 76] Mullis C.T., Roberts R.A. *Synthesis of minimum round-off noise fixed point digital filters.* IEEE Transactions on Circuits and Systems, Vol. CAS-**23**, pp. 555-562, 1976.

[NA 81] Nakamura H., Akaike H. *Statistical Identification for Optimal Control of Supercritical Power Plants.* Automatica, Vol. **17**, no. 1, pp. 143-155, 1981.

[Nor 86] Norton J.P. *An introduction to identification.* Academic Press, London, 1986.

[Ott 89] Ottersten B. *Parametric subspace fitting methods for array signal processing.* Ph.D. Thesis, Information Systems Laboratory, Department of Electrical Engineering, Stanford University, CA, USA, 1989.

[OVK 92] Ottersten B., Viberg M., Kailath T., *Analysis of subspace fitting and ML techniques for parameter estimation from sensor array data*, IEEE Tr. on SP, Vol. **40**, no. 3, pp. 590-600, 1992.

[OVSN 93] Ottersten B., Viberg M., Stoica P., Nehorai A., *Exact and large sample maximum likelihood techniques for parameter estimation and detection in array processing*, in Radar array processing, S.Haykin, J.Litva, T.J.Sherpherd (Eds.), Springer Verlag, pp. 99-151, 1993.

[OV 94] Ottersten B., Viberg M., *A Subspace Based Instrumental Variable Method for State Space System Identification.* Proc. of SYSID '94, Vol. **2**, 4-6 July, Copenhagen, Denmark, pp.139-144, 1994.

[Pal 82] Pal D., *Balanced stochastic realization and model reduction.* Master's thesis, Washington State University, Electrical Engineering, 1982.

[Pap 84] Papoulis A., *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, NY, 1984.

[RA 92] Rao B., Arun K.S. *Model Based Processing of Signals : A State Space Approach.* Proceedings of the IEEE, Vol. **80**, no. 2, pp. 283-309, 1992.

[SDM 94] Schelfhout G., De Moor B. *Inverse Outer Factor Weighted Balanced Truncation.* ESAT-SISTA Report 94-43I, Department of Electrical Engineering, Katholieke Universiteit Leuven, 1994. Submitted for publication.

[Sch 07] Schmidt E. *Zur Theorie der linearen und nichtlinearen Integralgleichungen. I Tiel. Entwicklung willkürlichen Funktionen nach System vorgeschriebener.* Mathematische Annalen, **63**, pp. 433-476, 1907.

[Sch 91] Schrama R., *An Open-Loop Solution to the Approximate Closed-Loop Identification Problem.* Proc. 9th IFAC/IFORS Symp. on Identification and System Parameter Estimation, Budapest, Hungary, pp. 1602-1607, 1991.

[SS 89] Söderström T., Stoica P. *System Identification.* Prentice Hall International Series in Systems and Control Engineering, Prentice Hall, New York, 1989.

[Sil 71] Silverman L. *Realization of linear dynamical systems.* IEEE Transactions on Automatic Control, Vol. AC-**16**, pp. 554-567, 1971.

[SROK 92] Swindlehurst A., Roy R., Ottersten B., Kailath T. *System identification via weighted subspace fitting.* Proc. of the American Control Conference, pp. 2158-2163, 1992.

[Syl 1889] Sylvester J.J. *Sur la réduction biorthogonale d'une forme linéo-lin'eaire à sa forme canonique.* Comptes Rendus, CVIII., pp. 651-653, 1889

[Sys 1994] *System Identification Competition, Benchmark tests for estimation methods of thermal characteristics of buildings and building components.* Organization : J. Bloem, Joint Research Centre, Ispra, Italy, 1994.

[Vac 87] Vaccaro R.J. *A State Space Approach to Positive Sequences.* proceedings of ICASSP, pp. 1304-1307, 1987.

[VV 93] Vaccaro R.J., Vukina T. *A solution to the positivity problem in the state-space approach to modeling vector-valued time series.*Journal of Economic Dynamics and Control, Vol. **17**, pp. 401-421, 1993.

[VTS 95] Vaccaro R.J., Tufts D.W., Shah A.A. *Parameter Estimation and Order Determination in the Low-Rank Linear Stochastical Model.* SVD and Signal Processing: Algorithms, Applications and Architectures, M. Moonen, B. De Moor (Eds.), Elsevier Science Publishers B.V. (North-Holland), 1995.

[VVDVOV 94] Vanrolleghem P.A., Van Daele M., Van Overschee P., Vansteenkiste G.C., *On-Line Model Structure Characterization of Nonlinear Wastewater Treatment Systems.* Proc. of SYSID '94, Vol. **1**, 4-6 July, Copenhagen, Denmark, pp. 279-284, 1994.

[VDHS 93] Van Den Hof P., Schrama R.J.P. *An Indirect Method for Transfer Function Estimation From Closed Loop Data.* Automatica, Vol. **29**, no. 6, pp. 1523-1527, 1993.

[VDS 93] Van Der Veen A., Deprettere E.F., Swindlehurst A.L. *Subspace-Based Signal Analysis Using Singular Value Decompositions.* Proceedings of the IEEE, Vol. **81**, no. 9, pp. 1277-1308, 1993.

[VOL 82] van Overbeek A.J.M., Ljung L. *On-line structure selection for multivariable state space models.* Automatica, **18**, no. 5, pp. 529-543, 1982.

[VODM 91a] Van Overschee P., De Moor B., *Subspace approach to stochastic realization and positive real sequences.*, ESAT-SISTA Report 91-02I, Department of Electrical Engineering, Katholieke Universiteit Leuven, 1991.

[VODMS 91] Van Overschee P., De Moor B., Suykens J. *Subspace algorithms for system identification and stochastic realization.* Proc. Conf. on Mathematical Theory for Networks and Systems, MTNS, Kobe, Japan, pp. 589-595, Mita Press, 1991.

[VODM 91b] Van Overschee P., De Moor B. *Subspace algorithms for the stochastic identification problem.* 30th IEEE Conference on Decision and Control, Brighton, UK, pp. 1321-1326, 1991.

[VODM 92] Van Overschee P., De Moor B. *Two subspace algorithms for the identification of combined deterministic stochastic systems.* Proc. of the 31st IEEE Conference on Decision and Control, December 16-18, Tucson, USA, Vol. **1**, pp. 511-516, 1992.

[VODM 93a] Van Overschee P., De Moor B. *Subspace algorithms for the stochastic identification problem.* Automatica, Vol. **29**, no. 3, 1993, pp. 649-660.

[VODM 93b] Van Overschee P., De Moor B. *N4SID: Numerical Algorithms for State Space Subspace System Identification* Proc. of the World Congress of the International Federation of Automatic Control, IFAC, Vol. **7**, pp. 361-364, Sydney, Australia, 1993.

[VODM 93c] Van Overschee P., De Moor B. *Subspace identification of a glass tube manufacturing process.* Proc. of the second European Control Conference, June 28-July 1, Groningen, The Netherlands, pp. 2338-2343, 1993.

[VODM 94a] Van Overschee P., De Moor B. *N4SID: Subspace Algorithms for the Identification of Combined Deterministic-Stochastic Systems.* Automatica, Special Issue on Statistical Signal Processing and Control, Vol. **30**, no. 1, pp. 75-93, 1994.

[VODM 94b] Van Overschee P., De Moor B. *A Unifying Theorem for three Subspace System Identification Algorithms.* Proc. of SYSID '94, Vol. **2**, 4-6 July, Copenhagen, Denmark, pp. 145-150, 1994.

[VODMBAK 94] Van Overschee P., De Moor B., Boyd S., Aling H., Kosut R. *A fully interactive system identification module for Xmath (ISID)*, Proc. of SYSID '94, Vol. **4**, Copenhagen, Denmark, pp. 1, 1994.

[VODM 94c] Van Overschee P., De Moor B. *A Unifying Theorem for three Subspace System Identification Algorithms.* American Control Conference, June 29-July 1, Baltimore, pp. 1645-1649, 1994.

[VODMAKB 94] Van Overschee P., De Moor B., Aling H., Kosut R., Boyd S. *Xmath Interactive System Identification Module, Part 2.* Integrated Systems Inc., Santa Clara, CA, USA, 1994.

[VO 94] Van Overschee P., *Data Processing for System Identification.* In: J. Van Impe, P. Vanrolleghem and D. Iserentant (Eds.), *Advanced Instrumentation, Data Interpretation, and Control of Biotechnological Processes*, Kluwer Academic Publishers, 1994.

[VODM 95a] Van Overschee P., De Moor B., *A Unifying Theorem for three Subspace System Identification Algorithms.* Automatica, Special Issue on Trends in System Identification, Vol. **31**, no. 12, pp. 1853-1864, 1995.

[VODM 95b] Van Overschee P., De Moor P. *About the choice of State Space Bases in Combined Deterministic-Stochastic Subspace Identification*, Automatica, Special Issue on Trends in System Identification, Vol. **31**, no. 12, pp. 1877-1883, 1995.

[VDM 93] Vanvuchelen P., De Moor B. *A Multi-Objective Optimization Approach for Parameter Setting in System and Control Design*. Proc. of the 16th IFIP Conference on system modeling and optimization, Compiegne, July 5-9, France pp. 63-66, 1993.

[Ver 91] Verhaegen M. *A novel non-iterative MIMO state space model identification technique.* Proc. 9th IFAC/IFORS Symp. on Identification and System Parameter Estimation, Budapest, Hungary, pp. 1453-1458, 1991.

[VD 91] Verhaegen M., Deprettere E. *A Fast, Recursive MIMO State Space Model Identification Algorithm.* Proc. of the 30th IEEE Conference on Decision and Control, Brighton, England, 11-13 Dec, Vol. **2**, pp 1349-1354, 1991.

[VD 92] Verhaegen M., Dewilde P. *Subspace model identification, Part I: The output-error state space model identification class of algorithms.* Int. J. Control, Vol. **56**, 1187-1210, 1992.

[Ver 93] Verhaegen M. *Application of a Subspace Model Identification Technique to Identify LTI Systems Operating in Closed-Loop.* Automatica, Vol. **28**, no. 4, pp. 1027-1040, 1993.

[Ver 94] Verhaegen M. *Identification of the deterministic part of MIMO state space models given in innovations form from input-output data*, Automatica (Special Issue on Statistical Signal Processing and Control), Vol **30**, no. 1, pp. 61-74, 1994.

[Vib 89] Viberg M. *Subspace fitting concepts in sensor array processing.* Ph.D. thesis, Department of Electrical Engineering, Linköping University, Sweden, 1989.

[VO 91] Viberg M., Ottersten B. *Sensor array processing based on subspace fitting.* IEEE Transactions on Acoustics, Speech and Signal Processing, ASSP-**39**, 1991.

[VOWL 91] Viberg M., Ottersten B., Wahlberg B., Ljung L. *A Statistical Perspective on State-Space Modeling Using Subspace Methods.* Proc. of the 30th IEEE Conference on Decision and Control, Brighton, England, 11-13 Dec, Vol **2**, pp 1337-1342, 1991.

[VOWL 93] Viberg M., Ottersten B., Wahlberg B., Ljung L., *Performance of Subspace Based State Space System Identification Methods.* Proc. of the 12th IFAC World Congress, Sydney, Australia, 18-23 July, Vol **7**, pp 369-372, 1993.

[Vib 94] Viberg M. *Subspace Methods in System Identification.* Proc. of SYSID '94, Vol. **1**, 4-6 July, Copenhagen, Denmark, pp. 1-12, 1994.

[Wan 94] Wang J., *Algorithms for Approximate Closed-Loop Identification of State Space Representations.* Master's Thesis, Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium, 1994.

[WS 91] Wang W., Safanov M.G. *Relative-error bound for discrete balanced stochastic truncation.* Int. J. Control, Vol. **54**, no. 3, pp. 593-612, 1991.

[WB 92] Wortelboer P.M.R., Bosgra O.H. *Generalized frequency weighted balanced reduction.* Selected Topics in Identification, Modeling and Control, Delft University Press, Vol **5**, pp. 29-36, 1992.

[Wil 86] Willems J. *From time series to linear systems.* Automatica, Part I: Vol. **22**, no. 5, pp. 561-580, 1986, Part II: Vol. **22**, no. 6, pp. 675-694, 1986, Part III: Vol. **23**, no. 1, pp. 87-115, 1987.

[ZM 74] Zeiger H., McEwen A. *Approximate linear realizations of given dimension via Ho's algorithm.* IEEE Transactions on Automatic Control, **19**, pp. 153, 1974.

[ZVODML 94] Zhu Y., Van Overschee P., De Moor B., Ljung L. *Comparison of three classes of identification methods.* Proc. of SYSID '94, Vol. **1**, 4-6 July, Copenhagen, Denmark, pp. 175-180, 1994.

INDEX

D

E

F

N

O

P

R

S