State space identification
Theory and practice

L.R.J. Haverkamp
State space identification

Theory and practice

Bert Haverkamp
State space identification

Theory and practice

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.F. Wakker,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 13 februari 2001 om 13:30 uur
door Lambertus Roelof Jan HAVERKAMP
elektrotechnisch ingenieur
geboren te Den Helder.
Dit proefschrift is goedgekeurd door de promotoren:
Prof. ir. H.B. Verbruggen
Prof. dr. ir. M.H.G. Verhaegen

Samenstelling promotiecommissie:

Rector Magnificus,
Prof. ir. H.B. Verbruggen,
Prof. ir. M.H.G. Verhaegen,
Prof. dr. ir. A.C.P.M. Backx,
Prof. dr. J.H. van Schuppen,
Dr. B. Hanzon,
Prof. dr. R. Johansson,
Prof. dr. ir. P.M.J. van den Hof,

Technische Universiteit Delft, voorzitter
Technische Universiteit Delft, promotor
Universiteit Twente, promotor
Technische Universiteit Eindhoven
Centrum voor Wiskunde en Informatica
Vrije Universiteit Amsterdam
Lund University
Technische Universiteit Delft

Copyright © 2001 by Bert Haverkamp. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission in writing from the author.
Preface

This book is the result of five years of research and writing. The task is almost done, and new challenges lie ahead. But I can’t move on without thanking all the people involved in this work.

First of all I would like to thank Prof. Michel Verhaegen for offering me the Ph.D position and pulling me into the field of system identification and modeling. Michel has been a source of motivation. His knowledge on the subject keeps amazing and new ideas come sometimes faster than his students can process them. He has been the major driving force for this thesis work. First as mentor, later as promoter.

Secondly, I would like to thank Prof. Henk Verbruggen. Henk has supported and motivated me, especially in the later period of my thesis work. We have had many discussions on many topics, both in and outside the field of research.

Then I have to mention my roommates at the 9th floor of the Electrotechnical faculty of Delft University. Tung Chou and David Westwick were wonderful colleagues who taught me a lot about topics ranging from spiders to Sylvester equations.

I immediately have to add the other people of the group. Vincent Verdult, Gerard Nijssse, Ton van den Boom and Jacquelyn Scherpen have provided me with lots of input and discussion, and became good friends. Also Marco Lovera, who spend a period at the Control Laboratory left a lasting impression.

Outside the university I would like to thank the Fam. Haring, for “adopting” me during my life in Schipluiden, all the Explo’s of scouting Schipluiden for taking my mind of technical issues and of course my family, who sometimes had to cope with a radio-silence of months.

Finally I’d like to thank Selena, whose care and love have enabled me to cope with the combination of a full-time job and a Ph.D thesis during the past year.

Thanks!

Bert Haverkamp
Maastricht, December 2000

vii
Contents

Preface vii

1 Introduction 1

2 Subspace Model Identification 9

2.1 The state space realization problem 10
  2.1.1 Problem definition 11
  2.1.2 Kung’s algorithm 11

2.2 The noise-free identification problem 14
  2.2.1 Problem definition 14
  2.2.2 The ordinary MOESP method 16

2.3 The white output-error identification problem 20
  2.3.1 Problem definition 20
  2.3.2 Ordinary MOESP and the white output-error problem 21

2.4 The output-error identification problem 24
  2.4.1 Problem definition 25
  2.4.2 The past input instrumental variable method 26

2.5 The innovations model identification problem 30
  2.5.1 Problem definition 30
  2.5.2 The past output instrumental variable method 31
2.5.3 Estimating the stochastic part of the innovations model ........ 36

2.6 The errors-in-variables identification problem ..................... 38
  2.6.1 Problem definition ........................................ 40
  2.6.2 The errors-in-variables method ................................ 41

2.7 The closed loop identification problem ............................. 42
  2.7.1 Problem definition ........................................ 42
  2.7.2 Errors-in-variable MOESP and the closed loop problem ........ 43

2.8 Persistence of excitation ......................................... 45
  2.8.1 Requirements on the input for OM-MOESP ........................ 45
  2.8.2 Requirements on the input for PI-MOESP ........................ 46
  2.8.3 Requirements on the input for PO-MOESP ........................ 47
  2.8.4 Requirements on the input for EIV-MOESP ...................... 49

2.9 Estimation of $B$ and $D$ and the initial state .................... 50
  2.9.1 Algorithm for estimation $B$, $D$ and $x_0$ .................... 50
  2.9.2 Estimating $B$, $D$ and $x_0$ in closed loop .................. 51

2.10 The Wiener model identification problem .......................... 54
  2.10.1 WIE-MOESP with odd non-linear functions .................... 56
  2.10.2 WIE-MOESP with even non-linearities ........................ 57
  2.10.3 Estimating $A_T$ and $D_TC_T$ ................................ 58
  2.10.4 Estimating $B_T$ and $D_TD_T$ ................................ 62
  2.10.5 Estimating the non-linear function ........................... 63

2.11 Conclusion .................................................. 63

3 Continuous time subspace identification .......................... 67

3.1 Introduction .................................................. 67

3.2 Continuous time systems ........................................ 71

3.3 Continuous time disturbances ..................................... 71
3.3.1 Semi-white noise ........................................ 73  
3.3.2 Wiener processes ....................................... 75  
3.3.3 Discretizing stochastic differential equations .............. 79  
3.4 Laguerre and all-pass filters ................................ 86  
3.4.1 The Laguerre filter ...................................... 86  
3.4.2 The all-pass filter ........................................ 88  
3.4.3 Relation with the z-transformation ......................... 90  
3.4.4 All-pass domain state space description .................... 92  
3.4.5 All-pass domain data equation ............................ 94  
3.5 The continuous time noise-free identification problem ......... 96  
3.5.1 Problem definition ....................................... 96  
3.5.2 The continuous time ordinary MOESP method ............... 97  
3.5.3 Approximation of Laguerre filters ........................ 99  
3.5.4 Error analysis .......................................... 103  
3.6 The innovations model identification problem ................ 106  
3.6.1 Problem definition ....................................... 106  
3.6.2 The continuous time PO-MOESP method .................... 107  
3.6.3 Error analysis .......................................... 113  
3.7 Estimation of B, D and the initial state ...................... 118  
3.8 Identification of the stochastic part of the innovations model . 120  
3.8.1 The continuous time Kalman filter ........................ 120  
3.8.2 The all-pass domain Kalman filter ........................ 121  
3.8.3 Identification of the continuous time stochastic system .... 124  
3.9 Identification of continuous time Wiener systems ............. 126  
3.9.1 Problem definition ....................................... 128  
3.9.2 The CWIE-MOESP algorithm .............................. 130  
3.10 Examples .................................................. 132
3.10.1 Example of using non-equidistant sampled data applied to the identification of stiff systems ........................................ 132
3.10.2 Example of the use of CPO-MOESP on a real life data-set .......................................................... 135
3.10.3 Example of the Wiener model identification .......................................................... 138
3.11 Conclusion ......................................................................................................................... 138

4 Separable Least Squares in state space identification ................................................................. 141

4.1 Introduction ................................................................................................................... 141
4.2 Separable Least Squares .............................................................................................. 143
4.3 Separation of parameters in discrete time state space models ........................................ 144
4.3.1 Tri-diagonal parameterization ............................................................................. 146
4.3.2 Output normal parameterization ........................................................................ 147
4.4 Gauss-Newton optimization and Separable Least Squares ........................................ 151
4.5 Examples of SLS optimization .................................................................................... 153
4.5.1 Example of parameter space reduction ............................................................. 154
4.5.2 Comparison between non-SLS and SLS based optimization ................................ 155
4.6 Recursive optimization using SLS .................................................................................. 157
4.6.1 Recursion of $H_k^{-1}(\hat{\theta}_u(k - 1))$ .................................................................. 159
4.6.2 Recursion of $\hat{\phi}_k$ ............................................................................................ 160
4.6.3 Recursion of $\hat{\theta}_1(k)$ ..................................................................................... 161
4.6.4 Recursion of $\hat{\phi}_k$ ............................................................................................ 161
4.7 Separable Least Squares and continuous time state space models .................................. 163
4.7.1 Continuous time output normal parameterization ............................................. 165
4.7.2 Recursive optimization of continuous time model .............................................. 166
4.8 prediction-error optimization and SLS ............................................................................ 169
4.9 Wiener models and SLS .............................................................................................. 170
4.10 Numerical example of recursive MIMO identification .................................................... 171
4.11 Example of the use of continuous time SLS on a real life data set .................................. 175
5 Implementation issues

5.1 Introduction ......................................................... 177
5.2 Efficient implementation of MOESP by using the Cholesky factorization ........................................... 178
5.3 Efficient implementation of the estimation of $B$ and $D$ ................................................................. 182
  5.3.1 Implementation of $Y_{0,N,1}$ ............................................. 183
  5.3.2 Implementation of $\Gamma_N$ ............................................... 184
  5.3.3 Implementation of $\mathcal{Y}$ .............................................. 185
  5.3.4 Implementation of $\mathcal{U}$ ............................................... 188
  5.3.5 Benchmark test .......................................................... 189
5.4 Solving for $B$ and $D$ .................................................... 190
  5.4.1 The Penrose-Moore pseudo-inverse .................................... 190
  5.4.2 Left matrix divide ...................................................... 191
  5.4.3 The pseudo inverse ....................................................... 191
  5.4.4 The QR factorization ..................................................... 192
  5.4.5 Benchmark test .......................................................... 194
5.5 Comparing different SMI algorithms ...................................... 194
  5.5.1 Modulizing MOESP and N4SID .......................................... 195
5.6 Combining MOESP and N4SID ............................................... 197
  5.6.1 Benchmark test .......................................................... 197
5.7 Conclusions ............................................................. 201

6 The SMI-toolbox ................................................................... 203

6.1 Introduction .............................................................. 203
6.2 Example 1: Using the SMI-toolbox ..................................... 204
6.3 Example 2: Using the SMI-toolbox with multiple data-sets ........ 208
6.4 Example 3: Identifying a Wiener model ............................... 214
6.5 Example 4: Identifying in a closed loop setting 219
6.6 Conclusion 223

7 Conclusions 225

A Important basic concepts 229
A.1 Projection matrices 229
A.2 The QR factorization 230
A.3 The Singular Value Decomposition 232
A.4 The Kronecker product 234
A.5 Matrix derivations 235
A.6 The State-space equation 236
A.7 Controllability 237
A.8 Observability 238
A.9 Minimality 238
A.10 Relation to impulse response functions 239
A.11 Relation to transfer functions 239
A.12 Matrix inversion lemma 239

B User guide to SMI 2.0 241
B.1 List of functions in SMI 2.0 241
cestac 244
cestbd 245
cestx 246
chebest 247
chebsim 248
cordom 249
cordpi 250
<table>
<thead>
<tr>
<th>Content</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>cordpo</td>
<td>251</td>
</tr>
<tr>
<td>css2th</td>
<td>252</td>
</tr>
<tr>
<td>cth2ss</td>
<td>253</td>
</tr>
<tr>
<td>destac</td>
<td>254</td>
</tr>
<tr>
<td>destbd</td>
<td>255</td>
</tr>
<tr>
<td>destk</td>
<td>257</td>
</tr>
<tr>
<td>destx</td>
<td>258</td>
</tr>
<tr>
<td>dfunlin</td>
<td>259</td>
</tr>
<tr>
<td>dfunwie</td>
<td>260</td>
</tr>
<tr>
<td>dmoesp</td>
<td>261</td>
</tr>
<tr>
<td>dordeiv</td>
<td>262</td>
</tr>
<tr>
<td>dordom</td>
<td>264</td>
</tr>
<tr>
<td>dordpi</td>
<td>265</td>
</tr>
<tr>
<td>dordpo</td>
<td>266</td>
</tr>
<tr>
<td>dordrs</td>
<td>267</td>
</tr>
<tr>
<td>drpi</td>
<td>268</td>
</tr>
<tr>
<td>drpo</td>
<td>269</td>
</tr>
<tr>
<td>drslslin</td>
<td>270</td>
</tr>
<tr>
<td>dslslin</td>
<td>271</td>
</tr>
<tr>
<td>dslswie</td>
<td>273</td>
</tr>
<tr>
<td>dss2th</td>
<td>275</td>
</tr>
<tr>
<td>dth2ss</td>
<td>276</td>
</tr>
<tr>
<td>kronekf</td>
<td>277</td>
</tr>
<tr>
<td>ncdestac</td>
<td>278</td>
</tr>
<tr>
<td>ncdestbd</td>
<td>279</td>
</tr>
<tr>
<td>ncdlsim</td>
<td>281</td>
</tr>
<tr>
<td>orderselect</td>
<td>282</td>
</tr>
</tbody>
</table>
Contents

prbn ................................................. 283
shave ................................................. 284
tlsim ................................................. 285
vaf ...................................................... 286

Index ................................................. 289

Biographies of scientists ......................... 289

Bibliography ......................................... 298

Summary .............................................. 299

Samenvatting ........................................ 303

Publications ......................................... 307

Curriculum Vitae .................................... 309
Chapter 1

Introduction

This thesis is on the subject of state space identification and in particular subspace model identification. The goal of my research has been to extend the subspace model identification technique that was developed at the Systems and Control Engineering group in both theory and practice. During the course of the past four years, I have worked on many topics in that area. However, this thesis can contain only a few of them. Therefore only three contributions have been selected. These contributions are driven by the following questions. 1) How can we expand the class of SMI algorithms to the identification of continuous time systems? 2) How can we combine the SMI and PEM directions in system identification? 3) How can we provide the SMI tools, that were developed over the years and during the course of my research, to a broader public in an useful form. Before going into these questions in more detail, we will give a short overview of the history of system identification.

System identification is the process of finding mathematical equations that relate, in some approximate sense, measured data that is obtained from the system under observation. The traces of system identification can be dated back to Gauss' estimation of a planetary trajectory. Gauss used system identification to predict the trajectory of a newly discovered “small planet” named Ceres. During the discovery of Ceres in 1801 only nine degrees of its orbit were observed, before it disappeared behind the sun. A number of predictions were published about the possible trajectory, among them was the one from Gauss. When the “planet” was rediscovered eleven months later, it was almost exactly where Gauss had predicted it would appear. Gauss did not disclose his method at that time, but for his prediction he had used the least squares method that he had invented.

More recently, the use of the statistical setting of maximum likelihood for system parameter estimation was introduced by people such as Aström, Bohlin and Eykhoff in the 1960's[1, 2, 4, 12, 13, 27]. This allowed for a proper mathematical foundation for system identification problems. Therefore we can state that the field of system
identification has its real roots in the 1960's. The field has expanded enormously in the last 40 years. System identification has become a popular field of research and many articles have been published on the topic. This has led to rapid advances in the theoretical foundation as well as in the practical application of system identification. Many new methods have been introduced. However, this large variety of methods has made it difficult for the user to choose the best suited method for the problem at hand. Detailed knowledge is still required to use the system identification tools presently available. We will come back to this later.

Let us continue with the historical outline and describe the development over the past 40 years. In the 1960's two important pioneering contributions were made. The first contribution is the prediction-error optimization framework for system identification by Åström. The second contribution is the state space realization scheme, developed by Ho and Kalman. Since both directions play an important role in this thesis, we will briefly describe both approaches.

The system identification method introduced by Åström is based on parameter optimization. He proposed the use of the maximum likelihood framework to derive what is now known as the Prediction-Error Method (PEM). The one-step ahead prediction $\hat{y}(k)$ is the minimum variance estimate of the output $y(k)$, given the (unknown) model parameters, the present input $u(k)$ and the past values of the input $u(\kappa)$ and the output $y(\kappa)$ for $\kappa < k$. The prediction-error is the difference between the measured output and the predicted output. In the PEM framework the prediction-error is used to create a cost-function. This cost-function is then optimized with respect to the parameters of the model. The prediction-error for a given set of input and output measurements can be made smaller, by tuning the model parameters. The values of the parameters for which the cost-function is minimal are taken as the final model parameters. With the following model

$$y(k) = G(q)u(k) + H(q)e(k)$$

where $G(q)$ is the deterministic model and $H(q)$ the noise model ($q$ being the time shift operator and $H(q)$ minimum phase), the predicted output is given by [59, page 69]

$$\hat{y}(k) = H(q)^{-1}G(q)u(k) + (I - H(q)^{-1})y(k)$$

The cost-function is generally defined as the average of the two-norm of the prediction-error:

$$\frac{1}{N} \sum_{k=1}^{N} |y(k) - \hat{y}(k)|^2$$

In the 1970's the PEM approach was further extended by Ljung [59]. He defined different classes of models (ARX, OE, ARMAX, BJ, etc), which allowed different representations of the noise contribution. With some of these models the prediction-error is linear in the model parameters. This allows for a simple optimal solution
using the linear least squares method. However, for most of the cases the problem is not linear in the model-parameters and a non-linear optimization routine has to be used. The initial estimate of the parameters becomes of crucial importance to find the global optimum for this type of optimization.

In order to find an optimal model in the PEM framework, a number of choices have to be made during the identification process. These choices include the selection of the model class, the order of the different polynomials in the deterministic model and the noise model and the starting value of model parameters. Much research has focused on finding a systematic method for assisting the non-expert in making these choices[59].

The diagram in figure 1.1 shows a flow chart that can guide the identification process[59]. First of all the experiment design is important. The experimental environment, type of input, its frequency content, the sampling rate and the experiment duration need to be determined. All these can influence the outcome of a identification experiment. Secondly, the measured data needs to be pre-processed. This includes filtering, detrending, decimating, etc. The third step in the identification process is the determining of the model class. The model class is the set of systems that can be represented by a certain model structure. The choice of model class requires some knowledge of the system under observation. If the system is not within the model class, (or sufficiently close to it), the estimated model will not be useful. Different classes of models can be used, such as the OE, ARX or the BJ model. However, these are all linear models. More generally, types of non-linear models can be considered. Also the representation has to be determined. Possible choices include the state space model, a frequency response function or a transfer function model. Transfer function models are most common due to their low number of parameters. The transfer function model also allows us to restrict some of the parameters in the deterministic and noise part of the model, resulting in an ARX, OE, ARMAX or other type of model. However, this flexibility has a drawback. A difficult problem in identifying transfer function models is the choice of the model set and the orders of the polynomials. For instance, in the following Single Input Single Output (SISO) model

\[ G(q) = \frac{B(q)}{A(q)} \quad \text{and} \quad H(q) = \frac{D(q)}{C(q)} \]

the orders of four polynomials need to be chosen. In a Multi Input Multi Output (MIMO) model even more variables need to be chosen. Since the choice is not trivial, this can become a troublesome process of trial and error. One recipe for the choice of polynomials is to start with a high order ARX model to find the order of the deterministic model \( G(q) \) [59]. By examining the pole/zero cancellation in this model, an indication of the system order is obtained. Tools like Akaike's criterion can also be used for determining the order of the model.

After the model set and the order of the polynomials have been chosen, the model parameters are optimized with respect to the prediction-error cost-function (see step four in figure 1.1). When the model performs satisfactory at the validation test, the identification is successful. Otherwise we need to iterate the process. For example,
one can try a higher order model or use a different noise model.

Tools exist now that try to guide the system engineer in these steps. The matlab IDENT toolbox is probably the most familiar of these tools [59]. The toolbox has a graphical user interface and allows the user to try out multiple models and different methods. Many more tools exist such as those developed by Kollár et al.[55]. Also many commercial products are available. Those tools, however, can not compensate completely for the inherently complex nature of PEM identification.

A second contribution to system identification in the 60's was made by Kalman and Ho[42]. In 1966 they introduced their famous realization method for state space systems. This method shows how a state space model can be retrieved from the impulse response function coefficients. The method uses the geometrical properties of a system and the structure of matrices, rather than the optimization of a cost function. In 1978 Kung published his improvement of the Ho-Kalman algorithm. This improvement is based on the Singular Value Decomposition(SVD) which allows for a numerically stable solution.

The realization algorithm is based on the property that the Hankel matrix which is build from impulse response parameters is equal to the product of the extended observability and the extended controllability matrix. Kronecker had already shown that when this Hankel matrix is of finite rank, an equivalent autoregressive model
of finite order can be constructed [56]. The state space description that was introduced by Kalman, however, allows for a much more direct and numerically robust determination of the model [53]. With \( L(i) \) denoting the impulse response parameters (Markov parameters) of a system and \( A, B \) and \( C \) (\( D \) equals \( L(0) \)) the matrices of the LTI state space description of the system we have

\[
\begin{bmatrix}
L(1) & L(2) & \cdots & L(i) \\
L(2) & L(3) & \cdots & L(i + 1) \\
\vdots & \vdots & \ddots & \vdots \\
L(i) & L(i + 1) & \cdots & L(2i - 1)
\end{bmatrix}
= \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{i-1} \end{bmatrix}
\begin{bmatrix} B & AB & \cdots & A^{i-1}B \end{bmatrix}
\]

The left hand side of this equation is a Hankel matrix, composed of the impulse response parameters. This low rank Hankel matrix can be decomposed by, for instance, a SVD, into two full rank matrices. These matrices are the extended observability and controllability matrices. (The term "extended" indicates that the exponent in the last term goes beyond the system order). The state space matrices \( A \) and \( C \) can be retrieved from these two matrices using the structure of the extended observability and controllability matrices. \( B \) can similarly be obtained from the extended controllability matrix.

The approach shown by Kalman, Ho and Kung had limited practical relevance for system identification, because it only allowed for the determination of a state space model from impulse responses. But the potential of this approach was clear. Via the SVD, information about the system order was extracted from the data. Therefore less tuning parameters were necessary than in the PEM approach. Moreover, MIMO models were included in a straightforward manner. It was the start of a different system identification approach.

Despite the potential of realization methods, it took another 25 years before the next step in state space identification was made. This step was the development of Subspace Model Identification (SMI) algorithms [58, 67, 86]. These methods were closely related to Kung’s algorithm for state space realization and differed significantly from the prediction-error methods. The order selection could be performed much simpler than in the PEM framework and in many cases even fully automatic. No explicit cost-function was optimized and the solution was based on geometrical properties of signal spaces. Experimental evidence and theoretical exploration have shown that some SMI algorithms are close to optimal, in the sense that they approach the Kramer-Rao bound even though no explicit cost-function is optimized [6]. Moreover, the use of a state space description instead of transfer functions is often numerically more reliable when we deal with MIMO systems [95]. The SMI research direction resulted in a set of extremely powerful identification methods [84, 88, 92, 93].

Nowadays, system identification is used in many different areas, such as control engineering, system design, economics, biology and the analysis of physical phenomena. In all these fields different problems are solved and therefore different types of models are needed.

In control engineering, for instance, an important task is to design a model-based
controller. This controller must stabilize the plant or ameliorate its performance according to certain imposed requirements. In order to do this, a model of the plant is required. This model is typically a discrete time model, as the controller is implemented as a digital one. A noise model and information about the model uncertainty is needed in this case if one wants to design a robust controller.

In fields such as astronomy and biology the analysis of real-life phenomena is the central issue. To this aim models are constructed, either by physical insight or from measurements using system identification tools. In the latter case the identified models should give insight in the structure of the system that is studied. It can also provide values to certain parameters in the system, such as mass, delay, etc. This means that the parameters in the identified model must represent physical quantities. This can only be done using continuous time models. An example of this can be found in biomechanical engineering where researchers and physicians are interested in the structure of human joints like the knee or the ankle [54].

As the above examples indicate, one can be interested in different types of models. This means that different identification methods have to be developed. In the past, the main focus lay on the identification of discrete time linear time-invariant models. However, as explained above, the identification of continuous time models is of interest in many cases. Moreover, the problem of identifying models under special circumstances, such as the identification of stiff systems, the use of non-equidistantly sampled data or special types of non-linear models make it necessary to develop special algorithms that can deal with these cases.

The development of these identification methods is a challenge of its own. However, in order for such methods to be used in practice they need to be incorporated in an identification tool. This tool should be able to identify different model classes such as discrete time models or continuous time models, with or without noise models and uncertainty regions, stiff systems and non-equidistantly sampled data. Moreover, in many fields of application, system identification is only a part of a bigger project. The identification tools are therefore often used by a person whose prime expertise is not system identification. To help this person, the identification tools should be easy to use and the number of "tuning" parameters should be kept to a minimum. The tuning that needs to be done should be well-understandable.

As we stated at the start of this chapter, the research in this thesis is guided by three questions. The first question was how to expand the class of SMI algorithms to the identification of continuous time systems. This question stems from the need for identification algorithms that can deal with a variety of model classes. For different problems, different model classes need to be identified. The SMI methods thus far only dealt with the identification of discrete time models. However, in many circumstances continuous time models are needed or more desirable. Therefore we have developed a SMI algorithm that is able to identify a continuous time model from sampled data. For this we have extended a particular group of SMI algorithms, namely the MIMO Output-Error State Space (MOESP) family. Although there exist a number of other SMI methods such as N4SID and CVA, the focus of this thesis lies on the MOESP
algorithm, to which our research group has contributed significantly in the past. The extension of the MOESP algorithms to the identification of continuous time models is accomplished through the use of Laguerre filters. This allows us to develop a method that deals with noise in a similar way as their discrete time counterparts.

The second question is about the combination of the SMI and PEM directions in system identification. Although these two directions in system identification have led to a mature set of algorithms and methods, both methods have their limitations. On the one hand the PEM suffers from the complex choice of initial settings. On the other the SMI methods suffer from the lack of optimality. Exploration of the strong and weak points of both approaches in system identification suggests a combination of the two methods. In this thesis we propose a way to accomplish this. In a combined method, SMI allows for the detection of system orders and the identification of an accurate initial estimate. The PEM approach will then be used to find an optimal model, starting with this initial estimate provided by the SMI method. In this combined method we optimize the parameters of a state space model rather than of a transfer function model. This way we do not lose the advantage that state space models have over transfer functions when applied to MIMO systems. In order to minimize the number of parameters in the optimization procedure and increase the computational speed and robustness, the use of Separable Least Squares (SLS) is proposed. By searching a suitable parameterization it is possible to keep the number of parameters in the state space model equal to the number that would be used in a parameter optimization of a transfer function model.

The third question is how we can provide the SMI tools, that were developed over the years and the course of this research, to a broader public in an useful form. To this aim we have developed a subspace model identification toolbox. This toolbox is called the SMI-toolbox and consists of the functions that are described in this thesis. It is based on the MOESP family of algorithms. These algorithms can deal with a wide range of discrete time identification problems. In addition to the discrete time MOESP functions, the newly developed continuous time MOESP and the SLS optimization functions are added to the toolbox. Together these functions form a flexible set of system identification routines. The aim is to provide a toolbox which is easy to use and understand by a person whose prime expertise is not system identification. To guide this person, the identification tools have a consistent interface and provide additional information when possible. This helps the user in making choices during the identification process.

Summarizing, in this thesis we shall give an overview of the present state of the art of the MOESP family of SMI algorithms and further extend this family to the identification of continuous time models. We will bridge the gap between PEM and SMI methods by applying the SLS technique for the optimization of state space models. Finally we will deal with the implementation of the functions and algorithms that were described in this thesis and introduce the SMI-toolbox for the identification of state space models.

These topics are covered in this thesis as follows. In chapter 2 we present an overview
and theoretical treatment of existing members of the MOESP family of subspace identification methods. The MOESP methods can solve a large number of identification problems, both linear and non-linear, that are of practical relevance. We will explain the identification methods at the hand of the identification problem that they solve. First the state space realization problem and its solution, Kung’s algorithm, is discussed. Then the noise-free identification problem and the ordinary MOESP algorithm is treated. Next, a number of more complex identification problems and their solution using a member of the MOESP family are treated. Finally some examples are given.

In chapter 3 we derive a new subspace algorithm. This algorithm allows the identification of continuous time models from sampled data, allowing us to estimate a continuous time model where discrete time models are not possible or not desired. This is for instance the case when only non-equidistant sampled data is available, or when one is interested in the continuous time characteristics of a system, such as resonance frequencies, damping, elasticity, mass, etc.

Next, in chapter 4 we bridge the gap between PEM and SMI methods. The optimization of the state space model is discussed. Here the initial estimate obtained with MOESP is further optimized using an output-error or prediction-error criterion. The optimization method is based on the Separable Least Squares method. A recursive version of the algorithm is derived, which allows on-line tracking of changes in the system under observation.

Chapter 5 goes into detail about implementation issues and improvements of the computational speed of the algorithms. The speed and accuracy of the SMI algorithms that are described in this thesis depend highly on the way they are implemented. In this chapter we investigate a number of possible implementations. The different implementations are tested for computational complexity and speed.

Chapter 6 will give an introduction to the SMI toolbox. This toolbox consists of a library of MOESP identification functions that are written for MATLAB and SCILAB. The functions allow a structured way of identifying a dynamical system. A number of examples are given to highlight some of the features in the toolbox. The toolbox is available on Internet at http://lceww.et.tudelft.nl/~crweb/software/.

Finally chapter 7 will give the major conclusions of this thesis and discuss the major contributions made in this thesis and present suggestions for further research.
Chapter 2

Subspace Model Identification

In this chapter we give an overview of the existing MIMO Output-Error State sPace (MOESP) algorithms. These algorithms were developed over the past decade and are members of the family of Subspace Model Identification (SMI) algorithms. The advantage of the MOESP algorithms is that it can be applied to a wide variety of models in a unified manner. As with most SMI schemes only a limited number of user specified parameters need to be specified a priori. Moreover, multi input multi output (MIMO) models are treated in a straightforward manner.

This chapter gives an overview of the discrete time MOESP algorithms that can be found in literature [92, 93, 88, 22, 86, 87, 94, 90, 89, 44, 45]. Although this chapter does not give any novel results in this field, it gives an uniform presentation of the several versions of MOESP that exists. This uniform presentation is new. Moreover, the presentation of MOESP in this manner gives an overview of the underlying principles of the moesp techniques. These are used again in chapter 3, where a new MOESP algorithm is introduced.

We start the chapter with the state space realization algorithm. This algorithm solves the problem of constructing a minimal state space model from given impulse response parameters. The realization algorithm is the basis for the MOESP algorithms as it demonstrates the key step that characterizes many SMI algorithms, among which MOESP. In contrast to prediction-error identification methods, in the realization algorithm no particular parameterization of the state space model is used and no optimization of a parametric model is required. Instead, the geometric structure of the Hankel matrix, constructed from the impulse response parameters is exploited to obtain a model.

After the state space realization problem has been explained, we will introduce the MOESP family of SMI algorithms. The first algorithm of the moesp family, ONI-MOESP deals with the noise-free identification problem and the white output-error identification problem. This algorithm is very elegant, since it only requires a QR factorization
and a Singular Value Decomposition (SVD). The OM-MOESP algorithm shows the basic steps of the MOESP family of algorithms and its relation to the realization algorithm.

Other members of MOESP are available to solve more general identification problems. These are the output-error identification problem which is solved by PI-MOESP, the innovations model problem (PO-MOESP) and the errors-in-variables problem (EIV-MOESP). All these algorithms allow us to deal with a wide range of noise conditions, without introducing extra tuning parameters in the identification process. Moreover, we will show how the PO-MOESP algorithm allows us to estimate a Kalman filter. This filter can be used as an one-step ahead predictor. Finally we show how the PI-MOESP algorithm can be used to identify a special class of non-linear models, namely the Wiener models and how the EIV-MOESP algorithm allows the identification of closed loop models.

The chapter is structured as follows. In section 2.1 the state space realization algorithm is explained and Kung’s algorithm is treated. In section 2.2 the noise-free identification problem is defined. This problem is solved by the OM-MOESP algorithm, which is introduced in this section. In section 2.3 the white noise output-error identification problem is defined. This section shows that the OM-MOESP algorithm can be used for this special case of output-error identification problems. Section 2.4 defines the general output-error identification problem and introduces the PI-MOESP algorithm that solves this identification problem. Section 2.5 defines the innovation model identification problem. This identification problem is solved by the PO-MOESP algorithm that is described in this section. Section 2.6 deals with the errors-in-variable problem and EIV-MOESP. Section 2.7 describes how the closed loop identification problem can be treated with the EIV-MOESP algorithm. In section 2.8 the conditions posed on the input signal for a successful identification with the algorithms described in previous sections are described. These conditions are given by the degree of persistence of excitation. In section 2.9 the estimation of the B and D matrices and the initial state of the state space model is described. Finally in section 2.10 the identification of the non-linear Wiener system with the use of the PI-MOESP algorithm is treated.

2.1 The state space realization problem

In this section we discuss the state space realization problem. This is the problem of constructing a minimal state space model from a given set of impulse response parameters or Markov parameters. Although this is in itself an interesting problem to address, since there exist several methods to estimate impulse response functions, the realization problem can also be viewed as a very specific identification problem. When an impulse is applied to a system, the output is equal to the impulse response. Therefore the state space realization algorithm can be seen as a SMI algorithm for one specific type of input signals: the impulse response.

The state space realization algorithm was introduced by Ho and Kalman in 1966[42] and refined by Kung[57] in 1978 by using the Singular Value Decomposition. The
algorithm exploits the geometrical properties of the Hankel matrix that is constructed from Markov parameters. In this respect, the realization algorithm is similar to SMI algorithms like MOESP.

Andrei Andreyevich Markov (1856-1922) was a graduate of Saint Petersburg University (1878), where he began as a professor in 1886. Markov is particularly remembered for his study of Markov chains, sequences of random variables in which the future variable is determined by the present variable but is independent of the way in which the present state arose from its predecessors. This work launched the theory of stochastic processes. Markov's son (of the same name) followed his father in also becoming a renowned mathematician.

2.1.1 Problem definition

The state space realization problem is defined as follows:

Let the set of impulse response parameters \( L_k \in \mathbb{R}^{t \times m} \), \( k \in [0, 2i - 1] \) be given of a system described by the following state space equations:

\[
\begin{align*}
x(k + 1) &= Ax(k) + Bu(k) \quad &\text{(2.1a)} \\
y(k) &= Cx(k) + Du(k) \quad &\text{(2.1b)}
\end{align*}
\]

with \( x(k) \in \mathbb{R}^n \), \( y(k) \in \mathbb{R}^t \) and \( u(k) \in \mathbb{R}^m \). The system matrices \( A, B, C \) and \( D \) are of appropriate dimensions. The parameter \( i \) equals the number of block rows in the Hankel matrix defined from \( L_k \). This parameter should be larger than the system order \( n \). The system is assumed to be minimal, as defined in Appendix A.9 and stable.

The state space realization problem is then to determine:

1. The order \( n \) of the system.
2. The system matrices \( A_T, B_T, C_T \) and \( D_T \) where the additional index \( T \) refers to the determination of the system matrices up to a similarity transformation.

2.1.2 Kung's algorithm

The following algorithm solves the above stated realization problem. The algorithm is known as Kung's algorithm[57].
Theorem 2.1 Let $L(k) \in \mathbb{R}^{t \times m}$, $k \in [1, 2i - 1]$ be the first $2i$ Markov parameters of a finite dimensional Linear Time Invariant (LTI) system where $i$ is larger than the system order $n$. Construct the matrix

$$M_i = \begin{bmatrix}
L(1) & L(2) & \cdots & L(i) \\
L(2) & L(3) & \cdots & L(i+1) \\
\vdots & \vdots & \ddots & \vdots \\
L(i) & L(i+1) & \cdots & L(2i-1)
\end{bmatrix} \quad (2.2)$$

Then

1. The minimal order $n$ of the system is equal to $\text{rank}(M_i)$.

2. We can factorize the matrix $M_i$ into

$$M_i = \Gamma_{T_i} C_{T_i} \quad (2.3)$$

with

$$\Gamma_{T_i} = \begin{bmatrix}
C_T \\
C_T A_T \\
\vdots \\
C_T A_T^{i-1}
\end{bmatrix}$$

$$C_{T_i} = \begin{bmatrix}
B_T \\
A_T B_T \\
\vdots \\
A_T^{i-1} B_T
\end{bmatrix}$$

the extended observability matrix and extended controllability matrix of an equivalent state space description.

Remark 2.1 The structure of $M_i$ is special in the sense that it has constant (block) anti-diagonals. This structured type of matrix is called a Hankel matrix. The Hankel matrix structure will return frequently in the remaining part of this thesis.

Remark 2.2 The order of the system $n$ is not known in advance. Therefore, in practice $i$ cannot be determined. We have to determine $i$ as an upper bound of the expected system order, based on a priori knowledge of the system and/or experience.

Proof:
Using the relation between the Impulse response function and the state space matrices

$$L(i) = \begin{cases}
D & i = 0 \\
CA^{i-1}B & i \neq 0
\end{cases} \quad (2.4)$$
Hermann Hankel (1839-1873) studied with Möbius, then with Weierstrass and Kronecker and became a student of Riemann. He worked on the theory of complex numbers, the theory of functions and the history of mathematics. Hankel wrote an important work which was published in 1873 “Theorie der complexen Zahlensysteme” which did much to make Grassmann’s ideas better known. He is remembered for the Hankel transformation which occurs in the study of functions which depend only on the distance from the origin. He also studied functions, now named Hankel functions or Bessel functions of the third kind.

we can decompose $M_i$ as $\Gamma_i C_i$ with

$$\Gamma_i = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{i-1} \end{bmatrix} \quad \text{and} \quad C_i = \begin{bmatrix} B & AB & \cdots & A^{i-1}B \end{bmatrix}$$

Since the system is minimal and $i > n$ rank($C_i$) = rank($\Gamma_i^T$) = $n$ and therefore rank($M_i$) = $n$.

The factorization (2.3) can be done using an SVD of the matrix $M_i$.

$$M_i = U \Sigma V^T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}$$

with $U_1 \in \mathbb{R}^{i \times n}$, $\Sigma_1 \in \mathbb{R}^{n \times n}$, $V_1^T \in \mathbb{R}^{n \times i}$ and the other matrices of appropriate dimension. Since $M_i$ is of rank $n$, the matrix $\Sigma_2$ is zero and we have

$$M_i = U_1 \Sigma_1 V_1^T$$

We can then find the desired factorization as

$$\Gamma_{Ti} = U_1 \Sigma_1^{\frac{1}{2}}$$

$$C_{Ti} = \Sigma_1^{\frac{1}{2}} V_1^T$$

We now need to show that $\Gamma_{Ti}$ and $C_{Ti}$ are indeed the extended observability and extended controllability matrices of an equivalent system. Using the relation

$$\Gamma_i C_i = \Gamma_{Ti} C_{Ti}$$

we see that $\Gamma_{Ti}$ can be explicitly written as

$$\Gamma_{Ti} = \Gamma_i T$$
where \( T = C_i C_i^T (C_i C_i^T)^{-1} \). The matrix \( T \) is a matrix of dimension \( n \) by \( n \). Since \( \Gamma_i \) and \( \Gamma_{Ti} \) are both of rank \( n \), \( T \) must also be of rank \( n \) and thus invertible.

Using the structure of \( \Gamma_i \) we have

\[
\Gamma_{Ti} = \Gamma_i T = \begin{bmatrix}
CT \\
C A T \\
\vdots \\
C A^{i-1} T \\
\end{bmatrix} = \begin{bmatrix}
C_T \\
C_T A_T \\
\vdots \\
C_T A_T^{i-1} \\
\end{bmatrix}
\]

From this we obtain \( C_T = CT \) and \( A_T = T^{-1} A T \). This shows that \( \Gamma_{Ti} \) is indeed the extended observability matrix of an equivalent system. With the same line of reasoning we can show that \( C_T \) is the extended controllability matrix of equivalent system and \( B_T = BT^{-1} \).

Kung’s algorithm is summarized in algorithm 2.1. This algorithm shows in short how the matrices \( A_T, B_T, C_T \) and \( D_T \) can be obtained from the impulse responses \( L_k, \) \( k = 0...2i - 1 \).

## 2.2 The noise-free identification problem

In the previous section we showed that the system matrices could be retrieved from data when the input to the system is an impulse. In this case the output is the impulse response and we can apply Kung’s algorithm to find the state space matrices.

In this section we derive a SMI algorithm that identifies the state space matrices of a system from the input and output measurements using a general input. The algorithm is part of the MOESP family of subspace identification algorithms.

The identification problem that is solved in this section is the noise-free identification problem. This is still a fairly restrictive problem since in practice measurements are always contaminated with noise. However, this problem serves as a step-up to more elaborated identification problems that are treated in later sections.

### 2.2.1 Problem definition

\[
\begin{array}{c}
u(k) \rightarrow \\
A, B, C, D \rightarrow y(k)
\end{array}
\]

Figure 2.1: Schematic representation of the noise-free model.

This problem is stated with the help of figure 2.1.
Algorithm 2.1 Kung's algorithm

1. Build the matrix $M_i$ as in equation (2.2).

2. Perform the SVD

$$M_i = U \Sigma V^T$$

3. find the order of the system, $n$, from the number of nonsingular values in $\Sigma$ and partition the matrices $U$, $\Sigma$ and $V^T$ as in equation (2.5).

4. calculate

$$\Gamma_{Ti} = U_1 \Sigma_{i}^{\frac{1}{2}}$$

$$C_{Ti} = \Sigma_{i}^{\frac{1}{2}} V_1^{T}$$

5. The state space matrices can be found as follows: Construct $U_1$ from the top $(i-1)\ell$ rows of $\Gamma_{Ti}$ and $U_2$ from the lower $(i-1)\ell$ rows. The matrix $A_T$ can be solved by taking advantage of the structure of $\Gamma_{Ti}$.

$$\begin{bmatrix}
C_T \\
C_T A_T \\
\vdots \\
C_T A_T^{i-2} T
\end{bmatrix} A_T = \begin{bmatrix}
C_T A_T \\
C_T A_T^2 \\
\vdots \\
C_T A_T^{i-1} T
\end{bmatrix}$$

The other state space matrices can be found as follows

$$B_T = \text{the first } m \text{ columns of } C_T,$$

$$C_T = \text{the upper } \ell \text{ rows of } \Gamma_{Ti},$$

$$D_T = L(0)$$
Let the input/output data-set \( \{ u(k), y(k) \}, \ k \in [0, N_m - 1] \) with \( u(k) \in \mathbb{R}^m \) and \( y(k) \in \mathbb{R}^f \), be given of a system described by the following state space equations:

\[
\begin{align*}
x(k + 1) &= Ax(k) + Bu(k) \quad (2.5a) \\
y(k) &= Cx(k) + Du(k) \quad (2.5b)
\end{align*}
\]

with \( x(k) \in \mathbb{R}^n \) and the system matrices \( A, B, C \) and \( D \) of appropriate dimensions. Let the system be minimal and stable and the number of samples much larger than the order of the system \( (N_m \gg n) \).

Let the input be sufficiently persistently exciting. The precise definition of Persistence of excitation is treated in section 2.8. For now we suffice with the fact that the input should be fluctuating enough to be able to recover the system dynamics.

Then the noise-free identification problem is to determine:

1. The order \( n \) of the system.
2. The matrices \( A_T, B_T, C_T \) and \( D_T \), where the additional index \( T \) refers to the determination of the system matrices up to the freedom in similarity transformation.

### 2.2.2 The ordinary MOESP method

The noise-free identification problem is solved by the ordinary MOESP algorithm. We will denote this algorithm with OM-MOFSP. It was first described in 1992[92]. As we will see, this algorithm is closely related to Kung’s realization algorithm that was treated in the previous section.

Given \( N_m \) measurements of the input, \( u(k) \in \mathbb{R}^m, \ k \in [0, N_m - 1] \), the output of the system can be written as follows:

\[
\begin{align*}
y(0) &= Cx(0) + Du(0) \\
y(1) &= Cx(1) + Du(1) \\
&= CAx(0) + CBu(0) + Du(1) \\
y(2) &= CA^2x(0) + CABu(0) + CBu(1) + Du(2) \\
&\vdots \\
y(k) &= CA^kx(0) + CA^{k-1}Bu(0) + \cdots + CBu(k-1) + Du(k)
\end{align*}
\]
with $N = N_m - i + 1$ this can be cast into the following matrix form:

$$
\begin{bmatrix}
  y(0) & y(1) & \cdots & y(N - 1) \\
  y(1) & y(2) & \cdots & y(N) \\
  \vdots & \vdots & \ddots & \vdots \\
  y(i - 1) & y(i) & \cdots & y(N + i - 2)
\end{bmatrix}
= \begin{bmatrix}
  C \\
  CA \\
  \vdots \\
  CA^{i-1}
\end{bmatrix}
\begin{bmatrix}
  x(0) \\
  x(1) \\
  \vdots \\
  x(N - 1)
\end{bmatrix}
+ \begin{bmatrix}
  D & 0 & \cdots & 0 \\
  CB & D & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  CA^{i-2}B & CA^{i-3}B & \cdots & D
\end{bmatrix}
\begin{bmatrix}
  u(0) \\
  u(1) \\
  \vdots \\
  u(i - 1)
\end{bmatrix}
\begin{bmatrix}
  u(1) \\
  u(2) \\
  \vdots \\
  u(i)
\end{bmatrix}
\begin{bmatrix}
  u(N - 1) \\
  u(N) \\
  \vdots \\
  u(N + i - 2)
\end{bmatrix}
$$

(2.6)

The above equation is called the data equation. This equation relates all measured data in one big equation. Since the number of measurements $N_m$ is generally much larger than the value of $i$, the data matrices containing the input and output are very flat matrices, (i.e. more columns than rows). The input and output matrices have constant block-anti-diagonal shape. This is the Hankel structure which we already encountered in the previous section. We will use the following shorthand notation for a Hankel matrix constructed from $y(k)$.

$$
Y_{i,j,N} = \begin{bmatrix}
  y(i) & y(i + 1) & \cdots & y(i + N - 1) \\
  y(i + 1) & y(i + 2) & \cdots & y(i + N) \\
  \vdots & \vdots & \ddots & \vdots \\
  y(i + j - 1) & y(i + j) & \cdots & y(i + j + N - 2)
\end{bmatrix}
$$

(2.7)

The three subscripts of $Y_{i,j,N}$ denote respectively the index in the left upper entry of the Hankel matrix, the number of (block) rows and the number of (block) columns. The use of three subscripts may seem a bit overdone in this case, but it allows us to be consistent in our notation over all the algorithms that are being described in this thesis.

For a row-vector constructed from $x(k)$, we will use a two-subscript notation, where the first subscript denotes the starting index and the second subscript the length of the vector.

$$
X_{i,N} = \begin{bmatrix}
  x(i) \\
  x(i + 1) \\
  \vdots \\
  x(i + N - 1)
\end{bmatrix}
$$

(2.8)

Furthermore we define

$$
\Gamma_i = \begin{bmatrix}
  C \\
  CA \\
  \vdots \\
  CA^{i-1}
\end{bmatrix}
\quad \text{and} \quad
H_i = \begin{bmatrix}
  D & 0 & \cdots & 0 \\
  CB & D & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  CA^{i-2}B & CA^{i-3}B & \cdots & D
\end{bmatrix}
$$

With these matrices we write the data equation (2.6) in condensed form as

$$
Y_{0,i,N} = \Gamma_i X_{0,N} + H_i U_{0,i,N}
$$

(2.9)
Our goal now is to find the state space matrices $A$ and $C$ (up to a similarity transformation). The MOESP algorithm does this in the following way. First we try to estimate the matrix $\Gamma_i$. From this matrix we then find estimates of the matrices $A$ and $C$.

To find $\Gamma_i$, MOESP performs two steps. First we try to remove the term with the input from equation (2.9). Then in the second step we find the column-space of the remaining part, which will give us the estimate of $\Gamma_i$.

An intuitive reason for these two steps, is the following: The output of the system is caused by two factors. These are the present input and the past input. The past input is cumulated in the memory or state of the system. The present input and the state both span a different vector space. The way in which these two spaces together form the output vector space is characteristic for the underlying system. If we can find this mapping, we can reconstruct the system. MOESP first tries to find the mapping from the state to the output. For this cause we first eliminate the part of the output space that is caused by the input. From the remaining part of the output space we then estimate the desired mapping.

In the first step we remove the term $H_iU_{0,i,N}$ in the data equation (2.6). The column-space of a matrix is not altered by right multiplication with a second matrix. Therefore we can eliminate $H_iU_{0,i,N}$ by right multiplication with an appropriate matrix. The matrix that has this property is the projection on the orthogonal complement of $U_{0,i,N}$, given by

$$\Pi_{U_{0,i,N}}^\perp = I - U_{0,i,N}^T(U_{0,i,N}U_{0,i,N}^T)^{-1}U_{0,i,N}$$

It can be easily seen from this expression that indeed $U_{0,i,N}\Pi_{U_{0,i,N}}^\perp = 0$ by substitution of $\Pi_{U_{0,i,N}}^\perp$.

To assure that $\Pi_{U_{0,i,N}}^\perp$ exists we have to introduce a requirement on $u(k)$, namely that $U_{0,i,N}U_{0,i,N}^T$ has full rank. Only then can we guaranty that $(U_{0,i,N}U_{0,i,N}^T)^{-1}$ exists. More generally we need to pose certain requirements on the input, in order to successfully identify the system. To this aim we introduce the notion of Persistence of Excitation (PE). It shall be obvious that not every input is suited for an identification experiment. A simple counter example is a zero input. This input does not excite the system enough to be able to identify the system uniquely. In this section we give a formal definition of persistence of excitation. In section 2.8 we will examine the MOESP algorithms for the exact requirements on the input signal to guaranty consistency.

**Definition 2.1** The signal $u(k)$ is called persistently exciting of order $i$ if the following holds

$$\mathbb{E}\{U_{0,i,N}U_{0,i,N}^T\} > 0$$

where $U_{0,i,N}$ is the Hankel matrix constructed of $u(k)$ according to equation (2.7).
2.2 The noise-free identification problem

After multiplying equation (2.9) with $\Pi_{U_{0,i,N}}^\dagger$ we obtain

$$Y_{0,i,N}\Pi_{U_{0,i,N}}^\dagger = \Gamma_i X_{0,N}\Pi_{U_{0,i,N}}^\dagger$$

(2.10)

From which we can recover the column-space to obtain $\Gamma_i$. The multiplication with $\Pi_{U_{0,i,N}}^\dagger$ is the projection onto the orthogonal complement of $U_{0,i,N}$. As shown in appendix A.2, this can be done in a numerically reliable way by performing the following LQ factorization.

$$
\begin{bmatrix}
U_{0,i,N} \\
Y_{0,i,N}
\end{bmatrix} = 
\begin{bmatrix}
L_{11} & 0 \\
L_{21} & L_{22}
\end{bmatrix}
\begin{bmatrix}
Q_1 \\
Q_2
\end{bmatrix}
$$

(2.11)

where $U_{0,i,N} \in \mathbb{R}^{m \times N}$ and $Y_{0,i,N} \in \mathbb{R}^{i \times N}$ $Q_1$ is of the same size as $U_{0,i,N}$ and $Q_2$ of the same size as $Y_{0,i,N}$. The blocks in the lower triangular matrix are of the following dimensions: $L_{11} \in \mathbb{R}^{m \times m}$, $L_{21} \in \mathbb{R}^{i \times m}$ and $L_{22} \in \mathbb{R}^{i \times i}$.

**Theorem 2.2 (om-moesp)** Given the state space model (2.5a) - (2.5b), with $u(k)$ PE of sufficient order. Construct the Hankel matrices $U_{0,i,N}$ and $Y_{0,i,N}$ according to equation (2.7). Then with the LQ factorization (2.11) the following holds:

$$L_{22} = \Gamma_i X_{0,N}Q_2^T$$

(2.12)

**Proof:**

From the LQ factorization, we have

$$Y_{0,i,N}Q_2^T = L_{21}Q_1Q_2^T + L_{22}Q_2Q_2^T = L_{22}$$

Using the data equation (2.9) we get

$$Y_{0,i,N}Q_2^T = \Gamma_i X_{0,N}Q_2^T + H_i U_{0,i,N}Q_2^T = \Gamma_i X_{0,N}Q_2^T + H_i L_{11}Q_1Q_2^T = \Gamma_i X_{0,N}Q_2^T$$

which completes the proof.

In the second step we find the column-space of $L_{22}Q_2^T$. Since this column-space is not changed by $Q_2^T$ we do not need to calculate this matrix. We only need to calculate $L_{22}$. To find the desired column-space we perform the following SVD:

$$L_{22} = U \Sigma V^T$$

(2.13)
The number of non-zero singular values gives the order of the system, \( n \). The column-space of \( L_{22} \) is equal to the first \( n \) columns of \( U \), which we denote by \( U_n \). \( A_T \) and \( C_T \) can now be found in the same way as in Kung's algorithm: Construct \( U_1 \) to be the upper \( (i - 1)\ell \) rows of \( U_n \) and \( U_2 \) the lower \( (i - 1)\ell \) rows. Then
\[
\begin{align*}
C_T &= \text{the upper } \ell \text{ rows of } U_n \\
A_T &= U_1^\dagger U_2
\end{align*}
\]

where \((\cdot)^\dagger\) denotes the pseudo inverse. Unfortunately \( B_T \) and \( D_T \) can not be found directly anymore, as in Kung's algorithm. This is due to the influence of the input term in equation (2.9). In section 2.9 the estimation of these matrices is discussed in detail. Algorithm 2.2 gives an overview of OM-MOESP.

### 2.3 The white output-error identification problem

The OM-MOESP algorithm solves the noise-free identification problem. There is however one noisy identification problem which can also be solved with ordinary MOESP: The white output-error problem. In this section we prove that the Ordinary MOESP method gives consistent estimates of the state space matrices \( A \) and \( C \), in the case where the output is perturbed with a white measurement noise.

#### 2.3.1 Problem definition

The white output-error problem is stated as follows

Let the input/output data-set \( \{u(k), y(k)\} \), \( k \in [0, N_m - 1] \) with \( u(k) \in \mathbb{R}^m \) and \( y(k) \in \mathbb{R}^\ell \), be given of a system described by the state space model:

\[
\begin{align*}
x(k + 1) &= Ax(k) + Bu(k) \\
y(k) &=Cx(k) + Du(k) \\
\tilde{y}(k) &= y(k) + v(k)
\end{align*}
\]

with \( x(k) \in \mathbb{R}^n \). Let \( v(k) \) be an additive white zero mean perturbation with variance \( \sigma \) which is independent of the input. Let \( u(k) \) be PE of sufficient order. Then the white output-error identification problem is to consistently estimate:

1. The order \( n \) of the system.
2. The matrices \( A_T, B_T, C_T \) and \( D_T \), where the additional index \( T \) refers to the determination of the system matrices up to the freedom in similarity transformation.
2.3.2 Ordinary MOESP and the white output-error problem

In this section we prove that the ordinary MOESP scheme gives consistent estimates of the matrices $A$ and $C$ for the white output error identification problem.

**Theorem 2.3** Given the state space model (2.14a) - (2.14c), with $u(k)$ PE of sufficient order. Construct the Hankel matrices $U_{0,i,N}$ and $Y_{0,i,N}$ according to equation (2.7). Then with the LQ factorization

$$
\begin{bmatrix}
U_{0,i,N} \\
\tilde{Y}_{0,i,N}
\end{bmatrix} =
\begin{bmatrix}
L_{11} & 0 \\
L_{21} & \tilde{L}_{22}
\end{bmatrix}
\begin{bmatrix}
Q_1 \\
Q_2
\end{bmatrix}
$$

(2.15)

the following holds:

$$
\lim_{N \to \infty} \frac{1}{\sqrt{N}} \tilde{L}_{22} = \lim_{N \to \infty} \frac{1}{\sqrt{N}} Y_{0,i,N} Q_2^T X_{0,i,N} Q_2^T
$$

(2.16)

**Proof:**

The key in the proof of this theorem is the fact that when $N \to \infty$ the disturbance $v(k)$ does not influence the matrix $U_n$ of the SVD. Define the perturbed output Hankel matrix

$$
\tilde{Y}_{0,i,N} = Y_{0,i,N} + V_{0,i,N}
$$

where $V_{0,i,N}$ is a Hankel matrix constructed from $v(k)$ and $N - N_m - i + 1$. Consider the perturbed LQ factorization 2.15. The matrix $\tilde{L}_{22}$ is explicitly given by

$$
\tilde{L}_{22} = Y_{0,i,N} \Pi_{U_{0,i,N}} \tilde{Q}_2^T
$$

Then

$$
\tilde{L}_{22} \tilde{L}_{22}^T = \tilde{Y}_{0,i,N} \Pi_{U_{0,i,N}} \Pi_{U_{0,i,N}}^T \tilde{Y}_{0,i,N}^T
$$

$$
= \tilde{Y}_{0,i,N} \Pi_{U_{0,i,N}} \tilde{Y}_{0,i,N}^T
$$

$$
= Y_{0,i,N} \Pi_{U_{0,i,N}} Y_{0,i,N}^T + Y_{0,i,N} \Pi_{U_{0,i,N}} V_{0,i,N}^T
$$

$$
+ V_{0,i,N} \Pi_{U_{0,i,N}} V_{0,i,N}^T
$$

(2.17)

Here we used that $\Pi_{U_{0,i,N}}$ equals $\Pi_{U_{0,i,N}}$, which is a property of projection matrices. The expectation of $U_{0,i,N} U_{0,i,N}^T$ is $\mathcal{O}(N)$. Since $Q_1$ has orthogonal rows, the magnitude of $L_{11}$ is $\mathcal{O}(\sqrt{N})$ and the matrix $\lim_{N \to \infty} L_{11}$ is infinite. The matrix $\lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{11}$, however, is bounded [92] and when $u(k)$ satisfies the PE condition it is even invertible. Therefore we can divide equation (2.17) by $\sqrt{N}$ to obtain

$$
\lim_{N \to \infty} \frac{1}{N} \tilde{L}_{22} \tilde{L}_{22}^T
$$

$$
= \lim_{N \to \infty} \frac{1}{N} \left( Y_{0,i,N} \Pi_{U_{0,i,N}} Y_{0,i,N}^T + Y_{0,i,N} \Pi_{U_{0,i,N}} V_{0,i,N}^T + V_{0,i,N} \Pi_{U_{0,i,N}} V_{0,i,N}^T \right)
$$

(2.18)
When $N$ goes to infinity the crossproducts between $Y_{0,i,N}$ and $Y_{0,i,N}$ are zero, since $y(k)$ is uncorrelated with $v(k)$. Moreover the last term becomes $\sigma I$ because of the white noise property of $v(k)$ and because it is uncorrelated with $u(k)$. We thus obtain

$$\lim_{N \to \infty} \frac{1}{N} \hat{L}_{22} \hat{L}_{22}^T = \lim_{N \to \infty} \frac{1}{N} Y_{0,i,N} \Pi_{U_{0,i,N}}^\perp Y_{0,i,N}^T + \sigma I$$

Since $Y_{0,i,N} \Pi_{U_{0,i,N}}^\perp$ equals $L_{22}Q_2$ we have

$$\lim_{N \to \infty} \frac{1}{N} \hat{L}_{22} \hat{L}_{22}^T = \lim_{N \to \infty} \frac{1}{N} L_{22} L_{22}^T + \sigma I$$

$$= \lim_{N \to \infty} \frac{1}{N} U \Sigma U^T + \sigma I$$

$$= \lim_{N \to \infty} \frac{1}{N} U (\Sigma + \sigma I) U^T$$

This shows that a perturbation of $y(k)$ in the limit $N \to \infty$ does not affect the matrix $U$, but only the singular values. Therefore the system matrices $A$ and $C$ are estimated consistently using the ordinary MOESP method.

Algorithm 2.2 OM-MOESP

1. Build the matrices $U_{0,i,N}$ and $Y_{0,i,N}$ according to equation (2.7), with $N = N_m - i + 1$.

2. Perform the LQ factorization equation (2.11).

3. Compute the SVD of the matrix $L_{22}$:

$$L_{22} = U \Sigma V^T$$

We can now find the order $n$ of the system as the number of non-zero singular values.

4. From the first $n$ left singular vectors of $U$, denoted by $U_n$ we can estimate the matrices $A_T$ and $C_T$. Take $U_1$ as the upper $(i-1)\ell$ rows of $U_n$ and $U_2$ the lower $(i-1)\ell$ rows of $U_n$. Compute $\hat{A}_T$ and $\hat{C}_T$ as follows

$$\hat{C}_T = \text{the upper } \ell \text{ rows of } U_n$$

$$\hat{A}_T = U_1^\dagger U_2$$
Example 2.1 We consider the system

\[
A = \begin{bmatrix}
0.1 & 0 \\
0 & 0.9 \\
\end{bmatrix}, \quad B = \begin{bmatrix}
1 \\
-0.5 \\
\end{bmatrix}, \quad C = \begin{bmatrix}
1 & 0 \\
0 & 1 \\
\end{bmatrix}, \quad D = \begin{bmatrix}
2 \\
-2 \\
\end{bmatrix}, \quad x(0) = \begin{bmatrix}
0 \\
0 \\
\end{bmatrix}
\]

The input is given by a random sequence of 11 samples: \( u(0) = -0.3017 \), \( u(1) = 0.9570 \), etc. \( U_{0.3,9} \) is given by the following Hankel matrix

\[
\begin{bmatrix}
-0.3017 & 0.9570 & -0.5334 & -0.9011 & -0.8926 & 0.2787 & -0.7458 & 1.6035 & 0.5743 \\
0.9570 & -0.5334 & -0.9011 & -0.8926 & 0.2787 & -0.7458 & 1.6035 & 0.5743 & 0.3207 \\
-0.5334 & -0.9011 & -0.8926 & 0.2787 & -0.7458 & 1.6035 & 0.5743 & 0.3207 & -0.1514
\end{bmatrix}
\]

\( Y_{0.3,9} \) is constructed similarly but because we have two outputs, it is a bigger (6 by 9) matrix. We perform the LQ factorization of \( \begin{bmatrix} U_{0.3,9} \\ Y_{0.3,9} \end{bmatrix} \) and construct \( L_{22} \) from the lower right 6 by 6 part of \( L \).

\[
L_{22} = \begin{bmatrix}
-2.4286 & 0 & 0 & 0 & 0 \\
0.7761 & -0.8558 & 0 & 0 & 0 \\
0.2429 & 0 & 0 & 0 & 0 \\
0.6987 & -0.7702 & 0 & 0 & 0 \\
0.0243 & 0 & 0 & 0 & 0 \\
0.6289 & -0.6932 & 0 & 0 & 0
\end{bmatrix}
\]

Here we already see the low rank of \( L_{22} \).

The singular value decomposition \( L_{22} = U \Sigma V^T \) gives the following singular values: \( \begin{bmatrix} 2.8082 & 1.1682 & 0 & 0 & 0 & 0 \end{bmatrix} \). We therefore build \( U_n \) from the first two columns of \( U \).

\[
U_n = \begin{bmatrix}
-0.8350 & 0.5410 \\
0.3463 & 0.5344 \\
-0.0835 & 0.0541 \\
0.3116 & 0.4810 \\
-0.0084 & 0.0054 \\
0.2805 & 0.4329
\end{bmatrix}
\]

From this we find

\[
\hat{A}_T = \begin{bmatrix}
0.3365 & 0.3651 \\
0.3651 & 0.6635
\end{bmatrix}, \quad \hat{C}_T = \begin{bmatrix}
-0.8350 & 0.5410 \\
0.3463 & 0.5344
\end{bmatrix}
\]

Of course we don’t find back the original matrices \( A \) and \( C \). This would be a tremendous coincidence. However, \( \hat{A}_T \) and \( \hat{C}_T \) represent the same system. An invariant which we can check is the set of eigenvalues of \( \hat{A}_T \). These should be the same as the eigenvalues of \( A \). Indeed, both \( A \) and \( \hat{A}_T \) have eigenvalues 0.1 and 0.9.

You can try the above example yourself in MATLAB with the following commands
% Construct system
A=[0.1 0; 0 0.9]
B=[1; -0.5]
C=eye(2)
D=[2; -2]
% generate data
u=randn(11,1)
y=dlsim(A,B,C,D,u)
% identify model from data
U=[u(1:9),u(2:10),u(3:11)]
Y=[y(1:9,:),y(2:10,:),y(3:11,:)]
[Q,R]=qr([U3,Y3]);L=R'
L22=L(4:9,4:9)
[U,Sigma,V]=svd(L22)
Un=U(:,1:2)
AT=Un(1:4,:)
\Un(3:6,:)
CT=Un(1:2,:)
eig(AT),eig(A)

2.4 The output-error identification problem.

In the previous section we assumed a simple noise model. The OM-MOESP algorithm can only deal with white output noise. In practice however, measurements are normally always polluted by non-white noise from different sources. In order to find an accurate model of the system, we need to take these noise sources into account. In the following sections we will discuss three subspace identification methods that deal with more elaborate noise models.

The first problem that we discuss is the output-error problem. This is the case where only the output is disturbed by a colored noise signal. The second problem is the case where the output is contaminated with white measurement noise and the state equation has an additive white process noise term. This is called the innovations model problem. The third problem is the case where on top of the process and output measurement noise, also the input is perturbed by a measurement noise. This problem is called the errors-in-variables problem. This problem includes the closed loop identification problem.

In this thesis we will use an ergodic-algebraic framework[92]. That is, when u and v are ergodic processes and u(1),...,u(N) and v(1),...,v(N) realizations of these processes,
for $N \to \infty$ the following expression holds:

$$\mathbb{E}\{u(k)v(k-l)\} = \lim_{N \to \infty} \frac{1}{N} \sum_{k=0}^{N-1} u(k)v(k-l)$$

We will assume all signals in the identification problems to be ergodic processes.

### 2.4.1 Problem definition

We will start with the output-error identification problem. This problem is of interest when one is mainly searching for the deterministic characteristics of the underlying system. Applications can be found in biomedical modeling, see e.g. [54] where analysis of the system is the main goal.

The fact that we can assume the input to be unperturbed is because it is often generated for the sole purpose of identifying the system. The input can be, for instance, a recorded sequence or a signal that is generated digitally. In these circumstances the input is undisturbed. The output is always measured from the system. This means noise will contaminate the output. Also discrepancies between the model and the system will lead to output-errors, that can be regarded as noise from a system identification point of view.

![Figure 2.2: Schematic representation of the output-error model.](image)

The output-error model is depicted in figure 2.2. The output-error identification problem is the following:

Let the input/output data-set $\{u(k), y(k)\}$, $k \in [0, N_m - 1]$ with $u(k) \in \mathbb{R}^m$ and $y(k) \in \mathbb{R}^l$, be given of a system described by the state space model:

$$
\begin{align*}
x(k+1) &= Ax(k) + Bu(k) \\
y(k) &= Cx(k) + Du(k) + v(k)
\end{align*}
$$

with $x(k) \in \mathbb{R}^n$. Let $v(k)$ be an additive zero mean perturbation (stochastic or deterministic) which is uncorrelated with the input, such that:

$$
\lim_{N_m \to \infty} \frac{1}{N_m} \sum_{k=0}^{N_m-1} u(k)v(k+l)^T = 0 \quad \forall \quad l
$$
Finally, let the system be minimal and stable. Then the problem is to consistently estimate:

1. The order $n$ of the system.
2. The matrices $A_T$, $B_T$, $C_T$ and $D_T$, where the additional index $T$ refers to the determination of the state space matrices up to a similarity transformation.

### 2.4.2 The past input instrumental variable method

We will now introduce the past input instrumental variable MOESP algorithm. This algorithm is able to solve the output-error identification problem. The method is a member of the MOESP family of subspace algorithms and is denoted by PI-MOESP. This algorithm was first introduced in [88]. To deal with the noise, PI-MOESP introduces an instrumental variable that removes the influence of the noise on the output.

The data equation corresponding to output-error model description (2.19a), (2.19b) is given by

$$Y_{i,j,N} = \Gamma_j X_{i,N} + H_j U_{i,j,N} + V_{i,j,N}$$  \hspace{1cm} (2.20)

$V_{i,j,N}$ is the noise Hankel matrix, constructed from $v(k)$ according to equation (2.7) and $N = N_m - i - j + 1$.

As we saw in the previous section, the goal of MOESP is to find the column-space which is spanned by the extended observability matrix $\Gamma_j$. The first step towards finding this column-space is the same as with OM-MOESP. We multiply the data equation with $\Pi_{U_{i,j,N}}^{\perp}$. This removes the term $H_j U_{i,j,N}$. But it does not remove the noise term $V_{i,j,N}$ from the data equation. Therefore we need a second trick to remove this term. We will try to find a matrix $Z$, that is correlated with the state, but uncorrelated with the noise. We want to project the data equation onto this $Z$. The matrix should have the following two properties. First it should be uncorrelated with the noise. Secondly it should be correlated with the state, such that the extended observability matrix can still be recovered. These two requirements for the instrumental variable are covered by:

$$\lim_{N \to \infty} \frac{1}{N} V_{i,j,N} \Pi_Z = 0$$  \hspace{1cm} (2.21)

$$\text{rank} \left( \lim_{N \to \infty} \frac{1}{N} X_{i,N} \Pi_Z \right) = n$$  \hspace{1cm} (2.22)

The matrix $Z$ is called the instrumental variable. It is used as a tool (instrument) to remove the noise term. In the case of the output error model a good choice of $Z$ is a matrix that is constructed from past input values. Because the noise is uncorrelated with the input, this choice of instrument will clearly conform to equation (2.21). We
2.4 The output-error identification problem.

can, under PE conditions on the input guaranty that equation (2.22) holds, since the state is dependent on the past inputs.

We define two types of Hankel matrices. The past Hankel matrix of the input is $U_{0,i,N}$ and the future Hankel matrix is $U_{i,j,N}$. For the output and the noise we have similar past and future matrices: $Y_{0,i,N}, Y_{i,j,N}, V_{0,i,N}$ and $V_{i,j,N}$. The values of the indices $i$ and $j$ have to be specified by the user. Often they are chosen equal and one or two times the expected order of the system.

Note that, although the data from which the past and future Hankel matrices are built do overlap, the data in corresponding columns do not. This way, the product $U_{0,i,N}U_{i,j,N}^T$ only consists of multiplications between past and future values of the data. This is the reason for denoting the matrices as past and future.

We will use the future Hankel matrices for the data equation and the past input Hankel matrix for the instrumental variable. The instrumental variable of pi-moesp is therefore given by

$$Z = U_{0,i,N}$$

(2.23)

We right multiply the data equation with $\Pi^T_{U_{i,j,N}}$ and $\Pi_Z$. When the number of samples goes to infinity we obtain:

$$\lim_{N \to \infty} \frac{1}{N} Y_{i,j,N} \Pi^T_{U_{i,j,N}} \Pi_{U_{0,i,N}} = \lim_{N \to \infty} \frac{1}{N} \Gamma_j X_{i,N} \Pi_{U_{i,j,N}} \Pi_{U_{0,i,N}}$$

(2.24)

From this matrix we can estimate the column-space to retrieve $\Gamma_j$.

The right-multiplication with $\Pi^T_{U_{i,j,N}}$ and $\Pi_Z$ can be efficiently calculated by using a single LQ factorization. For this, we build three Hankel matrices: $U_{i,j,N}$ and $Y_{i,j,N}$ and $U_{0,i,N}$. We construct the data matrix by stacking the Hankel matrices of the input and the output, as with the ordinary moesp. But this time we stack the instrumental variable in between.

$$
\begin{bmatrix}
U_{i,j,N} \\
Z \\
Y_{i,j,N}
\end{bmatrix} = 
\begin{bmatrix}
U_{i,j,N} \\
U_{0,i,N} \\
Y_{i,j,N}
\end{bmatrix} = 
\begin{bmatrix}
L_{11} & 0 & 0 \\
L_{21} & L_{22} & 0 \\
L_{31} & L_{32} & L_{33}
\end{bmatrix} 
\begin{bmatrix}
Q_1 \\
Q_2 \\
Q_3
\end{bmatrix}
$$

(2.25)

**Theorem 2.4 (pi-moesp)** Given the state space system (2.19a) - (2.19b), with $v(k)$ uncorrelated with the input $u(k)$ and the initial state $x(0)$. Let $u(k)$ be PE of sufficient order. Construct the Hankel matrices $U_{0,i,N}, U_{i,j,N}$ and $Y_{i,j,N}$ according to equation (2.7). Then with the LQ factorization (2.25) the following holds:

$$\lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{32} = \lim_{N \to \infty} \frac{1}{\sqrt{N}} \Gamma_j X_{i,N} Q_2^T$$
Proof:

From the LQ factorization we have (after dividing by $\sqrt{N}$):

$$\lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{32} = \lim_{N \to \infty} \frac{1}{\sqrt{N}} Y_{i,j,N} Q_2^T$$  \hspace{1cm} (2.26)

From the data equation (2.20) we have:

$$\lim_{N \to \infty} \frac{1}{\sqrt{N}} Y_{i,j,N} Q_2^T =$$

$$\lim_{N \to \infty} \frac{1}{\sqrt{N}} \Gamma_i X_{i,N} Q_2^T + \lim_{N \to \infty} \frac{1}{\sqrt{N}} H_j U_{i,j,N} Q_2^T + \lim_{N \to \infty} \frac{1}{\sqrt{N}} V_{i,j,N} Q_2^T$$  \hspace{1cm} (2.27)

Because $\lim_{N \to \infty} \frac{1}{\sqrt{N}} U_{i,j,N} = \lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{11} Q_1$ and $Q_1 Q_2^T = 0$,
we find that $\lim_{N \to \infty} \frac{1}{\sqrt{N}} H_j U_{i,j,N} Q_2^T$ is zero.

Since $u(k)$ is independent of $v(k)$ we have

$$\lim_{N \to \infty} \frac{1}{N} U_{i,j,N} V_{i,j,N}^T = \lim_{N \to \infty} \frac{1}{N} L_{11} Q_1 V_{i,j,N}^T = 0$$

Multiplying with the inverse of $\lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{11}$ gives

$$\lim_{N \to \infty} \frac{1}{\sqrt{N}} Q_1 V_{i,j,N}^T = 0$$  \hspace{1cm} (2.28)

Because past values of $u(k)$ are independent of future values of the noise $v(k)$ we have

$$\lim_{N \to \infty} \frac{1}{N} Z V_{i,j,N}^T = 0$$

$$\lim_{N \to \infty} \frac{1}{N} \left( L_{21} Q_1 + L_{22} Q_2 \right) V_{i,j,N}^T = 0$$

$$\lim_{N \to \infty} \frac{1}{N} L_{22} Q_2 V_{i,j,N}^T = 0$$

where we have used equation (2.28) to get the last result. Because of the persistence of excitation of $u(k)$ also $\frac{1}{\sqrt{N}} L_{22}$ is invertible. This results in

$$\lim_{N \to \infty} \frac{1}{\sqrt{N}} Q_2 V_{i,j,N}^T = 0$$  \hspace{1cm} (2.29)

Combining equations (2.27) and (2.29) finally leads to

$$\lim_{N \to \infty} \frac{1}{\sqrt{N}} Y_{i,j,N} Q_2^T = \lim_{N \to \infty} \frac{1}{\sqrt{N}} \Gamma_i X_{i,N} Q_2^T$$  \hspace{1cm} (2.30)

which together with equation (2.26) concludes the proof.
Algorithm 2.3 PI-MOESP

1. Build the matrices $U_{i,j,N}$ and $Y_{i,j,N}$ and $U_{0,i,N}$ according to equation (2.7) with $N = N_m - i - j + 1$.

2. Perform the LQ factorization equation (2.25).

3. Compute the SVD of the matrix $L_{32}$:

   \[ L_{32} = U \Sigma V^T \]

By inspecting the singular values (s.v.'s) in $\Sigma$ we can now estimate the order $n$ of the system. In the noise-free case, only $n$ singular values are non-zero. In the output-error case this does not hold anymore. But depending on the signal-to-noise ration, we can still distinguish the system related s.v.'s. We can group the singular values in a set of large s.v.'s that belong to system modes and a set of small singular values, that belong to noise contribution.

4. After selecting the order of the system we take the first $n$ columns of $U$ denoted by $U_n$. This is the estimate of $\Gamma_i$ from which we can estimate the matrices $\hat{A}_T$ and $\hat{C}_T$. Take $U_1$ as the upper $(i - 1)\ell$ rows of $U_n$ and $U_2$ the lower $(i - 1)\ell$ rows of $U_n$. Compute $\hat{A}_T$ and $\hat{C}_T$ as follows

   \[
   \hat{C}_T = \text{the upper } \ell \text{ rows of } U_n
   \]
   \[
   \hat{A}_T = U_1^\dagger U_2
   \]
2.5 The innovations model identification problem

The second identification method we treat is the innovations model identification problem. In this model we assume the output to be perturbed by two separate noise signals. The first is a direct perturbation by a white noise signal, the measurement noise. The second is a white noise signal that enters into the process and goes through the system dynamics. This is called process noise.

The innovation model identification problem is of interest when one wants to jointly model the deterministic and stochastic part of the system and is therefore of importance in control and prediction. With a model of the stochastic part of the system, a robust controller can be implemented or an one-step ahead predictor can be constructed.

2.5.1 Problem definition

\[ \begin{array}{c}
  w(k) \\
  v(k)
\end{array} \xrightarrow{+} \begin{array}{c}
  u(k) \\
  A, B, C, D
\end{array} \xrightarrow{+} y(k) \]

Figure 2.3: Schematic representation of the innovations model.

The innovations model identification problem is stated with the help of figure 2.3.

Let the input/output data-set \( \{u(k), y(k)\}, k \in [0, N_m - 1] \) with \( u(k) \in \mathbb{R}^m \) and \( y(k) \in \mathbb{R}^f \), be given of a system described by the state space model:

\[
\begin{align*}
  x(k + 1) &= Ax(k) + Bu(k) + w(k) \\
  y(k) &= Cx(k) + Du(k) + v(k)
\end{align*}
\]

with \( x(k) \in \mathbb{R}^n \).

The state space description (2.31a) - (2.31b) has two noise sources. The signal \( v(k) \) is called the measurement noise and \( w(k) \) is called the process noise. Let \( w(k) \) and \( v(k) \) be zero-mean white noise sequences that are statistically independent of the input \( u(k) \), with covariance matrices given by,

\[
E\left\{ \begin{bmatrix} w(k) \\ v(k) \end{bmatrix} \begin{bmatrix} w(l)^T \\ v(l)^T \end{bmatrix} \right\} = \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \delta_{kl}
\]
where $\delta_{kl}$ is the Kronecker delta function defined by

$$
\delta_{kl} = \begin{cases} 
0 & k \neq l \\
1 & k = l 
\end{cases}
$$

Let the pair $(A, [B \quad Q^{\frac{1}{2}}])$ be controllable, the pair $(A, C)$ be observable and let the system be stable and let $u(k)$ be PE of sufficient order.

The following system description is called the innovations model form.

$$
\dot{x}(k+1) = Ax(k) + Bu(k) + Ke(k) \quad (2.32a)
$$

$$
\hat{y}(k) = C\dot{x}(k) + Du(k) \quad (2.32b)
$$

$$
y(k) = \hat{y}(k) + e(k) \quad (2.32c)
$$

The system description (2.31a)-(2.31b) is then equivalent to the innovations model form. For every choice of $v(k)$ and $w(k)$ there exists an unique matrix $K$ and a white noise sequence $e(k)$ such that the input output relation is the same. The matrix $K$ is called the Kalman filter gain. The signal $e(k)$ is called the innovation. $e(k)$ is a white noise signal, independent of past values of the input and output. The signal $\hat{y}(k)$ is a minimum variance estimate of the output $y(k)$.

Then the problem is to consistently estimate:

1. the order $n$ of the system.
2. The matrices $A_T$, $B_T$, $C_T$ and $D_T$, where the additional index $T$ refers to the determination of the system matrices up to the freedom in similarity transformation.
3. An estimate of the Kalman gain $K$ such that an approximate minimum variance estimator for $\hat{y}(k)$ can be constructed.

Note that the deterministic part of the innovations model identification problem is a special case of the output-error problem, since the influence of the two white noise terms can be combined to one output noise term. The PI-MOESP method will then give a consistent estimate of the system matrices $A_T$ and $C_T$ if the deterministic part of the system is minimal. However, for this special case where the output-error can be separated into a white measurement noise and process noise contribution, the PO-MOESP algorithm derived in this section will give better results.

### 2.5.2 The past output instrumental variable method

The general data equation corresponding to the innovations model description contains two extra terms due to the noise $e(k)$ and $w(k)$:

$$
Y_{i,j,N} = \Gamma_{j}X_{i,N} + H_{j}U_{i,j,N} + G_{j}W_{i,j,N} + V_{i,j,N} \quad (2.33)
$$
$V_{i,j,N}$ and $W_{i,j,N}$ are again Hankel matrices, constructed as from $v(k)$ and $w(k)$ according to equation (2.7). $G_j$ is a matrix constructed from $C$ and $A$ as follows

$$G_j = \begin{bmatrix}
0 & 0 & \cdots & 0 \\
C & 0 & & \\
CA & C & & \ddots \\
& \vdots & & \\
CA^{j-2} & CA^{j-3} & \cdots & 0
\end{bmatrix}$$

As with the output-error problem, we try to remove all the terms in the right hand side of the data-equation except for the term with the extended observability matrix. We will do this again in two steps. First we project the data equation on the orthogonal complement of the input Hankel matrix. This removes the term $H_1U_{i,j,N}$. Secondly we project the data equation onto the instrumental variable, such that the two noise terms $G_1W_{i,j,N}$ and $V_{i,j,N}$ are removed from equation (2.33).

We again define the past Hankel matrices for the input and output as $U_{0,i,N}$ and $Y_{0,i,N}$, and the future input and output Hankel matrices $U_{i,j,N}$ and $Y_{i,j,N}$.

We can, as with the output-error problem, use the past input as instrumental variable for this purpose, since

$$\lim_{N \to \infty} \frac{1}{N} W_{i,j,N} U_{0,i,N}^T = 0$$
$$\lim_{N \to \infty} \frac{1}{N} V_{i,j,N} U_{0,i,N}^T = 0$$

A second choice that we can make for the instrumental variable is the past output $Y_{0,i,N}$. This is shown in the following lemma.

**Lemma 2.1** Given the system description (2.31a)-(2.31b). Let $Y_{0,i,N}$, $W_{i,j,N}$ and $V_{i,j,N}$ be defined as in equation (2.7) then

$$\lim_{N \to \infty} \frac{1}{N} Y_{0,i,N} W_{i,j,N}^T = 0$$
$$\lim_{N \to \infty} \frac{1}{N} Y_{0,i,N} V_{i,j,N}^T = 0$$

**Proof:**

A block element $(k+1, l+1)$, $k = [0, i-1]$, $l = [0, j-1]$ in the matrix $\lim_{N \to \infty} \frac{1}{N} Y_{0,i,N} W_{i,j,N}^T$ is given by $\lim_{N \to \infty} \frac{1}{N} y(k) w(i + l)^T$. To compute the value of this element, we write the solution $y(k)$ explicitly as:

$$y(k) = CA^k x(0) + C \sum_{\tau=0}^{k-1} A^{k-1-\tau} Bu(\tau) + C \sum_{\tau=0}^{k-1} A^{k-1-\tau} v(\tau)$$
2.5 The innovations model identification problem

With this solution for \( y(k) \), the above block element becomes

\[
\lim_{N \to \infty} \frac{1}{N} y(k)w(i + l)^T = C \cdot \lim_{N \to \infty} \frac{1}{N} x(0)w(i + l)^T + C \sum_{\tau=0}^{k-1} A^{k-1-\tau} B \lim_{N \to \infty} \frac{1}{N} u(\tau)w(i + l)^T
\]

\[
+C \sum_{\tau=0}^{k-1} A^{k-1-\tau} \lim_{N \to \infty} \frac{1}{N} w(\tau)w(k + l)^T + D \lim_{N \to \infty} \frac{1}{N} u(k)w(i + l)^T + \lim_{N \to \infty} \frac{1}{N} \epsilon(k)w(i + l)^T
\]

Because \( k < i \), the independence between \( x(0) \), \( u(k) \) and \( w(k) \) and the white noise property of \( \epsilon(k) \) and \( w(k) \) all terms in the right hand side are zero and we have

\[
\lim_{N \to \infty} \frac{1}{N} y(k)w(i + l)^T = 0 \quad \forall l > 0
\]

Since this holds for every block element in the matrix \( \mathbf{E}\{Y_{0,i,N}W_{i,j,N}^T\} \), the matrix itself is zero. Similarly we can prove \( \mathbf{E}\{Y_{0,i,N}V_{i,j,N}^T\} = 0 \).

The instrumental variable we use in PO-MOESP is based on a combination of the past input and the past output. This is stated in the following corollary:

**Corollary 2.1** Given the system description (2.31a)-(2.31b). Let \( Y_{0,i,N}, W_{i,j,N} \) and \( V_{i,j,N} \) be defined as in equation (2.7). Let the instrumental variable \( \mathbf{Z} \) be defined as

\[
\mathbf{Z} = \begin{bmatrix}
U_{0,i,N} \\
Y_{0,i,N}
\end{bmatrix}
\]

Then the following holds:

\[
\lim_{N \to \infty} \frac{1}{N} Y_{i,j,N} \Pi_{U_{i,j,N}} \mathbf{Z} = \lim_{N \to \infty} \frac{1}{N} \Gamma_{i,j} X_{i,N} \Pi_{U_{i,j,N}} \mathbf{Z}
\]

To distinguish the method from the PI-MOESP algorithm of the previous section, yet keep its acronym short, we denote the algorithm simply **Past Output MOESP** or **PO-MOESP**.

Instead of computing the projection \( Y_{i,j,N} \Pi_{U_{i,j,N}} \mathbf{Z} \) we can more efficiently do the following LQ factorization

\[
\begin{bmatrix}
U_{i,j,N} \\
Y_{i,j,N}
\end{bmatrix} = 
\begin{bmatrix}
U_{i,j,N} \\
U_{0,i,N} \\
Y_{0,i,N} \\
Y_{i,j,N}
\end{bmatrix} = 
\begin{bmatrix}
L_{11} & 0 & 0 & 0 \\
L_{21} & L_{22} & 0 & 0 \\
L_{31} & L_{32} & L_{33} & 0 \\
L_{41} & L_{42} & L_{43} & L_{44}
\end{bmatrix} \begin{bmatrix}
Q_1 \\
Q_2 \\
Q_3 \\
Q_4
\end{bmatrix}
\]

(2.31)
Theorem 2.5  Given the state space system (2.31a) - (2.31b), Let \( u(k) \) be PE of sufficient order. Construct the Hankel matrices \( U_{0,i,N} \), \( Y_{0,i,N} \), \( U_{i,j,N} \) and \( U_{i,j,N} \) as equation (2.7). Then, with the LQ factorization (2.34) the following holds:

\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} \begin{bmatrix} L_{42} & L_{43} \end{bmatrix} = \lim_{N \to \infty} \frac{1}{\sqrt{N}} \Gamma_j X_{i,N} \begin{bmatrix} Q_2 \\ Q_3 \end{bmatrix}^T
\]

(2.35)

Proof:
From the LQ factorization (2.34), we have

\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} \begin{bmatrix} L_{42} & L_{43} \end{bmatrix} = \lim_{N \to \infty} \frac{1}{\sqrt{N}} Y_{i,j,N} \begin{bmatrix} Q_2 \\ Q_3 \end{bmatrix}^T
\]

(2.36)

From the data equation (2.33) we have:

\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} Y_{i,j,N} \begin{bmatrix} Q_2 \\ Q_3 \end{bmatrix}^T = \lim_{N \to \infty} \frac{1}{\sqrt{N}} \Gamma_j X_{i,N} \begin{bmatrix} Q_2 \\ Q_3 \end{bmatrix}^T
\]

+ \lim_{N \to \infty} \frac{1}{\sqrt{N}} H_j U_{i,j,N} \begin{bmatrix} Q_2 \\ Q_3 \end{bmatrix}^T + \lim_{N \to \infty} \frac{1}{\sqrt{N}} G_j W_{i,j,N} \begin{bmatrix} Q_2 \\ Q_3 \end{bmatrix}^T
\]

+ \lim_{N \to \infty} \frac{1}{\sqrt{N}} V_{i,j,N} \begin{bmatrix} Q_2 \\ Q_3 \end{bmatrix}^T
\]

(2.37)

where the term containing the input has disappeared because \( Q_1 \begin{bmatrix} Q_2 \\ Q_3 \end{bmatrix}^T = 0. \)

We will now prove that the last two terms of equation (2.37) will disappear as \( N \) goes to infinity. We thus have to prove:

\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} W_{i,j,N} Q_2^T = 0
\]

(2.38)

\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} W_{i,j,N} Q_3^T = 0
\]

(2.39)

\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} V_{i,j,N} Q_2^T = 0
\]

(2.40)

\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} V_{i,j,N} Q_3^T = 0
\]

(2.41)

Because of the PE condition of the input \( u(k) \) \( \lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{11} \) and \( \lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{22} \) are invertible. Then, with the use of the independence between \( u(k) \) and \( w(k) \) and the ergodicity of these signals we can prove (2.38) as follows.
From the first block-row of the LQ factorization we obtain

\[
\lim_{N \to \infty} \frac{1}{N} U_{i,N} W_{i,N}^T = 0 \\
\lim_{N \to \infty} \frac{1}{N} L_{11} Q_1 W_{i,N}^T = 0 \\
\lim_{N \to \infty} \frac{1}{\sqrt{N}} Q_1 W_{i,N}^T = 0 
\]

Using the second block-row leads to

\[
\lim_{N \to \infty} \frac{1}{N} U_{0,i,N} W_{i,N}^T = 0 \\
\lim_{N \to \infty} \frac{1}{N} \left( L_{21} Q_1 + L_{22} Q_2 \right) W_{i,N}^T = 0 \\
\lim_{N \to \infty} \frac{1}{N} L_{21} Q_2 W_{i,N}^T = 0 \\
\lim_{N \to \infty} \frac{1}{\sqrt{N}} Q_2 W_{i,N}^T = 0 
\]

which is the transpose of (2.38). To prove equation (2.39), we take the result of lemma 2.1:

\[
\lim_{N \to \infty} \frac{1}{N} Y_{0,i,N} W_{i,N}^T = 0 \\
\lim_{N \to \infty} \frac{1}{N} \left( L_{31} Q_1 + L_{32} Q_2 + L_{33} Q_3 \right) W_{i,N}^T = 0 \\
\lim_{N \to \infty} \frac{1}{N} L_{33} Q_3 W_{i,N}^T = 0 
\]

Because of the noise contribution \( \lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{33} \) is invertible and we obtain

\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} Q_3 W_{i,N}^T = 0 
\]

The proof of equation (2.40) and (2.41) is identical to the proof of (2.39) and will be omitted.

Equation (2.37) now reduces to

\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} Y_{i,N} \begin{bmatrix} Q_2 \\ Q_3 \end{bmatrix}^T = \lim_{N \to \infty} \frac{1}{\sqrt{N}} \Gamma_{j',N} \begin{bmatrix} Q_2 \\ Q_3 \end{bmatrix}^T 
\]

which together with equation (2.36) completes the proof. \( \blacksquare \)
Algorithm 2.4 PO-MOESP (deterministic part)

1. Build the matrices $U_{i,j,N}$, $Y_{i,j,N}$, $U_{0,i,N}$ and $Y_{0,i,N}$ according to equation (2.7) with $N = N_m - i - j + 1$.

2. Perform the LQ factorization equation (2.34).

3. Compute the SVD of the matrix \[
\begin{bmatrix}
L_{42} & L_{43}
\end{bmatrix}
\]
\[
\begin{bmatrix}
L_{42} & L_{43}
\end{bmatrix} = U\Sigma V^T
\]

We can now estimate the order $n$ from the gap in the singular values.

4. From the first $n$ left singular vectors of $U$, denoted by $U_n$ we can estimate the matrices $A_T$ and $C_T$. Take $U_1$ as the upper $(i-1)\ell$ rows of $U_n$ and $U_2$ the lower $(i-1)\ell$ rows of $U_n$. Compute $\hat{A}_T$ and $\hat{C}_T$ as follows

\[
\hat{C}_T = \text{the upper } \ell \text{ rows of } U_n
\]
\[
\hat{A}_T = U_1^TU_2
\]

2.5.3 Estimating the stochastic part of the innovations model

In the previous section we estimated the deterministic part of the innovations model. For the construction of a robust controller or a predictor, we also need an estimate of the stochastic part of the model. With the estimates of the covariance matrices of the noise, a LQG controller or a Kalman filter can be constructed.

In this section we estimate the stochastic part of the innovations model. For this we follow the approach of Van Overschee [84].

The data equation of the innovations model is

\[
Y_{i,j,N} = \Gamma_j\hat{X}_{i,N} + H_jU_{i,j,N} + M_jE_{i,j,N}
\]

with $M_i$ defined by

\[
M_j = \begin{bmatrix}
I_\ell & 0 & \cdots & 0 \\
CK & I_\ell & \cdots & \\
\vdots & \ddots & \ddots & 0 \\
CA^{-2}K & CK & I_\ell
\end{bmatrix}
\]

In order to estimate the stochastic matrices $Q$, $R$ and $S$, we first need to estimate the state of the innovation process. As in the estimation of the deterministic part we
define the past Hankel matrices for the input and output as \( U_{0,i,N} \) and \( Y_{0,i,N} \), and the future input and output Hankel matrices \( U_{i,j,N} \) and \( Y_{i,j,N} \). This allows us to estimate the state sequence \( \hat{X}_{i,N} \) as follows.

By substituting equation (2.32c) into (2.32a) we obtain the following state equation

\[
\dot{x}(k+1) = A\hat{x}(k) + Bu(k) + Ky(k)
\]  
(2.44)

with

\[
\hat{A} = A - KC \quad \text{and} \quad \hat{B} = B - KD
\]

The state sequence \( \hat{X}_{i,N} \) can then be written as a function of the initial state \( \hat{X}_{0,N} \) and the two input and output Hankel matrices.

\[
\hat{X}_{i,N} = \hat{A}^i \hat{X}_{0,N} + C_{u,i}U_{0,i,N} + C_{y,i}Y_{0,i,N}
\]  
(2.45)

where

\[
C_{u,i} = \begin{bmatrix} \hat{A}^{i-1} \hat{B} & \hat{A}^{i-2} \hat{B} & \cdots & \hat{B} \\ \hat{A}^{i-1} K & \hat{A}^{i-2} K & \cdots & K \end{bmatrix}
\]

Because \( \hat{A} \) is stable, the term \( \hat{A}^i \hat{X}_{0,N} \) will disappear as \( i \to \infty \). When we substitute equation (2.45) into equation (2.42) and let \( i \) go to infinity, we have

\[
\lim_{i \to \infty} Y_{i,j,N} = \Gamma_j \lim_{i \to \infty} \hat{X}_{i,N} + \lim_{i \to \infty} H_jU_{i,j,N} + \lim_{i \to \infty} M_jE_{i,j,N}
\]

\[
= \Gamma_j \left( \lim_{i \to \infty} \left( C_{u,i}U_{0,i,N} + C_{y,i}Y_{0,i,N} \right) + \lim_{i \to \infty} H_jU_{i,j,N} + \lim_{i \to \infty} M_jE_{i,j,N} \right)
\]  
(2.46)

Since we already have estimated the matrices \( \hat{A}_T, \hat{B}_T, \hat{C}_T \) and \( \hat{D}_T \), we can construct estimates of \( \Gamma_j \) and \( H_j \). This allows us to recover the state sequence \( \hat{X}_{i,N} \) from equation (2.46) by projecting \( Y_{i,j,N} \) onto \( U_{i,j,N}, U_{0,i,N} \) and \( Y_{0,i,N} \). Define \( Z \) as

\[
Z = \begin{bmatrix} U_{i,j,N} \\ U_{0,i,N} \\ Y_{0,i,N} \end{bmatrix}
\]

Then we obtain

\[
\lim_{i \to \infty} Y_{i,j,N} \Pi Z = \lim_{i \to \infty} \Gamma_j \hat{X}_{i,N} \Pi Z + \lim_{i \to \infty} H_jU_{i,j,N} \Pi Z + \lim_{i \to \infty} M_jE_{i,j,N} \Pi Z
\]

The matrix \( E_{i,j,N} \) is independent from \( U_{i,j,N}, U_{0,i,N} \) and \( Y_{0,i,N} \), and therefore the noise term will disappear. For the other two terms we observe the following. Since by equation (2.45) in the limit the row-space of the state is completely spanned by the row-space of the matrices \( Y_{0,i,N} \) and \( U_{0,i,N} \), we have

\[
\lim_{i \to \infty} \hat{X}_{i,N} \Pi Z = \lim_{i \to \infty} \hat{X}_{i,N}
\]  
(2.47)
Equally for the input Hankel matrix we have

$$\lim_{i \to \infty} U_{i,j,N} \Pi Z = \lim_{i \to \infty} U_{i,j,N}$$

(2.48)

For a fixed value of $i$ and $j$ we can now estimate $\hat{X}_{i,N}$ as

$$\hat{X}_{i,N} = \hat{\Gamma}^T_j (Y_{i,j,N} \Pi Z - \hat{H}_j U_{i,j,N})$$

(2.49)

The state sequence $\hat{X}_{i+1,N}$ can be estimated similarly using

$$\hat{X}_{i+1,N} = \hat{\Gamma}^T_{j-1} (Y_{i+1,j-1,N} \Pi U_{i+1,j-1,N}) - \hat{H}_{j-1}U_{i+1,j-1,N})$$

(2.50)

Once we have found $\hat{X}_{i,N}$ and $\hat{X}_{i+1,N}$ we can solve

$$\begin{bmatrix} \hat{X}_{i+1,N} \\ Y_{i+1,N} \end{bmatrix} = \begin{bmatrix} \hat{A}_T & \hat{B}_T \\ \hat{C}_T & \hat{D}_T \end{bmatrix} \begin{bmatrix} \hat{X}_{i,N} \\ U_{i+1,N} \end{bmatrix} + \begin{bmatrix} \hat{W}_{i+1,N} \\ \hat{V}_{i+1,N} \end{bmatrix}$$

(2.51)

for $\hat{W}_{i+1,N}$ and $\hat{V}_{i+1,N}$

The covariances are then estimated by

$$\begin{bmatrix} \hat{Q} & \hat{S} \\ \hat{S}^T & \hat{R} \end{bmatrix} = \frac{1}{N} \begin{bmatrix} \hat{W}_{i+1,N} \\ \hat{V}_{i+1,N} \end{bmatrix} \begin{bmatrix} \hat{W}_{i+1,N} \\ \hat{V}_{i+1,N} \end{bmatrix}^T$$

(2.52)

and with these estimates, we can calculate the Kalman filter gain $K$ with

$$\hat{K} = (\hat{A}_T \hat{P} \hat{C}_T^T + \hat{S})(\hat{C}_T \hat{P} \hat{C}_T^T + \hat{R})^{-1}$$

(2.53)

$$\hat{P} = (\hat{A}_T \hat{P} \hat{A}_T^T + \hat{Q} - (\hat{A}_T \hat{P} \hat{C}_T^T + \hat{S})(\hat{C}_T \hat{P} \hat{C}_T^T + \hat{R})^{-1}(\hat{A}_T \hat{P} \hat{C}_T^T + \hat{S})^T$$

(2.54)

This algorithm can be efficiently implemented using the LQ factorization. This factorization is the same as in PO-MOESP. The $L$ factor that is computed in that algorithm can be reused in the approximate estimation of $K$. Algorithm 2.5 shows the steps involved. For infinite $i$ and $j$ the algorithm determines $K$ asymptotically unbiased.

### 2.6 The errors-in-variables identification problem

The third identification problem is the errors-in-variables problem. In this identification problem the input is perturbed, just like the output, with measurement noise. In this case the previously described methods can not estimate the system consistently, since the disturbance is not uncorrelated with the input.
Algorithm 2.5 PO-MOESP (stochastic part)

1. Estimate \( \hat{\mathbf{A}}_T, \hat{\mathbf{B}}_T, \hat{\mathbf{C}}_T \) and \( \hat{\mathbf{D}}_T \) using PO-MOESP and DESTBDSX.

2. Construct the matrices \( \hat{\mathbf{G}}_j, \hat{\mathbf{G}}_{j-1}, \hat{\mathbf{H}}_j \) and \( \hat{\mathbf{H}}_{j-1}, \), \( \mathbf{U}_{0,i,N} \), \( \mathbf{Y}_{0,i,N} \), \( \mathbf{U}_{i,j,N} \) and \( \mathbf{Y}_{i,j,N} \) with \( N = N_m - i - j + 1 \).

3. Perform the LQ factorization (2.34).

4. Compute
   \[
   \Xi_i = \mathbf{G}_j^\dagger \left( \begin{bmatrix} \mathbf{L}_{11} & \mathbf{L}_{12} & \mathbf{L}_{13} \end{bmatrix} - \mathbf{L}_j \begin{bmatrix} \mathbf{L}_{11} & 0 & 0 \end{bmatrix} \right)
   \]

5. Repartition the \( \mathbf{L} \) matrix according to
   \[
   \begin{bmatrix}
   \mathbf{U}_{i+1,j-1,N} \\
   \mathbf{U}_{i+1,i+1,N} \\
   \mathbf{Y}_{i+1,j+1,N} \\
   \mathbf{Y}_{i+1,i+1,N}
   \end{bmatrix} = \begin{bmatrix}
   \mathbf{L}_{11}' & 0 & 0 & 0 \\
   \mathbf{L}_{21}' & \mathbf{L}_{22}' & 0 & 0 \\
   \mathbf{L}_{31}' & \mathbf{L}_{32}' & \mathbf{L}_{33}' & 0 \\
   \mathbf{L}_{41}' & \mathbf{L}_{42}' & \mathbf{L}_{43}' & \mathbf{L}_{44}'
   \end{bmatrix} \begin{bmatrix}
   \mathbf{Q}_1' \\
   \mathbf{Q}_2' \\
   \mathbf{Q}_3' \\
   \mathbf{Q}_4'
   \end{bmatrix}
   \]

   and compute
   \[
   \Xi_{i+1} = \mathbf{G}_j^\dagger \left( \begin{bmatrix} \mathbf{L}_{41}' & \mathbf{L}_{42}' & \mathbf{L}_{43}' \end{bmatrix} - \mathbf{H}_{j-1} \begin{bmatrix} \mathbf{L}_{11}' & 0 & 0 \end{bmatrix} \right)
   \]

   Take \( \mathbf{L}_u \) to be the row in \( \mathbf{L} \) corresponding to the \( \mathbf{U}_{i,1,N} \) and \( \mathbf{L}_y \) the row corresponding to \( \mathbf{Y}_{i,1,N} \). In MATLAB notation this becomes
   \[
   \begin{align*}
   \mathbf{L}_u &= L(1:m,1:m(i+j)+\ell i+\ell) \\
   \mathbf{L}_y &= L(m(i+j)+\ell i+1:m(i+j)+\ell i+1,1:m(i+j)+\ell i+\ell)
   \end{align*}
   \]

6. Compute the residuals from the equation
   \[
   \mathbf{W} = \begin{bmatrix} \Xi_{i,N} \\ \mathbf{L}_y \end{bmatrix} - \begin{bmatrix} \hat{\mathbf{A}}_T & \hat{\mathbf{B}}_T \\ \hat{\mathbf{C}}_T & \hat{\mathbf{D}}_T \end{bmatrix} \begin{bmatrix} \Xi_{i,N} \\ \mathbf{L}_u \end{bmatrix}
   \]

7. The estimated covariance matrices are computed as
   \[
   \begin{bmatrix}
   \mathbf{\hat{Q}} \\
   \mathbf{\hat{S}}^T \\
   \mathbf{\hat{R}}
   \end{bmatrix} = \mathbf{W} \mathbf{W}^T
   \]

8. The estimated Kalman filter gain \( \mathbf{K} \) is now given by equation (2.53) - (2.54).
This problem is of interest in situations where both the input and the output have to be measured. Therefore, both are contaminated with noise. This situation is very common in closed loop identification. When the system is operated in closed loop, the noise that enters the system at any point in the loop will contaminate both the input and output.

In this section we describe a member of the MOESP family of algorithms that can consistently estimate a system when the input and output are both contaminated with measurement noise. The algorithm is called the errors-in-variables (EIV) method [22].

### 2.6.1 Problem definition

The identification problem is described with the help of figure 2.4.

![Figure 2.4: Schematic representation of the errors-in-variables model.](image)

Let the input/output data-set \( \{u(k), y(k)\}, \ k \in [0, N_m - 1] \) with \( u(k) \in \mathbb{R}^m \) and \( y(k) \in \mathbb{R}^l \), be given of a system described by the state space model:

\[
\begin{align*}
x(k + 1) &= Ax(k) + Bu(k) + w(k) \quad (2.55a) \\
y(k) &= Cx(k) + D\tilde{u}(k) + v(k) \quad (2.55b) \\
u(k) &= \tilde{u}(k) + f(k) \quad (2.55c)
\end{align*}
\]

The signal \( \tilde{u}(k) \) is the input to the system. The signal \( u(k) \) however is the measured signal, which is the input disturbed by \( f(k) \). The noise terms \( w(k), v(k) \) and \( f(k) \) denote zero mean white noise. The input is assumed to be uncorrelated with future values of noise, such that \( \mathbb{E}\{\tilde{u}(k)f(l)^T\} = \mathbb{E}\{\tilde{u}(k)v(l)^T\} = \mathbb{E}\{\tilde{u}(k)w(l)^T\} = 0 \) for all \( l \leq k \). The initial state \( x(0) \) is assumed to be independent of the noise. Finally, the system is assumed to be stable and minimal. Then the problem is to consistently estimate:

1. The order \( n \) of the system.
2. The quadruple \( A_T, B_T, C_T \) and \( D_T \), where the additional index \( T \) refers to the determination of the state space matrices up to a similarity transformation.
Remark 2.3 The input measurement noise \( f(k) \) can also be modeled as part of the process disturbance, using

\[
x(k + 1) = Ax(k) + Bu(k) + \hat{w}(k)
\]

with

\[
\hat{w}(k) = w(k) - Bf(k)
\]

But in that case, the noise and the input are not independent anymore, and the instrumental variable methods, as used in the previous subspace identification algorithms will not give consistent estimates.

2.6.2 The errors-in-variables method

In this section we describe a member of the MOESP family, called the errors-in-variables MOESP method or EIV-MOESP[22]. This algorithm will give consistent estimates of \( A_T \) and \( C_T \) in the case of input measurement noise.

With the model description (2.55a) - (2.55c), the noise term \( f(k) \), introduces an extra term in the data equation for the errors-in-variables model.

\[
Y_{i,j,N} = \Gamma_j X_{i,N} + H_j U_{i,j,N} - H_j F_{i,j,N} + G_j W_{i,j,N} + V_{i,j,N}
\]  

(2.56)

\( F_{i,j,N} \) is again a Hankel matrix, constructed from \( f(k) \) according to equation (2.7) and \( N = N_m - i - j + 1 \).

In EIV-MOESP we consider the following LQ factorization:

\[
\begin{bmatrix} U_{i,j,N} \\ Y_{i,j,N} \end{bmatrix}^T = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \]  

(2.57)

with

\[
Z = \begin{bmatrix} U_{0,i,N} \\ Y_{0,i,N} \end{bmatrix}
\]

Theorem 2.6 Given the state space equations (2.55a)-(2.55c) with \( w(k), \nu(k) \) and \( f(k) \) zero mean white noise, independent of the input \( u(k) \) and the initial state \( x(0) \). Let the input be PE of sufficient order. Construct the Hankel matrices \( U_{0,i,N} \) and \( Y_{0,i,N} \) and \( Z \). Then with the LQ factorization, equation (2.57), the following holds

\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{22} = \lim_{N \to \infty} \frac{1}{\sqrt{N}} \Gamma_j X_{i,N} Z^T Q_2^T
\]
Proof:
From the LQ factorization, we have
\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{22} = \lim_{N \to \infty} \frac{1}{\sqrt{N}} Y_{i,j,N} Z^T Q_2^T
\]

From the data equation (2.56) we have:
\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} Y_{i,j,N} Z^T Q_2^T = \lim_{N \to \infty} \frac{1}{\sqrt{N}} \Gamma_i X_{i,N} Z^T Q_2^T - \lim_{N \to \infty} \frac{1}{\sqrt{N}} H_i F_{i,j,N} Z^T Q_2^T \\
+ \lim_{N \to \infty} \frac{1}{\sqrt{N}} G_i W_{i,j,N} Z^T Q_2^T + \lim_{N \to \infty} \frac{1}{\sqrt{N}} V_{i,j,N} Z^T Q_2^T
\]

(2.58)

where the term containing the input \( u(k) \) has already disappeared because of the post-multiplication with \( Q_2^T \). Because the past input and output are independent of the noise sources \( f(k), w(k) \) and \( v(k) \) we have
\[
\lim_{N \to \infty} \frac{1}{N} F_{i,j,N} Z^T = 0 \\
\lim_{N \to \infty} \frac{1}{N} W_{i,j,N} Z^T = 0 \\
\lim_{N \to \infty} \frac{1}{N} V_{i,j,N} Z^T = 0
\]

This reduces equation (2.58)
\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} Y_{i,j,N} Z^T Q_2^T = \lim_{N \to \infty} \frac{1}{\sqrt{N}} \Gamma_i X_{i,N} Z^T Q_2^T
\]

(2.59)

which concludes the proof. ■

With the EIV-MOESP algorithm the matrices \( A \) and \( C \) can be estimated. The estimation of \( B \) and \( D \) in an errors-in-variable setting is treated in section 2.9.

2.7 The closed loop identification problem

As described in the previous section, the EIV algorithm can be applied to the errors-in-variables problem. In the proof of the previous section, we only assumed the input to be uncorrelated with future noise. Since this also holds in the closed loop identification problem, the EIV algorithm can be applied to closed loop data as well.

2.7.1 Problem definition

The closed loop identification problem is shown in figure 2.5.
Let the input/output data-set \( \{u(k), y(k)\}, k \in [0, N_m - 1] \) with \( u(k) \in \mathbb{R}^m \) and \( y(k) \in \mathbb{R}^l \), be given of a system described by the state space model:

\[
\begin{align*}
x(k+1) &= Ax(k) + Bu(k) + w(k) \\
y(k) &= Cx(k) + Du(k) + v(k) \\
\hat{u} &= r(k) + u_c(k) \\
n(k) &= \hat{u}(k) + f(k)
\end{align*}
\]

And with the controller given by

\[
\begin{align*}
x_c(k+1) &= A_c x_c(k) + B_c y(k) \\
u_c(k) &= C_c x_c(k)
\end{align*}
\]

The signal \( r(k) \) is the reference input to the system and \( u_c \) the controller input. The signal \( u(k) \) is the measured signal. This signal is perturbed by \( f(k) \). The output is perturbed by the noise signals \( w(k) \) and \( v(k) \). \( w(k) \), \( v(k) \) and \( f(k) \) denote zero mean white noise. The reference input \( r(k) \) and the initial states \( x(0) \) and \( x_c(0) \) are assumed to be uncorrelated with future values of noise. The system is assumed to be minimal and to have at least one delay. The controller is assumed to be stabilizing and causal. Then the problem is to consistently estimate:

1. The order \( n \) of the system.
2. The quadruple \( A_T, B_T, C_T \) and \( D_T \), where the additional index \( T \) refers to the determination of the state space matrices up to a similarity transformation.

### 2.7.2 Errors-in-variable MOESP and the closed loop problem

The errors-in-variables method can be applied to the closed loop identification problem. In order to show this, we need to assess that the requirement \( \mathbb{E}\{\hat{u}(k)f_l^T\} = \)
\[ \mathbb{E}\{\tilde{u}(k)v(l)^T\} = \mathbb{E}\{\tilde{u}(k)w(l)^T\} = 0 \quad \text{for all } l \leq k \text{ holds.} \] The output of the system is given by

\[
y(k) = CA^k x(0) + \sum_{\tau=0}^{k-1} CA^{k-1-\tau} Br(\tau) + \sum_{\tau=0}^{k-1} CA^{k-1-\tau} Bu_c(\tau) + \sum_{\tau=0}^{k-1} CA^{k-1-\tau} w(\tau) + Dr(k) + Du_c(k) + v(k)
\]

The controller signal \( u_c(k) \) is given by

\[
u_c(k) = C_c A_c^k x_c(0) + \sum_{\tau=0}^{k-1} C_c A_c^{k-1-\tau} B_c y(\tau)
\]

Since the closed loop system is stable, \( \tilde{u}(k) \) is a stationary signal. The requirement \( \mathbb{E}\{\tilde{u}(k)f(l)^T\} = \mathbb{E}\{\tilde{u}(k)v(l)^T\} = \mathbb{E}\{\tilde{u}(k)w(l)^T\} = 0 \) for all \( l \leq k \) holds since \( \tilde{u}(k) \) depends only on past values of \( y(k) \) and therefore on past values of the noise. In this case we can use the EIV-MOESP algorithm to estimate \( A \) and \( C \). Note that we do not need any information about the controller, other than that there is at least one sample delay.

**Algorithm 2.6 EIV-MOESP**

1. Build the matrices \( U_{i,j,N}, Y_{i,j,N}, U_{0,i,N} \) and \( Y_{0,i,N} \) according to equation (2.7) and \( N = N_m - i - j + 1 \).
2. Perform the LQ factorization equation (2.57).
3. Compute the SVD of the matrix \( L_{22} \):

\[
L_{22} = U \Sigma V^T
\]
4. Estimate the order \( n \) using the singular values. From the first \( n \) left singular vectors of \( U \), denoted by \( U_n \) we can estimate the matrices \( A_T \) and \( C_T \). Take \( U_1 \) as the upper \((i - 1)\ell\) rows of \( U_n \) and \( U_2 \) the lower \((i - 1)\ell\) rows of \( U_n \).
5. Compute \( \hat{A}_T \) and \( \hat{C}_T \) as follows

\[
\hat{C}_T = \text{the upper } \ell \text{ rows of } U_n
\]
\[
\hat{A}_T = U_1^T U_2
\]
2.8 Persistence of excitation

In the previous sections we have frequently referred to the persistence of excitation property of the input signal. In section 2.2.2 we gave the basic definition of Persistence of Excitation (PE). As mentioned there, we need a certain degree of excitation to be able to identify the system successfully. However, we did not go into detail about the exact requirements put on the input for successful identification. These conditions were researched in [6, 24, 45, 46, 96]. In this section we will give an overview of the results obtained in these papers. We have adapted the results to the different MOESP algorithms that were treated in this chapter and to the notation that is used here.

2.8.1 Requirements on the input for OM-MOESP

The first algorithm we examine is the ordinary MOESP algorithm. In OM-MOESP the matrix $\Gamma_i$ is estimated as the column-space of the following matrix

$$Y_{0,i,N} \Pi_{U_{0,i,N}}^T$$

(2.62)

From equation (2.10) we have that this matrix must equal

$$\Gamma_i X_{0,N} \Pi_{U_{0,i,N}}^T$$

When $i$ is chosen larger than $n$, the extended observability matrix $\Gamma_i$ is of rank $n$. This matrix can only be recovered from the column-space of $Y_{0,i,N} \Pi_{U_{0,i,N}}^T$ if this matrix is also of rank $n$. The covariance of $Y_{0,i,N} \Pi_{U_{0,i,N}}^T$ is given by

$$\mathbb{E}\{Y_{0,i,N} \Pi_{U_{0,i,N}}^T Y_{0,i,N}^T\} = \Gamma_i \mathbb{E}\{X_{0,N} \Pi_{U_{0,i,N}}^T X_{0,N}^T\} \Gamma_i^T$$

$$= \Gamma_i (R_x - R_{xu} R_u^{-1} R_{xu}^T) \Gamma_i^T$$

(2.63)

with

$$R_x = \lim_{N \to \infty} \frac{1}{N} X_{0,N} X_{0,N}^T$$

$$R_{xu} = \lim_{N \to \infty} \frac{1}{N} X_{0,N} U_{0,i,N}^T$$

$$R_u = \lim_{N \to \infty} \frac{1}{N} U_{0,i,N} U_{0,i,N}^T$$

Equation (2.63) has rank $n$ if, and only if, $R_x - R_{xu} R_u^{-1} R_{xu}^T$ is positive definite. Using the Schur complement, this requirement is equal to the statement that

$$\begin{bmatrix} R_x & R_{xu} \\ R_{xu}^T & R_u \end{bmatrix} = \mathbb{E}\begin{bmatrix} X_{0,N} \\ U_{0,i,N} \end{bmatrix} \begin{bmatrix} X_{0,N} \\ U_{0,i,N} \end{bmatrix}^T > 0$$
It is shown in [46] that, since \( x(k) \) depends on past values of the input, this matrix is positive definite when \( u(k) \) PE of order \( n + i \). This is formally stated in the following lemma.

**Lemma 2.2 (Jansson [46])** Given the state space system (2.5a)-(2.5b) Then

\[
\lim_{N \to \infty} \frac{1}{N} \begin{bmatrix} X_{0,N} \\ U_{0,i,N} \end{bmatrix}^T \begin{bmatrix} X_{0,N} \\ U_{0,i,N} \end{bmatrix}^T > 0
\]

if the input \( u(k) \) is PE of order \( n + i \).

\[\square\]

### 2.8.2 Requirements on the input for PI-MOESP

For the PI-MOESP and PO-MOESP algorithms we have the following result. The matrix \( \Gamma_j \) is estimated as the column-space of

\[
Y_{i,j,N} \Pi_{U_{i,j,N}}^T \Pi_Z
\]

This matrix equals

\[
(\Gamma_j X_{i,N} + H_j U_{i,j,N} + V_{i,j,N}) \Pi_{U_{i,j,N}}^T \Pi_Z = (\Gamma_j X_{i,N} + V_{i,j,N}) \Pi_{U_{i,j,N}}^T \Pi_Z
\]

where the instrumental variable \( Z \) for PI-MOESP is equal \( U_{0,i,N} \).

The covariance of \( Y_{i,j,N} \Pi_{U_{i,j,N}}^T \Pi_Z \) is given by

\[
\Gamma_j (R_{xz} - R_{xu} R_u^{-1} R_{uz}) R_z^{-1} (R_{xz} - R_{xu} R_u^{-1} R_{uz})^T \Gamma_j^T
\]

with

\[
R_x = \lim_{N \to \infty} \frac{1}{N} X_{i,N} X_{i,N}^T
\]

\[
R_u = \lim_{N \to \infty} \frac{1}{N} U_{i,j,N} U_{i,j,N}^T
\]

\[
R_z = \lim_{N \to \infty} \frac{1}{N} Z Z^T
\]

\[
R_{uz} = \lim_{N \to \infty} \frac{1}{N} U_{i,j,N} Z^T
\]

\[
R_{zu} = \lim_{N \to \infty} \frac{1}{N} X_{i,N} U_{i,j,N}^T
\]

\[
R_{xz} = \lim_{N \to \infty} \frac{1}{N} X_{i,N} Z^T
\]

The PI-MOESP algorithm gives consistent results if \((R_{xz} - R_{xu} R_u^{-1} R_{uz}) R_z^{-1} (R_{xz} - R_{xu} R_u^{-1} R_{uz})^T > 0\). This requirement is fulfilled if the following two conditions hold:
2.8 Persistence of excitation

(a) \( \text{rank} \left( R_z^{-1} \right) = \text{full} \)

(b) \( \text{rank} \left( R_{xz} - R_{xu} R_u^{-1} R_u \right) = n \)

Requirement (a) is met if \( R_z > 0 \), which is true if the input is PE of order \( i \). Requirement (b) is equivalent with

\[
\text{rank} \left( \begin{bmatrix} R_{xz} & R_{xu} \\ R_{uz} & R_u \end{bmatrix} \right) = n + im
\]  \( (2.64) \)

The above matrix can be written as

\[
\begin{bmatrix} R_{xz} & R_{xu} \\ R_{uz} & R_u \end{bmatrix} = \lim_{N \to \infty} \frac{1}{N} \begin{bmatrix} X_{i,N} \\ U_{i,j,N} \end{bmatrix} \begin{bmatrix} U_{0,i,N} \\ U_{i,j,N} \end{bmatrix}^T
\]  \( (2.65) \)

Therefore \text{PI-MOESP} is consistent if the input is PE of order \( i \) and if the matrix in equation (2.65) is full rank. Unlike for the \text{OM-MOESP} case no single PE requirement for the input is known in this case, to the authors knowledge.

2.8.3 Requirements on the input for PO-MOESP

In the \text{PO-MOESP} case the extended observability matrix \( \Gamma_j \) is estimated from matrix

\[
Y_{i,j,N} \Pi_{U_{i,j,N}}^1 \Pi_Z
\]

This gives rise to the same covariance matrix as in the \text{PI-MOESP} case (i.e. equation (2.64)) but with a different instrumental variable \( Z \):

\[
Z = \begin{bmatrix} U_{0,i,N} \\ Y_{0,i,N} \end{bmatrix}
\]

Identical to the \text{PI-MOESP} case we have two requirements in order to assure this matrix to be positive definite:

(a) \( \text{rank} \left( R_z^{-1} \right) = \text{full} \)

(b) \( \text{rank} \left( R_{xz} - R_{xu} R_u^{-1} R_u \right) = n \)

We will first investigate what consequences requirement (b) has for the persistence of excitation assumption of the input for \text{PO-MOESP} algorithm. The above requirement is equivalent with the condition

\[
\text{rank} \left( \begin{bmatrix} X_{i,N} \\ U_{i,j,N} \end{bmatrix} \begin{bmatrix} Y_{0,i,N} \\ U_{0,i,N} \end{bmatrix}^T \right) = n + mi
\]
Both the state \( x(k) \) and the output \( y(k) \) consist of a deterministic part, caused by \( u(k) \) and a stochastic part, caused by \( w(k) \). Separating these, using the superscripts \( d \) and \( s \), gives

\[
\begin{bmatrix}
X_{i,N}^d \\
U_{i,j,N}
\end{bmatrix}^T 
= \begin{bmatrix}
X_{i,N}^d \\
U_{i,j,N}^d \\
X_{i,N}^s \\
U_{i,j,N}
\end{bmatrix}^T 
+ \begin{bmatrix}
X_{i,N}^s \\
U_{i,j,N}^s
\end{bmatrix}
\]

\( (2.66) \)

We will now derive the condition on the input that guarantees that the deterministic part of the above equation is of rank \( n + mi \).

The state sequence \( X_{i,N}^d \) in this expression is equal to

\[
X_{i,N}^d = A^i X_{0,N}^d + \sum_{k=0}^{i-1} A^{i-k-1} B u(k)
\]

\[
= A^i X_{0,N}^d + C^r_i U_{0,i,N}
\]

\( (2.67) \)

with \( C^r_i = [A^i B \quad A^{i-1} B \quad \cdots \quad B] \).

Similarly for \( Y_{0,i,N}^d \) we can write

\[
Y_{0,i,N}^d = \Gamma_i X_{0,N}^d + H_i U_{0,i,N}
\]

\( (2.68) \)

Substituting this in equation (2.65) gives

\[
\begin{bmatrix}
X_{i,N}^d \\
U_{i,j,N}
\end{bmatrix}^T 
= \begin{bmatrix}
A^i & C^r_i & 0 \\
0 & 0 & I_{mj}
\end{bmatrix} \lim_{N \to \infty} \frac{1}{N} \begin{bmatrix}
X_{0,N}^d \\
X_{0,i,N}^d \\
X_{0,i,j,N}
\end{bmatrix}^T 
\begin{bmatrix}
\Gamma_i^T & 0 & 0 \\
H_i^T & I_{mi} & 0 \\
0 & 0 & I_{mj}
\end{bmatrix}
\]

The matrices

\[
\begin{bmatrix}
A^i & C^r_i & 0 \\
0 & 0 & I_{mj}
\end{bmatrix}
\quad \text{and} \quad 
\begin{bmatrix}
\Gamma_i^T & 0 & 0 \\
H_i^T & I_{mi} & 0 \\
0 & 0 & I_{mj}
\end{bmatrix}
\]

are of both of full row rank if the system is controllable and \( i \geq n \). Therefore it is sufficient for (2.64) to hold that

\[
\lim_{N \to \infty} \frac{1}{N} \begin{bmatrix}
X_{0,N} \\
X_{0,i,N} \\
X_{0,i,j,N}
\end{bmatrix}^T > 0
\]

Using lemma 2.2 we find that this is the case if \( u(k) \) is persistently exciting of order \( i + j + n \).
It can be shown that under special conditions on the disturbance, the second term in equation (2.66) can destroy the rank of the complete matrix [46, 96]. In general this is however unlikely if \( j \ell >> n \).

Condition a) is met when

\[
\begin{bmatrix}
U_{0,i,N} \\
Y_{0,i,N}
\end{bmatrix}
\begin{bmatrix}
U_{0,i,N} \\
Y_{0,i,N}
\end{bmatrix}^T > 0
\]  
(2.69)

Using the same method as for requirement b) we can write the deterministic part of this matrix as

\[
\begin{bmatrix}
0 & I_{m_j} \\
\Gamma & H_i
\end{bmatrix}
\lim_{N \to \infty} \frac{1}{N}
\begin{bmatrix}
X_{0,N}^d \\
U_{0,i,N}
\end{bmatrix}
\begin{bmatrix}
X_{0,N}^d \\
U_{0,i,N}
\end{bmatrix}^T
\begin{bmatrix}
\Gamma_i^T & 0 \\
H_i & I_{m_j}
\end{bmatrix}
\]

which is true if the input is PE of order \( i + n \) Combining condition a) and b) we find that PO-MOESP gives consistent estimates if the input is PE of order \( i + j + n \)

### 2.8.4 Requirements on the input for EIV-MOESP

In the EIV-MOESP case the extended observability matrix \( \Gamma_j \) is estimated from matrix

\[
Y_{i,j,N} \mathbf{Z}_i^T \Pi_{U_{i,j,N}}^{-1} \mathbf{Z}_i
\]

Again we take the covariance of this expression. This leads to

\[
\Gamma_j \left( R_{xz} R_{xz}^T - R_{xz} R_{uz}^T \left( R_{uz} R_{uz}^T \right)^{-1} R_{uz} R_{xz}^T \right) \Gamma_j^T
\]

This results in the following requirement for consistent estimate

\[
R_{xz} R_{xz}^T - R_{xz} R_{uz}^T \left( R_{uz} R_{uz}^T \right)^{-1} R_{uz} R_{xz}^T > 0
\]

which is equivalent with

\[
\begin{bmatrix}
R_{xz} R_{xz}^T \\
R_{uz} R_{uz}^T
\end{bmatrix}
\begin{bmatrix}
R_{xz} \\
R_{uz}
\end{bmatrix}^T > 0
\]

This matrix is positive definite if, and only if, the matrix \( \begin{bmatrix} R_{xz} \\ R_{uz} \end{bmatrix} \) is of rank \( n + m_i \).

This means that the EIV-MOESP algorithm is consistent if

\[
\text{rank} \left( \begin{bmatrix}
X_{i,N} \\
U_{i,j,N}
\end{bmatrix}
\begin{bmatrix}
X_{0,i,N} \\
U_{0,i,N}
\end{bmatrix}^T \right) = n + m_i
\]  
(2.70)

As with the PI-MOESP algorithm, a single PE requirement for the input is not known in EIV-MOESP case, to the authors knowledge.
2.9 Estimation of $B$ and $D$ and the initial state

The MOESP algorithms that were treated thus far found estimates of the order of the system and the matrices $A$ and $C$. In a second step we will estimate the matrices $B$, $D$ and the initial state. In the calculation of these matrices we assume that the matrices $A$ and $C$ are known. In practice we will use their estimates. When these estimates are consistent, this will result in a consistent estimate of $B$, $D$ and $x_0$.

We will first derive the basic algorithm. This algorithm deals with the estimation of $B$, $D$ and $x_0$ for the noise-free, the output error and the innovations model identification problem. For the errors-in-variables problem a slightly different approach has to be taken to deal with the input measurement noise. This is treated afterwards.

2.9.1 Algorithm for estimation $B$, $D$ and $x_0$

The output $y(k)$ of a system given by the state space system description (2.31a)-(2.31b) can be rewritten as,

$$y(k) = CA^kx(0) + \sum_{\tau=0}^{k-1} CA^{k-1-\tau}Bu(\tau) + Du(k) + v(k)$$

This description includes model description for the noise-free and the innovations model identification problems. Using the Kronecker product (see appendix A.4) we can rewrite the above equation as follows:

$$y(k) = CA^kx(0) + \left[ \sum_{\tau=0}^{k-1} u(\tau)^T \otimes CA^{k-1-\tau} \right] \text{vec}(B)$$

$$+ [u(k)^T \otimes I_i] \text{vec}(D) + v(k) \quad (2.71)$$

where $\otimes$ and $\text{vec}(\cdot)$ respectively denote the Kronecker product and the vector resulting by storing the columns of the matrix $(\cdot)$ on top of each other. $e_i$ is the $i$-th column
of the identity matrix \( I_n \). When we define the following matrices:

\[
Y_{0,N,1} = \begin{bmatrix}
y(0) \\
y(1) \\
\vdots \\
y(N-1)
\end{bmatrix} \quad \Gamma_N = \begin{bmatrix}
C \\
CA \\
\vdots \\
C^{N-1}
\end{bmatrix}
\]

\[
\mathcal{Y} = \begin{bmatrix}
0 \\
u(0)^T \otimes C \\
\vdots \\
\sum_{\tau=0}^{N-2} u(\tau)^T \otimes C A^{N-2-\tau}
\end{bmatrix} \quad \mathcal{U} = \begin{bmatrix}
u(0)^T \otimes I_l \\
u(1)^T \otimes I_l \\
\vdots \\
u(N-1)^T \otimes I_l
\end{bmatrix}
\]

\[
E = \begin{bmatrix}
u(0) \\
u(1) \\
\vdots \\
u(N-1)
\end{bmatrix} \quad B = \text{vec}(B) \quad D = \text{vec}(D)
\]

then we can write equation (2.71) as a matrix equation:

\[
Y_{0,N,1} = \begin{bmatrix}
\Gamma_N \\
\mathcal{Y} \\
\mathcal{U}
\end{bmatrix} \begin{bmatrix}
x_0 \\
B \\
D
\end{bmatrix} + E
\]  
(2.74)

Because \( x_0 \) and \( u(k) \) are independent of the noise \( v(k) \), the coefficients of \( B, D \) and \( x_0 \) can be retrieved unbiased by solving the above equation in a least square sense.

\[
\begin{bmatrix}
x_0 \\
B \\
D
\end{bmatrix} = \left( \begin{bmatrix}
\Gamma_N \\
\mathcal{Y} \\
\mathcal{U}
\end{bmatrix} \right)^\dagger Y_{0,N,1}
\]

(2.75)

In practice we don’t have the matrices \( A \) and \( C \) but their estimates \( \hat{A}_T \) and \( \hat{C}_T \). When we substitute the real matrices with their consistent estimates we will estimate the matrices \( \hat{B}_T \) and \( \hat{D}_T \) and \( \hat{x}_0 \) consistently with the same similarity transformation as \( A_T \) and \( C_T \). Algorithm 2.7 gives a summary of the estimation procedure. More details about methods to solve the LS problem can be found in section 5.3.

### 2.9.2 Estimating \( B, D \) and \( x_0 \) in closed loop

In order to adapt the basic algorithm to the errors-in-variables we need to take the input measurement noise into account. The basic algorithm assumes the noise to be independent from the input. This is no longer the case.

With the model description (2.55a)-(2.55b) we get the following expression for the
Leopold Kronecker (1823-1891) was taught mathematics at school by Kummer and it was due to him that Kronecker became interested in mathematics. He studied under Jacobi, Dirichlet and Eisenstein. He wrote his Ph. D. thesis on algebraic number theory under Dirichlet’s supervision.

His primary contributions were in the theory of equations and higher algebra.

Kronecker believed in the reduction of all mathematics to arguments involving only the integers and a finite number of steps. Kronecker is well known for his remark “God created the integers, all else is the work of man.”

$$y(k) = CA^k x(0) + \sum_{\tau=1}^{k-1} CA^{k-1-\tau} Bu(\tau) - \sum_{\tau=1}^{k-1} CA^{k-1-\tau} Bf(\tau) + Du(k)$$

$$+ \sum_{\tau=1}^{k-1} CA^{k-1-\tau} w(\tau) + v(k)$$

In matrix form we can write this as

$$Y_{0,N,1} = \Gamma_N x_0 + \begin{bmatrix} \mathcal{Y} & \mathcal{U} \end{bmatrix} \begin{bmatrix} x_0 \\ B \\ D \end{bmatrix} + E + E_f$$

$$\quad \tag{2.76}$$

with

$$E = \begin{bmatrix} v(0) \\ Cw(0) + v(1) \\ \vdots \\ \sum_{\tau=0}^{N-2} CA^{N-2-\tau} w(\tau) + v(N-1) \end{bmatrix}$$

$$E_f = \begin{bmatrix} 0 \\ -CBf(0) \\ \vdots \\ -\sum_{\tau=0}^{N-2} CA^{N-2-\tau} Bf(\tau) \end{bmatrix}$$

Since the $E_f$ is correlated with $\mathcal{Y}$ and $\mathcal{U}$ we can not solve this equation directly for $x_0$, $B$ and $D$. We therefore pre-multiply equation (2.76) with an instrumental variable $Z$. This instrumental variable has to meet the following requirements.

$$\lim_{N \to \infty} \frac{1}{N} Z^T E_f = 0$$

$$\lim_{N \to \infty} \frac{1}{N} Z^T E = 0$$

$$\lim_{N \to \infty} \frac{1}{N} Z^T \begin{bmatrix} \Gamma_N & \mathcal{Y} & \mathcal{U} \end{bmatrix} \text{ is full rank}$$
An instrumental variable that satisfies these requirements is

\[ Z = \begin{bmatrix} Y_{N-i_1-j+1,N,i_1}^T \otimes I_p & U_{N-i_1-j+2,N,i_2}^T \otimes I_p \end{bmatrix} \]

Here \( i_1 \) and \( i_2 \) are design parameters that are specified by the user. The matrix \( Z \) consists of values of the input and output with time indices less than 0. The input used in equation (2.76) consist of data with indices larger or equal to zero. Because \( u(k) \) and \( y(k) \) are uncorrelated with future noise the first two requirements are met.

The condition under which the third requirement holds is more difficult to evaluate. This depends on the choice of input and the system itself [22].

An undesired side effect of this instrumental variable is that it disallows the estimation of the initial state. Since \( \lim_{N \to -\infty} \frac{1}{N} Z^T \Gamma_N = 0 \) the estimation of \( x_0 \) is not possible. \( B \) and \( D \) can however be estimated by solving the following equation

\[
\begin{bmatrix} B \\ D \end{bmatrix} = (Z^T \begin{bmatrix} \mathcal{Y} & \mathcal{U} \end{bmatrix})^\dagger (Z^T y_{0,N,1})
\]

(2.77)

**Algorithm 2.7 DESTBDX**

1. Construct the matrices \( U_{0-N-i_1+1,N,i_1}, Y_{0-N-i_2+1,N,i_2}, Y_{0,N,1}, \Gamma_N, \mathcal{U} \) and \( \mathcal{Y} \) using equations (2.72) and (2.73) with \( A \) and \( C \) substituted with their estimates \( \hat{A}_T \) and \( \hat{C}_T \).

2. For the basic algorithm solve

\[
\begin{bmatrix} \hat{x}_{0,T} \\ \hat{B}_T \\ \hat{D}_T \end{bmatrix} = \begin{bmatrix} \Gamma_N & \mathcal{Y} & \mathcal{U} \end{bmatrix}^\dagger y_{0,N,1}
\]

For the instrumental variable method construct

\[ Z = \begin{bmatrix} Y_{N-i_1-j+1,N,i_1}^T \otimes I_p & U_{N-i_1-j+2,N,i_2}^T \otimes I_p \end{bmatrix} \]

and solve

\[
\begin{bmatrix} \hat{B}_T \\ \hat{D}_T \end{bmatrix} = (Z^T \begin{bmatrix} \mathcal{Y} & \mathcal{U} \end{bmatrix})^\dagger (Z^T y_{0,N,1})
\]

3. Reconstruct \( \hat{B}_T, \hat{D}_T \) and if available \( \hat{x}_{0,T} \) from the above solution.
2.10 The Wiener model identification problem.

So far we have only treated linear identification problems. However, a number of nonlinear problems such as the Wiener, Hammerstein and bilinear model identification problem [28, 85, 94, 101] can also be solved by subspace based identification methods. In this section we will derive a SMI algorithm for the Wiener model identification problem. Here we additionally, see figure 2.6, assume the error free output of the linear part to be transformed by a square non-linear mapping $f(\cdot) : \mathbb{R}^\ell \rightarrow \mathbb{R}^\ell$. Based on this figure, we define the following Wiener model identification problem.

Let the input/output data-set $\{u(k), z(k)\}$, $k \in [0, N_m - 1]$, with $u(k) \in \mathbb{R}^m$ and $z(k) \in \mathbb{R}^\ell$, be given of a system described by the state space model:

\begin{align}
    x(k+1) &= Ax(k) + Bu(k) \\
    y(k) &= Cx(k) + Du(k) \\
    z(k) &= f(y(k)) + v(k)
\end{align}

(2.78a)  
(2.78b)  
(2.78c)

with $x(k) \in \mathbb{R}^n$ and $y(k) \in \mathbb{R}^\ell$.

Let $v(k)$ be an additive perturbation (stochastic or deterministic) which
is independent of the input, such that:
\[
\lim_{N_m \to \infty} \frac{1}{N_m} \sum_{k=1}^{N_m} u(k)v(k)^T = 0
\]
Let \(f(\cdot)\) be the non-linear function, described by
\[
f(y(k)) = \sum_{q=1}^{\infty} \Psi_q Y_q
\]
where \(\Psi_q\) is a constant matrix \(Y_q\) consists of all possible \(q\)-th order powers and cross-powers of the elements of \(y(k)\). Finally, let the system be minimal and stable.

Then the problem is to consistently estimate:

1. The order \(n\) of the system.
2. The matrices \(A_T, B_T, C_T\) and \(D_T\), where the additional index \(T\) refers to the determination of the system matrices up to a similarity transformation and the matrix \(D_T\) refers to an additional unidentifiable scaling of the output of the linear system.
3. The static non-linearity
\[
f(D_T^{-1}(\cdot))
\]

Such type of problems arise in many practical applications, e.g. in the identification of distillation columns [91]. The Wiener model can also be used as an approximation of a general non-linear system.

The following lemma is a general result for zero-mean jointly Gaussian signals. We will use this lemma to prove theorem 2.7 which in turn will help us to solve the Wiener model problem.

**Lemma 2.3 ([8])** Given \(n\) zero-mean jointly Gaussian random variables \(x_1, x_2, \ldots, x_n\) then the expected value of the product of these variables
\[
E\{x_1x_2 \cdots x_n\} \quad (2.80)
\]
is zero for odd values of \(n\) and for even values of \(n\), the expected value is equal to the sum over all possible permutations of the product of expected values of products of pairs of \(x_1, x_2, \ldots, x_n\). \[\Box\]

**Example 2.2** For the expectation of the product of 4 zero mean jointly Gaussian signals, the above lemma results in:
\[
E\{x_1x_2x_3x_4\} = E\{x_1x_2\}E\{x_3x_4\} + E\{x_1x_3\}E\{x_2x_4\} + E\{x_1x_4\}E\{x_2x_3\} \quad (2.81)
\]
\[\Box\]
Johann Carl Friedrich Gauss (1777-1855) worked in a wide variety of fields in both mathematics and physics including number theory, analysis, differential geometry, geodesy, magnetism, astronomy and optics. Gauss studied at Brunswick Collegium Carolinum in Germany and received his degree in 1799. Gauss devoted himself to research on a stipend from the Duke of Brunswick. In 1807 he became director of the Göttingen observatory.

Gauss’s work never seemed to suffer from his personal tragedy. His first wife died after giving birth to their second son, who was to die soon after her. In 1831 Gauss’s second wife died after a long illness.

### 2.10.1 WIE-MOESP with odd non-linear functions

Using lemma 2.3 we can prove the following theorem, which is a special case of Bussgang’s theorem [8, 18].

**Theorem 2.7 (Westwick [101])** Given the non-linear mapping (2.79) where \( f(\cdot) \) has at least one odd non-linear term (i.e. \( \Psi_q \) is nonzero for at least one odd \( q \)). Let \( u(k) \) and \( z(k) \) be the input and output of a Wiener system described by (2.78a)-(2.78c). Then the following holds:

\[
E\{z(k)u^T(\tau)\} = D_f E\{y(k)u^T(\tau)\}
\]

(2.82)

where \( D_f \) is a constant matrix that only depends on the non-linear mapping \( f(\cdot) \) and the covariance of \( y(k) \).

In words, the theorem states that the covariance between the non-linear output \( z(k) \) and the input \( u(k) \) is equal to the covariance of the linear output \( y(k) \) and the input \( u(k) \) up to a constant scaling matrix.

**Proof:**

The \( i \)-th element of \( z(k) \) can be written as

\[
z_i(k) = \sum_{q=1}^{\infty} \sum_{D_q} \psi_{qij} y_1^{q_1}(k)y_2^{q_2}(k) \cdots y_{\ell}^{q_{\ell}}(k)
\]

(2.83)

with \( \psi_{qij} \) the appropriate entry of \( \Psi_q \) and \( D_q \) the set of all combinations of \( \ell \) natural numbers that add up to \( q \):

\[
D_q = \{\{q_1, q_2, \cdots, q_{\ell}\} \in \mathbb{N}^\ell | q_1 + q_2 + \cdots q_{\ell} = q\}
\]

(2.84)

The expectation \( E\{z_i(k)u^T(\tau)\} \) can now be written as

\[
E\{z_i(k)u^T(\tau)\} = \sum_{q=1}^{\infty} \sum_{D_q} \psi_{qij} E\{y_1^{q_1}(k)y_2^{q_2}(k) \cdots y_{\ell}^{q_{\ell}}(k)u^T(\tau)\}
\]
Using lemma 2.3 we have

\[
\mathbf{E}\{y_1^{q_1}(k)y_2^{q_2}(k)\cdots y_\ell^{q_\ell}(k)u^T(\tau)\} = \\
q_1\mathbf{E}\{y_1^{q_1-1}(k)y_2^{q_2}(k)\cdots y_\ell^{q_\ell}(k)\} \mathbf{E}\{y_1(k)u^T(\tau)\} \\
+ q_2\mathbf{E}\{y_1^{q_1}(k)y_2^{q_2-1}(k)\cdots y_\ell^{q_\ell}(k)\} \mathbf{E}\{y_2(k)u^T(\tau)\} \\
+ \cdots \\
+ q_\ell\mathbf{E}\{y_1^{q_1}(k)y_2^{q_2}(k)\cdots y_\ell^{q_\ell-1}(k)\} \mathbf{E}\{y_\ell(k)u^T(\tau)\} \\
0 q \text{ odd} \\
0 q \text{ even}
\]

Therefore, when the non-linear mapping \(f(\cdot)\) contains at least one odd term, we can write the output \(z_i(k)\) as a linear combination of the expectations \(\mathbf{E}\{y_i(k)u^T(\tau)\}\) \(i = 1, 2, \ldots, \ell\). In matrix form this reads as

\[
\mathbf{E}\{z(k)u^T(\tau)\} = D_f \mathbf{E}\{y(k)u^T(\tau)\}
\] (2.85)

2.10.2 WIE-MOESP with even non-linearities

In the case where no odd non-linear term is present in the function \(f(\cdot)\), the above scheme fails, because the covariance \(\mathbf{E}\{z(k)u^T(\tau)\}\) is zero. In this case we have to resort to a different scheme. The following theorem helps us to find the linear system in the case where the non-linearity is an even function.

Theorem 2.8 Let \(u(k)\) and \(z(k)\) be the input and output of a Wiener system described by (2.78a)-(2.78c) Let the non-linear function \(f(\cdot)\) contain at least one even term. Define

\[
\hat{z}_i(k) = w_i(k)(z_i(k) - \mathbf{E}\{z_i(k)\}) \quad i = 1, 2, \ldots, \ell
\] (2.86)

where \(w_i(k)\) is a linear function of the input. Then the following holds:

\[
\mathbf{E}\{\hat{z}(k)u^T(\tau)\} = D_f \mathbf{E}\{y(k)u^T(\tau)\}
\] (2.87)

where \(D_f\) is a constant matrix that only depends on the non-linear mapping \(f(\cdot)\) and the covariance of \(y(k)\) and \(w(k)\). □

Proof:

\[
\mathbf{E}\{\hat{z}_i(k)u^T(\tau)\} = \mathbf{E}\{w(k)_iz_i(k)u^T(\tau)\} - \mathbf{E}\{z_i(k)\} \mathbf{E}\{w_i(k)u^T(k)\}
\] (2.88)
Since \( u(k) \) is zero-mean Gaussian, so are \( y(k) \) and \( w(k) \). Therefore, for odd powers of \( q \), these are two products involving an odd number Gaussian variables. Thus, this term is zero according to lemma 2.3.

For even values of \( q \), the first term can be written as

\[
\mathbb{E}\{w_i(k)z_i(k)u^T(\tau)\} = \mathbb{E}\{w_i(k)\left(\sum_{q=1}^{\infty} \sum_{q_i} \sum_{q_{i-1}} \cdots \sum_{q_1} y_1^{q_1}(k)y_2^{q_2}(k)\cdots y_l^{q_l}(k)\right) u^T(k)\}
\]

\[
= \sum_{q=1}^{\infty} \sum_{q_i} \mathbb{E}\{w_i(k)y_1^{q_1}(k)y_2^{q_2}(k)\cdots y_l^{q_l}(k)u^T(k)\}
\]

for any value of \( q \) a single term in this summation becomes

\[
\mathbb{E}\{w_i(k)y_1^{q_1}(k)y_2^{q_2}(k)\cdots y_l^{q_l}(k)u^T(\tau)\} = \left\{
\begin{array}{l}
0 \quad \text{if } q \text{ odd}, \\
q_1 \mathbb{E}\{w_i(k)y_1^{q_1-1}(k)y_2^{q_2}(k)\cdots y_l^{q_l}(k)\} \mathbb{E}\{y_1(k)u^T(\tau)\} \\
+ q_2 \mathbb{E}\{w_i(k)y_1^{q_1}(k)y_2^{q_2-1}(k)\cdots y_l^{q_l}(k)\} \mathbb{E}\{y_2(k)u^T(\tau)\} \\
+ \cdots \\
+ q_l \mathbb{E}\{w_i(k)y_1^{q_1}(k)y_2^{q_2}(k)\cdots y_l^{q_l-1}(k)\} \mathbb{E}\{y_l(k)u^T(\tau)\}
\end{array}
\right.
\]

This holds for all elements of \( \tilde{z}(k) \) and thus, if the non-linearity contains at least one even term, we can write

\[
\mathbb{E}\{\tilde{z}(k)u^T(\tau)\} = D_f \mathbb{E}\{y(k)u^T(\tau)\}
\]  

(2.89)

The choice of the multiplier \( w(k) \) depends on the linear system we try to identify. We seek a multiplier that will maximize the correlation between \( w(k) \) and \( y(k) \). Clearly \( y(k) \) itself would be a good choice for \( w(k) \), if we had access to it. A possible choice of \( w(k) \) is the delayed input at \( u(k - \delta) \) or a linear combination of different delayed inputs. With this choice for \( w(k) \) we can get an initial estimate of the system. Given this initial estimate of the system we can then use the estimated \( \hat{y}(k) \) as multiplier in a second estimation.

### 2.10.3 Estimating \( A_T \) and \( D_f C_T \)

Theorem 2.7 can be used to directly extend the PI-MOESP algorithm to the case of Wiener models with odd non-linearity. The data equation for the Wiener model is given by

\[
Z_{i,j,N} = F(Y_{i,j,N}) + V_{i,j,N}
\]

\[
= F(\Gamma_j X_{i,N} + H_i U_{i,j,N}) + V_{i,j,N}
\]
Here the function $F(\cdot)$ is a large matrix function that applies $f(\cdot)$ on every element of the matrix argument.

Theorem 2.7 now allows us to use the PI-MOESP algorithm to solve the Wiener model problem. Since the static non-linearity does not alter the correlation between $y(k)$ and $u(k)$, except for a constant factor, we can use the same LQ factorization as we did in the output-error case:

$$
\begin{bmatrix}
  U_{i,j,N} \\
  U_{0,i,N} \\
  Z_{i,j,N}
\end{bmatrix}
= \begin{bmatrix}
  L_{11} & 0 & 0 \\
  L_{21} & L_{22} & 0 \\
  L_{31} & L_{32} & L_{33}
\end{bmatrix}
\begin{bmatrix}
  Q_1 \\
  Q_2 \\
  Q_3
\end{bmatrix}
\quad (2.90)
$$

**Theorem 2.9** Given the state space model (2.78a) - (2.78c) with $u(k)$ PE of at least order $n + i + j$. Construct the Hankel matrices $U_{i,j,N}, Z_{i,j,N}$ and $U_{0,i,N}$ according to definition (2.7) and $N = N_m - i - j + 1$. Then with the LQ factorization (2.90) we have the following result:

$$
\lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{32} = \lim_{N \to \infty} \frac{1}{\sqrt{N}} \kappa \Gamma_i X_{i,N} Q_2^T
$$

where

$$
\kappa = I_i \otimes D_f
\quad (2.91)
$$

**Proof:**

From the LQ factorization we obtain

$$
\lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{32} = \lim_{N \to \infty} \frac{1}{\sqrt{N}} Z_{i,j,N} Q_2^T
$$

Due to the PE condition on the input $u(k)$ the matrix $\begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}$ is nonsingular.

From the data equation (2.20) we have

$$
\lim_{N \to \infty} \frac{1}{\sqrt{N}} Z_{i,j,N} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}^T
= \lim_{N \to \infty} \frac{1}{\sqrt{N}} Z_{i,j,N} \begin{bmatrix} U_{i,j,N} \\ U_{0,i,N} \end{bmatrix}^T \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}^{-T}
= \lim_{N \to \infty} \frac{1}{\sqrt{N}} \kappa \Gamma_i Y_{i,j,N} \begin{bmatrix} U_{i,j,N} \\ U_{0,i,N} \end{bmatrix}^T \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}^{-T}
= \lim_{N \to \infty} \frac{1}{\sqrt{N}} \kappa \Gamma_i X_{i,N} + H_i U_{i,j,N} \begin{bmatrix} U_{i,j,N} \\ U_{0,i,N} \end{bmatrix}^T \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}^{-T}
= \lim_{N \to \infty} \frac{1}{\sqrt{N}} \kappa \Gamma_i X_{i,N} + H_i L_{11} Q_1 \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}^T
$$
And by orthogonality between $Q_1$ and $Q_2$ we obtain

$$
\lim_{N \to \infty} \frac{1}{\sqrt{N}} Z_{i,j,N} Q_2^T = \lim_{N \to \infty} \frac{1}{\sqrt{N}} \kappa \Gamma_i X_{i,N} Q_2^T
$$

From $L_{32}$ we can now, using the singular value decomposition, get an estimate of the column-space:

$$
\kappa \Gamma_i = \begin{bmatrix}
D_f C \\
D_f CA \\
\vdots \\
D_f CA_{i-1}
\end{bmatrix}
$$

From this column-space, we can then find the combination $D_f C_T$ and $A_T$ in the same way as we did in previous subspace algorithms. Algorithm 2.8 summarizes the steps taken.

**Algorithm 2.8 weie-moesp (odd non-linearity)**

1. Build the matrices $U_{i,j,N}$, $Z_{i,j,N}$ and $U_{0,i,N}$ according to equation (2.7) with $N = N_m - i - j + 1$.

2. Perform the LQ factorization equation (2.90).

3. Compute the SVD of the matrix $L_{32}$:

$$
L_{32} = U \Sigma V^T
$$

We can now estimate the order $n$ from the gap in the singular values.

4. From the first $n$ left singular vectors of $U$, denoted by $U_n$ we can estimate the matrices $A_T$ and $D_f C_T$. Take $U_1$ as the upper $(i - 1)\ell$ rows of $U_n$ and $U_2$ the lower $(i - 1)\ell$ rows of $U_n$. Compute $\hat{A}_T$ and $\hat{D}_f C_T$ as follows

$$
\begin{align*}
D_f \hat{C}_T &= \text{the upper } \ell \text{ rows of } U_n \\
\hat{A}_T &= U_1^T U_2
\end{align*}
$$

In the case of an even non-linearity, the PI-MOESP algorithm needs to be slightly modified. In this case the instrumental variables have to be constructed from $\tilde{z}(k)$ as in theorem 2.8. Algorithm 2.9 shows the steps for this case.
Algorithm 2.9 WIE-MOESP (even non-linearity)

1. Construct the signal \( w(k) \) from past values of the input and the signal \( \tilde{z}(k) \) using equation (2.86).

2. Build the matrices \( U_{i,j,N}, \hat{Z}_{i,j,N} \) and \( U_{0,i,N} \) with \( N = N_m - i - j + 1 \) according to equation (2.7).

3. Perform the LQ factorization

\[
\begin{bmatrix}
U_{i,j,N} \\
U_{0,i,N} \\
\hat{Z}_{i,j,N}
\end{bmatrix} =
\begin{bmatrix}
L_{11} & 0 & 0 \\
L_{21} & L_{22} & 0 \\
L_{31} & L_{32} & L_{33}
\end{bmatrix}
\begin{bmatrix}
Q_1 \\
Q_2 \\
Q_3
\end{bmatrix}
\]  \hspace{1cm} (2.93)

4. Compute the SVD of the matrix \( L_{32} \):

\[
L_{32} = U \Sigma V^T
\]

We can now estimate the order \( n \) from the gap in the singular values.

5. From the first \( n \) left singular vectors of \( U \), denoted by \( U_n \) we can estimate the matrices \( A_T \) and \( D_f C_T \). Take \( U_1 \) as the upper \( (i-1)\ell \) rows of \( U_n \) and \( U_2 \) the lower \( (i-1)\ell \) rows of \( U_n \). Compute \( \hat{A}_T \) and \( \hat{D}_f C_T \) as follows

\[
\hat{C}_T = \text{the upper } \ell \text{ rows of } U_n
\]
\[
\hat{A}_T = U_1^\dagger U_2
\]
2.10.4 Estimating $B_T$ and $D_f D_T$

The next step is to find the matrix $B_T$ and the combination $D_f D_T$. In this section we shall show that the method used in the output-error case can also be used in the Wiener model case. For this, we use the notation of equation (2.71), without the noise contribution

$$
y(k) = CA^{k-1}x(0) + \left[ \sum_{\tau=1}^{k-1} u(\tau)^T \otimes CA^{k-1-\tau} \right] \operatorname{vec}(B) + [u(k)^T \otimes I_t] \operatorname{vec}(D)
$$

(2.94)

However, in the Wiener model identification we can measure nor reconstruct $y(k)$, since there exist a non-observable scaling $D_f$. Therefore we have to look at the following equation.

$$
D_f y(k) = D_f CA^{k-1}x(0) + \left[ \sum_{\tau=1}^{k-1} u(\tau)^T \otimes D_f CA^{k-1-\tau} \right] \operatorname{vec}(B) + [u(k)^T \otimes I_t] \operatorname{vec}(D_f D)
$$

For $k \to \infty$ the influence of the initial state $x(0)$ is negligible and we can disregard this term. If the signal $D_f y(k)$ was known we could solve for $B_T$ and $D_f D_T$ as follows

$$
\begin{bmatrix}
\operatorname{vec}(B) \\
\operatorname{vec}(D_f D_T)
\end{bmatrix} = [\mathcal{V} \ U]^{+} Y_{D_f}
$$

(2.95)

with $\mathcal{V}$ and $U$ constructed using (2.73) with $C$ and $A$ replaced by the estimates $\hat{A}_T$ and $D_f \hat{C}_T$. $Y_{D_f}$ is given by

$$
Y_{D_f} =
\begin{bmatrix}
D_f y(0) \\
D_f y(1) \\
\vdots \\
D_f y(N-1)
\end{bmatrix}
$$

(2.96)

The signal $D_f y(k)$ is not known. However, using theorem 2.9 we have the following result:

$$
\lim_{N \to \infty} \frac{1}{N} \mathcal{V}^{T} Z_{0,N,1} = \lim_{N \to \infty} \frac{1}{N} \mathcal{V} Y_{D_f}
$$

$$
\lim_{N \to \infty} \frac{1}{N} U^{T} Z_{0,N,1} = \lim_{N \to \infty} \frac{1}{N} U Y_{D_f}
$$

This is easy to see, since both $U$ and $\mathcal{V}$ depend linearly on $u(k)$. With this result we can replace the matrix $Y_{D_f}$ in equation (2.95) with $Z_{0,N,1}$.

$$
\begin{bmatrix}
\operatorname{vec}(B) \\
\operatorname{vec}(D_f D_T)
\end{bmatrix} = [\mathcal{V} \ U]^{+} Z_{0,N,1}
$$

(2.97)
2.10.5 Estimating the non-linear function

Once the linear part of the Wiener model is estimated, we want to find the static non-linear function between $D_f y(k)$ and $z(k)$. Since $D_f y(k)$ is unknown, we have to work with the simulated value $D_f \hat{y}(k)$, using the estimated linear state space model.

The non-linear part can be modeled by a number of basis functions, such as Chebyshev polynomials, wavelets, sigmoids, radial basis functions etc. As an example we will take the Chebyshev polynomials. The output of the Wiener model can be written as

$$z(k)^T = \mathcal{M}(D_f \hat{y}(k)^T)\Theta$$

where $\mathcal{M}(x)$ consists of the Chebyshev polynomials of $y(k)$ up to order $n_n$, according to

$$\mathcal{M}(x) = \begin{bmatrix} T_1(x) & T_2(x) & \cdots & T_{n_n}(x) \end{bmatrix}$$

and the Chebyshev polynomials are given by

$$T_1(x) = 1$$
$$T_2(x) = x$$
$$T_n(x) = 2x \cdot T_{n-1}(x) - T_{n-2}(x)$$

where for vectors the multiplication has to be performed element-wise. $\Theta$ is a matrix of dimension $\ell n_n \times \ell$ consisting of weightings of the individual Chebyshev polynomials. When we define

$$Z_{cheb} = \begin{bmatrix} z(0)^T \\ z(1)^T \\ \vdots \\ z(N-1)^T \end{bmatrix} \quad \hat{Y}_{cheb} = \begin{bmatrix} [D_f \hat{y}(0)]^T \\ [D_f \hat{y}(1)]^T \\ \vdots \\ [D_f \hat{y}(N-1)]^T \end{bmatrix}$$

then we can estimate the matrix $\Theta$ as

$$\hat{\Theta} = \mathcal{M}(\hat{Y}_{cheb})^\dagger Z_{cheb} \quad (2.98)$$

2.11 Conclusion

In this chapter we have presented an overview and theoretical treatment of existing members of the MOESP family of subspace identification methods. We have shown how the basic MOESP algorithm is related to the realization algorithm of Kung. The MOESP algorithm solves the identification problems for the state space matrices $A$ and $C$ in the same way as Kung's algorithm, namely by selecting a subspace that equals the column-space which contains the extended observability matrix.
Pafnuty Lvovich Chebyshev (1821-1894) is largely remembered for his investigations in number theory. In 1847 Chebyshev was appointed to the University of St Petersburg in Russia. He became a foreign associate of the Institute de France in 1874 and also of the Royal Society. In 1845 Bertrand conjectured that there was always at least one prime between $n$ and $2n$ for $n > 3$. Chebyshev proved Bertrand's conjecture in 1850. Chebyshev wrote about many subjects, including probability theory, quadratic forms, orthogonal functions, the theory of integrals, the construction of maps and the calculation of geometric volumes.

We have described a number of different algorithms, that allow us to solve a range of linear identification problems. The ordinary MOESP algorithm, called OM-MOESP, solves the noise-free and white noise output-error identification problem. The past input MOESP (PI-MOESP) solves the output-error identification problem. The innovations model identification problem is solved by the past output MOESP (PO-MOESP). Finally, the errors-in-variables identification problem is solved by the errors-in-variables variant of MOESP (EIV-MOESP).

Next to the above linear identification problems, we also have shown how the PI-MOESP algorithm can be applied to the non-linear Wiener model identification problem.

For the above listed linear identification problems we have given a detailed analysis of the requirements that are posed on the input for a successful identification. This requirement, the order of persistence of excitation (PE) was first derived for PO-MOESP in [46] and is derived in this chapter for all the MOESP algorithms that were treated here.

The treatment of MOESP that is given in this chapter is the most comprehensive treatment given to date, to the author’s knowledge. It presents a description of all the main MOESP algorithms. However, it is not exhaustive. Some variants were not covered. Much research has been done, for instance, on the use of MOESP for non-linear models such as Hammerstein systems [94] and bilinear systems [28, 85]. In [36, 90] the identification of non-causal systems using MOESP was treated. The use of MOESP in a recursive manner was the topic of [61]. Analysis of the optimality of SMI algorithms was theoretically and practically analyzed in [7].

The basic structure of all MOESP algorithms is identical. This allows us to build a toolbox around the MOESP family of algorithms that allows a wide choice of model structures, yet remains consistent in its usage. The functions that were treated in this chapter, as well as most of the functions mentioned above are gathered in the SMI-toolbox which is treated in section 6.

Despite the work that is done, many problems still remain to be solved or explored in the field of SMI. Expressions for the optimal values for the block matrix parameters
i and j that are used in most MOESP algorithms are desirable, but thus far unknown. Estimation of the Kalman filter in the errors-in variables identification problem is a non-solved problem. Finally the extension of MOESP to different classes of identification problems, such as the identification of continuous time, time-varying, parameter varying and classes of non-linear models are pursued.

In the next chapter we will deal with the extension of MOESP to the class of continuous time state space models. The extension allows us to directly identify a continuous state space model from sampled data.
Chapter 3

Continuous time subspace identification

3.1 Introduction

In chapter 2 we have dealt with the identification of discrete time state space systems. We have looked at a number of different identification problems with different assumptions on the disturbances. For all these identification problems variants of the family of MOESP algorithms were described that identify the system consistently. In this chapter we will introduce a number of new members of the MOESP family. These algorithms allow the direct identification of continuous time state space models from sampled data. The data can be either equidistant or non-equidistant and might be contaminated with process and measurement noise.

The interest in continuous time systems is motivated by three major reasons. First of all, most physical phenomena are viewed in a continuous time setting. Discrete time models can be used to describe such systems, for instance, to simulate the system or to design a digital controller. However, in a number of practical applications the use of continuous time models is preferable. This is the case, for example, when one is looking for a model which represents the underlying continuous time physical system. The objective in this case is to find a model from which we can extract parameters that have a physical meaning such as time constants, reaction time, elasticity, mass, etc. These parameters are directly linked to the continuous time model. Discrete time models on the other hand, do not have the same interpretability as do the continuous time representations, in this case. While the parameters in the second order continuous time transfer function

\[ \frac{1}{ms^2 + bs + k} \]
can represent mass, elasticity and friction, the parameters in a discrete time model of the same process

\[
\frac{b_0 z + b_1}{a_0 z^2 + a_1 z + a_2}
\]

do not have a physical meaning. Moreover, additional parameters are introduced in the numerator by the sampling process. In areas such as astrophysics, economics or biophysics one is interested in the analysis of the physical system [9, 71, 54]. In these cases identification of continuous time models is advantageous.

The second reason for using continuous time identification is, that in some situations it is difficult to obtain equidistant sampled data. This problem arises in medicine, transport and traffic systems, astrophysics and other areas where the instant of a measurement is not under control of the experimenter. In these cases the continuous time model, which is independent of a sampling period, is more flexible than a discrete time model. A discrete linear time-invariant (LTI) model can not represent the system, since it is inherently based on a constant sampling period. The continuous time model, however, represents the system at every time instance. The measurements are basically just points on a continuous line, which do not need to be equidistantly spaced.

The third reason for our interest in continuous time systems is the identification of stiff systems. These are systems with time constants that are of different order of magnitude. A stiff system contains both slow and fast dynamics. Traditionally this requires the acquisition of a large amount of data. On the one hand we need to sample fast, to capture the fast dynamics. On the other hand we need to sample over a long period to capture the slow dynamics. Combined, this will lead to a large number of samples. The large amount of data leads to long calculation times, even with modern computer equipment. As we will show in this chapter, the proposed continuous time identification algorithm allows to sample for each time-constant separately. Since the algorithm does not require equidistantly sampled data, the sampling rates and measurement times can be adjusted during the experiment. This allows us to take every time constant or group of time constants into account separately. Because of this, we can significantly reduce the amount of data and the computation time that is needed for the identification of stiff systems.

Continuous time identification methods have received wide attention. In literature [82, 103] two main groups of continuous time identification methods are distinguished: the indirect and the direct methods. The indirect methods try to estimate continuous time parameters from either a model that is estimated in discrete time or from a non-parametric model, such as a step response or a frequency response [54, 83]. The other group consists of methods, that try to identify the continuous time model directly from input/output data. An example is the State Variable Filter (SVF) method.

For the indirect identification, it is common to identify a discrete time model from data. Subsequently this model is transformed to continuous time. However, the conversion of a discrete time model to a continuous time one is in general not trivial.
The relation between the continuous time Laplace operator $s$ and the discrete time $z$ operator is given as $s = \ln(z)/\Delta$, where $\Delta$ is the sampling period of the discrete time model. This relation provides the tool for these conversions. But this conversion involves taking a logarithm. This may yield complex arithmetics when poles with a negative real part are present in the discrete time model. These poles, which lie in the left half of the unit circle are well defined in the discrete time case. But it is not possible to give a physical meaning to them, since taking the logarithm of such a pole results in a complex pole without a complex conjugate in continuous time.

Further difficulties are encountered in the choice of the sampling time $\Delta$. A large sampling time leads to loss of information. A small sampling time on the other hand clusters the poles of the discrete time model near $z = 1$. This leads to numerical ill-conditioning [76].

Direct identification methods often approximate the differentiation operator by a discrete filter [78]. A parametric model is then fitted on these filtered signals. The SYF [64, 68, 72] methods are examples of this approach. The advantage of this approach is that the continuous time parameters are found directly. However, high order discrete time filters are necessary for the approximation of the differentiation operator. Scaling is often needed to prevent numerical problems with the filtered signals. The use of the differentiation operator, or its digital approximate, also amplifies the noise in higher frequency regions. Measures against this, such as noise filtering, have to be taken in order to arrive at a reliable estimate.

The Subspace Model Identification for continuous time models was first proposed by Moonen et.al. [66]. His approach uses linear filters of large order to obtain a continuous time data equation. The data equation that was discussed in chapter 2 gives a relationship between matrices constructed from input and output data and is the basis of the SMI identification. Moonen uses approximate integration rather than differentiation. This is more reliable from a numerical point of view. Extra band-pass filters can be used if a model in a limited frequency range is desired. Apart from the difficulty in selecting those high order filters the paper of Moonen does not treat the case where process noise is present.

In this chapter we introduce a different approach. We propose to replace differentiation as represented by the Laplace operator $s$ by the operator $w$. These operators are related via the bilinear relationship $w = \frac{s-a}{s+a}$ [36]. The new $w$-domain model can be estimated directly from data, using filtered signals. Afterwards the parameters of this model are translated back to the parameters of the ordinary continuous time domain model, using simple algebraic relations. The $w$ operator introduced here is an all-pass filter. This filter does not alter the frequency content of signals and only influences the phase. Additional scaling of the filtered signals is therefore not necessary.

As such, the proposed method has an indirect and a direct part. On one hand we identify the parameters of the continuous time model from data. This is the direct part, since the $w$-domain model is a continuous time representation. The obtained model however, is then transformed to the ordinary state space model. This is the
indirect part. The advantage over many other indirect methods is firstly, that the \( w \)
operator is more suitable for the identification of continuous time models than the
Laplace operator. Secondly, the transformation does not require taking the logarithm,
but consists of a simple algebraic relation. A third advantage of the use of the bilinear
operator \( w \) is the fact that it results in the use of Laguerre filters. The \( w \)-domain
model is estimated using a bank of Laguerre filters. The orthogonality of the Laguerre
filters enables us to cope with both process and measurement noise in a simple and
effective way, as will be shown in section 3.6.

The proposed method is similar to the one used by Johansson [48, 49]. In Johansson’s
method however a filter \( f(s) = \frac{s^a}{s+a} \) is used. The drawback of his approach is that the
repeated filtering, \( f^2(s), f^3(s), \ldots \) which is used in the algorithm results in a strong
attenuation of signals above the cut-off frequency of \( f(s) \).

The use of Laguerre filters for continuous time identification is not new[82]. It has,
for instance, been used in a “measures” based identification scheme. The use of “mea-
sures” is a typical feature of direct continuous time identification methods. A measure
is generated from input and output signals using a linear dynamic operation. The
“measures”, instead of the input and output signals are then used in the identifica-
tion method. Laguerre filters can be used to generate the “measures” from the input
and output signals. The input and output “measures” are then used to minimize the
parameters of a model, based on those measures, with respect to a certain criterion.
This method has strong similarities with a Fourier based method. Our method is
related to the approach based on measures, as it is also based on the generation of a
set of filtered versions of the input and output signals using a bank of Laguerre filters.
However, we do use the minimization of a cost-function to arrive at the model param-
eters. In our method, the system itself is transformed to a different form. Instead of
the difference operator, the allpass operator is used for the state space representation.
From this transformation of the model, the Laguerre filter arrives automatically. The
parameters of the model are found using the Subspace Model Identification technique.

The algorithms described in this chapter are a counterpart of the discrete time PO-
MOESP algorithm. These algorithms give consistent estimates of both the determinis-
tic part and the stochastic part of the continuous time innovations model. The use of
Laguerre filters plays a key role in the proof of this property. Furthermore, the pro-
posed algorithm can be used to identify models from non-equidistant sampled data.
This property can be used to our advantage for the identification of stiff systems as
is shown in an example at the end of this chapter.

The chapter is structured as follows. In section 3.2 we formally introduce the con-
tinuous time system. In section 3.3 we discuss how the continuous time disturbances
can be modeled. Next, in section 3.4 the Laguerre filter and the all-pass filter are
introduced. The first identification problem, the noise-free case, is treated in section
3.5. The second identification problem, the continuous time innovation problem, is
covered in section 3.6. Section 3.7 covers the identification of the \( B \) and \( D \) matrices
in the continuous time case. Section 3.8 deals with the identification of the stochastic
part of the innovation model and section 3.9 treats the identification of continuous
time Wiener models. Finally section 3.10 shows some applications of the continuous
time Subspace Model Identification algorithm on several identification problems.

3.2 Continuous time systems

In this section we introduce the structure of a mixed deterministic/stochastic con-
tinuous time system description. The description is very similar to the deterministic
continuous time description. Two disturbance signals are added to model the un-
certainty in the system. It is assumed that the disturbances \( w(t) \) and \( v(t) \) are not
measurable. In general they can have any type of coloring and even be determin-
istic signals. However, in chapter 2 where discrete time subspace identification was
discussed, we introduced white noise disturbances to model the uncertainty in the
signals \( u(k) \) and \( y(k) \). In the continuous time case we would like to do the same,
but the direct extension of the discrete time white noise definition to the continuous
time results in a non-physical noise signal. This is solved by the notion of Wiener
processes.

**Definition 3.1** The general mixed deterministic/stochastic continuous time system
with both process and measurement noise is given by the following equations:

\[
\begin{align*}
\dot{x}(t) &= Ax(t) + Bu(t) + w(t) \\
y(t) &= Cx(t) + Du(t) + v(t)
\end{align*}
\]

Here \( x \in \mathbb{R}^n \), \( u \in \mathbb{R}^m \) and \( y \in \mathbb{R}^l \). The system matrices \( A \), \( B \), \( C \) and \( D \) are of
appropriate dimensions. The signals \( w(t) \in \mathbb{R}^n \) and \( v(t) \in \mathbb{R}^l \) are respectively the
process and measurement noise.

The problems that are encountered when defining continuous time white noise, will
be elucidated in the next section. We will first define what we mean by continuous
time white noise. Then we will give an example of the difficulties encountered when
using the concept of continuous time white noise. Next we will give some possible
(approximate) solutions. We finish this section with a formal definition of the Wiener
process which we will use as continuous time disturbance model throughout the rest
of this thesis.

3.3 Continuous time disturbances

In this section we will introduce and discuss several ways to model continuous time
stochastic signals. The key problem in modeling a stochastic signal in continuous time
is the fact, that one searches for a stochastic signal which is uncorrelated at different
time instances, but with a bounded (and non-zero) variance and power.

In order to analyze properties of continuous time stochastic signals, we need to be
able to use derivatives, limits and integrals of the signals. For this we need to have a
definition of convergence and continuity of a stochastic signal. The following theorem
defines what we mean with continuity of a stochastic signal.

**Definition 3.2 ([3])** A second order stochastic process \( v(t) \) is said to be continuous
in the mean square at time \( t \) if

\[
\lim_{\Delta \to 0} \mathbb{E}\{(v(t + \Delta) - v(t))^2\} = 0
\]

(3.2)

There are many other possible definitions of continuity of stochastic signals, such as
"continuity with probability one". The definition of continuity in the means square
is, however, a suitable tool because it often leads to very simple analysis [3].

For our disturbance model we would like to have a continuous time white noise signal
\( v(t) \) that has similar correlation properties as the discrete time white noise, yet is
continuous in time (in some sense). Therefore we search for a signal that combines
the properties listed in the following definition:

**Definition 3.3 (Continuous time white noise) [3]**. A continuous time signal \( v(t) \)
is called continuous time zero mean white noise if it has the following three properties:

1. \( v(t) \) is continuous in the mean square sense and has bounded variance.
2. \( v(t) \) is independent from \( v(s) \) for \( t \neq s \).
3. \( v(t) \) has zero mean.

However, a signal with the above properties does not exist. This is shown in the
following lemma.

**Lemma 3.1** A continuous time white noise signal according to definition 3.3 has
variance \( \mathbb{E}\{v(t)^2\} = 0 \).

**Proof:**
Let \( R_v(\tau) \) be the covariance function of \( v(t) \). Because of requirements 1 and 2 on \( v(t) \)
we have

\[
\mathbb{E}\{v(t)v(t - \tau)\} = R_v(\tau) = Q\delta_\tau(\tau)
\]
with $\delta_s(\tau)$ the selector function defined by

$$
\delta_s(\tau) = \begin{cases} 
0 & \tau \neq 0 \\
1 & \tau = 0
\end{cases}
$$

Then the power spectral density (PSD) is given by

$$
S_v(\omega) = \int_{-\infty}^{\infty} e^{-j\omega \tau} R_v(\tau) d\tau = 0
$$

This means that $v(t)$ has zero power at every frequency. Therefore the variance variance $Q$ is defined by

$$
Q = R_v(0) = \int_{-\infty}^{\infty} S_v(\omega) d\omega = 0
$$

which means that $v(t)$ must be zero for all $t$.

We see that in order for the PSD $S_v(\omega)$ to be non-zero, the covariance $R_v(\tau)$ must be equal to the $Q\delta(\tau)$. Here $\delta(\tau)$ is the Dirac delta. With this definition we have

$$
S_v(\omega) = \int_{-\infty}^{\infty} e^{-j\omega \tau} R_v(\tau) d\tau \int_{-\infty}^{\infty} e^{-j\omega \tau} Q\delta(\tau) d\tau = Q
$$

But this would violate requirement 1 since the covariance is infinite.

### 3.3.1 Semi-white noise

Different methods exist to circumvent the problem with infinite variance of the continuous time white noise. Most methods allow the signal to be not completely white, but to have a certain covariance function. In this way it is possible to define a semi-white noise signal $v(t)$ which has finite variance. We shall give two examples of such semi-white noise signals which are frequently used in literature.

The first possible definition is band-limited white noise. The spectrum of the noise signal is hard limited at a certain frequency $\omega_0$ that is preferably much higher than the dynamics of the model. The PSD of a band-limited white noise signal is given by

$$
S_v(\omega) = \begin{cases} 
Q & |\omega| < \omega_0 \\
0 & \omega \geq \omega_0
\end{cases}
$$

(3.3)

With this definition of the power spectral density, the covariance function becomes

$$
R_v(\tau) = 2\omega_0 QS\text{sinc}(\omega_0 \tau)
$$

(3.4)

Figure 3.1 shows the PSD and the covariance function of a semi-white noise signal of this type.
Figure 3.1: Power spectral density (left) and covariance function (right) of band-limited continuous time white noise.

With the above definition of the covariance function, the signal $v(t)$ is uncorrelated at intervals $\frac{\pi}{\omega_0}$. Thus when sampling this signal with a sampling rate $\frac{\omega_0}{\pi}$ we obtain a discrete time white noise sequence. Moreover, when we let $\omega_0$ tend to infinity, the covariance $R_v(\tau)$ will tend to the Dirac delta function $\delta(\tau)$ and the variance $R_v(0) = 2\omega_0Q$ to infinity. This scheme is used for instance in [47].

Paul Dirac (1902-1984) is famous as the creator of the complete theoretical formulation of quantum mechanics. He studied electrical engineering at the University of Bristol before doing research in mathematics at St John’s College Cambridge. His first major contribution to quantum theory was a paper written in 1925. He published “The principles of Quantum Mechanics” in 1930 and for this work he was awarded the Nobel Prize for Physics in 1933. Dirac was appointed Lucasian professor of mathematics at the University of Cambridge in 1932, a post he held for 37 years.

A second method that is used, is low-pass filtered continuous time white noise. The PSD of low-pass filtered continuous time white noise is given by

$$S_v(\omega) = Q\frac{\omega_0^2}{\omega^2 + \omega_0^2}$$

(3.5)

This function has the following covariance function

$$R_v(\tau) = \frac{\omega_0Q}{2} Q e^{-\omega_0|\tau|}$$

(3.6)

Figure 3.2 shows the power spectral density and the covariance function of low-pass filtered continuous time white noise. As with band-limited white noise, we have that for $\omega_0 \to \infty$ the covariance function becomes a Dirac pulse.
3.3 Continuous time disturbances

Figure 3.2: Power spectral density (left) and covariance function (right) of low-pass filtered continuous time white noise

3.3.2 Wiener processes

The examples of semi-white noise given above, show that it is possible to define continuous time disturbance signals when we loosen the requirements posed on these signals in definition 3.3. However the semi-white noise signals can not have a Dirac pulse as covariance function, since this results in an infinite variance. We will now introduce the Wiener process which is a solution to this problem. The Wiener process is also called the Brownian motion process.

Robert Brown (1773-1858) is acknowledged as Britain's greatest botanist, although his name is commemorated only in a basic natural phenomenon. Brown studied medicine and joined the army as a surgeon in 1795. He joined Flinders's voyage, which surveyed the coast of Australia from 1802 until 1803. He published the results of his collecting in his famous "Prodromus Florae Novae Hollandiae" in 1810. During experiments in 1827 Robert Brown noticed that small particles (with diameters in the order of 0.001mm) submerged in a fluid, have a very irregular motion. Only in 1905 Einstein showed that the motion could be explained by assuming that it was caused by collisions with molecules of the fluid. A rigorous mathematical analysis of the process was given by Wiener in 1923.

Before defining the Wiener process we first have to define the concept of orthogonal stationary increments and independent stationary increments. We will derive expressions for the mean, covariance and power spectral density of the filtered Wiener process.

Definition 3.4 (Orthogonal stationary increments) [3] The stochastic process
\( v(t), \ t \in T \) has orthogonal increments if for all \( t_i \in T \) with \( t_1 < t_2 < \cdots < t_k \)

\[
v(t_1), v(t_2) - v(t_1), \ldots, v(t_k) - v(t_{k-1})
\]

are mutually uncorrelated. If the distribution of \( v(t_k) - v(t_{k-1}) \) depends only on \( t_k - t_{k-1} \), the process is said to have stationary increments. If the increments \( v(t_k) - v(t_{k-1}) \) are independent, \( v(t) \) is said to have independent increments.

For a stochastic process \( v(t) \) with orthogonal increments we define the following measures:

The incremental mean is defined by

\[
dm_v(\tau) = E\{dv(\tau)\} \tag{3.7}
\]

The incremental covariance is given by

\[
dR_v(\tau) = E\{dv(\tau), dv^T(\tau)\} \tag{3.8}
\]

For a process with stationary increments, the increments \( dm_v(\tau) \) and \( dR_v(\tau) \) are independent from the time instance \( \tau \) and only depend on the increment \( d\tau \). Therefore they can be written as

\[
\begin{align*}
\ dm_v(\tau) &= m_v d\tau \\
\ dR_v(\tau) &= R_v d\tau 
\end{align*}
\]

After we have given the definition of independent stationary increments the Wiener process is defined as follows:

**Definition 3.5 (Wiener process)** [3] A Wiener process \( v(t) \) is a stochastic signal with the following properties.

- \( v(0) = 0 \)
- \( v(t) \) is normal
- \( E\{v(t)\} = 0 \ \forall \ t \geq 0 \)
- \( v(t) \) has independent stationary increments

**Theorem 3.1** [3] Let \( v(t) \) be a Wiener process. Its incremental mean is \( m_v d\tau \) and its incremental covariance is \( R_v d\tau \). Let \( y(t) \) be the output of a stable filter with transfer function \( H(s) \) and impulse response function \( h(\tau) \) where \( v(t) \) is the input.

\[
y(t) = \int_{-\infty}^{t} h(t - \tau)dv(\tau)
\]
Then the expected value of \( y(t) \) equals

\[
m_y(\tau) = \mathbb{E}\{y(\tau)\} = \int_{-\infty}^{t} h(t-\tau)m_v d\tau
\]

The covariance function of \( y(t) \) equals

\[
R_y(\tau) = \mathbb{E}\{y(t)y^T(t-\tau)\} = \int_{0}^{\infty} h(\tau)R_v h^T(\nu-\tau) d\nu
\]

The spectral density of \( y(t) \) equals

\[
S_y(\omega) = H(\omega)R_v H^T(-\omega)
\]

**Proof:**

For the incremental mean of \( y(t) \) we can write

\[
m_y(\tau) = \mathbb{E}\{y(\tau)\} = \int_{-\infty}^{t} h(t-\tau) E\{d\nu(\tau)\}
\]

\[
= \int_{-\infty}^{t} h(t-\tau) E\{d\nu(\tau)\}
\]

\[
= \int_{-\infty}^{t} h(t-\tau) m_v(\tau)
\]

\[
= \int_{-\infty}^{t} h(t-\tau) m_v d\tau
\]

Similarly for the covariance function of \( y(t) \):

\[
R_y(\tau) = \mathbb{E}\{y(t)y^T(t-\tau)\}
\]

\[
= \int_{-\infty}^{t} \int_{-\infty}^{t-\tau} h(t-t') E\{d\nu(t')d\nu(T')\} h^T(t-\tau) d\nu
\]

\[
= \int_{-\infty}^{t} \int_{-\infty}^{t-\tau} h(t-t') E\{d\nu(t')d\nu(T')\} h^T(t-\tau) d\nu
\]

\[
= \int_{-\infty}^{t} h(t-t') R_v \delta(t'-\tau') h^T(t-\tau-\tau') dt' d\tau'
\]

\[
= \int_{-\infty}^{t} h(t-t') R_v h^T(t-\tau-t') dt'
\]

\[
= \int_{0}^{\infty} h(\nu) R_v h^T(\nu-\tau) d\nu
\]
Using the expression for the covariance function we can write the spectral density of \( y(t) \) as

\[
S_y(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-j\omega\tau} R_y d\tau \\
= \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-j\omega\tau} \int_{0}^{\infty} h(\tau + \nu)R_y h^T(\nu) d\nu d\tau \\
= \frac{1}{2\pi} \int_{0}^{\infty} \int_{-\infty}^{\infty} e^{-j\omega(\tau + \nu)} h(\tau + \nu)R_y h^T(\nu) e^{j\omega\nu} d\tau d\nu \\
= H(\omega)R_y H^T(-\omega)
\]

Using the concept of Wiener processes we define the following deterministic stochastic state space model description. The increment in the state \( dx \) consists of three parts: the present state, the contribution of the input and a random part. The disturbance part is modeled by a Wiener process.

A general continuous time system with both process and measurement noise is then given by the following deterministic/stochastic differential equations:

\[
dx(t) &= Ax(t)dt + Bu(t)dt + dw(t) \\
dz(t) &= Cx(t)dt + Du(t)dt + dv(t)
\]  

(3.9a)  

(3.9b)  

(3.9c)

The noise \( w(t) \in \mathbb{R}^n \) and \( v(t) \in \mathbb{R}^l \) are Wiener processes. The incremental covariance of the noise is given by

\[
E\{ \begin{bmatrix} dw(t) \\ dv(t) \end{bmatrix} \begin{bmatrix} dw(t) \\ dv(t) \end{bmatrix}^T \} = \begin{bmatrix} Q & S \\ ST & R \end{bmatrix} dt
\]

The above system description is related to the system description (3.1a)-(3.1b). By “dividing” system description (3.9a)-(3.9b) by \( dt \), the deterministic part reduces to the well-known ordinary continuous time state space equations.

\[
\dot{x}(t) = Ax(t) + Bu(t) + \frac{dw}{dt}(t) \\
y(t) = \dot{z}(t) = Cx(t) + Du(t) + \frac{dv}{dt}(t)
\]

With the noise signals present, this “division” is formally not possible. Strictly speaking these signals do not exist, since \( \frac{dw}{dt} \) and \( \frac{dv}{dt} \) are signals with infinite covariance.
This notation can however be used in some situations. $Q$ and $R$ should then be regarded as the spectral density of $\frac{du(t)}{dt}$ and $\frac{dv(t)}{dt}$ respectively. We will try to avoid the last notation and stick with the formal framework. In some cases however we can use the less formal description to simplify notation.

### 3.3.3 Discretizing stochastic differential equations

Before we proceed with the identification of continuous time systems, we introduce the theory behind sampling of continuous time systems. In the undisturbed case this is a trivial problem, but since the disturbances have infinite variance, we need to introduce a formal framework on sampling.

In order to obtain samples of the output $y(t)$ from the system (3.9a)-(3.9b), a pre-filter $h(\tau)$ has to be applied. When this pre-filter is strictly proper ($h(0) = 0$), the filtered output which will be denoted by $y_f(t)$, has finite variance.

$$y_f(t) = \int_{-\infty}^{\infty} h(t - \tau) dz(\tau)$$  \hspace{1cm} (3.10)

A commonly used sampling pre-filter is the integrate-and-reset pre-filter:

$$h(\tau) = \begin{cases} \frac{1}{\Delta} & 0 \leq \tau \leq \Delta \\ 0 & \text{otherwise} \end{cases}$$  \hspace{1cm} (3.11)

**Theorem 3.2** Let $y(t)$ be the output of the system described by the state space equations (3.9a) - (3.9b).

Let $y_f(t)$ be the output $y(t)$ pre-filtered by (3.11). Then the sampled $y_f(t)$ is described by the following discrete time state space description:

$$x_q(k + 1) = A_q x_q(k) + u_1(k) + w_q(k)$$  \hspace{1cm} (3.12a)

$$y_q(k) = C_q x_q(k) + u_2(k) + v_q(k)$$  \hspace{1cm} (3.12b)
\[ x_q(k) = x(k\Delta) \]  
(3.13)

\[ y_q(k) = y_f(k\Delta - \Delta) \]  
(3.14)

\[ u_1(k) = \int_{k\Delta}^{k\Delta + \Delta} e^{A(k\Delta + \Delta - \tau)} Bu(\tau) d\tau \]  
(3.15)

\[ w_2(k) = \frac{1}{\Delta} \int_{k\Delta}^{k\Delta + \Delta} \theta_1(\tau) Bu(\tau) d\tau + \frac{1}{\Delta} \int_{k\Delta}^{k\Delta + \Delta} Du(t) dt \]  
(3.16)

\[ w_q(k) = \int_{k\Delta}^{k\Delta + \Delta} e^{A(k\Delta + \Delta - \tau)} dw(\tau) \]  
(3.17)

\[ v_q(k) = \frac{1}{\Delta} \int_{k\Delta}^{k\Delta + \Delta} \theta_1(\tau) dw(\tau) + \frac{1}{\Delta} \int_{k\Delta}^{k\Delta + \Delta} dv(t) \]  
(3.18)

\[ A_q = e^{A\Delta} \]  
(3.19)

\[ C_q = \frac{1}{\Delta} C \int_0^\Delta e^{A\tau} d\tau \]  
(3.20)

with

\[ \theta_1(\tau) = C \int_0^{k\Delta + \Delta} e^{A(t-\tau)} dt \]  
(3.21)

The cross- and auto-covariances of \( w_q(k) \) and \( v_q(k) \) are given by

\[ \mathbb{E}\left\{ \begin{bmatrix} w_q(k) \\ v_q(k) \end{bmatrix} \right\} \begin{bmatrix} w_q(l) \\ v_q(l) \end{bmatrix}^T = \begin{bmatrix} Q_q & S_q \\ S_q^T & R_q \end{bmatrix} \delta_{kl} \]  
(3.22)

with

\[ Q_q = \mathbb{E}\{w_q(k)w_q^T(k)\} = \int_0^\Delta e^{A\tau} Q e^{A^\tau} d\tau \]  
(3.23)

\[ R_q = \mathbb{E}\{v_q(k)v_q^T(k)\} \]  

\[ = \frac{1}{\Delta^2} \int_0^\Delta \theta_2(\tau)Q\theta_2^T(\tau) d\tau + \frac{1}{\Delta^2} \int_0^\Delta \theta_2(\tau)S d\tau \]  
+ \frac{1}{\Delta^2} \int_0^\Delta S^T \theta_2^T(\tau) d\tau + \frac{R}{\Delta} \]  
(3.24)

\[ S_q = \mathbb{E}\{w_q(k)v_q^T(k)\} \]  

\[ = \frac{1}{\Delta} \int_0^\Delta e^{A(\Delta-\tau)}Q\theta_2^T(\tau) d\tau + \frac{1}{\Delta} \int_0^\Delta e^{A(\Delta-\tau)}S d\tau \]  
(3.25)

with

\[ \theta_2(\tau) = C \int_\tau^\Delta e^{A(t-\tau)} dt \]  
(3.26)
Notice that the pre-filtering introduces a delay of one sample; one obtains the output sample \( y_f(k\Delta) \) one sample instance later, at time \( k\Delta + \Delta \) as \( y_q(k) \).

**Proof:**

*From the state transition equation*

\[
x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A(t-\tau)} Bu(\tau) d\tau + \int_{t_0}^{t} e^{A(t-\tau)} dw(\tau)
\]

*substituting \( t_0 = k\Delta \) and \( t = k\Delta + \Delta \) we have*

\[
x(k\Delta + \Delta) = e^{A\Delta}x(k\Delta) + \int_{k\Delta}^{k\Delta+\Delta} e^{A(k\Delta+\Delta-\tau)} Bu(\tau) d\tau + \int_{k\Delta}^{k\Delta+\Delta} e^{A(k\Delta+\Delta-\tau)} dw(\tau)
\]

*The output equation gives*

\[
z(k\Delta + \Delta) = z(k\Delta) + \int_{k\Delta}^{k\Delta+\Delta} dz(t)
\]

\[
= z(k\Delta) + \int_{k\Delta}^{k\Delta+\Delta} \left(Cx(t)dt + Du(t)dt + dv(t)\right) dt
\]

\[
= z(k\Delta) + \int_{k\Delta}^{k\Delta+\Delta} Ce^{A(t-k\Delta)} dt \, x(k\Delta)
\]

\[
+ \int_{k\Delta}^{k\Delta+\Delta} C \int_{k\Delta}^{t} e^{A(t-\tau)} Bu(\tau) d\tau dt + \int_{k\Delta}^{k\Delta+\Delta} Du(t) dt
\]

\[
+ \int_{k\Delta}^{k\Delta+\Delta} C \int_{k\Delta}^{t} e^{A(t-\tau)} dw(\tau) dt + \int_{k\Delta}^{k\Delta+\Delta} dv(t)
\]

\[
= z(k\Delta) + \int_{k\Delta}^{k\Delta+\Delta} Ce^{A(t-k\Delta)} dt \, x(k\Delta)
\]

\[
+ \int_{k\Delta}^{k\Delta+\Delta} C \int_{k\Delta}^{k\Delta} e^{A(t-\tau)} dt Bu(\tau) d\tau + \int_{k\Delta}^{k\Delta+\Delta} Du(t) dt
\]

\[
+ \int_{k\Delta}^{k\Delta+\Delta} C \int_{k\Delta}^{k\Delta} e^{A(t-\tau)} dt dw(\tau) + \int_{k\Delta}^{k\Delta+\Delta} dv(t)
\]

\[
= z(k\Delta) + \int_{k\Delta}^{k\Delta+\Delta} Ce^{A(t-k\Delta)} dt \, x(k\Delta)
\]

\[
+ \int_{k\Delta}^{k\Delta+\Delta} \theta_1(\tau) Bu(\tau) d\tau + \int_{k\Delta}^{k\Delta+\Delta} Du(t) dt
\]

\[
+ \int_{k\Delta}^{k\Delta+\Delta} \theta_1(\tau) dw(\tau) + \int_{k\Delta}^{k\Delta+\Delta} dv(t)
\]
Applying the pre-filter to $y(t)$ we obtain

$$
y_f(k\Delta + \Delta) = \frac{1}{\Delta} \int_{k\Delta}^{k\Delta + \Delta} h(k\Delta + \Delta - \tau) dz(\tau)
= \frac{z(k\Delta + \Delta) - z(k\Delta)}{\Delta}
= \frac{1}{\Delta} \int_{k\Delta}^{k\Delta + \Delta} C e^{A(\tau - k\Delta)} d\tau \ x(k\Delta)
+ \frac{1}{\Delta} \int_{k\Delta}^{k\Delta + \Delta} \theta_1(\tau) Bu(\tau) d\tau + \frac{1}{\Delta} \int_{k\Delta}^{k\Delta + \Delta} Du(t) dt
+ \frac{1}{\Delta} \int_{k\Delta}^{k\Delta + \Delta} \theta_1(\tau) dw(\tau) + \frac{1}{\Delta} \int_{k\Delta}^{k\Delta + \Delta} dv(t)
$$

Finally by defining $y_q(k\Delta) = y_f(k\Delta + \Delta)$ gives the discrete time system (3.12a) - (3.12b).

The covariance matrices equals

$$
Q_q = \mathbb{E}\{w_q(k)w_q^T(k)\}
= \mathbb{E}\left\{ \left( \int_{k\Delta}^{k\Delta + \Delta} e^{A(k\Delta + \Delta - \tau)} dw(\tau) \right) \left( \int_{k\Delta}^{k\Delta + \Delta} e^{A(k\Delta + \Delta - \tau)} dw(\tau) \right)^T \right\}
= \int_{k\Delta}^{k\Delta + \Delta} e^{A(k\Delta + \Delta - \tau)} Q e^{T(k\Delta + \Delta - \tau)} d\tau
= \int_0^\Delta e^{A\tau} Q e^{A^T\tau} d\tau
$$

$$
R_q = \mathbb{E}\{v_q(k)v_q^T(k)\}
= \mathbb{E}\left\{ \frac{1}{\Delta} \left( \int_{k\Delta}^{k\Delta + \Delta} \theta_1(\tau) dw(\tau) + \frac{1}{\Delta} \int_{k\Delta}^{k\Delta + \Delta} dv(t) \right) \times \left( \frac{1}{\Delta} \int_{k\Delta}^{k\Delta + \Delta} \theta_1(\tau) dw(\tau) + \frac{1}{\Delta} \int_{k\Delta}^{k\Delta + \Delta} dv(t) \right)^T \right\}
= \frac{1}{\Delta^2} \int_{k\Delta}^{k\Delta + \Delta} \theta_1(\tau) Q \theta_1^T(\tau) d\tau + \frac{1}{\Delta^2} \int_{k\Delta}^{k\Delta + \Delta} \theta_1(\tau) S d\tau
+ \frac{1}{\Delta^2} \int_{k\Delta}^{k\Delta + \Delta} S^T \theta_1^T(\tau) d\tau + \frac{1}{\Delta^2} \int_{k\Delta}^{k\Delta + \Delta} R d\tau
= \frac{1}{\Delta^2} \int_0^\Delta \left( \theta_2(\tau) Q \theta_2^T(\tau) + \theta_2(\tau) S + S^T \theta_2^T(\tau) + R \right) d\tau
$$
\[ S_q = \mathbf{E}\{ w_q(k)v_q^T(k) \} \]

\[ = \mathbf{E}\left\{ \left( \int_{k\Delta}^{k\Delta+\Delta} e^{A(k\Delta+\Delta-\tau)} dw(\tau) \right) \times \left( \frac{1}{\Delta} \int_{k\Delta}^{k\Delta+\Delta} \theta_1(\tau) dw(\tau) + \frac{1}{\Delta} \int_{k\Delta}^{k\Delta+\Delta} dv(l) \right)^T \right\} \]

\[ = \frac{1}{\Delta} \int_{k\Delta}^{k\Delta+\Delta} e^{A(k\Delta+\Delta-\tau)} Q\theta_1^T(\tau) d\tau + \frac{1}{\Delta} \int_{k\Delta}^{k\Delta+\Delta} e^{A(k\Delta+\Delta-\tau)} S d\tau \]

\[ = \frac{1}{\Delta} \int_{0}^{\Delta} \left( e^{A(\Delta-\tau)} Q\theta_2^T(\tau) + e^{A(\Delta-\tau)} S \right) d\tau \]

The covariance matrices in theorem 3.2 are expressed as integrals. For implementation purposes, we derive the following expressions for the matrices \( Q_q, S_q \) and \( R_q \).

**Corollary 3.1** Let \( Q_q, R_q \) and \( S_q \) be given by the equations (3.23), (3.24) and (3.25). Then \( Q_q \) can be solved from the Lyapunov equation

\[ e^{A\Delta}Qe^{A^T\Delta} - Q = QA_q + Q_A^T \]  

(3.27)

and \( R_q \) and \( S_q \) can be calculated as

\[ R_q = C A^{-1} (Q_q - Q e^{A^T\Delta} + Q - e^{A\Delta}Q + Q + \Delta Q) A^{-T} C^T + R \Delta \]

(3.28)

\[ + (S^T e^{A^T\Delta} - S^T - S^T \Delta) A^{-T} C^T + C A^{-1} (e^{A\Delta} S - S - S \Delta) \]

\[ S_q = Q_A A^{-T} C^T - A^{-1} e^{A\Delta} Q A^{-T} C^T + A^{-1} Q A^{-T} C^T + S \Delta \]

(3.29)

**Aleksandr Mikhailovich Lyapunov (1857-1918)** was a school friend of Markov and later a student of Chebyshev. He did important work on differential equations, potential theory, stability of systems and probability theory. His work concentrated on the stability of equilibrium and motion of a mechanical system and the stability of an uniformly rotating fluid. He devised important methods of approximation. Lyapunov's methods, introduced by him in 1899, provide ways of determining the stability of sets of ordinary differential equations.

**Proof:**

The matrix \( Q_q \) is given by

\[ Q_q = \int_{0}^{\Delta} e^{A(\Delta-\tau)} Q e^{A^T(\Delta-\tau)} d\tau = \int_{0}^{\Delta} e^{A\tau} Q e^{A^T(\tau)} d\tau \]
By using \( \int_0^\Delta f'(x)dx = f(X) - f(0) \), we get

\[
\int_0^\Delta \frac{d}{d\tau}(e^{A\tau}Qe^{A^T\tau})d\tau \\
= e^{A\Delta}Qe^{A^T\Delta} - Q \\
= \int_0^\Delta A e^{A\tau}Qe^{A^T\tau}d\tau + \int_0^\Delta e^{A\tau}Qe^{A^T\tau}A^Td\tau \\
= AQ_q + Q_qA^T
\]

We can therefore find the matrix \( Q_q \) by solving the Riccati equation

\[
e^{A\Delta}Qe^{A^T\Delta} - Q = AQ_q + Q_qA^T
\]

The matrix \( R_q \), given by

\[
R_q = \frac{1}{\Delta^2} \int_0^\Delta \left( \theta_2(\tau)Q\theta_2^T(\tau) + S^T\theta_2^T(\tau) + \theta_2(\tau)S + R \right)d\tau \\
= \frac{1}{\Delta^2} \int_0^\Delta CA^{-1}(e^{A\tau} - I)Q(e^{A\tau} - I)^TA^{-T}CTd\tau \\
+ \frac{1}{\Delta^2} \int_0^\Delta S^T(e^{A\tau} - I)^TA^{-T}CTd\tau \\
+ \frac{1}{\Delta^2} \int_0^\Delta CA^{-1}(e^{A\tau} - I)Sd\tau \\
+ \frac{1}{\Delta^2} \int_0^\Delta Rd\tau \\
= \frac{1}{\Delta^2} \int_0^\Delta CA^{-1}(e^{A\tau}Qe^{A^T\tau} - Qe^{A^T\tau} - e^{A\tau}Q + Q)A^{-T}CTd\tau \\
+ \frac{1}{\Delta^2} \int_0^\Delta ST e^{A^T\tau}A^{-T}CTd\tau + \frac{1}{\Delta^2} \int_0^\Delta ST A^{-T}CTd\tau \\
+ \frac{1}{\Delta^2} \int_0^\Delta CA^{-1}e^{A\tau}Sd\tau + \frac{1}{\Delta^2} \int_0^\Delta CA^{-1}Sd\tau \\
+ \frac{1}{\Delta^2} \int_0^\Delta Rd\tau
\]
\[ \begin{align*} 
&= \frac{1}{\Delta} \left( Q - Q A^{-T} (e^{A^T \Delta} - I) - (e^{A \Delta} - I) A^{-1} Q + \Delta Q \right) A^{-T} C^T \\
&+ \frac{1}{\Delta} S^T A^{-T} (e^{A^T \Delta} - I) A^{-T} C^T - \frac{1}{\Delta} S^T A^{-T} C^T \\
&+ \frac{1}{\Delta} C A^{-1} (e^{A \Delta} - I) A^{-1} S - \frac{1}{\Delta} C A^{-1} S \\
&+ \frac{1}{\Delta} R
\end{align*} \]

Finally, the matrix \( S_q \) can be calculated as

\[ S_q = \frac{1}{\Delta} \int_0^\Delta \left( e^{A(\Delta - \tau)} Q g^T \right) d\tau \\
= \frac{1}{\Delta} \int_0^\Delta \left( e^{A(\Delta - \tau)} Q (e^{A(\Delta - \tau)} - I)^T A^{-T} C^T + e^{A(\Delta - \tau)} S \right) d\tau \\
= \frac{1}{\Delta} \int_0^\Delta \left( (e^{A(\Delta - \tau)} Q e^{A(\Delta - \tau)} - e^{A(\Delta - \tau)} Q) A^{-T} C^T + e^{A(\Delta - \tau)} S \right) \\
= \frac{1}{\Delta} \left( Q - A^{-1} (e^{A \Delta} - I) Q \right) A^{-T} C^T + \frac{1}{\Delta} (e^{A \Delta} - I) A^{-1} S \]

When no measurement noise is present a different approach can be taken. In this case, there is no need to introduce a pre-filter, since the output is of finite variance. An instantaneous sampling scheme is now possible. In this case we can match the PSDs of the continuous time description by a discrete time state space model[77].

Let the continuous time system be described by the transfer function

\[ H(s) = \frac{B(s)}{A(s)} \]  
(3.30)

with \( \deg B < \deg A \). Assume that the input is white noise such that its PSD is constant \( (S_u = Q) \). Then the spectrum of the output of a continuous time system with is given by

\[ S_y(\omega) = \frac{B(j\omega)}{A(j\omega)} Q \frac{B(-j\omega)}{A(-j\omega)} \]  
(3.31)

The spectrum of a discrete time model is given by

\[ S_\tilde{y}(\omega) = \frac{B_d(e^{j\omega})}{A(e^{j\omega})} Q \frac{B(e^{-j\omega})}{A(e^{-j\omega})} \]  
(3.32)

By equating \( S_y(\omega) \) with \( S_\tilde{y}(\omega) \) we can find a discrete time model which produces the same output spectrum as the continuous one by spectral factorization.
3.4 Laguerre and all-pass filters

In this section we introduce the Laguerre filter and the first order all-pass filter that will be used in the continuous time MOESP algorithm. The Laguerre filter is widely used in system identification as a basis-function for both continuous time and discrete time systems [30, 97, 98]. The Laguerre filters are an orthogonal basis for the function space $L^2(\mathbb{R}^+^+)$. They have many properties that can be exploited in system identification. Some of these properties will be demonstrated in this section. The Laguerre filter has a close relation to the first order all-pass filter. A bank of Laguerre filters can be viewed as a single low-pass filter in combination with a bank of all-pass filters. It will be shown that the all-pass filter and the discrete time domain z-operator have certain similarities which can be exploited in the identification of continuous time systems.

Edmond Laguerre (1834-1886) attended the École Polytechnique in Paris but only ranked 46th in his class. He was an artillery officer from 1854 to 1864 when he returned to the École Polytechnique where he remained for the rest of his life. Laguerre studied approximation methods and is best remembered for the Laguerre polynomials which are solutions of the Laguerre differential equations.

3.4.1 The Laguerre filter

The transfer function of the $i$-th order continuous time Laguerre filter is given by

$$
\mathcal{L}_i(s) = \sqrt{2a} \frac{(s - a)^i}{(s + a)^{i+1}}
$$

(3.33)

For $a > 0$ this filter is stable and causal. For $a < 0$ the filter is unstable. We can however regard the unstable filter as an anti-causal filter which is used backwards in time. In this setting the filter is stable. In real-time this is of course not possible, but on a recorded data-set, anti-causal filters can be used.

The stable Laguerre filters constitute an orthonormal basis for the space $L^2(\mathbb{R}^+^+)$ [63]. Therefore the following holds

**Lemma 3.2** Let $\ell_q(\tau)$ and $\ell_r(\tau)$ be the Impulse response functions (IRF)s of the $q$-th and $r$-th order Laguerre filter respectively then

$$
\int_0^\infty \ell_q(\tau) \ell_r(\tau) d\tau = \delta_{qr}
$$
Lemma 3.3 ([63]) Every function $h(t) \in L^2(\mathbb{R}^+)$ can be written as a weighted sum of Laguerre filters

$$h(t) = \sum_{i=0}^{\infty} \alpha_i \ell_i(t)$$ (3.34)

where $\alpha_i = \int_0^\infty h(t) \ell_i(t) dt$.

We will use the following notation in this thesis: $\ell_i(t)$ denotes the time domain representation (impulse response) of the $i$-th Laguerre filter. $[\ell_i u](t)$ denotes the convolution of the signal $u(t)$ with the impulse response $\ell_i(t)$ i.e. $[\ell_i u](t) = \int_0^t \ell_i(t-\tau)u(\tau)d\tau$.

**Theorem 3.3** Let $\mathcal{L}_q(s)$ and $\mathcal{L}_r(s)$ be the $q$-th and $r$-th order Laguerre filter defined by equation (3.33) and $\ell_q(t)$ and $\ell_r(t)$ the IRFs belonging to these filters. Let $v(t)$ and $w(t)$ be two Wiener processes with incremental cross covariance Sdt. Let $[\ell_q v](t)$ denote the filtering of $v(t)$ with $\ell_q(t)$ and $[\ell_r w](t)$ the filtering of $w(t)$ with $\ell_r(t)$. Then

$$\mathbf{E}\{[\ell_q v](t)[\ell_r w]^T(t)\} = S\delta_{qr}$$ (3.35)

with $\delta_{qr}$ the Kronecker delta.

**Proof:**

For the expectation (3.35) we can write

$$\mathbf{E}\{[\ell_q v](t)[\ell_r w]^T(t)\}$$ (3.36)

$$= \mathbf{E}\{\int_0^\infty \ell_q(t-\tau)dv(\tau)\left(\int_0^\infty \ell_r(t-s)dw(s)\right)^T\}$$ (3.37)

$$= \int_0^\infty \int_0^\infty \ell_q(t-\tau)\mathbf{E}\{dv(\tau)dw(s)^T\}\ell_r(t-s)$$ (3.38)

$$= \int_0^\infty \int_0^\infty \ell_q(t-\tau)S\delta(\tau-s)\ell_r(t-\tau)d\tau ds$$ (3.39)

$$= \int_0^\infty \ell_q(t-\tau)S\ell_r(t-\tau)d\tau$$ (3.40)

$$= S\delta_{qr}$$ (3.41)

where, in going from equation (3.40) to (3.41), we have used the result of lemma 3.2.

This theorem shows an important correspondence between the Laguerre filters and the discrete time shift operator. Let $v(k)$ and $w(k)$ be two discrete time white noise
sequences with cross covariance $S\delta(k)$. Let $z$ denote the forward shift operator with the property $[z^iv](k) = v(k + i)$, then

$$E\{[z^iv](k)[z^iw]^T(k)\} = S\delta_{qr}$$

This relation is very similar to equation (3.35). We can therefore see the Laguerre filter as a replacement of the $z$ operator in the continuous time domain.

Figure 3.3 shows the IRFs of the first 5 Laguerre filters.

![Figure 3.3: IRFs of the Laguerre filters $\mathcal{L}_0(s)$ up to $\mathcal{L}_4(s)$](image)

Now we have introduced the Laguerre filter and some of its properties, we will move to the all-pass filters. Both the Laguerre filter and the first order all-pass filter will show up in the derivation of the continuous time identification algorithm presented later. We will show that Laguerre filters have a close connection with the first order all-pass filter and show some similarities between the all-pass filter in the continuous time domain and the $z$-operator in the discrete time domain.

### 3.4.2 The all-pass filter

In this section we restrict ourselves to the first order all-pass filter, which is given by the following equation:

$$w(s) = \frac{s - a}{s + a}$$  \hspace{1cm} (3.42)

On the imaginary axis, the magnitude of $w(s)$ is constant. This means that the filter has a constant amplitude amplification. Figure 3.4 shows the bode plot of the first order all-pass filter.
3.4 Laguerre and all-pass filters

From now on we will use the all-pass filter as an operator and leave away the argument $s$ for brevity. The notation $w$ will therefore denote an operator. We will further use the following notation: $w_i(t)$ denotes the time domain representation (impulse response) of a concatenation of $i$ first order all-pass filters. $[w_i u](t)$ denotes the convolution of $u(t)$ with $w_i(t)$ i.e. $[w_i u](t) = \int_0^t w_i(t - \tau)u(\tau)d\tau$.

With the all-pass filter we can define a bank of Laguerre filters in the following way. The zero-th order Laguerre filter $L_0(s) = \frac{\sqrt{2a}}{s + a}$ equals $\frac{1-w}{\sqrt{2a}}$. Multiplying this with $w$, results in the first order Laguerre filter $L_0(s)$. By repetitively multiplying, we obtain a bank of Laguerre filters. The structure of this filter bank is shown in figure 3.5.

![Block diagram of a Laguerre filter bank, constructed from all-pass filters.](image)

To simplify notation in the rest of this chapter we introduce the filter $L_i(s)$.

$$L_i(s) = 2a \frac{(s - a)^i}{(s + a)^{i+1}}$$
This filter equals the Laguerre filter $L_i(s)$ up to the constant factor $\sqrt{2a}$. However, we will take the liberty to keep using the name Laguerre filter for this filter, even though the normality property is lost. Using this definition we have the relation

$$ L_i(s) = (1 - w)w^i $$

This relation will be used frequently in the next sections.

The choice for the all-pass operator $w$ we introduced here, is guided by its simplicity and the orthogonality properties in the Laguerre filters that result from it. These properties make it a very suitable choice for system identification [30, 36, 39, 40]. Many other choices of operators have been considered in literature, for instance low-pass filtering[49] or PMF[75], leading to a wide family of similar identification methods. A problem, one encounters with more complex operators, is the increase in the number of parameters the user has to specify. Moreover, most of these methods have problems when dealing with noise. As we shall see in the next sections, disturbances can be handled well within the Laguerre filter setting.

### 3.4.3 Relation with the $z$-transformation

The $z$-transformation maps a function that is defined on the discrete time axis to a complex function, defined on the $z$ plane. The Laplace transformation does something similar. It transforms a function, defined on the continuous time axis to a complex function on the $s$ plane. A relation between the two transformations exists. This relation is given by $z = e^{s\Delta}$. This equation transforms the complex valued variable $s$ to another complex variable $z$.

The $z$-transformation maps the discrete time parameter $t$ to a complex variable $z$. Since the Laplace transformation does the same, a relation between the two transformations exists. This relation is given by $z = e^{s\Delta}$. This equation transforms the complex valued variable $s$ to another complex variable $z$.

The relation between $z$ and $s$ closely resembles the relation between $w$ and $s$. The all-pass operator $w$ is related to the Laguerre operator as $w = \frac{s-a}{s+a}$. Both these equations give a mapping from the $s$ domain to a new domain. The mappings transform the entire left half plane (LHP) of the $s$-plane to the area within the unit circle, as shown in figure 3.7. The right half plane (RHP) is mapped outside the unit circle.

This section discusses some of the similarities and the differences between the all-pass transformation and the $z$-transformation.

This is similar to the all-pass transformation which also maps the LHP to the inside of the unit circle. However, the transformation from $s$-domain to $z$-domain is not injective, it maps the points $s = (x + jy) \mod \frac{2\pi}{\Delta}$ to the same point in the $z$-plane. This means that the inverse transformation from the $z$-domain to the $s$-domain is not
3.4 Laguerre and all-pass filters

Figure 3.6: Mapping from s-plane left half plane to the z-plane inside the unit circle.

unique. The inverse transformation is given by

\[ s = \frac{1}{\Delta} \ln(z) \]

Every horizontal band in the z-domain, of height \( \frac{2\pi}{\Delta} \), maps to the entire s-domain. This is shown in figure 3.6.

Figure 3.7: Mapping from s-plane left half plane to the z-plane inside the unit circle.

The all-pass transformation also maps the LHP to the inside of the unit circle, as in figure 3.7, but unlike the z-transformation it is a bijective mapping, for which the inverse mapping

\[ s = a \frac{1 + w}{1 - w} \]

is an unique mapping from the w-domain to the s-domain.

The all-pass transformation is also a conformal mapping. Given two intersecting curves \( S_1(s) \) and \( S_2(s) \) in the s-domain and their all-pass transforms in the w-domain. Then the angle between those curves at the intersection in the s-domain is equal to the angle in the w-domain. Figure 3.8 gives an example of a conformal mapping.
3.4.4 All-pass domain state space description

Lemma 3.4 Let the operators $w$ and $s$ be related as in equation (3.42), then the model description (3.9a) - (3.9b) can be transformed into the following state space model:

\[
\begin{align*}
[wx](t) &= A_w x(t) + B_w [l_0 u](t) + [l_0 w](t) + K_1 x_0 l_0(t) \\
l_0 y(t) &= C_w x(t) + D_w [l_0 u](t) + [l_0 v](t) + K_2 x_0 l_0(t)
\end{align*}
\] (3.43a)

(3.43b)

with

\[
\begin{align*}
A_w &= (A + aI)^{-1}(A - aI) \\
B_w &= (A + aI)^{-1}B \\
C_w &= 2aC(A + aI)^{-1} \\
D_w &= D - C(A + aI)^{-1}B \\
K_1 &= (A - aI)^{-1} \\
K_2 &= C(A + aI)^{-1} \\
w_w(t)dt &= (A + aI)^{-1}dw(t) \\
v_w(t)dt &= dv(t) - C(A + aI)^{-1}dw(t)
\end{align*}
\] (3.44)

(3.45)

(3.46)

(3.47)

(3.48)

(3.49)

(3.50)

(3.51)

Proof:

Starting from the state space equations (3.9a) - (3.9b), we have in the Laplace domain notation

\[
\begin{align*}
sX(s) &= AX(s) + BU(s) + W(s) + x(0) \\
Y(s) &= CX(s) + DU(s) + V(s)
\end{align*}
\]
substituting \( s = a \frac{1+w}{1-w} \) in the state equation we obtain

\[
\begin{align*}
\frac{1+w}{1-w}X(s) &= AX(s) + BU(s) + W(s) + x(0) \\
a(1+w)X(s) &= A(1-w)X(s) + B(1-w)U(s) + (1-w)W(s) + (1-w)x(0) \\
aX(s) + awX(s) &= AX(s) - wAX(s) + B(1-w)U(s) + (1-w)W(s) + (1-w)x(0) \\
w(A + aI)X(s) &= (A - aI)X(s) + B(1-w)U(s) + (1-w)W(s) + (1-w)x(0) \\
wX(s) &= (A + aI)^{-1}A(1-w)X(s) + (A + aI)^{-1}B(1-w)U(s) + (1-w)W(s) + (1-w)x(0) \\
&= A_wX(s) + B_w(1-w)U(s) + (1-w)W_w(s) + K_1(1-w)x(0)
\end{align*}
\]

This gives us

\[
\begin{align*}
A_w &= (A + aI)^{-1}(A - aI) \\
B_w &= (A + aI)^{-1}B \\
W_w(s) &= (A + aI)^{-1}W(s) \\
K_1 &= (A + aI)^{-1}
\end{align*}
\]

The output equation gives

\[
\begin{align*}
(1-w)Y(s) &= C(1-w)X(s) + D(1-w)U(s) + (1-w)V(s) \\
&= CX(s) - CA_wX(s) - CB_w(1-w)U(s) - CK_1(1-w)W(s) \\
&= CK_1(1-w)x(0) + D(1-w)U(s) + (1-w)V(s) \\
&= (C - CA_w)X(s) + (D - CB_w)(1-w)U(s) + (1-w)V(s) \\
&= CK_1W(s) - CK_1(1-w)x(0) \\
&= C_wX(s) + D_wU(s) + V_w(s) + K_2(1-w)x(0)
\end{align*}
\]

This results in

\[
\begin{align*}
C_w &= C - CA_w \\
&= C - C(A + aI)^{-1}(A - aI) \\
&= C(A + aI)^{-1}(A + aI) - C(A + aI)^{-1}(A - aI) \\
&= 2aC(A + aI)^{-1} \\
D_w &= (D - CB_w) \\
&= D - C(A + aI)^{-1}B \\
V_w(s) &= V(s) - C(A + aI)^{-1}W(s) \\
K_2 &= C(A + aI)^{-1}
\end{align*}
\]

Translating the result back to the time domain gives (3.43a) - (3.43b).
In the derivation we have used the fact that the Laplace transforms of the noise signals \( v(t) \) and \( w(t) \) exist. In fact they do not. However, the signal for the \([l_0v](t)\) we can write

\[

\nu(t) = [l_0\dot{v}](t)

\]

with \( L_0(s) = \frac{2a}{s+a} \) this can be written as

\[
\frac{d}{dt} \nu(t) + a\nu(t) = 2a \frac{d}{dt} v(t)
\]

"Multiplying" the equation with \( dt \) results in

\[
d\nu(t) = -a\nu(t)dt + a + 2adv(t)
\]

This shows that the signal \([l_0\dot{v}](t)\) is the output of a first order stochastic differential equation, that has the Wiener process \( v(t) \) as input. The signals \([l_0w_w](t)\) and \([l_0v_w](t)\) are therefore well defined and of finite variance.

### 3.4.5 All-pass domain data equation

With the new all-pass system description (3.43a) - (3.43b) we can construct the following set of equations:

\[
[(1-w)y](t) = C_w x(t) + D_w [(1-w)u](t) + [(1-w)v_w](t) + K_0 x_0[(1-w)](t)
\]

\[
[(1-w)w y](t) = C_w [w x](t) + D_w [(1-w)wu](t) + [(1-w)w v_w](t) + K_2 x_0[(1-w)](t)
\]

\[
= C_w A_w x(t) + C_w B_w [(1-w)u](t) + C_w [(1-w)w w](t) + C_2 K_0 x_0[(1-w)](t) + D_w [(1-w)w u](t) + (1-w)w v_w(t) + K_2 x_0[(1-w)w](t)
\]

etc..
When we use the fact that \( L_i(s) = (1 - w)w^i \), we can define the following continuous time data equation:

\[
\begin{bmatrix}
[l_0 y](t) \\
[l_1 y](t) \\
\vdots \\
[l_{i-1} y](t)
\end{bmatrix} = \begin{bmatrix}
C_w & C_w A_w & \cdots & C_w A_w^{i-1} \\
C_w B_w & D_w & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
C_w A_w^{i-2} B_w & \cdots & C_w B_w & D_w
\end{bmatrix} x(t) + \begin{bmatrix}
D_w & 0 & \cdots & 0 \\
C_w B_w & D_w & \cdots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
C_w A_w^{i-2} B_w & \cdots & C_w B_w & D_w
\end{bmatrix} \begin{bmatrix}
[l_0 u](t) \\
[l_1 u](t) \\
\vdots \\
[l_{i-1} u](t)
\end{bmatrix}
\]

\[
+ \begin{bmatrix}
0 & 0 & \cdots & 0 \\
C_w & 0 & \cdots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
C_w A_w^{i-2} & \cdots & C_w & 0
\end{bmatrix} \begin{bmatrix}
[l_0 w](t) \\
[l_1 w](t) \\
\vdots \\
[l_{i-1} w](t)
\end{bmatrix}
\]

\[
+ \begin{bmatrix}
K_2 x_0 & 0 & \cdots & 0 \\
C_w K_1 x_0 & K_2 x_0 & \cdots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
C_w A_w^{i-2} K_1 x_0 & \cdots & K_2 x_0 & 0
\end{bmatrix} \begin{bmatrix}
l_i(t) \\
l_{i+1}(t) \\
\vdots \\
l_{i+j-1}(t)
\end{bmatrix}
\]

Introduce the following notation:

\[
U_{i,j}^w(t) = \begin{bmatrix}
[l_i u](t) \\
[l_{i+1} u](t) \\
\vdots \\
[l_{i+j-1} u](t)
\end{bmatrix}
\]

(3.52)

\[
H_j^w = \begin{bmatrix}
D_w & 0 & \cdots & 0 \\
C_w B_w & D_w & \cdots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
C_w A_w^{i-2} B_w & \cdots & C_w B_w & D_w
\end{bmatrix}
\]

\[
\Gamma_j^w = \begin{bmatrix}
C_w \\
C_w A_w \\
\vdots \\
C_w A_w^{i-1}
\end{bmatrix}
\]

\[
G_j^w = \begin{bmatrix}
0 & \cdots & \cdots & 0 \\
C_w & \cdots & \vdots \\
\vdots & \ddots & \vdots \\
C_w A_w^{i-2} & \cdots & C_w & 0
\end{bmatrix}
\]

\[
F_j^w = \begin{bmatrix}
K_2 x_0 & 0 & \cdots & 0 \\
C_w K_1 x_0 & K_2 x_0 & \cdots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
C_w A_w^{i-2} K_1 x_0 & \cdots & K_2 x_0 & 0
\end{bmatrix}
\]

\[
\Psi_{i,j}(t) = \begin{bmatrix}
l_i(t) \\
l_{i+1}(t) \\
\vdots \\
l_{i+j-1}(t)
\end{bmatrix}
\]
The first subscript $i$ of the matrix $U_{i,j}^w(t)$ is the order of the Laguerre filter in the first row. The second subscript $j$ is the number of (block) rows. $Y_{i,j}^w(t)$, $W_{i,j}^w(t)$ and $V_{i,j}^w(t)$ are defined similar to $U_{i,j}^w(t)$.

With this notation we can write the continuous time data equation in compact form:

$$Y_{i,j}^w(t) = \Gamma_j^u[w_i x](t) + H_j^w U_{i,j}^w(t) + G_j^w W_{i,j}^w(t) + V_{i,j}^w(t) + F_j^w \Psi_{i,j}(t) \quad (3.53)$$

Using the sampled data at sampling times $t_1, t_2, \ldots, t_N$ we obtain the following sampled data matrices

$$U_{i,j,N}^w = \begin{bmatrix} [l_i u](t_1) & [l_i u](t_2) & \cdots & [l_i u](t_N) \\ [l_{i+1} u](t_1) & [l_{i+1} u](t_2) & \cdots & [l_{i+1} u](t_N) \\ \vdots & \vdots & \ddots & \vdots \\ [l_{i+j-1} u](t_1) & [l_{2i-1} u](t_2) & \cdots & [l_{i+j-1} u](t_N) \end{bmatrix}$$

(3.54)

$$X_{i,N}^w = \begin{bmatrix} [w_i x](t_1), [w_i x](t_2), \ldots, [w_i x](t_N) \end{bmatrix}$$

(3.55)

The third subscript $N$ of $U_{i,j,N}^w$ denotes the number of columns in the matrix. $Y_{i,j,N}^w$, $W_{i,j,N}^w$ and $V_{i,j,N}^w$ are again defined similarly to $U_{i,j,N}^w$. With these matrices we have the sampled data equation

$$Y_{i,j,N}^w = \Gamma_j^u X_{i,N}^w + H_j^w U_{i,j,N}^w + G_j^w W_{i,j,N}^w + V_{i,j,N}^w + F_j^w \Psi_{i,j,N} \quad (3.56)$$

This notation is very similar to the discrete time data equation (2.33). The main difference is the term $F_j^w \Psi_{i,j,N}$, which is due to the filtering of the initial state of the state space system.

### 3.5 The continuous time noise-free identification problem

In this section we will derive a continuous time subspace identification algorithm for the noise-free case. The algorithm identifies a continuous time model from sampled data. For consistency with chapter 2 and to keep the outline of the algorithm clear, we will first consider the noise-free identification problem.

First we give the problem definition. Then the algorithm is derived. Next we discuss how to approximate the Laguerre filtered signals using sampled input and output data and we give an analysis of the approximation error.

#### 3.5.1 Problem definition

Let the data-set $\{u(t_k), y(t_k)\}$, $k \in [1, N]$, with $u(t) \in \mathbb{R}^m$ and $y(t) \in \mathbb{R}^l$, be the (possibly non-equidistantly) sampled input/output data at the
sampliing instances \( t_k \) from a system described by the following state space equations:

\[
\begin{align*}
    \dot{x}(t) &= Ax(t) + Bu(t) \\
    y(t) &= Cx(t) + Du(t)
\end{align*}
\]  

where \( x(t) \in \mathbb{R}^n \) and the system matrices \( A, B, C \) and \( D \) are of appropriate dimensions. The system is assumed to be minimal and stable. Let \( N \gg n \).

Then the continuous time noise-free identification problem is to consistently (i.e. \( N \to \infty \) and \( t_{k+1} - t_k \to 0 \forall k \in [1, N] \)) estimate:

1. The order \( n \) of the system.
2. The system matrices \( A_T, B_T, C_T \) and \( D_T \), where the additional index \( T \) refers to the determination of this quadruple up to a similarity transformation.

### 3.5.2 The continuous time ordinary MOESP method

The derivation of this algorithm will be done in two steps. First we derive the algorithm, based on continuous time Laguerre filters. This allows us to show, that when the signals \( [l_0u](t), [l_0u](t), ..., [l_iu](t) \) are available, the algorithm will consistently estimate the system. This setting is pictured in figure 3.9. In the second step, in section 3.5.3, we deal with the approximation error of these signals when discretized Laguerre filters are used. It will be shown under which conditions the algorithm still can estimate the system consistently. This setup is shown in figure 3.10. Finally, in section 3.5.4 we will make an error analysis for a commonly used filter approximation.

**Theorem 3.4** Let \( U_{0,i,N}^w \) and \( Y_{0,i,N}^w \) be constructed as in equation (3.54). Consider following LQ factorization.

\[
\begin{bmatrix}
    U_{0,i,N}^w \\
    Y_{0,i,N}^w
\end{bmatrix}
= 
\begin{bmatrix}
    L_{11} & 0 \\
    L_{21} & L_{22}
\end{bmatrix}
\begin{bmatrix}
    Q_1 \\
    Q_2
\end{bmatrix}
\]  

then

\[
L_{22} = \Gamma_i X_{0,N}^w Q_2^T
\]

**Proof:**

From the LQ factorization we have

\[
Y_{0,i,N}^w Q_2^T = L_{21} Q_1 Q_2^T + L_{22} Q_2 Q_2^T = L_{22}
\]
Figure 3.9: Block diagram of the filtering with a bank of continuous time Laguerre filters.

Figure 3.10: Block diagram of the filtering of sampled input and output signals with digital filters.
Using the data equation (3.56) we obtain

\[ Y_{0,i,N}^u Q_2^T = \Gamma_i^u X_{0,N}^u Q_2^T + H_i^u U_{0,i,N}^u Q_2^T \]

\[ = \Gamma_i^u X_{0,N}^u Q_2^T + H_i^u L_{11} Q_1 Q_2^T \]

\[ = \Gamma_i^u X_{0,N}^u Q_2^T \]

which finishes the proof.

\[ \square \]

### 3.5.3 Approximation of Laguerre filters

The above theorem allows us to estimate the extended observability matrix \( \Gamma_i^u \) from the sampled data. However, we started with the assumption that we could sample the filtered signals. In practice however, we only have the availability of the sampled signals \( u(t_k) \) and \( y(t_k) \). From these signals we need to reconstruct the filtered signals \([l_i u](t_k)\) and \([l_i y](t_k)\) as accurate as possible.

When the data is non-equidistantly sampled, but sampled with a variable sampling time, we can not use one single discrete time filter to replace the continuous time Laguerre filter. In this case we need to use an interpolation scheme, or a higher order integration method to estimate the filtered signal \([l_i u](t)\) at the required time instances by \([\hat{l}_i u](t)\)

\[ \hat{\Gamma}_{i,j,N}^u = \begin{bmatrix} [\hat{l}_i u](t_1) & [\hat{l}_i u](t_2) & \cdots & [\hat{l}_i u](t_N) \\ [l_{i+1} u](t_1) & [l_{i+1} u](t_2) & \cdots & [l_{i+1} u](t_N) \\ \vdots \\ [l_{i+j-1} u](t_1) & [l_{2i-j+1} u](t_2) & \cdots & [l_{i+j-1} u](t_N) \end{bmatrix} \] (3.59)

When the data is equidistantly sampled we can use a bank of discrete time filters to approximate the bank of continuous time Laguerre filters. This setup is shown in figure 3.10. These discrete time approximations are used to estimate the filtered signal from sampled data. In the case of batch data, the digital approximations can even be anti-causal, using, for instance, higher order Padé approximations. The filter \( F(s) \) band-limits the output signal, to reduce aliasing and noise.

The approximation of the continuous time Laguerre filters by discrete time ones can be done by different methods. Chen and Francis [20] showed how we can formulate this approximation as an optimization problem. The optimality is expressed as a model matching problem in a mixed continuous time sampled data framework. The setup of the problem is illustrated in figure 3.11. Given the continuous time filter \( L_i(s) \) to be discretized, an impulse input \( u(t) \) and a pre-filter, we want to find the optimal approximated filter \( L_i^d(z) \). The pre-filter can be used to influence the optimization
process. The idea is, that this pre-filter passes only those parts of the input, for which we would like the error to be small. The optimization now consists of finding a discrete time filter \( L_i^d(z) \) that minimizes \( \|\gamma\|_2 \).

![Diagram](image)

Figure 3.11: Model matching problem setting.

However, the optimal approximated filter depends on the input signal \( u(t) \). To remove the dependency on \( u(t) \) we have to make assumptions about properties of this signal. Moreover, using the model matching approach, we need to find an approximation for every Laguerre filter separately. This can be very time consuming and results in high order filters.

An alternative option is the approximation of the continuous Laguerre filter, using an approximation technique such as the First Order Hold (FOH) approximation, the Tustin’s approximation or the Padé approximation. [23].

We can discretize the filters using one of these approximations. Then the data matrices are replaced by their approximated versions. We will use the following notation for the approximated data matrices. Let discrete time filter \( L_i^d(z) \) be an approximation of the continuous time filter \( L_i(s) \). \( [I_i^d u](t_k) \) will denote the filtering of the sampled signal \( u(t_k) \) with the discrete time filter \( L_i^d(z) \). Finally, the approximated data matrices become

\[
\hat{U}_{i,j,N}^w = \begin{bmatrix}
[I_i^d u](t_1) & [I_i^d u](t_2) & \cdots & [I_i^d u](t_N) \\
[I_i^{d+1} u](t_1) & [I_i^{d+1} u](t_2) & \cdots & [I_i^{d+1} u](t_N) \\
\vdots & \vdots & \ddots & \vdots \\
[I_i^{d+j-1} u](t_1) & [I_i^{d+j-1} u](t_2) & \cdots & [I_i^{d+j-1} u](t_N)
\end{bmatrix}
\] (3.60)

and \( \hat{Y}_{i,j,N}^w \) which is constructed similarly. The bar denotes that we are using approximated filters to construct the matrices.

**Theorem 3.5** Let the discrete time approximations of the i-th order Laguerre filter \( L_i(s) \) be given by \( L_i^d(z) \) and the discrete time filtered signals by \( [I_i u](t_k) \). Let \( u(t) \) be persistently exciting of order \( i \). Furthermore, let the discrete time filter be chosen
such that the following limit holds:

\[
\frac{1}{N} \sum_{k=0}^{N-1} [l_i u](t_k)[l_j u](t_k) = \lim_{\Delta \to 0} \frac{1}{N} \sum_{k=0}^{N-1} [l_i^d u](t_k)[l_j^d u](t_k)
\]

(3.61)

\[
\frac{1}{N} \sum_{k=0}^{N-1} [l_i y](t_k)[l_j y](t_k) = \lim_{\Delta \to 0} \frac{1}{N} \sum_{k=0}^{N-1} [l_i^d y](t_k)[l_j^d y](t_k)
\]

(3.62)

This means that the correlation function of the discrete time filtered (sampled) signals is in the limit \( \Delta \to 0 \) identical to the correlation of the (sampled) continuous time filtered signals.

Now construct the LQ factorization.

\[
\begin{bmatrix}
\tilde{U}_{0,i,N}^w \\
\tilde{Y}_{0,i,N}^w
\end{bmatrix} =
\begin{bmatrix}
\tilde{L}_{11} & 0 \\
\tilde{L}_{21} & \tilde{L}_{22}
\end{bmatrix}
\begin{bmatrix}
\tilde{Q}_1 \\
\tilde{Q}_2
\end{bmatrix}
\]

(3.63)

Then

\[
\lim_{\Delta \to 0} \tilde{L}_{22} = \Gamma_i X_{0,N}^w Q_2^T
\]

Proof:

Using equations (3.61) - (3.62) we have

\[
\lim_{\Delta \to 0} \frac{1}{N} \begin{bmatrix}
\tilde{U}_{0,i,N}^w \\
\tilde{Y}_{0,i,N}^w
\end{bmatrix}^T
\begin{bmatrix}
\tilde{U}_{0,i,N}^w \\
\tilde{Y}_{0,i,N}^w
\end{bmatrix} = \frac{1}{N} \begin{bmatrix}
\tilde{U}_{0,i,N}^w \\
\tilde{Y}_{0,i,N}^w
\end{bmatrix}^T
\begin{bmatrix}
\tilde{U}_{0,i,N}^w \\
\tilde{Y}_{0,i,N}^w
\end{bmatrix}
\]

From the LQ factorization equation (3.63)

\[
\begin{bmatrix}
\tilde{U}_{0,i,N}^w \\
\tilde{Y}_{0,i,N}^w
\end{bmatrix}^T
\begin{bmatrix}
\tilde{U}_{0,i,N}^w \\
\tilde{Y}_{0,i,N}^w
\end{bmatrix} =
\begin{bmatrix}
\tilde{L}_{11} & 0 \\
\tilde{L}_{21} & \tilde{L}_{22}
\end{bmatrix}
\begin{bmatrix}
\tilde{L}_{11} & 0 \\
\tilde{L}_{21} & \tilde{L}_{22}
\end{bmatrix}^T
\]

From this it follows that

\[
\lim_{\Delta \to 0} \frac{1}{\sqrt{N}} \tilde{L}_{22} = \frac{1}{\sqrt{N}} L_{22}
\]

and using theorem 3.4 finishes the proof.

The procedure described in this section is summarized in algorithm 3.1. This is an extension of the ordinary MOESP scheme [92] to the identification of continuous time systems and denoted by Continuous time Ordinary MOESP or COM-MOESP. The algorithm solves the first part of the noise free continuous time identification problem, namely the estimation of \( n, A \) and \( C \).
Algorithm 3.1 Continuous time Ordinary MOESP

1. Build the matrices: \( Y_{0,i,N}^w \) and \( U_{0,i,N}^w \) according to equation (3.59) or (3.60).

2. Perform the LQ factorization (3.63). Under the conditions that \( u(t) \) is persistently exciting, the initial state is zero and \( i \) is chosen larger than the system order, the column-space of \( L_{22} \) approximates that of \( \Gamma_i^w \).

3. Compute the column-space of the matrix \( L_{22} \) via its SVD given as
   \[
   L_{22} = USV^T
   \]

4. From this decomposition, the number of non-zero singular values determine the order \( n \) of the system and the first \( n \) columns of \( U \), gathered in \( U_n \), approximate the column-space of \( \Gamma_i^w \).
   Take \( U_1 \) as the upper \((i - 1)\ell\) rows of \( U_n \) and \( U_2 \) the lower \((i - 1)\ell\) rows of \( U_n \).
   Compute \( \hat{A}_w^T \) and \( \hat{C}_w^T \) as follows
   \[
   \hat{C}_w^T = \text{the upper } \ell \text{ rows of } U_n
   \]
   \[
   \hat{A}_w^T = U_1^T U_2
   \]

5. After we have found \( \hat{A}_w^T \) and \( \hat{C}_w^T \), the matrices \( \hat{A}_T \) and \( \hat{C}_T \) can be calculated using the relation
   \[
   \hat{A} = a(I + \hat{A}_w)(I - \hat{A}_w)^{-1}
   \]
   \[
   \hat{C} = \hat{C}_w(I - \hat{A}_w)^{-1}
   \]
3.5.4 Error analysis

In this section we try to get insight in the effects of the approximation error for the case where \( \{u(t_k), y(t_k)\} \) is equidistantly sampled and for a particular approximation method, namely the Tustin's approximation.

**Lemma 3.5 (Tustin’s approximation)** Consider the state space model (3.57a) - (3.57b), repeated here for convenience:

\[
\dot{x}(t) = Ax(t) + Bu(t) \\
y(t) = Cx(t) + Du(t)
\]

Let the output \( y(t) \) be filtered by an \( i \)-th order continuous time Laguerre filter. In state space notation this Laguerre filter is given by:

\[
\dot{\xi}(t) = F\xi(t) + Hy(t) \\
\nu(t) = G\xi(t)
\]

Let the Tustin's approximation of this Laguerre filter be given by:

\[
\dot{\xi}(k\Delta + \Delta) = F_q\xi(k\Delta) + H_qy_f(k\Delta) \\
\nu(k\Delta) = G\xi(k\Delta) \\
F_q = \left(\frac{2}{\Delta} - F\right)^{-1}\left(\frac{2}{\Delta} + F\right) \\
H_q = \left(\frac{2}{\Delta} - F\right)^{-1}H
\]

Then the error \( \nu(k\Delta) - \nu(k\Delta) \) is of \( O(\Delta^2) \).

**Proof:**

Since the Tustin's approximation has order of accuracy equal to \( \Delta^3 \), the global error in simulation is \( O(\Delta^2) \) [60]

To establish a link between the perturbation of the filtered signals and the estimated system matrices, we first need to look at the perturbation of the QR factorization and the SVD. The following two lemmas deal with that.

**Lemma 3.6 (QR perturbation)** [80]

Given the matrices \( A \) of full row rank and its QR factorization

\[ A = QR \]
Let the matrix $A$ be perturbed with $E$ satisfying

$$
\|A^t\|_F \|E\|_F < \frac{1}{2}
$$

such that perturbed QR factorization satisfies

$$
A + E = (Q + W)(R + F)
$$

Then the perturbations $F$ and $W$ are bounded by

$$
\|W\|_F \leq \frac{3\kappa(A)\|E\|_F}{\|A\|_F - 2\kappa(A)\|E\|_F}
$$

$$
\|F\|_F \leq \|E\|_F + \|W\|_F (\|A\|_F + \|E\|_F)
$$

Here $\kappa(A)$ is defined by $\kappa(A) = \|A\|_F \|A^{-1}\|_F$.

Lemma 3.7 (SVD perturbation) [31]

Let the matrix $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) and its SVD be given by

$$
A = U S V^T = \begin{bmatrix} U_1 & U_2 \\ k & m-k \end{bmatrix} \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \quad k \leq m-k
$$

and let range($V_1$) and range($U_1$) form respectively the row-space and column-space of $A$. Let $E \in \mathbb{R}^{m \times n}$ be a perturbation on $A$ and

$$
U^T E V = \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \\ k & n-k \end{bmatrix} \quad k \leq m-k
$$

and define $\epsilon = \| [E_{21} E_{12}^T] \|_F$ and $\delta = \min_{\sigma \in \sigma(S_1)} |\sigma - \gamma| - \|E_{11}\|_2 - \|E_{22}\|_2$.

If $\delta > 0$ and $\frac{\epsilon}{\delta} \leq \frac{1}{2}$, then there exists a matrix $P \in \mathbb{R}^{(n-k) \times k}$ and $Q \in \mathbb{R}^{(m-k) \times k}$ satisfying

$$
\| \begin{bmatrix} Q \\ P \end{bmatrix} \|_F \leq 2 \frac{\epsilon}{\delta}
$$

such that range($V_1 + V_2 Q$) and range($U_1 + U_2 P$) are the row-space and column-space of $A + E$.

For the perturbation of the singular values the following holds:

$$
\|\sigma_i(A + E) - \sigma_i(A)\| \leq \|E\|_2
$$

Here $\sigma(A)$ denotes the singular values of $A$ and $\sigma_i(A)$ the $i$-th singular value of $A$. 

The continuous time noise-free identification problem

3.5 The continuous time noise-free identification problem

The following theorem finally gives the main result. Theorem 3.6 shows how the error in the estimated column space of the extended observability matrix is related to the sampling time. To this aim the theorem uses the result of lemmas 3.5, 3.6 and 3.7.

Theorem 3.6 Let $L_i(s)$ be the $i$-th order continuous time Laguerre filter and let $L_i^T(z)$ be its Tustin’s approximation with sampling time $\Delta$. Construct the data matrices $U_{i,N}^w$ and $Y_{i,N}^w$ using equation (3.54) and $\tilde{U}_{i,N}^w$ and $\tilde{Y}_{i,N}^w$ according to equation (3.60) from the input and output signals obtained from the system described by equations (3.57a) - (3.57b). Perform the LQ factorizations (3.58) and (3.63) to obtain $L_{22}$ and $\tilde{L}_{22}$. Let

$$\lim_{N \to \infty} \frac{1}{N} L_{22} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}$$

Let $E = \lim_{N \to \infty} \frac{1}{\sqrt{N}} (L_{22} - \tilde{L}_{22})$, $\epsilon = \|E\|_F$, $\delta = \sigma - 2\|E\|_2$ and $\sigma = \min(\sigma(S_1))$. Let $\sigma > 0$ and $\frac{\sigma}{\delta} \leq \frac{1}{2}$.

Then there exists a matrix $P \in \mathbb{R}^{(n-k) \times k}$ satisfying

$$\lim_{N \to \infty} \|P\|_F \leq 2 \frac{\epsilon}{\delta} \leq \frac{C_1 \Delta^2}{\sigma - C_2 \Delta^2}$$

with $C_1$ and $C_2$ constants, independent of $\Delta$, such that range($U_1 + U_2 P$) is the estimated column-space of $\tilde{L}_{22}$.

Proof:

Using theorem 3.4 we find that $\lim_{N \to \infty} \frac{1}{\sqrt{N}} L_2$ is of rank $n$ and thus $S_2$ is zero. With lemma 3.5 we have

$$\begin{bmatrix} U_{i,N}^w \\ Y_{i,N}^w \end{bmatrix} - \begin{bmatrix} \tilde{U}_{i,N}^w \\ \tilde{Y}_{i,N}^w \end{bmatrix} = O(\Delta^2)$$

Using lemma 3.6, we find that the error on the lower triangular factor of the LQ factorization and therefore

$$\lim_{N \to \infty} \frac{1}{\sqrt{N}} \|L_{22} - \tilde{L}_{22}\|_F = O(\Delta^2)$$

Using lemma 3.7 establishes the proof.

The insight we obtain from this result is the following. The perturbation of the estimated column-space, due to discretization errors, is of order $\Delta^2$. But it is not only depending on the approximation scheme, but also on system properties and the experimental conditions, since both influence the magnitude of $\sigma$. 

3.6 The innovations model identification problem

In this section we will extend the results obtained in the previous section for the noise-free identification problem, to the case where the output of the system is disturbed by process and measurement noise. The noise sources are modeled as Wiener processes, allowing us to use analysis techniques from the discrete time domain to tackle the problem.

3.6.1 Problem definition

Let the data-set \( \{u(t_k), y(t_k)\} \), \( k \in [1, N] \), with \( u(t) \in \mathbb{R}^m \) and \( y(t) \in \mathbb{R}^f \), be the (possibly non-equidistantly) sampled input/output data at the sampling instances \( t_k \) from a system described by the following state space equations:

\[
\begin{align*}
    dx(t) &= Ax(t)dt + Bu(t)dt + dw(t) \\
    dz(t) &= Cx(t)dt + Du(t)dt + dv(t) \\
    y(t)dt &= dz(t)
\end{align*}
\]

(3.64)

where \( x(t) \in \mathbb{R}^n \) and the system matrices \( A, B, C \) and \( D \) are of appropriate dimensions. The noise \( w(t) \in \mathbb{R}^n \) and \( v(t) \in \mathbb{R}^f \) are Wiener processes with incremental covariance

\[
\mathbb{E}\left\{ \begin{bmatrix} dw(t) \\ dv(t) \end{bmatrix} \begin{bmatrix} dw(t) \\ dv(t) \end{bmatrix}^T \right\} = \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} dt
\]

The initial state, \( dw(t) \), \( dv(t) \) and \( u(t) \) are assumed to be mutually independent. \( \{A, C\} \) is assumed to be observable, \( \{A, [B \ Q^{1/2}]\} \) is assumed to be controllable and the system is assumed to be stable. Let \( N \gg n \).

The continuous time innovations model identification problem is to asymptotically consistently (i.e. \( N \to \infty \) and \( t_{k+1} - t_k \to 0 \forall k \in [1, N - 1] \)) estimate:

1. The order \( n \) of the system.
2. The system matrices \( A_T, B_T, C_T \) and \( D_T \) where the additional index \( T \) refers to the determination of this quadruple up to a similarity transformation.
3. An estimate of the noise covariance matrices and the Kalman gain \( K \) such that an approximate minimum variance estimator for \( \hat{y}(t_k) \) can be determined.
3.6.2 The continuous time PO-MOESP method

In this section we will develop a procedure to identify the matrices $A_T$ and $C_T$ of the state space model (3.64a)-(3.64c). As in the noise-free case we will first deal with the case where the Laguerre filtering is done in the continuous time domain by analog filters as depicted in figure 3.9. Then we investigate the digital approximation of the Laguerre filter as shown in figure 3.10.

Before we derive the algorithm, we first introduce two instrumental variables that allow us to cope with the process and measurement noise. In the discrete time analogous case, as analyzed e.g. in [89] the so-called instrumental variable (IV) matrix $Z$ was constructed from past input and output quantities. In the continuous time setting, two different IV matrices are proposed.

The first is the filtered output, using a bank of causal Laguerre filters. This scheme allows us to find the matrices $A$ and $C$ consistently, but does not allow for the determination of the stochastic matrices $Q_w$, $R_w$ and $S_w$.

The second instrumental variable that is investigated, is the filtered output using a bank of anti-causal Laguerre filters. With this IV we can solve the complete innovation model identification problem. The anti-causal Laguerre filter is similar to the discrete time $z$ operator, which is also anti-causal. This allows us to find the stochastic part of the innovations model, using a technique that parallels the discrete time method.

Identification using a causal IV

**Lemma 3.8** Let $L_r(s)$, $r \in [0, i + j - 1]$, for positive integers $i$, $j$, be a bank of causal Laguerre filters (Laguerre filter constant $a > 0$). Let the signals $y(t)$, $w(t)$ and $v(t)$ be the output, process noise and measurement noise according description (3.64a)-(3.64c). Let $Y^w_{i,j}(t)$, $W^w_{i,j}(t)$ and $V^w_{i,j}(t)$ be constructed from these signals as in equation (3.52). Then the following holds:

$$ \mathbb{E}\{W^w_{0,i}(t)Y^w_{i,j}(t)\} = 0 \quad (3.65) $$

$$ \mathbb{E}\{V^w_{0,i}(t)Y^w_{i,j}(t)\} = 0 \quad (3.66) $$

**Proof:**
To prove equation (3.65), we note that the block elements of this matrix can be expressed as:

$$ \mathbb{E}\{[l_qw_w](t)[l_ry](t)^T\}, \quad q \in [0, i - 1], \quad r \in [i, i + j - 1] $$
Using Parseval’s theorem, we can write the right hand side as
\[ E\{[l_q w_{w}](t)[l_r y]^T(t)\} = R_{[l_q w_{w}](t),[l_r y](t)} = \int_{-\infty}^{\infty} S_{[l_q w_{w}](t),[l_r y](t)}(j\omega) d\omega \]

where \( R_{[l_q w_{w}](t),[l_r y](t)} \) is the cross covariance and \( S_{[l_q w_{w}](t),[l_r y](t)}(j\omega) \) is the cross power spectral density between \([l_q w_{w}](t)\) and \([l_r y](t)\).

Elaborating the right hand side using the relationship between \( y(t) \) and \( w(t) \), denoted by \( H(s) = C(sI - A)^{-1} \) we get
\[
\int_{-\infty}^{\infty} S_{[l_q w_{w}](t),[l_r y](t)}(j\omega) d\omega \\
= \int_{-\infty}^{\infty} (A + aI)^{-1} L_q(j\omega)QL_r^T(-j\omega)(-j\omega - AT)^{-1}C^T dw \\
= \int_{-\infty}^{\infty} (A + aI)^{-1} 4a^2 \frac{(j\omega + a)^{r-q-1}}{(j\omega - a)^{r-q+1}} Q(j\omega + AT)^{-1}C^T dw
\]

Since the integrand is strictly proper for \( r > q \). When \( a > 0 \) and \( \text{Re}(\lambda(A)) < 0 \) (Stable system) all poles of the integrand are in the right half-plane. Therefore it is analytic in the left half-plane. Application of Cauchy’s theorem now proves that the integral is zero.

In the matrix \( E\{W_{w_{0,i}}^w(t)Y_{i,j}^w(t)^T\} \) all elements have \( r > q \). Therefore the matrix is zero. Equation (3.66) can be proven in an identical way.

**Theorem 3.7** Let \( L_i(s) \) be a bank of causal Laguerre filters \((a > 0)\). Let \( u(t) \) and \( y(t) \) be the input-output data of a system described by equations (3.64a) - (3.64c). Let \( U_{0,i,N}^w, Y_{0,i,N}^w, U_{i,j,N}^w \) and \( Y_{i,j,N}^w \) be constructed from \( u(t) \) and \( y(t) \) according to (3.54) and \( \Psi_{0,i,N} \) as in (3.53).

Let the instrumental variable be defined as
\[
Z_w = \begin{bmatrix} U_{i,j,N}^w \\ Y_{w_{0,i,N}}^w \end{bmatrix}
\]

Consider the LQ factorization
\[
\begin{bmatrix} \Psi_{0,i,N} \\ U_{0,i,N}^w \\ Z_w \\ Y_{0,i,N}^w \end{bmatrix} = \begin{bmatrix} U_{0,i,N}^w \\ \Psi_{0,i,N} \\ U_{i,j,N}^w \\ Y_{i,j,N}^w \end{bmatrix} = \begin{bmatrix} L_{11} & 0 & 0 & 0 & 0 \\ L_{21} & L_{22} & 0 & 0 & 0 \\ L_{31} & L_{32} & L_{33} & 0 & 0 \\ L_{41} & L_{42} & L_{43} & L_{44} & 0 \\ L_{51} & L_{52} & L_{53} & L_{54} & L_{55} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \\ Q_5 \end{bmatrix}
\]

(3.67)

then the following holds:
\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} \begin{bmatrix} L_{53} \\ L_{54} \end{bmatrix} = \lim_{N \to \infty} \frac{1}{\sqrt{N}} \Gamma_{i}^{w} X_{0,N}^{w} \begin{bmatrix} Q_3 \\ Q_4 \end{bmatrix}^T
\]

(3.68)
Proof:
From the LQ factorization (3.67) we have
\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} \begin{bmatrix} L_{53} & L_{54} \end{bmatrix} = \lim_{N \to \infty} \frac{1}{\sqrt{N}} Y_{0,i,N}^w \begin{bmatrix} Q_3 \\ Q_1 \end{bmatrix}^T
\]
From the data equation (3.56) we have
\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} Y_{0,i,N}^w \begin{bmatrix} Q_3 \\ Q_1 \end{bmatrix}^T = \lim_{N \to \infty} \frac{1}{\sqrt{N}} \Gamma_i^w X_{0,N}^w \begin{bmatrix} Q_3 \\ Q_1 \end{bmatrix}^T
+ \lim_{N \to \infty} \frac{1}{\sqrt{N}} \Gamma_{0,i}^w W_{0,i,N}^w \begin{bmatrix} Q_3 \\ Q_4 \end{bmatrix}^T + \lim_{N \to \infty} \frac{1}{\sqrt{N}} Y_{0,i,N}^w \begin{bmatrix} Q_3 \\ Q_4 \end{bmatrix}^T \tag{3.69}
\]
The terms due to the input and the initial state are zero because of the orthogonality between \( \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \) and \( \begin{bmatrix} Q_3 \\ Q_1 \end{bmatrix} \). We will now prove that the last two terms of equation (3.69) will also disappear as \( N \) goes to infinity. We thus have to prove:
\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} W_{0,i,N}^w Q_3^T = 0 \tag{3.70}
\]
\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} V_{0,i,N}^w Q_3^T = 0 \tag{3.71}
\]
\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} W_{0,i,N}^w Q_4^T = 0 \tag{3.72}
\]
\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} V_{0,i,N}^w Q_4^T = 0 \tag{3.73}
\]
Because of the PE condition of the input and due to the noise, we have that the matrices \( \lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{11} \), \( \lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{22} \), \( \lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{33} \) and \( \lim_{N \to \infty} \frac{1}{\sqrt{N}} L_{44} \) are invertible. Since \( w(t) \) and \( v(t) \) are independent from \( u(t) \) and \( x_0 \), we thus have
\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} Q_1 W_{0,i,N}^{wT} = 0 \quad \lim_{N \to \infty} \frac{1}{\sqrt{N}} Q_3 W_{0,i,N}^{wT} = 0
\]
\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} Q_1 V_{0,i,N}^{wT} = 0 \quad \lim_{N \to \infty} \frac{1}{\sqrt{N}} Q_3 V_{0,i,N}^{wT} = 0
\]
\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} Q_2 W_{0,i,N}^{wT} = 0 \quad \lim_{N \to \infty} \frac{1}{\sqrt{N}} Q_2 V_{0,i,N}^{wT} = 0
\]
Which proves (3.70) and (3.71). Using lemma 3.8 we find
\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} Y_{i,j,N}^w W_{0,i,N}^{wT} = 0
\]
\[
\lim_{N \to \infty} \frac{1}{N} \left( L_{41} Q_1 + L_{42} Q_2 + L_{43} Q_3 + L_{44} Q_4 \right) W_{0,i,N}^{wT} = 0
\]
\[
\lim_{N \to \infty} \frac{1}{N} L_{41} Q_1 W_{0,i,N}^{wT} = 0
\]
\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} Q_4 W_{0,i,N}^{wT} = 0
\]
which proves equation (3.72). Similarly equation (3.73) can be proven, which finishes the proof of equation (3.68).

Identification using an anti-causal IV

In this section we investigate the use of an anti-causal Laguerre filter bank to construct an instrumental variable for the continuous time PO-MOESP algorithm. The use of an anti-causal filter is possible since we work on recorded data batches. The anti-causal filter can be applied on this type of data by inverting the sequence of the measured samples. The use of an anti-causal operator is in close relation with the discrete time PO-MOESP algorithm where the \( z \) operator is used, which is, in principle, also an anti-causal operator.

**Lemma 3.9** Let \( L_r(s) \), \( r \in [0,i+j] \), for positive integers \( i, j \), be a bank of anti-causal Laguerre filters \( (a < 0) \). Let the signals \( y(t) \), \( w(t) \) and \( v(t) \) be the output, process noise and measurement noise according to system description (3.64a) - (3.64c). Let \( Y_{0,i}^w(t) \), \( W_{i,j}^w(t) \) and \( V_{i,j}^w(t) \) be constructed from these signals as in equation (3.52). Then the following holds:

\[
\mathbf{E}\{W_{i,j}^w(t)Y_{0,i}^w(t)\} = 0 \tag{3.74}
\]

\[
\mathbf{E}\{V_{i,j}^w(t)Y_{0,i}^w(t)\} = 0 \tag{3.75}
\]

**Proof:**

As with the proof of lemma 3.8 we will only prove equation (3.74). The block-elements of this matrix can be expressed as:

\[
\mathbf{E}\{[l_qw_w(t)][l_rq]^T(t)\}, \quad q \in [i,j], r \in [0,i-1]
\]

Using Parseval’s theorem the cross covariance block entry can be written as

\[
\mathbf{E}\{[l_qw_w(t)][l_rq]^T(t)\} = \int_{-\infty}^{\infty} S_{[l_qw_w(t)][l_rq](t)}(j\omega) d\omega
\]

\[
= \int_{-\infty}^{\infty} (A + aI)^{-1} 4a^2 (j\omega - a)^{q-r} (j\omega + a)^{q-r+1} Q(j\omega + A^T)^{-1} C^T d\omega
\]

The integrand is strictly proper for \( r < q \). When \( a < 0 \) and \( \text{Re}(\lambda(A)) < 0 \) all poles of the integrand are in the right half-plane. Therefore it is analytic in the left half-plane. Application of Cauchy’s theorem now proves that the integral is zero.

In the matrix \( \mathbf{E}\{W_{i,j}^w(t)Y_{0,i-1}^w(t)\} \) all elements have \( r < q \). Therefore the matrix is zero. The proof of equation (3.75) goes similarly.
With the result of lemma 3.9 we can state the following theorem, which shows how the column-space of the extended observability matrix can be estimated asymptotically consistently with the use of an anti-causal instrumental variable.

**Theorem 3.8** Let \( L_i(s) \) be a bank of anti-causal Laguerre filters with the Laguerre filter parameter \( a > 0 \). Let \( u(t) \) and \( y(t) \) be the input/output data of a system described by equation (3.64a) - (3.64c). Let \( U_i^{w,i,N}, Y_i^{w,i,N}, U_i^{w,i,N} \) and \( Y_i^{w,i,N} \) be constructed from \( u(t) \) and \( y(t) \) according to (3.54). \( \Psi_{i,j,N}^{w} \) as in (3.53) and let the instrumental variable be defined as

\[
Z_{w} = \begin{bmatrix}
U_{0,i,N}^{w} \\
Y_{0,i,N}^{w}
\end{bmatrix}
\]

Consider the LQ factorization

\[
\begin{bmatrix}
\Psi_{i,j,N}^{w} \\
U_{i,j,N}^{w} \\
Y_{i,j,N}^{w}
\end{bmatrix} = \begin{bmatrix}
\Psi_{i,j,N}^{w} \\
U_{i,j,N}^{w} \\
Y_{i,j,N}^{w}
\end{bmatrix} = \begin{bmatrix}
L_{11} & 0 & 0 & 0 & 0 \\
L_{21} & L_{22} & 0 & 0 & 0 \\
L_{31} & L_{32} & L_{33} & 0 & 0 \\
L_{41} & L_{42} & L_{43} & L_{44} & 0 \\
L_{51} & L_{52} & L_{53} & L_{54} & L_{55}
\end{bmatrix} \begin{bmatrix}
Q_{1} \\
Q_{2} \\
Q_{3} \\
Q_{4}
\end{bmatrix}
\]

(3.76)

then the following holds:

\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} \begin{bmatrix}
L_{53} \\
L_{54}
\end{bmatrix} = \lim_{N \to \infty} \frac{1}{\sqrt{N}} \Gamma_{i}^{w} X_{0,N}^{w} \begin{bmatrix}
Q_{3} \\
Q_{4}
\end{bmatrix}^T
\]

(3.77)

**Proof:**

The proof goes parallel to the proof of theorem 3.7 with the use of lemma 3.9 instead of lemma 3.8.

When using sampled data, we can not construct the matrices \( Y_{0,i}^{w}, Y_{i,j}^{w}, U_{0,i}^{w}, \) and \( U_{i,j}^{w} \). The filtered signals have to be approximated and we instead use the matrices \( \tilde{Y}_{0,i}^{w}, \tilde{U}_{0,i}^{w}, \tilde{Y}_{i,j}^{w}, \tilde{U}_{i,j}^{w} \) according to equation (3.59) or (3.60).

The complete continuous time PO-MOESP algorithm, denoted by CPO-MOESP, is given in algorithm 3.2.
Algorithm 3.2 continuous time PO-MOESP

1. Construct the matrices $\tilde{U}_{0,i,N}^w, \tilde{U}_{i,j,N}^w, \tilde{Y}_{i,j,N}^w$ and $\tilde{Y}_{0,i,N}^w$ according to equation (3.60) or (3.59) and $\hat{\Psi}_{i,j,N}^w$ or $\hat{\Psi}_{0,i,N}^w$ according to equation (3.53) using either a causal or an anti-causal Laguerre filter bank.

2. For the causal Laguerre filter bank perform the LQ decomposition

$$
\begin{bmatrix}
\hat{\Psi}_{0,i,N}^w \\
\tilde{U}_{0,i,N}^w \\
\tilde{U}_{i,j,N}^w \\
\tilde{Y}_{i,j,N}^w \\
\tilde{Y}_{0,i,N}^w
\end{bmatrix} =
\begin{bmatrix}
\tilde{L}_{11} & 0 & 0 & 0 & 0 \\
\tilde{L}_{21} & \tilde{L}_{22} & 0 & 0 & 0 \\
\tilde{L}_{41} & \tilde{L}_{32} & \tilde{L}_{33} & 0 & 0 \\
\tilde{L}_{41} & \tilde{L}_{42} & \tilde{L}_{43} & \tilde{L}_{44} & 0 \\
\tilde{L}_{51} & \tilde{L}_{52} & \tilde{L}_{53} & \tilde{L}_{54} & \tilde{L}_{55}
\end{bmatrix}
\begin{bmatrix}
\hat{\Phi}_1 \\
\hat{\Phi}_2 \\
\hat{\Phi}_3 \\
\hat{\Phi}_4 \\
\hat{\Phi}_5
\end{bmatrix}
$$

For the anti-causal Laguerre filter bank perform the LQ decomposition

$$
\begin{bmatrix}
\hat{\Psi}_{i,j,N}^w \\
\tilde{U}_{i,j,N}^w \\
\tilde{U}_{0,i,N}^w \\
\tilde{Y}_{0,i,N}^w \\
\tilde{Y}_{i,j,N}^w
\end{bmatrix} =
\begin{bmatrix}
\tilde{L}_{11} & 0 & 0 & 0 & 0 \\
\tilde{L}_{31} & \tilde{L}_{22} & 0 & 0 & 0 \\
\tilde{L}_{31} & \tilde{L}_{32} & \tilde{L}_{33} & 0 & 0 \\
\tilde{L}_{41} & \tilde{L}_{42} & \tilde{L}_{43} & \tilde{L}_{44} & 0 \\
\tilde{L}_{51} & \tilde{L}_{52} & \tilde{L}_{53} & \tilde{L}_{54} & \tilde{L}_{55}
\end{bmatrix}
\begin{bmatrix}
\hat{\Phi}_1 \\
\hat{\Phi}_2 \\
\hat{\Phi}_3 \\
\hat{\Phi}_4 \\
\hat{\Phi}_5
\end{bmatrix}
$$

3. Perform the following SVD to find the column-space of the matrix $\begin{bmatrix} \tilde{L}_{53} & \tilde{L}_{54} \end{bmatrix}$:

$$
\begin{bmatrix} \tilde{L}_{53} & \tilde{L}_{54} \end{bmatrix} = USVT
$$

4. Determine the order $n$ from the singular values in $S$ and construct $U_n$ from the first $n$ columns of $U$. Take $U_1$ as the upper $(i-1)\ell$ rows of $U_n$ and $U_2$ the lower $(i-1)\ell$ rows of $U_n$. Compute $\hat{A}_{wT}$ and $\hat{C}_{wT}$ as follows

$$
\hat{C}_{wT} = \text{the upper } \ell \text{ rows of } U_n \\
\hat{A}_{wT} = U_1^T U_2
$$

5. The matrices $\hat{A}_T$ and $\hat{C}_T$ can be computed using the relations

$$
A = a(I + A_w)(I - A_w)^{-1} \\
C = C_w(I - A_w)^{-1}
$$
3.6.3 Error analysis

A question that needs to be answered at this point is whether the CPO-MOESP algorithm has the same convergence properties as the COM-MOESP algorithm, when the sampling time $\Delta$ goes to zero. We will show that this is indeed the case, by analyzing the algorithm for a typical approximation scheme, namely Tustin’s approximation.

**Lemma 3.10** Consider a system described by the state space equations (3.64a) - (3.64c):

\[
\begin{align*}
    dx(t) & = Ax(t)dt + Bu(t)dt + dw(t) \\
    dz(t) & = Cx(t)dt + Du(t)dt + dv(t) \\
    y(t)dt & = dz(t)
\end{align*}
\]

Let $L_i(s)$ be the $i$-th order continuous time Laguerre filter, given in state space notation by

\[
\begin{align*}
    d\xi(t) & = F\xi(t)dt + Hdz(t) \\
    \nu(t) & = G\xi(t)
\end{align*}
\]

Let $y_f(k\Delta)$ be the pre-filtered and sampled output of the system $H(s)$.

Let $\nu(t)$ be the output of the concatenation of the system and the Laguerre filter $L_i(s)$ and let $\hat{\nu}(k\Delta)$ be the output of $\hat{L}_i(z)$ with the sampled output of the system $y_f(k\Delta)$ as input, where $\hat{L}_i(z)$ is the Tustin’s approximation of $L_i(s)$:

\[
\begin{align*}
    \dot{\xi}(k\Delta + \Delta) & = F_q\dot{\xi}(k\Delta) + H_qy_f(k\Delta) \\
    \hat{\nu}(k\Delta) & = G\dot{\xi}(k\Delta) \\
    F_q & = \left(\frac{2}{\Delta} - F\right)^{-1}\left(\frac{2}{\Delta} + F\right) \\
    H_q & = \left(\frac{2}{\Delta} - F\right)^{-1}H
\end{align*}
\]

Then $E\{(\hat{\nu}(k\Delta) - \nu(k\Delta))^2\}$ is proportional to $\Delta^3$.

**Proof:**

The output of the continuous time filter can be written as a function of the input by concatenating the system under investigation and the Laguerre filter. The concatenated system can be written as:
\[
\begin{bmatrix}
\frac{dx(t)}{dt} \\
\frac{d\xi(t)}{dt}
\end{bmatrix} =
\begin{bmatrix}
A & 0 \\
HC & F
\end{bmatrix}
\begin{bmatrix}
x(t) \\
\xi(t)
\end{bmatrix}
dt +
\begin{bmatrix}
B & I \\
HD & 0
\end{bmatrix}
u(t)dt +
\begin{bmatrix}
0 & 0 \\
0 & H
\end{bmatrix}
\begin{bmatrix}
dw(t) \\
dv(t)
\end{bmatrix}
\]

\[\nu(t) =
\begin{bmatrix}
0 & G
\end{bmatrix}
\begin{bmatrix}
x(t) \\
\xi(t)
\end{bmatrix}\]

Using theorem 3.2 we discretized this system as:

\[
\begin{bmatrix}
x(k\Delta + \Delta) \\
\xi(k\Delta + \Delta)
\end{bmatrix} =
\begin{bmatrix}
e^{A\Delta} & 0 \\
\Psi(\Delta) & e^{F\Delta}
\end{bmatrix}
\begin{bmatrix}
x(k\Delta) \\
\xi(k\Delta)
\end{bmatrix} +
\begin{bmatrix}
u_q(k) \\
\bar{u}_q(k)
\end{bmatrix} +
\begin{bmatrix}
w_q(k) \\
\bar{v}_q(k)
\end{bmatrix}
\]

\[\nu(k\Delta) =
\begin{bmatrix}
0 & G
\end{bmatrix}
\begin{bmatrix}
x(k\Delta) \\
\xi(k\Delta)
\end{bmatrix}\]

where

\[
\begin{bmatrix}
e^{A\Delta} & 0 \\
\Psi(\Delta) & e^{F\Delta}
\end{bmatrix} = e^{\begin{bmatrix}
A & 0 \\
HC & F
\end{bmatrix}\Delta}
\]

and

\[u_q(k) = \int_{k\Delta}^{k\Delta+\Delta} e^{A(k\Delta + \Delta - \tau)} Bu(\tau) d\tau\]
\[w_q(k) = \int_{k\Delta}^{k\Delta+\Delta} e^{A(k\Delta + \Delta - \tau)} dw(\tau)\]
\[\bar{u}_q(k) = \int_{k\Delta}^{k\Delta+\Delta} \Psi(k\Delta + \Delta - \tau) Bu(\tau) d\tau + \int_{k\Delta}^{k\Delta+\Delta} e^{F(k\Delta + \Delta - \tau)} u(\tau) d\tau\]
\[\bar{v}_q(k) = \int_{k\Delta}^{k\Delta+\Delta} \Psi(k\Delta + \Delta - \tau) dw(\tau) + \int_{k\Delta}^{k\Delta+\Delta} e^{F(k\Delta + \Delta - \tau)} dv(\tau)\]
\[u_q'(k) = \int_{k\Delta}^{k\Delta+\Delta} \theta_1(\tau) Bu(\tau) d\tau + \int_{k\Delta}^{k\Delta+\Delta} Du(t) dt\]
\[v_q(k) = \int_{k\Delta}^{k\Delta+\Delta} \theta_1(\tau) dw(\tau) + \int_{k\Delta}^{k\Delta+\Delta} dv(t)\]

with $\theta_1(\tau)$ defined by equation (3.21).
The concatenation of the discretized system and Tustin’s approximation of the Laguerre filter gives

\[
\begin{bmatrix}
    x(k\Delta + \Delta) \\
    \dot{x}(k\Delta + \Delta)
\end{bmatrix}
= \begin{bmatrix}
    A_g & 0 \\
    H_cG & F_q
\end{bmatrix}
\begin{bmatrix}
    x(k\Delta) \\
    \dot{x}(k\Delta)
\end{bmatrix}
+ \begin{bmatrix}
    I & 0 \\
    0 & H_c
\end{bmatrix}
\begin{bmatrix}
    u_g(k) \\
    v_g(k)
\end{bmatrix}

+ \begin{bmatrix}
    I & 0 \\
    0 & H_c
\end{bmatrix}
\begin{bmatrix}
    w_q(k) \\
    v_q(k)
\end{bmatrix}
\]

\[
\dot{v}(k\Delta + \Delta) = \begin{bmatrix}
    0 & G \\
\end{bmatrix}
\begin{bmatrix}
    \dot{x}(k\Delta) \\
    \dot{\xi}(k\Delta)
\end{bmatrix}
\]

To find the approximation error \(\delta_\nu(k) = v(k) - \hat{v}(k)\), we define the state error \(\delta_\xi(k) = \xi(k) - \hat{\xi}(k)\).

To examine this error we construct the following extended state space equation

\[
\begin{bmatrix}
    x(k\Delta + \Delta) \\
    \dot{x}(k\Delta + \Delta) \\
    \delta_\xi(k\Delta + \Delta)
\end{bmatrix}
= \begin{bmatrix}
    e^{A\Delta} & 0 & 0 \\
    \Psi(\Delta) - H_cF_q & e^{F\Delta} - F_q & 0 \\
    u_g(k) & v_g(k) & w_q(k)
\end{bmatrix}
\begin{bmatrix}
    x(k\Delta) \\
    \dot{x}(k\Delta) \\
    \delta_\xi(k\Delta)
\end{bmatrix}
+ \begin{bmatrix}
    u_q(k) - H_cu_g(k) \\
    \bar{v}_q(k) - H_cv_q(k)
\end{bmatrix}
\]

(3.78)

For brevity this state space system will be denoted by

\[
x_\nu(k+1) = A_\nu x_\nu(k) + u_\nu(k) + v_\nu(k) \\
\delta_\nu(k) = G_\nu x_\nu(k)
\]

(3.79a, 3.79b)

The variance of the approximation error due to the deterministic input \(u_\nu(t)\) was discussed in lemma 3.5 and of order of magnitude \(\Delta^2\).

To calculate the approximation error, corresponding to the stochastic input, we look at the variance of the signal \(\delta_\nu(k) = v(k) - \hat{v}(k)\), with only the stochastic input. Since \(\delta_\nu(k) = G\xi(k) - G\hat{\xi}(k) = G\delta_\xi(k)\), we will further only look at the state error \(\delta_\xi(k)\).

The variance of the signal \(\delta_\xi(k)\) can be calculated using the state covariance matrix \(P_\nu\) of the state space system (3.79a)-(3.79b). This state covariance matrix follows from the following Lyapunov equation.

\[
A_\nu P_\nu A_\nu^T - P_\nu = \Omega
\]

(3.80)

with

\[
\Omega = E\left\{\begin{bmatrix}
    w_q(k) \\
    \bar{v}_q(k) \\
    \bar{v}_q(k) - H_cv_q(k)
\end{bmatrix}
\begin{bmatrix}
    w_q(k) \\
    \bar{v}_q(k) \\
    \bar{v}_q(k) - H_cv_q(k)
\end{bmatrix}^T\right\}
\]
The (block-) entries of $\Omega$ are given by

$$
\mathbb{E}\{w_q(k)w_q^T(k)\} = \int_0^\Delta e^{A(\Delta - \tau)}Qe^{A^T(\Delta - \tau)}d\tau
$$

$$
\mathbb{E}\{\bar{v}_q(k)w_q^T(k)\} = \int_0^\Delta \Psi(\Delta - \tau)Qe^{A^T(\Delta - \tau)}d\tau + \int_0^\Delta e^{F(\Delta - \tau)}STe^{A^T(\Delta - \tau)}d\tau
$$

$$
\mathbb{E}\{(\bar{v}_q(k) - \frac{H_q}{\Delta}v_q(k))w_q^T(k)\} = \int_0^\Delta \Psi(\Delta - \tau)Qe^{A^T(\Delta - \tau)}d\tau + \int_0^\Delta e^{F(\Delta - \tau)}STe^{A^T(\Delta - \tau)}d\tau
$$

$$
+ \int_0^\Delta \frac{H_q}{\Delta} \theta_2(\tau)Qe^{A^T(\Delta - \tau)}d\tau + \int_0^\Delta \frac{H_q}{\Delta} STe^{A^T(\Delta - \tau)}d\tau
$$

$$
\mathbb{E}\{\bar{v}_q(k)\bar{v}_q^T(k)\} = \int_0^\Delta \Psi^T(\Delta - \tau)Q\Psi^T(\Delta - \tau)d\tau + \int_0^\Delta \Psi(\Delta - \tau)Se^{F^T(\Delta - \tau)}d\tau
$$

$$
+ \int_0^\Delta e^{F(\Delta - \tau)}ST\Psi^T(\Delta - \tau)d\tau + \int_0^\Delta e^{F(\Delta - \tau)}Re^{F^T(\Delta - \tau)}d\tau
$$

$$
\mathbb{E}\{(\bar{v}_q(k) - \frac{H_q}{\Delta}v_q(k))\bar{v}_q^T(k)\} = \int_0^\Delta \Psi(\Delta - \tau)Q\Psi^T(\Delta - \tau)d\tau + \int_0^\Delta \Psi(\Delta - \tau)Se^{F^T(\Delta - \tau)}d\tau
$$

$$
+ \int_0^\Delta e^{F(\Delta - \tau)}ST\Psi^T(\Delta - \tau)d\tau + \int_0^\Delta e^{F(\Delta - \tau)}Re^{F^T(\Delta - \tau)}d\tau
$$

$$
+ \int_0^\Delta \frac{H_q}{\Delta} \theta_2(\tau)Q\Psi^T(\Delta - \tau)d\tau + \int_0^\Delta \frac{H_q}{\Delta} ST\Psi^T(\Delta - \tau)d\tau
$$

$$
+ \int_0^\Delta \frac{H_q}{\Delta} ST\Psi^T(\Delta - \tau)d\tau + \int_0^\Delta \frac{H_q}{\Delta} Re^{F^T(\Delta - \tau)}d\tau
$$
\[ \mathbb{E}\{ \bar{v}_q(k) - \frac{H_q}{\Delta} v_q(k) \} (\bar{v}_q(k) - \frac{H_q}{\Delta} v_q(k))^T \] 

\[ = \int_0^\Delta \Psi(\Delta - \tau) Q \Psi^T(\Delta - \tau) d\tau + \int_0^\Delta \Psi(\Delta - \tau) S e^{F_T(\Delta - \tau)} d\tau \]

\[ + \int_0^\Delta \Psi(\Delta - \tau) Q \theta_2^T(\tau) \frac{H_q}{\Delta} d\tau + \int_0^\Delta \Psi(\Delta - \tau) S \frac{H_q}{\Delta} d\tau \]

\[ + \int_0^\Delta e^{F(\Delta - \tau)} S^T \Psi^T(\Delta - \tau) d\tau + \int_0^\Delta e^{F(\Delta - \tau)} R e^{F_T(\Delta - \tau)} d\tau \]

\[ + \int_0^\Delta \frac{H_q}{\Delta} \theta_2(\tau) Q \Psi^T(\Delta - \tau) d\tau + \int_0^\Delta \frac{H_q}{\Delta} \theta_2(\tau) S e^{F_T(\Delta - \tau)} d\tau \]

\[ + \int_0^\Delta \frac{H_q}{\Delta} \theta_2(\tau) Q \theta_2^T(\tau) \frac{H_q}{\Delta} d\tau + \int_0^\Delta \frac{H_q}{\Delta} \theta_2(\tau) S \frac{H_q}{\Delta} d\tau \]

\[ + \int_0^\Delta \frac{H_q}{\Delta} S^T \Psi^T(\Delta - \tau) d\tau + \int_0^\Delta \frac{H_q}{\Delta} R e^{F_T(\Delta - \tau)} d\tau \]

\[ + \int_0^\Delta \frac{H_q}{\Delta} S^T \theta_2^T(\tau) \frac{H_q}{\Delta} d\tau + \int_0^\Delta \frac{H_q}{\Delta} R \frac{H_q}{\Delta} d\tau \]

From these equations we find that the right hand side of the Lyapunov equation (3.80) equals

\[ \Omega = \begin{bmatrix} \Omega_{11} & \Omega_{12} & \Omega_{13} \\ \Omega_{21} & \Omega_{22} & \Omega_{23} \\ \Omega_{31} & \Omega_{32} & \Omega_{33} \end{bmatrix} \begin{bmatrix} \mathcal{O}(\Delta) & \mathcal{O}(\Delta) & \mathcal{O}(\Delta^2) \\ \mathcal{O}(\Delta) & \mathcal{O}(\Delta) & \mathcal{O}(\Delta^2) \\ \mathcal{O}(\Delta^2) & \mathcal{O}(\Delta^2) & \mathcal{O}(\Delta^3) \end{bmatrix} \]

The Lyapunov equation can now be written as:

\[ \begin{bmatrix} e^{A_T} & 0 & 0 \\ \Psi(\Delta) & e^{F_T} & 0 \\ \Psi(\Delta) - \frac{H_q}{\Delta} C_\Delta & e^{F_T} - F_q & F_q \end{bmatrix} \begin{bmatrix} P_{11} & P_{12}^T & P_{13}^T \\ P_{21} & P_{22} & P_{23}^T \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \times \begin{bmatrix} e^{A_T} & 0 & 0 \\ \Psi(\Delta) & e^{F_T} & 0 \\ \Psi(\Delta) - \frac{H_q}{\Delta} C_\Delta & e^{F_T} - F_q & F_q \end{bmatrix}^T = \Omega \]

The covariance matrix of the estimation error term \( \delta \xi(k) \) is given by \( P_{33} \). To find this matrix we first solve for \( P_{11}, P_{21}, \) etc. For the sake of brevity we outline the derivation for determining the order of magnitude of the terms \( P_{11}, P_{21} \) only, since the derivation of the order of magnitude of the other terms is similar (though more lengthy).

\( P_{11} \) has to satisfy the following equation:

\[ e^{A_T} P_{11} e^{A_T} - P_{11} = Q_q \]
Under the assumption that $A$ is asymptotically stable, it follows that $\|P_{11}\|$ is of $O(\Delta^{1.5})$.

The equation for $P_{21}$ equals:

$$e^{F_\Delta}P_{21}e^{F_\Delta^T} - P_{21} = \Omega_{21} - \Psi(\Delta)P_{11}\Psi^T(\Delta)$$

Since $\|\Omega_{21} - \Psi(\Delta)P_{11}\Psi^T(\Delta)\|$ is $O(\Delta^2)$, we can find the order of magnitude of $\|P_{21}\|$ in a similar way as for $P_{11}$. Eventually we find that $\|P_{33}\|$ is $O(\Delta^2)$, completing the proof.

Note: If we only have process noise and set the measurement noise to zero, the right-hand side becomes

$$\Omega = \begin{bmatrix}
O(\Delta) & O(\Delta^2) & O(\Delta^2) \\
O(\Delta^2) & O(\Delta^3) & O(\Delta^3) \\
O(\Delta^2) & O(\Delta^3) & O(\Delta^4)
\end{bmatrix}$$

In that case we find that $E\{(\hat{\nu}(k\Delta) - \nu(k\Delta))^2\}$ is proportional to $\Delta^4$ and the approximation error $\hat{\nu}(k\Delta) - \nu(k\Delta)$ of order of magnitude $\Delta^2$.

### 3.7 Estimation of $B$, $D$ and the initial state

In the previous sections we presented two algorithms to estimate $A$ and $C$. Now that these matrices are known, we continue with the estimation of $B$ and $D$. Given $A$ and $C$, the output $y(t)$ depends linearly on the elements of the matrices $B$, $D$ and the initial state.

Let $e_i$ be the $i$-th column of the $n \times n$ identity matrix $I_n$. Using the filtered input and output $[l_0u](t)$ and $[l_0\hat{y}](t)$ we can write:

$$[l_0\hat{y}](t) = C[l_0x](t) + D[l_0u](t) + [l_0\hat{v}](t)$$

$$= C \int_0^te^{A(t-\tau)}[l_0x](\tau)x_0d\tau + C \int_0^te^{A(t-\tau)}B[l_0u](\tau)d\tau + D[l_0u](t)$$

$$+ C \int_0^te^{A(t-\tau)}[l_0\hat{w}](\tau)d\tau + [l_0\hat{v}](t)$$

$$= C \int_0^te^{A(t-\tau)}[l_0x](\tau)d\tau + C \int_0^te^{A(t-\tau)} \sum_{i=1}^m \sum_{j=1}^m e_i b_{ij} [l_0u_j](\tau)d\tau$$

$$+ \sum_{i=1}^m \sum_{j=1}^m e_i d_{ij} [l_0u_j](t) + C \int_0^te^{A(t-\tau)}[l_0\hat{w}](\tau)d\tau + [l_0\hat{v}](t)$$
\[= C \int_0^t e^{A(t-\tau)}l_0(\tau)d\tau x_0 + C \int_0^t [l_0u]^T(\tau) \otimes e^{A(t-\tau)}d\tau \text{vec}(B) + [l_0u]^T(t) \otimes I_t \text{vec}(D) + C \int_0^t e^{A(t-\tau)}[l_0\dot{w}](\tau)d\tau + [l_0\dot{v}](t)\]

Note that the signals \([l_0\dot{v}](t)\) and \([l_0\dot{w}](t)\) are of finite variance, as we showed at the end of the proof of lemma 3.4. When we define the following matrices:

\[
Y = \begin{bmatrix}
[l_0y](0) \\
[l_0y](\Delta) \\
\vdots \\
[l_0y]((N-1)\Delta)
\end{bmatrix}
\quad (3.81)
\]

\[
\Theta = \begin{bmatrix}
0 \\
C \int_0^\Delta e^{A(\Delta-\tau)}l_0(\tau)d\tau \\
\vdots \\
C \int_0^{(N-1)\Delta} e^{A((N-1)\Delta-\tau)}l_0(\tau)d\tau
\end{bmatrix}
\quad (3.82)
\]

\[
\Upsilon = \begin{bmatrix}
0 \\
C \int_0^\Delta [l_0u]^T(\tau) \otimes e^{A(\Delta-\tau)}d\tau \\
\vdots \\
C \int_0^{(N-1)\Delta} [l_0u]^T(\tau) \otimes e^{A((N-1)\Delta-\tau)}d\tau
\end{bmatrix}
\quad (3.83)
\]

\[
\mathcal{U} = \begin{bmatrix}
[l_0u]^T(0) \otimes I_t \\
[l_0u]^T(\Delta) \otimes I_t \\
\vdots \\
[l_0u]^T((N-1)\Delta) \otimes I_t
\end{bmatrix}
\quad (3.84)
\]

\[
E = \begin{bmatrix}
[l_0\dot{w}](0) \\
\int_0^\Delta e^{A(\Delta-\tau)}[l_0\dot{w}](\tau)d\tau + [l_0\dot{w}](\Delta) \\
\vdots \\
\int_0^{(N-1)\Delta} e^{A((N-1)\Delta-\tau)}[l_0\dot{w}](\tau)d\tau + [l_0\dot{w}]((N-1)\Delta)
\end{bmatrix}
\quad (3.85)
\]

\[
B = \text{vec}(B)
\quad (3.86)
\]

\[
D = \text{vec}(D)
\quad (3.87)
\]

then we can write the following relation for the output matrix \(Y\).

\[Y = \begin{bmatrix} \Theta & \Upsilon & \mathcal{U} \end{bmatrix} \begin{bmatrix} x_0 \\ B \\ D \end{bmatrix} + E \quad (3.88)\]

Because \(x_0\) and \(u(t)\) are independent from the noise \(v(t)\) and \(w(t)\) the coefficients of \(B, D\) and \(x_0\) can be retrieved asymptotically consistent w.r.t. \(\Delta\) and \(N\) by taking the least squares solution of the above equation. This result is very similar to the one obtained in section 2.9. Algorithm 3.3 summarizes the results of this section.
Algorithm 3.3 Estimation of $B$, $D$ and the initial state

1. Construct the matrices $Y_{1,N,1}$, $\Theta$, $\mathcal{U}$ and $\mathcal{Y}$ according to equations (3.81) - (3.84), with $A$ and $C$ substituted with the estimates $\hat{A}_T$ and $\hat{C}_T$.

2. Solve the following least squares problem

$$ Y_{1,N,1} = \begin{bmatrix} \Theta & \mathcal{U} \end{bmatrix} \begin{bmatrix} \hat{x}_{0r} \\ \hat{B}_T \\ \hat{D}_T \end{bmatrix} $$

3. Reconstruct $\hat{B}_T$, $\hat{D}_T$ and $\hat{x}_{0r}$ from $\begin{bmatrix} \hat{x}_{0r} \\ \hat{B}_T \\ \hat{D}_T \end{bmatrix}$.

3.8 Identification of the stochastic part of the innovations model

In this section we describe a procedure to estimate the stochastic part of the innovations model from sampled input/output data. With the knowledge of these matrices, a continuous time Kalman filter can be constructed to minimize the error in a state estimate.

3.8.1 The continuous time Kalman filter

The Kalman filter in continuous time is defined as follows

$$ d\hat{x}(t) = A\hat{x}(t)dt + Bu(t)dt + K(t)d\epsilon(t) $$

$$ d\hat{z}(t) = C\hat{x}(t)dt + Du(t)dt $$

$$ dz(t) = d\hat{z}(t) + d\epsilon(t) $$

$d\epsilon(t)$ is called the innovation. The innovation is a zero mean signal with independent increments [3].

When the noise properties are known, the Kalman gain can be calculated with

$$ K(t) = P(t)C^TR^{-1} $$

$$ \frac{d}{dt}P(t) = AP(t) + P(t)A^T + Q - P(t)C^TR^{-1}CP(t) $$

$$ P(0) = \mathbb{E}\{x(0)x(0)^T\} $$

When $t \to \infty$ the Kalman filter gain $K$ and error covariance matrix $P$ reach a steady state value defined by
\[ K = P C^T R^{-1} \]
\[ 0 = A P + P A^T + Q - P C^T R^{-1} C P \]

### 3.8.2 The all-pass domain Kalman filter

The continuous time Kalman filter that we introduced in the previous section can be translated to the all-pass domain in the following manner.

**Theorem 3.9** Let the all-pass domain system be given by equation (3.43a) - (3.43b). Then:

The Kalman filter in the all-pass domain is given by

\[
[w \hat{x}](t) = A_w \hat{x}(t) + B_w[l_0 u](t) + K_w[l_0 e](t) \tag{3.95}
\]
\[
[l_0 \hat{y}](t) = C_w \hat{x}(t) + D_w[l_0 u](t) \tag{3.96}
\]
\[
[l_0 y](t) = [l_0 \hat{y}](t) + [l_0 e](t) \tag{3.97}
\]

The Kalman gain \(K_w\) is given recursively by

\[
P_w = A_w P_w A_w^T + Q_w \tag{3.98}
\]
\[
- (A_w P_w C_w + S_w)(C_w P_w A_w^T + R_w)^{-1}(C_w P_w A_w^T + S_w) \tag{3.99}
\]

The relation between this steady state all-pass domain Kalman gain \(K_w\) and the continuous time Kalman gain \(K\) in steady state is given by:

\[
K = 2a(I - A_w + K_w C_w)^{-1} K_w \tag{3.100}
\]

**Proof:**

By multiplying equation (3.95) - (3.97) by \(w^k\) we obtain:

\[
[w_{k+1} \hat{x}](t) = A_w[w_k \hat{x}](t) + K_k[l_k e](t)
\]
\[
[l_k \hat{y}](t) = C_w[w_k \hat{x}](t)
\]
\[
[l_k y](t) = C_w[w_k \hat{x}](t) + [l_k e](t)
\]

\[\square\]
We need to find the best approximation \([w_{i+1} \hat{x}](t)\) of the filtered state \([w_{k+1} \hat{x}](t)\) given the filtered output \([l_0 y](t), \ldots, [l_k y](t)\). For this we define:

\[
\begin{align*}
H^y_k & \triangleq \text{Span}([l_0 y](t), [l_1 y](t), \ldots, [l_k y](t)) \\
[w_k \hat{x}](t) & \triangleq [w_k x](t) \Pi_{H^y_{k-1}} \\
[w_k \hat{x}](t) & \triangleq [w_k x](t) - [w_k \hat{x}](t) \\
[l_k e](t) & \triangleq [l_k y](t) - C_w[w_k \hat{x}](t) \\
P_k & \triangleq \mathbb{E}\{[w_k \hat{x}](t)[w_k \hat{x}]^T(t)\}
\end{align*}
\]

Where the matrix \(\Pi_{H^y_{k-1}}\) denotes the projection matrix for projecting onto the space \(H^y_{k-1}\) in Hilbert space, as defined in [19].

From this it follows that,

\[
\begin{align*}
[w_k \hat{x}](t) \Pi_{H^y_{k-1}} &= 0 \\
[l_k e](t) &= [l_k y](t) - C_w[w_k \hat{x}](t) \\
&= C_w[w_k x](t) + [l_k v_w](t) - C_w[w_k \hat{x}](t) \\
&= C_w[w_k \hat{x}](t) + [l_k v_w](t) \\
[l_k e](t) \Pi_{H^y_{k-1}} &= 0
\end{align*}
\]

\[
H^y_k = \text{Span}([l_0 y](t), [l_1 y](t), \ldots, [l_{k-1} y](t), [l_k y](t))
\]

\[
= \text{Span}([l_0 y](t), [l_1 y](t), \ldots, [l_{k-1} y](t), C[w_k \hat{x}](t) + [l_k e](t))
\]

\[
= \text{Span}([l_0 y](t), [l_1 y](t), \ldots, [l_{k-1} y](t)) + \text{Span}([l_k e](t))
\]

\[
= H^y_{k-1} \oplus H^e_{(k)}
\]

Where \(\oplus\) denotes the direct sum [19] and \(H^e_{(k)} = \text{Span}([l_k e](t))\)

\[
[w_{k+1} \hat{x}](t) = [w_{k+1} x](t) \Pi_{H^y_k}
\]

\[
= [w_{k+1} x](t) \Pi_{H^y_{k-1}} + [w_{k+1} x](t) \Pi_{H^e_{(k)}}
\]

\[
= A_w[w_k x](t) + [w_{k+1} x](t) \Pi_{H^e_{(k)}}
\]

For \([w_{k+1} x](t)\) we can write

\[
[w_{k+1} x](t) \Pi_{H^e_{(k)}} = \mathbb{E}\{[w_{k+1} x](t)[l_k e]^T(t)\} \mathbb{E}\{[l_k e](t)[l_k e]^T(t)\}^{-1}[l_k e](t)
\]
3.8 Identification of the stochastic part of the innovations model

with

\[
\mathbb{E}\{[w_{k+1} x](t)[l_k e]^T(t)\} = \mathbb{E}\{(A_w[w_k x](t) + [l_k w_w](t))(C[w_k \tilde{x}](t) + [l_k v_w](t))^T\} \\
= \mathbb{E}\{(A_w(w_k \tilde{x})(t) + [l_k w_k](t)) (C[w_k \tilde{x}](t) + [l_k v_w](t))^T\} \\
= A_w \mathbb{E}\{[w_k \tilde{x}](t)[w_k \tilde{x}]^T(t)\} C_w^T + \mathbb{E}\{[l_k w_w](t)[l_k v_w]^T(t)\} \\
= A_w P_k C_w^T + S_w
\]

and

\[
\mathbb{E}\{[l_k e](t)[l_k e]^T(t)\} = \mathbb{E}\{(l_k y)(t) - C[w_k \tilde{x}](t))(l_k y)(t) - C_w[w_k \tilde{x}](t))^T\} \\
= \mathbb{E}\{(C_w[w_k \tilde{x}](t) + [l_k v_w](t)) (C_w[w_k \tilde{x}](t) + [l_k v_w](t))^T\} \\
= C \mathbb{E}\{[w_k \tilde{x}](t)[w_k \tilde{x}]^T(t)\} C_w^T + R_w \\
= C_w P_k C_w^T + R_w
\]

This leads to:

\[
[w_{k+1} \tilde{x}](t) = A_w[w_k \tilde{x}](t) + K_k(\bar{y}(t) - C_w[w_k \tilde{x}](t))
\]

with

\[
K_k = \left( A_w P_k C_w^T + S_w \right) \left( C P_k C^T + R_w \right)^{-1}
\]

To get a recursive expression for \( P_k \), we note that:

\[
[w_{k+1} \tilde{x}](t) = [w_k \tilde{x}](t) - [w_{k-1} \tilde{x}](t) \\
= (A_w[w_k \tilde{x}](t) + [l_k w_w](t)) - (A_w[w_k \tilde{x}](t) + K_k(\bar{y}(t) - C_w[w_k \tilde{x}](t))) \\
= A_w[w_k \tilde{x}](t) + [l_k w_w](t) - K_k C_w[w_k \tilde{x}](t) + K_k C_w[w_k \tilde{x}](t) \\
- A_w[w_k \tilde{x}](t) - K_k [l_k y](t) + K_k C_w[w_k \tilde{x}](t) \\
= (A_w - K_k C_w)[w_k \tilde{x}](t) + K_k C_w[w_k \tilde{x}](t) - K_k [l_k y](t) + [l_k w_w](t) \\
= (A_w - K_k C_w)[w_k \tilde{x}](t) - K_k [l_k v_w](t) + [l_k w_w](t)
\]

Using the orthogonality between \( \tilde{x}(t) \) and \( w_w(t) \) and \( v_w(t) \) we obtain:

\[
P_{k+1} = \mathbb{E}\{[w_k \tilde{x}](t)[w_k \tilde{x}]^T(t)\} \\
= (A_w - K_k C_w)P_k(A_w - K_k C_w)^T - K_k S_w - S_w K_k^T + K_k R_w K_k^T + Q_w \\
= A_w P_k A_w^T - (A_w P_k C_w + S_w)(C P_k C^T + R_w)^{-1}(C_w P_k A_w^T + S_w) + Q_w
\]
This proves the first point of the theorem. The sequence of matrices $P_k$ and $K_k$ will lead to a steady state value for $k \to \infty$. These steady state matrices are given by:

\[ P_w = \lim_{k \to \infty} P_k = A_w P_w A_w^T - (A_w P_w C_w + S_w) (C P_w C^T + R_w)^{-1} (C P_w A_w^T + S_w) + Q_w \]

\[ K_w = \lim_{k \to \infty} K_k = (A_w P_w C_w^T + S_w) (C P_w C^T + R_w)^{-1} \]

In order to find the relation between the all-pass domain Kalman gain $K_w$ and the continuous time domain Kalman gain $K$ we transform the Kalman filter equations to the Laplace domain:

\[ wX(s) = A_w X(s) + K_w ((1-w)Y(s) - C_w X(s)) \]

With $w = \frac{s-a}{s+a}$ and some algebraic manipulation we arrive at

\[ sX(s) = a(I - A_w + K_w C_w)^{-1} (I + A_w - K_w C_w) X(s) + 2a(I - A_w + K_w C_w)^{-1} K_w Y(s) \]

This equation can be equated, term by term, with the known continuous time Kalman filter equation:

\[ sX(s) = (A - KC) X(s) + KY(s) \]

From here we find directly

\[ K = 2a(I - A_w + K_w C_w)^{-1} K_w \]

The structure of this Kalman filter resembles the Kalman filter in discrete time. The shift operator is in this case replaced by the all-pass operator $w$. It is therefore interesting to investigate how the discrete time techniques as are described in [69, 84] and discussed in section 2.5.3, can be applied to the problem.

### 3.8.3 Identification of the continuous time stochastic system

In this section we show how the stochastic system in the all-pass domain can be estimated.

The data equation for the Kalman filter (3.95) - (3.97) is given by

\[ Y_{i,j}^w(t) = \Gamma_{i,j}^w [w_t x](t) + H_j^w U_{i,j}^w(t) + G_j^w E_{i,j}^w(t) + F_j^w \Psi_{i,j}(t) \]  

(3.101)
3.8 Identification of the stochastic part of the innovations model

Where $M_{j}^{w}$ is defined as

$$
M_{j}^{w} = \begin{bmatrix}
I_{\ell} & 0 & \cdots & 0 \\
C_{w}K_{w} & I_{\ell} & \ddots \\
& \ddots & \ddots \\
C_{w}A_{w}^{i-2}K_{w} & C_{w}K_{w} & I_{\ell}
\end{bmatrix}
$$

(3.102)

With the definition

$$
A_{w} = A_{w} - K_{w}C_{w}
$$

(3.103)

$$
B_{w} = B_{w} - K_{w}D_{w}
$$

(3.104)

the state $[w_{i+1}, \tilde{x}]_{i}^{(t)}$ can be written as

$$
[w_{i+1}, \tilde{x}]_{i}^{(t)} = \bar{A}_{i}^{w+1} \tilde{x}(t) + C_{u,i}^{w}U_{0,i}^{w}(t) + C_{y,i}^{w}Y_{0,i}^{w}(t) + C_{x,i}^{w}\Psi_{0,i}^{w}(t)
$$

(3.105)

where

$$
C_{u,i}^{w} = \begin{bmatrix}
\bar{A}_{i}^{w}B_{w} & \bar{A}_{i}^{w-1}B_{w} & \cdots & B_{w}
\end{bmatrix}
$$

(3.106)

$$
C_{y,i}^{w} = \begin{bmatrix}
\bar{A}_{i}^{w}K_{w} & \bar{A}_{i}^{w-1}K_{w} & \cdots & K_{w}
\end{bmatrix}
$$

(3.107)

$$
C_{x,i}^{w} = \begin{bmatrix}
\bar{A}_{i}^{w}K_{1} & \bar{A}_{i}^{w-1}K_{1} & \cdots & K_{1}
\end{bmatrix}
$$

(3.108)

The sampled matrix form of this equation is

$$
\hat{X}_{i+1,N}^{w} = \bar{A}_{i}^{w+1} \hat{X}_{0,N}^{w} + C_{u,i}^{w}U_{0,i,N}^{w} + C_{y,i}^{w}Y_{0,i,N}^{w} + C_{x,i}^{w}\Psi_{0,i,N}^{w}
$$

(3.109)

In the limit $i \to \infty$ the effect of the term $\bar{A}_{i}^{w}\hat{X}_{0,N}^{w}$ disappears since $\bar{A}_{w}$ is stable and the sampled form of the data equation becomes

$$
\lim_{i \to \infty} Y_{i+1,j,N}^{w} = \Gamma_{j}^{w} \lim_{i \to \infty} \left( C_{u,i}^{w}U_{0,i,N}^{w} + C_{y,i}^{w}Y_{0,i,N} + F_{j}^{w}\Psi_{0,j,N}^{w} \right)
$$

(3.110)

$$
+ \lim_{i \to \infty} H_{j}^{w}U_{i+1,j,N}^{w} + \lim_{i \to \infty} G_{j}^{w}E_{i+1,j,N}^{w} + \lim_{i \to \infty} E_{j}^{w}\Psi_{i+1,j,N}
$$

By projecting $Y_{i+1,j,N}^{w}$ onto the space spanned by $U_{0,i,N}^{w}, Y_{0,i,N}^{w}, \Psi_{0,i,N}^{w}, U_{i+1,j,N}^{w}$ and $\Psi_{i+1,j,N}^{w}$ the term $G_{j}^{w}E_{i+1,j,N}^{w}$ drops, since the innovation is orthogonal to the lesser filtered output and independent from the initial condition and the input. Define

$$
\Xi_{i,j,N} = \begin{bmatrix}
\Psi_{0,i,N}^{w} \\
U_{0,i,N}^{w} \\
U_{i,j,N}^{w} \\
Y_{0,i,N}^{w}
\end{bmatrix}
$$

(3.111)
Then

\[
\lim_{i \to \infty} Y_{i+1,j,N}^w \Pi \xi_{i,j,N} = \lim_{i \to \infty} \Gamma_j^w \hat{X}_{i+1,N}^w \Pi \xi_{i,j,N} + \lim_{i \to \infty} H_j^w U_{i+1,j,N}^w \Pi \xi_{i,j,N}
\]

(3.112)

Although the equality only holds in the limit case, it is a good approximation for a fixed but high value of \(i\). Since \(B_w, D_w\) and \(K_1\) are already estimated, we can find estimates of \(\hat{X}_{i+1,N}^w\) and \(\hat{X}_{i,N}^w\) as

\[
\hat{X}_{i+1,N}^w = \hat{\Gamma}_j^w \left( Y_{i+1,j,N}^w \Pi \xi_{i,j,N} - \hat{H}_j^w U_{i,j,N}^w \Pi \xi_{i,j,N} \right)
\]

(3.113)

\[
\hat{X}_{i,N}^w = \hat{\Gamma}_{j+1}^w \left( Y_{i,j+1,N}^w \Pi \xi_{i-1,j+1,N} - \hat{H}_{j+1}^w U_{i-1,j+1,N}^w \Pi \xi_{i-1,j+1,N} \right)
\]

(3.114)

With these two state estimates we can solve the following least squares problem

\[
\begin{bmatrix}
\hat{X}_{i+1,N}^w \\
Y_{i+1,N}^w
\end{bmatrix} =
\begin{bmatrix}
A_{wT} & B_{wT} \\
C_{wT} & D_{wT}
\end{bmatrix}
\begin{bmatrix}
\hat{X}_{i,N}^w \\
Y_{i,N}^w
\end{bmatrix} +
\begin{bmatrix}
W_{i+1,N}^w \\
V_{i+1,N}^w
\end{bmatrix}
\]

(3.115)

for \(W_{i+1,N}^w\) and \(V_{i+1,N}^w\)

\[
\begin{bmatrix}
Q_w & S_w \\
S_w^T & R_w
\end{bmatrix} =
\mathbb{E}\left( \begin{bmatrix}
[l_0 w](t) \\
[l_0 v](t)
\end{bmatrix} \begin{bmatrix}
[l_0 w](t) \\
[l_0 v](t)
\end{bmatrix}^T \right) =
\begin{bmatrix}
W_{i,0,N}^w \\
V_{i,0,N}
\end{bmatrix}^T
\]

(3.116)

With these estimates of the covariance matrices, we can finally compute the estimated Kalman filter gain \(\hat{K}_w\) using equation (3.98) and (3.99). Algorithm 3.4 gives the steps of the algorithm. This algorithm uses the same LQ factorization that is performed for the estimation of the deterministic state space matrices in algorithm 3.2.

### 3.9 Identification of continuous time Wiener systems

This section considers the problem of identifying continuous time Wiener models from sampled data. Wiener models are block oriented non-linear models with a continuous time linear time-invariant (LTI) block followed by a static non-linear block.(see figure 2.6).

We examine a novel approach to identify a continuous time Wiener model. This approach combines the method for identifying discrete time Wiener models and the
Algorithm 3.4 Continuous time estimate of the Kalman gain

1. Estimate $A_{wT}$ and $C_{wT}$, $B_T$ and $D_T$ using CPO-MOESP and CESTDXY. Calculate $\hat{B}_{wT}$, $\hat{D}_{wT}$ and $K_1$ using equation (3.45), (3.47) and (3.48).

2. Construct the matrices $\hat{U}_{0,i,N}^w$, $\hat{C}_{i,j,N}^w$, $\hat{Y}_{i,j,N}^w$ and $\hat{Y}_{i,1,N}^w$ according to equation (3.60) and $\Psi_{0,j,N}$ as in equation (3.33).

3. Compute

$$
\begin{bmatrix}
\Psi_{0,j,N} \\
\hat{U}_{0,i,N} \\
\hat{Y}_{0,i,N} \\
\hat{Y}_{i+1,j,N}
\end{bmatrix} =
\begin{bmatrix}
\tilde{L}_{11} & 0 & 0 & 0 \\
\tilde{L}_{21} & \tilde{L}_{22} & 0 & 0 \\
\tilde{L}_{31} & \tilde{L}_{32} & \tilde{L}_{33} & 0 \\
\tilde{L}_{41} & \tilde{L}_{42} & \tilde{L}_{43} & \tilde{L}_{44}
\end{bmatrix}
\begin{bmatrix}
\hat{Q}_1 \\
\hat{Q}_2 \\
\hat{Q}_3 \\
\hat{Q}_4
\end{bmatrix}
$$

4. Repartition this as

$$
\begin{bmatrix}
\Psi_{0,j,N} \\
\hat{U}_{0,i-1,N} \\
\hat{Y}_{0,i-1,N} \\
\hat{Y}_{i,j-1,N}
\end{bmatrix} =
\begin{bmatrix}
L'_{11} & 0 & 0 & 0 \\
L'_{21} & L'_{22} & 0 & 0 \\
L'_{31} & L'_{32} & L'_{33} & 0 \\
L'_{41} & L'_{42} & L'_{43} & L'_{44}
\end{bmatrix}
\begin{bmatrix}
\hat{Q}'_1 \\
\hat{Q}'_2 \\
\hat{Q}'_3 \\
\hat{Q}'_4
\end{bmatrix}
$$

These factorizations are the same as in CPO-MOESP, only the partitioning is different. Therefore we can reuse the $L$ from CPO-MOESP.

5. Construct

$$
\Xi_{i+1,N} = \hat{\Gamma}_j^{wT} \left( \begin{bmatrix} L_{51} & L_{52} & L_{53} & L_{54} \end{bmatrix} - \hat{H}_j^w \left[ \begin{bmatrix} L_{41} & L_{42} & L_{43} & L_{44} \end{bmatrix} \right] \right)
$$

$$
\Xi_{i,N} = \hat{\Gamma}_j^{wT} \left( \begin{bmatrix} L'_{51} & L'_{52} & L'_{53} & L'_{54} \end{bmatrix} - \hat{H}_j^w \left[ \begin{bmatrix} L'_{41} & L'_{42} & L'_{43} & L'_{44} \end{bmatrix} \right] \right)
$$

6. Take $L_u$ to be the row in $L$ that corresponds to $[l_iu](t)$, over the block-columns 1 to 4, and $L_y$ the row that corresponds to $[l_iu](t)$. In MATLAB notation this becomes $L_u = L(1 : m, 1 : (2m + l)s + l)$, $L_y = L((2m + l)s + 1 : (2m + l)s + l, 1 : (2m + l)s + l)$.

7. Compute the residuals from the equation

$$
\begin{bmatrix}
\rho_w \\
\rho_v
\end{bmatrix} =
\begin{bmatrix}
\Xi_{i+1,N} \\
L_y
\end{bmatrix} -
\begin{bmatrix}
\hat{A}_{wT} & \hat{B}_{wT} \\
\hat{C}_{wT} & \hat{D}_{wT}
\end{bmatrix}
\begin{bmatrix}
\Xi_{i,N} \\
L_u
\end{bmatrix}
$$

8. The estimated covariance matrices are computed as

$$
\begin{bmatrix}
\hat{Q}_w & \hat{S}_w \\
\hat{S}_w^T & \hat{R}_w
\end{bmatrix} =
\begin{bmatrix}
\rho_w \\
\rho_v
\end{bmatrix} \begin{bmatrix}
\rho_w \\
\rho_v
\end{bmatrix}^T
$$

9. Calculate $K_w$ and $K$ using equation (3.98), (3.99) and (3.100).
continuous time MOESP algorithms that were introduced in this chapter. The algorithm for identifying discrete time Wiener models was introduced by Westwick [101] and was discussed in section 2.10. It generally gives better results than methods based on estimating impulse responses. In this section we show how this algorithm can be combined with the subspace algorithm for the identification of continuous time linear models to form an algorithm to identify continuous time Wiener models.

The rest of this section is organized as follows: Section 3.9.1 gives the problem statement. In section 3.9.2 we present an identification algorithm. As before we first introduce the algorithm using analog filters to construct the Laguerre filtered signals. An example will be presented in section 3.10.3.

3.9.1 Problem definition

Let the data-set \( \{u(t_k), y(t_k)\}, k \in [1, N] \) with \( u(t) \in \mathbb{R}^m \) and \( y(t) \in \mathbb{R}^l \), be the (possibly non-equidistantly) sampled input/output data at the sampling instances \( t_k \) from a system described by the following Wiener model:

\[
\begin{align*}
\dot{x}(t) &= Ax(t) + Bu(t) \\
y(t) &= Cx(t) + Du(t) \\
\bar{z}(t) &= f(y(t)) \\
z(t) &= \bar{z}(t) + v(t)
\end{align*}
\]

(3.117a) (3.117b) (3.117c) (3.117d)

where \( x(t) \in \mathbb{R}^n \) and the system matrices \( A, B, C \) and \( D \) are of appropriate dimensions. The LTI part is assumed to be minimal and stable. \( f(.) \) is a static non-linear function and we assume \( f \) is well-behaved in the sense that \( \bar{z}(t) \) has finite variance. The deterministic input \( u(t) \) is assumed to be Gaussian distributed. The disturbance \( m(t) \) is zero-mean colored noise, uncorrelated with \( u(t) \).

The continuous time Wiener model identification problem is to asymptotically consistently (i.e. as \( N \to \infty \) and \( t_{k+1} - t_k \to 0 \forall k \in [1, N] \)) estimate:

1. The order \( n \) of the system.
2. The system matrices \( A_T, B_T, D_f C_T \) and \( D_f D_T \), where the index \( T \) refers to freedom in similarity transformation and a matrix \( D_f \) to the freedom in distribution of the scaling between the LTI block and non-linear function.
3. The static non-linearity \( f(D_f^{-1}(\cdot)) \).

We begin with the following lemma.
Lemma 3.11 Given the system description (3.117a)-(3.117d). Let \( R_{\tilde{z}u}(\tau) = \mathbb{E}\{\tilde{z}(t)u(t-\tau)^T\} \) and \( R_{yu}(\tau) = \mathbb{E}\{y(t)u(t-\tau)^T\} \), also \( S_{\tilde{z}u}(j\omega) \) and \( S_{yu}(j\omega) \) be the corresponding spectral density functions, then

\[
R_{\tilde{z}u}(\tau) = D_f R_{yu}(\tau) \tag{3.118}
\]
\[
S_{\tilde{z}u}(j\omega) = D_f S_{yu}(j\omega) \tag{3.119}
\]

where \( D_f = \mathbb{E}\{f(y(t))y(t)^T\}(\mathbb{E}\{y(t)y(t)^T\})^{-1} \) is a constant matrix that depends only on the static non-linearity \( f \) and covariance of \( y(t) \).

Proof:
We first prove the following relationship: Let \( \alpha \) and \( \beta \) be two zero-mean Gaussian vectors and \( f(\alpha) \) be a non-linear vector function of \( \alpha \) then

\[
\mathbb{E}\{f(\alpha)\beta^T\} = \mathbb{E}\{[f(\alpha)\alpha^T]\}(\mathbb{E}\{\alpha\alpha^T\})^{-1}\mathbb{E}\{\alpha\beta^T\} \tag{3.120}
\]

Let \( p(\cdot) \) denote the probability distribution function. Since \( \alpha \) and \( \beta \) are jointly Gaussian distributed, it can be shown [79, p.566] that the conditional expectation

\[
\mathbb{E}\{\beta^T|\alpha\} = \alpha^T(\mathbb{E}\{\alpha\alpha^T\})^{-1}\mathbb{E}\{\alpha\beta^T\} \tag{3.121}
\]

It follows then

\[
\mathbb{E}\{f(\alpha)\beta^T\} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha)\beta^T p(\alpha, \beta) d\beta d\alpha \tag{3.122}
\]
\[
= \int_{-\infty}^{\infty} f(\alpha) \left( \int_{-\infty}^{\infty} \beta^T p(\beta|\alpha) d\beta \right) p(\alpha) d\alpha \tag{3.123}
\]
\[
= \int_{-\infty}^{\infty} f(\alpha)\alpha^T (\mathbb{E}\{\alpha\alpha^T\})^{-1}\mathbb{E}\{\alpha\beta^T\} p(\alpha) d\alpha \tag{3.124}
\]
\[
= \mathbb{E}\{f(\alpha)\alpha^T\}(\mathbb{E}\{\alpha\alpha^T\})^{-1}\mathbb{E}\{\alpha\beta^T\} \tag{3.125}
\]

Since \( u(t) \) is assumed to be Gaussian distributed, \( y(t) \) also has a Gaussian distribution. The result can now be obtained by substituting \( \beta = u(t-\tau) \) and \( \alpha = y(t) \).

Equation (3.118) forms the basis of many Wiener model identification algorithms [10, 11, 43, 101] as it allows us to estimate the impulse response of the LTI part, up to a constant scaling, from input-output measurements. Similar relations in discrete time for the multi-variable case where \( f \) is assumed to be a polynomial are derived in [101]. The following lemma shows that this relation also holds for filtered input-output measurements.

Lemma 3.12 Let \( F_1 \) and \( F_2 \) be any two scalar continuous time stable linear filters. Define the correlation functions
\( R_{(F_1 \hat{z})(F_2 u)}(\tau) = E\{[F_1 \hat{z}](t)[F_2 u](t - \tau)^T\} \) and \( R_{(F_1 y)(F_2 u)}(\tau) = E\{[F_1 y](t)[F_2 u](t - \tau)^T\} \), then

\[
R_{(F_1 \hat{z})(F_2 u)}(\tau) = D_f R_{(F_1 y)(F_2 u)}(\tau)
\]  

(3.126)

where \( D_f \) is a constant matrix that depends only on the static non-linearity \( f \) and variance of \( y(t) \).

**Proof:**

Let \( S_{(F_1 \hat{z})(F_2 u)}(j\omega) \) and \( S_{(F_1 y)(F_2 u)}(j\omega) \) denote the relevant spectral density functions, then

\[
R_{(F_1 \hat{z})(F_2 u)}(\tau) = \int_{-\infty}^{\infty} S_{(F_1 \hat{z})(F_2 u)}(j\omega) \exp(j\omega\tau) d\omega
\]  

(3.127)

\[
= \int_{-\infty}^{\infty} F_1(j\omega) S_{\hat{z}u}(j\omega) F_2(-j\omega) \exp(j\omega\tau) d\omega
\]  

(3.128)

\[
= \int_{-\infty}^{\infty} F_1(j\omega) D_f S_{yu}(j\omega) F_2(-j\omega) \exp(j\omega\tau) d\omega
\]  

(3.129)

Since \( D_f \) is a constant matrix and \( F_1(j\omega) \) is a scalar, the matrix \( D_f \) can be taken out of the integral and what remains is simply \( R_{(F_1 y)(F_2 u)}(\tau) \).

### 3.9.2 The CWIE-MOESP algorithm

Since the LTI block is assumed to be stable and \( f \) is assumed to be well-behaved, the stochastic processes \( U_{i,j}^w, U_{0,i}^w, N \) and \( Z_{i,j}^w, N \) belong to the Hilbert space of stochastic processes with finite variance [19]. Apply the following LQ factorization:

\[
\begin{bmatrix}
\Psi_{i,j}^w \\
U_{i,j}^w \\
U_{0,i}^w \\
Z_{i,j}^w
\end{bmatrix} =
\begin{bmatrix}
L_{11} & 0 & 0 & 0 \\
L_{21} & L_{22} & 0 & 0 \\
L_{31} & L_{32} & L_{33} & 0 \\
L_{41} & L_{42} & L_{43} & L_{44}
\end{bmatrix}
\begin{bmatrix}
Q_1 \\
Q_2 \\
Q_3 \\
Q_4
\end{bmatrix}
\]  

(3.130)

**Theorem 3.10** Given equation (3.130), then the following equality holds

\[
\lim_{N \to \infty} \frac{1}{N} L_{43} = \lim_{N \to \infty} \frac{1}{N} \kappa \Gamma_i^w X_{i,N}^w Q_2^T
\]  

(3.131)

where

\[
\kappa =
\begin{bmatrix}
D_f & \cdots \\
& \ddots \\
& & \cdots \\
& & & D_f
\end{bmatrix}
\]  

(3.132)
Proof:
Let \( L_3 \) denote the top-left 3-by-3 block of \( L \).

\[
L_3 = \begin{bmatrix}
L_{11} & 0 & 0 \\
L_{21} & L_{22} & 0 \\
L_{31} & L_{32} & L_{33}
\end{bmatrix}
\]  \hspace{1cm} (3.133)

This matrix is invertible due to assumption on \( u(t) \) and definition of \( \Psi_{i,j,N} \). We have:

\[
\lim_{N \to \infty} \frac{1}{N} \begin{bmatrix}
L_{41} & L_{42} & L_{43} & L_{44}
\end{bmatrix}
\]  \hspace{1cm} (3.134)

\[
= \lim_{N \to \infty} \frac{1}{N} Z_{i,j,N}^w \begin{bmatrix}
Q_1^T & Q_2^T & Q_3^T & Q_4^T
\end{bmatrix}
\]  \hspace{1cm} (3.135)

\[
= \lim_{N \to \infty} \frac{1}{N} Z_{i,j,N}^w \begin{bmatrix}
U_{i,j,N}^w & \Psi_{i,j,N}^T & U_{0,i,N}^w & 0
\end{bmatrix} L_3^{-T}
\]  \hspace{1cm} (3.136)

\[
= \lim_{N \to \infty} \frac{1}{N} Z_{i,j,N}^w \begin{bmatrix}
U_{0,i,N}^w & \Psi_{i,j,N}^T & U_{0,i,N}^w & 0
\end{bmatrix} L_3^{-T}
\]  \hspace{1cm} (3.137)

\[
= \lim_{N \to \infty} \frac{1}{N} \tilde{Z}_{i,j,N}^w \begin{bmatrix}
U_{0,i,N}^w & \Psi_{i,j,N}^T & U_{0,i,N}^w & 0
\end{bmatrix} L_3^{-T}
\]  \hspace{1cm} (3.138)

\[
= \kappa \lim_{N \to \infty} \frac{1}{N} Y_{i,j,N}^w \begin{bmatrix}
U_{0,i,N}^w & \Psi_{i,j,N}^T & U_{0,i,N}^w & 0
\end{bmatrix} L_3^{-T}
\]  \hspace{1cm} (3.139)

\[
= \kappa \lim_{N \to \infty} \frac{1}{N} Y_{i,j,N}^w \begin{bmatrix}
U_{0,i,N}^w & \Psi_{i,j,N}^T & U_{0,i,N}^w & 0
\end{bmatrix}
\]  \hspace{1cm} (3.140)

Note:

1. The matrix \( L_3 \) is independent of the expectation operator.

2. In moving from equation (3.137) to (3.138), we use the fact that \( v(t) \) is uncorrelated with \( u(t) \).

3. Lemma 3.12 is used to arrive at equation (3.139). Note in particular that \( \kappa \) is common to all the terms since it depends only on the given \( f \) and the variance of \( y(t) \).

By making use of equation (3.56), where the process and measurement noise terms are zero and using the given LQ factorization equation (3.130), we have

\[
Y_{i,j,N}^w = \Gamma_j^w X_{i,N}^w + H_j^w U_{i,j,N}^w + F_j^w \Psi_{i,j,N}
\]

\[
= \Gamma_j^w X_{i,N}^w + H_j^w (L_{21} Q_1 + L_{22} Q_2) + F_j^w L_{11} Q_1
\]

By substituting this expression into equation (3.140), then the last block column gives the required result of this theorem.
The above theorem is the continuous time analog of theorem 2 in [101] where subspace identification for discrete time Wiener models is investigated. Thus, provided $D_f$ is invertible and \( \text{rank}(E(X_{i,N}^wQ_2^T)) = n \), then \( \text{rank}(L_{32}) = n \). This gives us information on the system order. Its column-space enables us to estimate $A_w$ and $C_w$ up to a similarity transform and a left multiplication on $C_w$.

The invertibility of $D_f$ depends on the map $f$. If $f$ is an even map, then $D_f$ is zero. The applicability of the above algorithm is therefore limited to the case where $f$ is not an even map. However, when the map is even, one may modify the algorithm following section 2.10.2.

Once the matrices $A_w$ and $C_w$ have been computed, $A$ and $C$ can be recovered, again up to a similarity transform and a left multiplication of $C$. The computation of $B$ and $D$ follows mutatis mutandis its discrete time counterpart.

As in previous sections, where we dealt with sampled data, the continuous time filter is replaced by an appropriate discrete time approximation. Instead of the continuous time data vectors $U_{i,j,N}^w$ and $Z_{i,j,N}^w$ we now get discrete time matrices $\tilde{U}_{i,j,N}^w$ and $\tilde{Z}_{i,j,N}^w$. Using theorem 3.5 we can show that the same algorithm can be applied using these matrices.

### 3.10 Examples

In this section three examples are shown of the application of continuous time Subspace Model Identification. First we show the use of the CPO-MOESP algorithm for the identification of a stiff system. In the second example we show the performance of continuous time MOESP on real life data. The third example shows how a non-linear Wiener system can be identified using continuous time MOESP.

#### 3.10.1 Example of using non-equidistant sampled data applied to the identification of stiff systems

In this section we show the use of continuous time MOESP algorithm to non-equidistantly sampled data. An interesting application of this is the identification of stiff systems. A stiff system has dynamics with time-constant that are of different order of magnitude. Suppose we have a system with two time constants $\tau_1$ and $\tau_2$, with $\tau_1 \gg \tau_2$. To identify the system with discrete time methods, we need to record a data-set which contains the effects of both the fast and the slow time constant. A good rule of thumb is to sample the data from a system with a sampling time of approximately one-tenth of the fastest time constant. In the noise-free case the duration of the measurement has to be approximately 10 times the slowest time constant. For noisy data this has to be even longer to be able to average out the noise.
With the continuous time MOESP, we can sample the system at two different sampling rates. First we sample for a short period with a short sampling time for the fast time constant. Then over a longer period with a large sampling time for the slow time constant.

In figure 3.12 an example is shown in which two different sampling times are used.

![Graph showing non-equidistant sample intervals for the identification of stiff systems.](image)

**Example 3.1** *Given the following system in state space notation:*

\[
A = \begin{bmatrix}
-\frac{2\pi}{\tau_1} & 0 \\
0 & -\frac{2\pi}{\tau_2}
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
1 \\
20
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
1 & 1
\end{bmatrix}
\]

\[
D = 0
\]

*with time constant \( \tau_1 = 1 \text{ sec.} \) and \( \tau_2 = 0.01 \text{ sec.} \)***

In an equidistant sampled discrete time identification experiment, the sampling time for this system should be 0.001 sec. and the duration 10 sec. This means that we would need 10,000 data points for the identification of this system.
With two sampling rates, we can sample for 0.1 sec. with a sampling time of 0.001 sec. and then for 10 sec. with a sampling rate of 0.1 sec. This way we only need to process 200 data points.

We simulated the above system using a filtered white Gaussian signal as input. No disturbance was added to the output. For the simulation we used a sampling time of 0.001 sec. and a duration of 10 seconds. We then took the first 100 measurements of the input and output as \( u_1 \) and \( y_1 \). The rest of the 10,000 data points were down sampled by a factor 100, to mimic the sampling at a lower sampling rate. We used a low-pass pre-filter to avoid aliasing. This way we obtain the slow sampled data sequences \( u_2 \) and \( y_2 \).

We then concatenated \( u_1 \) and \( u_2 \) and \( y_1 \) and \( y_2 \) and built the matrices \( \tilde{U}_{i,j,N}^w \) and \( \tilde{Y}_{i,j,N}^w \) as in equation (3.59). We used a simple first order hold integration scheme to compute the approximated signals \( \hat{\tilde{u}}(t) \) and \( \hat{\tilde{u}}(t) \).

With these data matrices we computed the matrices \( \hat{A}_T \) and \( \hat{C}_T \) in a Monte Carlo simulation of 100 trial. Figure 3.13 shows the resulting estimates of the the time constants. The average over 100 trial was \( \tau_1 = 0.011 \) and \( \tau_2 = 0.834 \). The variance was respectively \( 7.5 \times 10^{-6} \) and \( 2.5 \times 10^{-3} \).

![Figure 3.13: Estimated time constants \( \tau_1 \) and \( \tau_2 \) over 100 trial.](image-url)
3.10.2 Example of the use of CPO-MOESP on a real life data-set.

In this section we show the workings of CPO-MOESP on a real life data-set. This data is supplied by Leyland Technical Center, Great Britain. The data is measured from a small Multi Purpose Vehicle (MPV). The MPV is mounted on a test rig that is excited by four hydraulic actuators at the tires. In figure 3.14, a picture is shown of the setup. The input signals are the four hydraulic actuator drives. These are pink noise signals. The output of the system are four measurements of the axle accelerations, chassis accelerations and suspension displacement. This gives in total four inputs and twelve outputs. In this section we only deal with the accelerations of the axle. Therefore, the system we analyze is a four by four system. A discrete time identification effort was made in [14]. In this section we try to find a continuous time model.

Figure 3.14: Example of the test setup of a Vehicle on a four post hydraulic actuator

The data records are 10240 samples, measured with a sampling rate of 100Hz. We divided the data in two sets of 5120 samples. The first record was used for identification. The second for validation. Analysis in [14] showed that the data includes a
delay of 4 samples. By shifting the output data, we removed this delay. Furthermore the data was detrended to remove the mean. However, the data was not decimated to 60 Hz as in [14].

After some tuning of the filter parameter \( a \) and the block matrix parameter \( i \) and the order of the model \( n \), the best model we found was a model of order 10 with \( a \) and \( i \) set to respectively 12 and 16. As a measure of accuracy of our model we used the Variance Accounted For (VAF), which is given by the following formula:

\[
\left( 1 - \frac{\text{variance}(y - y_{\text{est}})}{\text{variance}(y)} \right) \times 100\%
\]

With these values we obtained VAFs of respectively 73.44, 67.45, 78.49 and 79.37 % on the identification data batch. On the validation batch we obtained VAFs of

Figure 3.15 shows the input signals over 100 samples. Figure 3.16 shows the output and estimated output over the 100 samples of the validation data-set.

![Input Signals](image)

Figure 3.15: Measured input for 100 samples of the validation data-set
Figure 3.16: Measured output (solid line) and estimated output (dashed line) for 100 samples of the validation data-set
3.10.3 Example of the Wiener model identification

In this section we show an example of the performance of the described algorithm for the identification of a simple second order non-linear system. The linear part of the system was taken to be

\[
\frac{1}{s^2 + 2s + 2}
\]

The non-linear element was a saturation function that flattens at -0.5 and 0.5. The system was simulated using a first order hold assumption with a sampling time of 0.1s. The input signal was taken to be discrete white noise with variance 16. We used a data-set of 1000 samples. To the non-linear output a disturbance that resulted in a signal-to-noise ratio of 14dB is added.

For the identification, we used a value of 6 for the block-row parameter \(i\) and 10 for the filter parameter \(a\). The non-linear element was estimated by fitting Chebyshev polynomials up to the fifth order.

Altogether 200 sets of data were simulated, each with a different input and noise realization. These data-sets were further divided into 100 estimation/validation data-sets. The singular values of the matrix \(L_{43}\) given by these 100 identification experiments are plotted in figure 3.17. The top two lines are the largest singular values that were found in the 100 trials. These are due to the system. Of course this is a priori information that is given by the fact that we are using simulated data. However, the fact that these singular values are significantly bigger than the other SV's over all trials shows that we can estimate the model order on the basis of the singular values even if this was measured data. Figure 3.18 shows the estimated pole locations of these 100 estimated models. Over the 100 estimation data-sets, the mean and standard deviation of the VAFs of the estimated output are respectively 95.00\% and 0.55\%; and, over the validation data-sets, they are 94.16\% and 2.91\% respectively.

3.11 Conclusion

In this chapter we have introduced a set of new members of the MOESP family of SMI algorithms. These new algorithms allow the identification of continuous time systems from sampled data. The data can be non-equidistant sampled and contaminated by continuous time white process and measurement noise. We are able to estimate a one-step ahead predictor directly from data, by extending the identification scheme for the Kalman filter from discrete time to the continuous time setting.

We have investigated the modeling of continuous time noise. The problem with a continuous time white noise signal is that the variance becomes infinite, if the signal has non-zero power. Several possible solutions have been discussed, such as band-limited white noise and filtered white noise. An alternative is to describe the noise
as Wiener processes. The use of Wiener processes or Brownian motion is based on incremental white noise. This allows us to deal with continuous time white noise in a mathematically correct way.

The continuous time MOESP algorithms that are introduced in this chapter use Laguerre filtering of the input and output. This filtering provides an orthogonal basis of signals. This filtering can substitute the shift operator that is used in the discrete time case to construct a orthogonal basis of signals. With the Laguerre filter, the continuous time state space description is converted to a new domain, called the all-pass domain. In this domain the MOESP methodology is used to estimate the system matrices.

We have developed a method for directly estimating the continuous time Kalman filter from sampled data, by extension of the discrete time method for the estimation of the Kalman filter. This estimation requires the use of anti-causal filters. This is parallel to the discrete time case, where the anti-causal shift-operator is used.

The accuracy of the continuous time estimates depends on the sampling time. We have made an error analysis of the proposed algorithm in which we have shown how the estimation error depends on the sampling time. We have proven that in the limit w.r.t the sampling time and the number of samples, the proposed algorithm is consistent in the estimation of the deterministic system matrices.
Finally, we have shown how the continuous time MOESP algorithm can be applied to the identification of non-equidistantly sampled data. This can be used to identify stiff systems by measuring the data with multiple sampling rates, which has been shown by an example in section 3.10.1. The application of continuous time MOESP on a simulated Wiener system and on a real life data-set was shown to give good results.
Chapter 4

Separable Least Squares in state space identification

4.1 Introduction

In this chapter we treat the identification of state space models using optimization of a cost-function. A cost-function specifies the discrepancy between the model and the information that is available of the system. The cost-function that is used in this chapter is the two-norm of either the output-error or the prediction-error.

In the previous chapters it was shown how a discrete time or continuous time state space model is obtained from sampled data using the Subspace Model Identification method MOESP. The subspace identification method however, does not explicitly optimize a cost-function. Whether SMI methods are optimal and if so, in what respect (i.e. which cost-function they optimize) is still an open research issue[6]. In general, to obtain a MIMO state space model which is optimal in, for instance, a prediction-error sense, we need to perform a non-linear optimization of the parameters describing the state space model.

We will address the problem of estimating MIMO discrete time as well as continuous time LTI state space models from measured data. Also the non-linear Wiener model is treated. The method can easily be extended to other classes of non-linear systems, such as Hammerstein models[16]. For the optimization we exploit the Separable Least Squares (SLS) technique of Golub and Pereyra [32]. With this technique we can reduce the dimension of the parameter space over which the non-linear optimization is performed. As shown in [17] this leads to a optimization over less parameters which increases the chance of finding the global optimum. Moreover, the Hessian of the cost-function problem is better conditioned.
The optimization can be performed in two ways, iteratively and recursively. In an *iterative* or off-line optimization a cost-function is optimized for the entire data batch in each iteration. In *recursive* identification, the optimization proceeds for every new measured input/output data. This is the case for instance, when we deal with on-line identification, where we want to improve or adapt the model using "fresh" data samples taken in real-time from a running system. Using the new samples of input and output data, we want to update the model, without having to redo the entire calculation on the old data plus the newly obtained data. Instead, we use as much information as possible from the previous step in the calculation of the updated model parameters.

The problem of iteratively estimating MIMO systems was previously addressed in [17]. Here only a continuous time state space model was used. The model was parameterized using the output-normal form, which parameterizes the matrices $A$ and $C$ independently from $B$ and $D$ (and $x_0$) with a minimum number of parameters. This parameterization was developed in [33, 34]. The method was extended to the iterative and recursive estimation of discrete time systems, using the output normal parameterization for discrete time SISO systems[35], by Edrissi et al.[26]. At that moment no MIMO output normal parameterization was available. The parameterization for discrete time MIMO systems was introduced recently in [70] and the optimization of MIMO discrete time state space models using this parameterization is presented in this thesis.

In all these cases, as with every non-linear optimization technique, the accuracy of the initial estimate is an important factor for finding the global optimum instead of a local optimum. The combination of MOESP and the SLS technique for the optimization of linear models is therefore a logical one. SLS optimization of an output-error model only needs the initial estimate of the matrices $A$ and $C$ of a state space model. In the prediction-error case also an estimate of the Kalman gain $K$ is required. MOESP allows for the estimation of $A$, $C$ and $K$ separately from $B$, $D$ and $x_0$. Therefore no computing time has to be wasted to calculate these matrices.

The chapter is structured as follows. In section 4.2, the general SLS optimization technique is explained. Section 4.3 presents the connection between SLS and linear discrete time state space models and introduces two discrete time parameterizations which can be used with the SLS technique. In section 4.4 the use of SLS in a gradient based optimization of a discrete time linear state space model is discussed. Section 4.5 shows an illustrative example of how SLS reduces the search space for optimization. In section 4.6 the recursive estimation algorithm for linear discrete time state space models is derived. In section 4.7 the optimization of continuous time state space models is discussed and in section 4.8 the use of SLS with innovation models for prediction error optimization is shown. Section 4.9 shows how SLS can be used for non-linear Wiener models. In section 4.10 a numerical example of the use of the recursive SLS scheme on a discrete time MIMO system is presented. Finally section 4.12 summarizes the results of this chapter.
4.2 Separable Least Squares

In this section we explain the SLS technique for optimization that was introduced by Golub and Pereyra[32]. Consider the following parameter optimization problem:

\[ \hat{\theta} = \arg \left( \min_{\theta} V_N(\theta) \right) = \arg \left( \min_{\theta} \frac{1}{2} ||Y_N - \hat{Y}_N(\theta)||^2_2 \right) \]  \hspace{1cm} (4.1)

where \( V_N(\theta) \) is the cost-function to be minimized. \( Y_N \) is the measured output over a time window of \( N \) samples and \( \hat{Y}_N(\theta) \) the estimated output, which depends on the parameter vector \( \theta \). In some cases the estimated output \( \hat{Y}_N(\theta) \) can be written as \( \Phi_N(\theta_n)\theta_l \). The parameter vector \( \theta \in \mathbb{R}^p \) is separated into two groups of parameters, with \( \theta_n \in \mathbb{R}^{p_n} \) and \( \theta_l \in \mathbb{R}^{p_l} \) such that \( p_n + p_l = p \). In these cases the principle of Separable Least Squares can be applied. With this division of \( \theta \) in \( \theta_l \) and \( \theta_n \) the original optimization problem becomes

\[ \min_{\theta_n, \theta_l} V_N(\theta_n, \theta_l) = \min_{\theta_n, \theta_l} ||Y_N - \Phi_N(\theta_n)\theta_l||^2_2 \]  \hspace{1cm} (4.2)

The principle of SLS, originally proposed in [32] is summarized in the following theorem.

**Theorem 4.1 (Golub and Pereyra[32])** Let the optimization problem be given by (4.2) and assume that the matrix \( \Phi_N(\theta_n) \) has constant rank over an open set \( \Omega \in \mathbb{R}^{p_n} \), then we have the following two results:

(i). If \( \hat{\theta}_n \) is a minimizer of

\[ V_2(\theta_n) = ||\left(I - \Phi_N(\theta_n)\Phi_N^\dagger(\theta_n)\right)Y||^2_2 \]  \hspace{1cm} (4.3)

with \( \Phi_N^\dagger \) denoting the pseudo-inverse of \( \Phi_N \), (i.e. \( \Phi_N^\dagger \Phi_N = I \)) and \( \hat{\theta}_l = \Phi_N^\dagger(\theta_n)Y \), then \( (\hat{\theta}_n, \hat{\theta}_l) \) also minimizes the original cost-function \( V(\theta_n, \theta_l) \) of equation (4.2).

(ii). If \( (\hat{\theta}_n, \hat{\theta}_l) \) minimizes \( V(\theta_n, \theta_l) \) for \( \hat{\theta}_n \in \Omega \), then \( \hat{\theta}_n \) minimizes \( V_2(\theta_n) \) in \( \Omega \) and \( V_2(\hat{\theta}_n) = V(\hat{\theta}_n, \hat{\theta}_l) \). Furthermore, if there is an unique \( \theta_l \) among the minimizing pair \( (\theta_n, \theta_l) \) of \( V(\theta_n, \theta_l) \), then this \( \theta_l \) must satisfy \( \theta_l = \Phi_N^\dagger(\theta_n)Y \).

The above theorem can be used to our advantage in system identification as will be shown in the next section.
4.3 Separation of parameters in discrete time state space models

For the output-error optimization, the system is assumed to be given by the following state space description.

\begin{align}
  x(k + 1, \theta) &= A(\theta)x(k, \theta) + B(\theta)u(k) \\
  y(k, \theta) &= C(\theta)x(k, \theta) + D(\theta)u(k) + v(k) \\
  x(0, \theta) &= x_0(\theta)
\end{align}

with \(x(k, \theta) \in \mathbb{R}^n\), input \(u(k) \in \mathbb{R}^m\) and output \(y(k) \in \mathbb{R}^t\). \(v(k)\) is the measurement noise which is independent of \(u(k)\). The system matrices \(A, B, C\) and \(D\) and the initial state \(x_0\) are of appropriate dimensions. We will further assume the dependency of \(x_0, A, B, C\) and \(D\) on \(\theta\) implicitly. The system is assumed to be stable and minimal.

Since \(v(k)\) is unknown, the estimated output is given by

\[ \hat{y}(k, \theta) = C(\theta)x(k, \theta) + D(\theta)u(k) \]

The output-error optimization problem thus becomes

\begin{align*}
  \hat{\theta} &= \arg \left( \min_{\theta} \frac{1}{2} \sum_{k=0}^{N-1} \|v(k)\|^2_2 \right) \\
  &= \arg \left( \min_{\theta} \frac{1}{2} \sum_{k=0}^{N-1} \|y(k) - \hat{y}(k, \theta)\|^2_2 \right) \\
  &= \arg \left( \min_{\theta} \frac{1}{2} \|Y_N - \hat{Y}_N(\theta)\|^2_2 \right)
\end{align*}

The estimated output of the model can be written explicitly as

\[ \hat{y}(k, \theta) = CA^kx_0 + \sum_{\tau=0}^{k-1} CA^{k-1-\tau}Bu(\tau) + Du(k) \]

Equation (4.5) can be rewritten as

\[ \hat{y}(k, \theta) = CA^kx_0 + \sum_{\tau=0}^{k-1} (u^T(\tau) \otimes CA^{k-1-\tau})\text{vec}(B) \]

\[ + (u^T(k) \otimes I_t)\text{vec}(D) \]

\[ = \left[ \begin{array}{c} CA^k \sum_{\tau=0}^{k-1} u^T(\tau) \otimes CA^{k-1-\tau} \\ u^T(k) \otimes I_t \end{array} \right] \times \left[ \begin{array}{c} x_0 \\ \text{vec}(B) \\ \text{vec}(D) \end{array} \right] \]

(4.6)
Here, the vec(·) operator stacks all elements of a matrix in one vector and ⊗ denotes the Kronecker matrix product (see appendix A.4). We see that the estimated output depends linearly on the initial state $x_0$ and the matrices $B$ and $D$, and non-linearly on the matrix $A$ and $C$. However, the system matrices and the initial state depend on the parameters $\theta$. Therefore we split the parameter vector $\theta$ in two parameter vectors. The first vector $\theta_n$ describes the matrices $A$ and $C$. The second vector $\theta_l$ describes the matrices $B$, $D$ and $x_0$. When we define $\theta_l$ as

$$
\theta_l = \begin{bmatrix}
x_0 \\
\text{vec}(B) \\
\text{vec}(D)
\end{bmatrix}
$$

then we can write the above estimated output as follows:

$$
\hat{y}(k) = \phi_k(\theta_n) \Theta(\theta_l)
$$

with

$$
\phi_k(\theta_n) = \left[ CA^k \sum_{\tau=0}^{k-1} u^T(\tau) \otimes CA^{k-1-\tau} \left( u^T(k) \otimes I_l \right) \right]
$$

Instead of equation (4.6) we can also make the estimated output linear in $C$ and $D$.

$$
\hat{y}(k, \theta) = \left( x_0 A^k \right)^T \otimes I_l \text{vec}(C) + \sum_{\tau=0}^{k-1} \left( A^{k-1-\tau} Bu(\tau) \otimes I_l \right) \text{vec}(C)
$$

$$
+ \left( u^T(k) \otimes I_l \right) \text{vec}(D)
$$

$$
= \begin{bmatrix}
(x_0 A^k)^T \otimes I_l \sum_{\tau=0}^{k-1} (A^{k-1-\tau} Bu(\tau) \otimes I_l) & u^T(k) \otimes I_l
\end{bmatrix} \times
\begin{bmatrix}
\text{vec}(C) \\
\text{vec}(D)
\end{bmatrix}
$$

(4.10)

In this case we need to parameterize $A$, $B$ and $x_0$ independently of the matrices $C$ and $D$. The number of parameters in the linear term for this form is $nl + ml$. In the previous form there are $n + nm + ml$ parameters in the linear term. Which form to chose depends mainly on the number of inputs and outputs of the model. In this chapter we will restrict ourselves to the first form, where the linear part consists of $B$, $D$ and $x_0$.

We now search for an appropriate parameterization $\theta$ of the class of stable and minimal state space systems. Moreover, this parameterization must be such that we can separate the parameters that define the system quadruple $(A,B,C,D)$ in a parameter vector $\theta_n$ that defines the matrices $A$ and $C$ and a vector $\theta_l$ that defines $B$, $D$ and $x_0$. When choose to parameterize the last set by all elements of the matrices $B$, $D$
and \( x_0 \), we are left with the choice of an appropriate parameterization of the set of stable and observable pairs \( A \) and \( C \).

There are many such parameterizations for the state space matrices \( A \) and \( C \). One way is to simply use all the elements of \( A \) and \( C \) as a parameter vector \( \theta_n \). This way we have \( n^2 + nl \) parameters in \( \theta_n \). However, this parameterization is not unique and requires much more parameters than necessary. The minimal number of parameters for describing \( A \) and \( C \) is \( nl \) as shown in[21].

When a non-minimal parameterization is chosen, dependencies will exist between different parameters that describe \( A \) and \( C \). However, since the elements of \( B, D \) and \( x_0 \) are still independent from the parameters describing \( A \) and \( C \), the application of SLS is still possible. Care must be taken in this case when a gradient based optimization method is used. Because of the dependency between the parameters in \( \theta_n \), the Hessian of the cost-function can become singular or close to singular and regularization techniques must be applied.

We will discuss two parameterizations. The first is a parameterization using a tri-diagonal \( A \) matrix. This parameterization does not result in the minimal number of parameters, but is easy and fast to compute. However, because the parameterization is not minimal, regularization must take place.

The second parameterization is the output normal form. In the output normal parameterization, the pair \((A, C)\) is first transformed such that the observability Gramian is equal to identity and the observability matrix is lower triangular with positive entries on the diagonal. This parameterization is minimal as it describes \( A \) and \( C \) with \( nl \) parameters.

There are many more possible parameterizations. However, we will restrict ourselves to the above mentioned two. They show the trade-off between two types of parameterizations that are either fast but non-minimal or minimal but complex and slow.

### 4.3.1 Tri-diagonal parameterization

In the tri-diagonal parameterization \( A \) is parameterized as a tri-diagonal matrix and the matrix \( C \) is fully parameterized. This results in \( 3n - 2 + nl \) parameters. This is not the minimal, but still a large reduction from the \( n^2 + nl \) parameters needed for a fully parameterized \( A \) and \( C \). Because the model is over-parameterized, we need to add a regularization term to the Hessian, to prevent singularity.

An example of the tri-diagonal parameterization for a 5th order system with two outputs is given by:
4.3 Separation of parameters in discrete time state space models

\[
A = \begin{bmatrix}
\theta_5 & \theta_1 & 0 & 0 & 0 \\
\theta_{10} & \theta_6 & \theta_2 & 0 & 0 \\
0 & \theta_{11} & \theta_7 & \theta_3 & 0 \\
0 & 0 & \theta_{12} & \theta_8 & \theta_1 \\
0 & 0 & 0 & \theta_{13} & \theta_9
\end{bmatrix}
\]  
(4.11)

\[
C = \begin{bmatrix}
\theta_{14} & \theta_{16} & \theta_{18} & \theta_{20} & \theta_{22} \\
\theta_{15} & \theta_{17} & \theta_{19} & \theta_{21} & \theta_{22}
\end{bmatrix}
\]  
(4.12)

The matrices \(B\) and \(D\) are fully parameterized.

In the optimization, the derivatives of the matrices \(A\) and \(C\) with respect to the parameter vector \(\theta_n\) are necessary. An advantage of the tri-diagonal parameterization is that these derivatives \(\frac{\partial A}{\partial \theta_n}\) and \(\frac{\partial C}{\partial \theta_n}\) can be calculated very fast because the elements of the matrices are independent from each other and directly related to the parameter vector.

### 4.3.2 Output normal parameterization

The representation we propose in this section is an output normal parameterization [35, 70] of the state space matrices. In this particular parameterization, the matrices \(A\) and \(C\) are chosen such that the output Gramian is equal to identity. This is equal to the requirement that

\[
A^T A + C^T C = I
\]  
(4.13)

In order to make the state space description unique, we then transform \(A\) and \(C\) such that \(\begin{bmatrix} C \\ A \end{bmatrix}\) is an upper triangular matrix with positive entries on the diagonal.

**Jorgen Pedersen Gram (1850-1916)** taught at the University of Copenhagen, Denmark, but he was also managing director of an insurance company and chairman of the Danish Insurance Council. In 1874 he published “Sur quelques théorèmes fondamentaux de l’algebre moderne” in Mathematische Annalen. Gram is best remembered for the Gram-Schmidt orthogonalisation process which constructs an orthogonal set of vectors from an independent one. He was not however the first to use this method. The process seems to be a result of Laplace and it was essentially used by Cauchy in 1836.

Once the matrices \(A\) and \(C\) are transformed to this form, a parameterization using only \(n\ell\) parameters is calculated. For this parameterization the transformed matrix
pair is brought into the following form:

$$\begin{bmatrix} C \\ A \end{bmatrix} = T_1 T_2 \cdots T_n \begin{bmatrix} 0 \\ I_n \end{bmatrix}$$  \hspace{1cm} (4.14)$$

where

$$T_k = \begin{bmatrix} I_{k-1} \\ U_k \\ I_{n-k} \end{bmatrix}$$  \hspace{1cm} (4.15)

$$U_k = \begin{bmatrix} -s_k \\ r_k \\ s_k \end{bmatrix}$$  \hspace{1cm} (4.16)

$$t_k = s_k^T s_k \hspace{0.5cm} r_k = \sqrt{1-t_k} \hspace{0.5cm} S_k = I_t - \frac{1-r_k}{t_k} s_k s_k^T$$  \hspace{1cm} (4.17)

The matrix $U_k$ is parameterized by the elements of the vector $s_k$, which contains $l$ elements. Since $k = 1, 2, \ldots, n$ the total number of parameters equals $n l$.

**Example 4.1** In this example we show the use of the output normal parameterization on a second order system.

$$H(z) = \frac{z + 0.5}{z^2 - 1.5z + 0.7}$$  \hspace{1cm} (4.18)

A state space realization of this system is given by

$$A = \begin{bmatrix} 1.5000 & -0.7000 \\ 1.0000 & 0 \end{bmatrix} \hspace{1cm} B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1.0000 & 0.5000 \end{bmatrix} \hspace{1cm} D = \begin{bmatrix} 0 \end{bmatrix}$$

First we bring the system in a form such that equation (4.13) holds and $$\begin{bmatrix} C \\ A \end{bmatrix}$$ in lower triangular. The similarity transformation

$$T_t = \begin{bmatrix} 0.3223 & 0.9466 \\ 0.9466 & -0.3223 \end{bmatrix}$$

transforms the system into

$$A = \begin{bmatrix} 0.6176 & 0.4706 \\ -0.3294 & 0.8824 \end{bmatrix} \hspace{1cm} B = \begin{bmatrix} 1.4003 \\ 4.1133 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.7141 & 0 \end{bmatrix} \hspace{1cm} D = \begin{bmatrix} 0 \end{bmatrix}$$

The matrix $$\begin{bmatrix} C \\ A \end{bmatrix}$$ can now be decomposed as

$$\begin{bmatrix} 0.7141 & 0 \\ 0.6176 & 0.4706 \\ -0.3294 & 0.8824 \end{bmatrix} = \begin{bmatrix} 0.7 & 0.7141 & 0 \\ 0.7141 & -0.7 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -0.8824 & 0.4706 \\ 0 & 0.4706 & 0.8824 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$
The matrices $A$ and $C$ are now completely parameterized by the following $\theta_n$: $s_2$:

$$
\theta_n = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} 0.7 \\ 0.8824 \end{bmatrix}
$$

The corresponding $\theta_1$, without $x_0$, is given by

$$
\theta_1 = \begin{bmatrix} \text{vec}(B) \\ \text{vec}(D) \end{bmatrix} = \begin{bmatrix} 1.4003 \\ 4.1133 \\ 0 \end{bmatrix}
$$

Algorithm 4.1 describes the procedure to transform a state space model to output normal form.

The calculation of the matrices $\frac{\partial A}{\partial \theta_n}$ and $\frac{\partial C}{\partial \theta_n}$, which is needed in the optimization procedure, can be done algebraically and proceeds in the following way:

$$
\begin{bmatrix}
\frac{\partial C}{\partial \theta_{n_1}} \\
\frac{\partial A}{\partial \theta_{n_2}} \\
\frac{\partial A}{\partial \theta_{n_3}} \\
\vdots \\
\frac{\partial A}{\partial \theta_{n_k}}
\end{bmatrix} = \frac{\partial Z_1}{\partial \theta_n} + \begin{bmatrix}
\frac{\partial Z_2}{\partial \theta_{n_1}} & \frac{\partial Z_2}{\partial \theta_{n_2}} & \cdots & \frac{\partial Z_2}{\partial \theta_{n_k}}
\end{bmatrix}
$$

(4.19)

In equation 4.14 the matrix $T_k$ is a function of the elements $(k-1)\ell + 1$ to $k\ell$ of $\theta_n$. Therefore in differentiating the matrices $A$ and $C$ with respect to $\theta_n$ we need to differentiate $T_k$ with respect to these elements of $\theta_n$. For $(k-1)\ell < i \leq k\ell$ the derivative of $Z$ with respect to the $i$-th element of $\theta_n$ is given by

$$
\frac{\partial Z}{\partial \theta_{n_i}} = T_1 \cdots \frac{\partial T_k}{\partial \theta_{n_i}} \cdots T_n \begin{bmatrix} 0 \\ I_n \end{bmatrix}
$$

(4.20)

Let $j$ denote the element in $s_k$ that corresponds to $\theta_{n_i}$, i.e. $j = \text{remainder} \left( \frac{i}{k} \right)$. Then the matrix $\frac{\partial T_k}{\partial \theta_{n_i}}$ is given by

$$
\frac{\partial T_k}{\partial \theta_{n_i}} = \begin{bmatrix} 0_{k-1} \\ \frac{\partial U_k}{\partial \theta_{n_i}} \\ 0_{n-k} \end{bmatrix}
$$

(4.21)

$$
\frac{\partial U_k}{\partial \theta_{n_i}} = \begin{bmatrix} -c_j \\ \frac{\partial s_k}{\partial \theta_{n_i}} \\ e_j^T \end{bmatrix}
$$

(4.22)

$$
\frac{\partial s_k}{\partial \theta_{n_i}} = \left( \frac{2\theta_{n_i} (1 - r_i)}{t_i^2} - \frac{\theta_{n_i}}{t_i r_i} \right) s_i s_i^T - \frac{1 - r_i}{t_i} (e_j s_i^T + s_i e_j^T)
$$

(4.23)

and $e_j$ an unit vector of length $n$ with an one as the $j$-th entry.
Algorithm 4.1 Output normal parameterization of a discrete time state space system

(i). Compute the observability Gramian $W_0$ of $A$ and $C$ and calculate the Cholesky factorization $W_0 = Q_1 Q_1^T$.

(ii). Apply the transformation $A_1 = Q_1 A Q_1^{-1}$ and $C_1 = C Q_1^{-1}$. The observability Gramian of $(A_1, C_1)$ now equals identity.

(iii). Compute the observability matrix $\Gamma$ of the pair $A_1$, $C_1$ and perform the QR factorization $\Gamma = R^T Q_1^T$.

(iv). Construct $Q_3$ as a diagonal matrix with elements $-1$ and $1$, according to the signs of the elements on the diagonal of $R$.

(v). Apply the transformation $A_2 = Q_3^{-1} Q_2^{-1} A_1 Q_2 Q_3$ and $C_2 = C_1 Q_2 Q_3$. The observability matrix of $(A_2, C_2)$ is lower triangular with positive entries on the diagonal.

(vi). Let $Z_1$ equal $\begin{bmatrix} C_2 \\ A_2 \end{bmatrix}$. Take $s_1$ to be the entries of the last row of $Z_1$, below the diagonal (in MATLAB notation this is: $s_1 = Z_1(n+1 : n+\ell,n)$). Take $r_1$ to be the first entry of the diagonal of $Z_1$ (MATLAB: $r_k = Z(n,n)$).

(vii). Construct $T_k$ using (4.15)- (4.17).

(viii). Perform:

$$T_1^T Z_1 = \begin{bmatrix} Z_2 \\ \vdots \\ 0 \end{bmatrix}$$

(ix). Repeat the same procedure with $Z_2, Z_3, ..., Z_n$ until we arrive at the following factorization:

$$\begin{bmatrix} 0 \\ I_n \end{bmatrix} = T_n^T \cdots T_2^T T_1^T Z_1$$

(x). The matrices $T_k$ are completely parameterized by the vectors $s_k$. Construct the parameter vector $\theta_n$ by stacking these.
Andre-Louis Cholesky (1875-1918) was a French military officer involved in geodesy and surveying in Crete and North-Africa just before World War I. He developed the method, now named after him, to compute solutions to the normal equations for some least squares data fitting problems arising in geodesy. His work was posthumously published on his behalf in 1924 by a fellow officer, Benoit, in the "Bulletin Geodesique".

4.4 Gauss-Newton optimization and Separable Least Squares

Given the parameterizations of the state space model, we can use a gradient based optimization method to optimize the model. Many different methods exist. In this section we restrict ourselves to the use of a Gauss-Newton based type of optimization strategy. The Gauss-Newton optimization is given by the following equation. Let \( \hat{\theta}(k-1) \) be the estimated parameter vector at iteration \( k-1 \), then the new vector \( \hat{\theta}(k) \) is found using the following updating scheme:

\[
\hat{\theta}(k) = \hat{\theta}(k-1) - \mu(k)H^{-1}(k-1)(\hat{\theta}(k-1))V'(\hat{\theta}(k-1))
\]

(4.24)

Here \( H(\hat{\theta}(k-1)) \) is an approximation of the Hessian \( V''(\hat{\theta}(k-1)) \) of the cost-function and \( V'(\hat{\theta}(k-1)) \) its gradient. The factor \( \mu \) is the step size parameter, which can be determined separately after the calculation of the approximate Hessian and gradient to further optimize the cost-function[16].

The cost-function is commonly defined as the squared output-error or prediction-error. With the definition

\[
Y_N = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} \quad \text{and} \quad \hat{Y}_N(\theta) = \begin{bmatrix} \hat{y}(1,\theta) \\ \hat{y}(2,\theta) \\ \vdots \\ \hat{y}(N,\theta) \end{bmatrix}
\]

the error \( E(\theta) \) becomes

\[
E(\theta) = Y_N - \hat{Y}_N(\theta)
\]

(4.25)

The cost-function can then be written as

\[
V_N(\theta_n) = \frac{1}{2}E_N^T(\theta)E_N(\theta)
\]
Otto Hesse (1811-1874) studied under Jacobi at Königsberg and spent a while as a teacher of physics and chemistry before he graduated from Königsberg in 1840. Hesse's main work was in the development of the theory of algebraic functions and the theory of invariants. He introduced the Hessian determinant in a paper in 1842 when he investigated cubic and quadratic curves.

Using the framework for Matrix Calculus as given in appendix A.5 and [15] the gradient of the cost function, $V'_N(\theta)$ is given by

$$
V'_N(\theta) = \frac{1}{2} \frac{\partial E^T_N(\theta)}{\partial \theta} E_N(\theta) + \frac{1}{2} \left( I_p \otimes E^T_N(\theta) \right) \frac{\partial E_N(\theta)}{\partial \theta} \\
= \frac{\partial E^T_N(\theta)}{\partial \theta} E_N(\theta) \\
= \left( \frac{\partial E_N(\theta)}{\partial \theta^T} \right)^T E_N(\theta) \\
= \Psi^T_N(\theta) E_N(\theta)
$$

(4.26)

The Hessian $V''_N(\theta)$ is equal to

$$
V''_N(\theta) = \frac{\partial^2 E^T(\theta)}{\partial \theta^T \partial \theta} \left( I_p \otimes E_N(\theta) \right) + \frac{\partial E^T_N(\theta)}{\partial \theta} \frac{\partial E_N(\theta)}{\partial \theta^T} \\
= \frac{\partial^2 E^T(\theta)}{\partial \theta^T \partial \theta} \left( I_p \otimes E_N(\theta) \right) + \left( \frac{\partial E_N(\theta)}{\partial \theta^T} \right)^T \frac{\partial E_N(\theta)}{\partial \theta^T} \\
= \frac{\partial^2 E^T_N(\theta)}{\partial \theta^T \partial \theta} \left( I_p \otimes E_N(\theta) \right) + \Psi^T_N(\theta) \Psi_N(\theta)
$$

(4.27)

where $\Psi_N(\theta)$ is defined as

$$
\Psi_N(\theta) = \frac{\partial E_N}{\partial \theta^T}
$$

(4.28)

The Gauss-Newton approximation consists of neglecting the first term of this expression. This is justified, since in the neighborhood of the optimum, the second derivative of the error and the error itself are weakly correlated. Therefore in the optimum the cost-function can be approximated by a quadratic function. We thus have

$$H_N(\theta) = \Psi^T_N(\theta) \Psi_N(\theta)
$$

When we use the separation of parameters from section 4.3 the estimated output becomes

$$\hat{Y}_N = \Phi(\theta_n) \theta_I$$
and the estimation error error
\[ E_N(\theta_n, \theta_l) = Y_N - \Phi(\theta_n)\theta_l \]
where we used the definition
\[ \Phi(\theta_n) = \begin{bmatrix} 
\phi_0(\theta_n) \\
\phi_1(\theta_n) \\
\vdots \\
\phi_N(\theta_n) 
\end{bmatrix} \]
and \( \phi_k(\theta_n) \) is given by equation (4.9). The Separable Least Squares theorem 4.1 then gives us the following result:

(i). We can optimize \( \theta_n \) using the estimation error
\[ E_N(\theta_n, \theta_l) = Y_N - \Phi(\theta_n)\Phi^\dagger(\theta_n)Y_N \]  \hspace{1cm} (4.29)

(ii). For every \( \theta_n \) the optimal \( \theta_l \) is found directly as
\[ \theta_l = \Phi^\dagger(\theta_n)Y_N \]  \hspace{1cm} (4.30)

The advantage of using SLS over a full parameterization is the lower dimension of the search space. For a full parameterization of the state space matrices by taking all the elements of the matrices \( A, B, C, D \) and \( x_0 \) the optimization needs to search the optimum in a \( n^2 + nm + n\ell + m\ell + n \) dimensional space. Even with a minimal canonical form, such as the output normal form of section 4.3.2, the optimization still needs to search over \( n\ell + nm + m\ell + n \) parameters. Using the proposed SLS method and a minimal canonical form the optimization of only \( n\ell \) parameters is necessary.

This reduction of the search space comes at a cost. The calculation of the cost-function is more complex, due to the pseudo inverse. However, in section 4.5.2 and [74] it is shown that the SLS scheme and the non-SLS scheme have similar properties with respect to numerical complexity. The higher complexity of the cost-function is partly compensated for by the lower number of iterations that is needed to reach the optimum. Moreover, it has been proven in [17] that in the SLS optimization the Hessian matrix is better conditioned than in an equivalent optimization without SLS.

### 4.5 Examples of SLS optimization

In this section we illustrate the principle of Separable Least Squares with two examples. In the first example we show an optimization problem in two parameters and how it can be reduced to an one-dimensional optimization problem. In the second example we do a comparison between SLS optimization and non-SLS optimization.
Sir Isaac Newton (1643-1727) was born on Christmas Day in Woolsthorpe. He entered Trinity College, Cambridge in 1661. In 1665 the plague closed the University. At home, Newton laid the foundation for differential and integral calculus. Newton's greatest achievement was his work in physics and celestial mechanics, which culminated in the theory of universal gravitation. In 1687 Newton published the "Philosophiae naturalis principia mathematica" or "Principia" as it is always known. After suffering a nervous breakdown in 1693, Newton retired from research to take up a government position in London becoming Warden of the Royal Mint (1696) and Master (1699). In 1703 he was elected president of the Royal Society. He was knighted in 1708 by Queen Anne.

4.5.1 Example of parameter space reduction

Regard the system from example 4.1. From this system, we obtain a set of data \( u(k), y(k), k \in [0, N_m - 1] \). Using this set of data we try to optimize a state space model. The model is parameterized using the output-normal form for \( A \) and \( C \), as explained in section 4.3.2. The parameter vectors \( \theta_n \) and \( \theta_l \) that describe the above system are given by

\[
\theta_n = \begin{bmatrix} 0.8824 \\ -0.7 \end{bmatrix} \quad \theta_l = \begin{bmatrix} 1.4003 \\ 4.1133 \\ 0 \end{bmatrix}
\] (4.31)

For visualization we restrict all but two of the parameters to their true values. The two free parameters were chosen to be the first element of \( \theta_n \) and the first element of the \( \theta_l \).

\[
\hat{\theta}_n = \begin{bmatrix} \alpha \\ -0.7 \end{bmatrix} \quad \hat{\theta}_l = \begin{bmatrix} \beta \\ 4.1133 \\ 0 \end{bmatrix}
\] (4.32)

First we do a parameter optimization with respect to the two free parameters. Figure 4.1 shows the surface spanned by the cost-function. The figure shows several local minima, corresponding to different values of \( \alpha \). In the \( \beta \) direction the function is parabolic. This shows that the model is indeed linear in this parameter.

Next we do the optimization using Separable Least Squares. Because the model is linear in \( \beta \) we can find, for every value of \( \alpha \), the optimal value by calculating the least squares estimate. This reduces the parameter space in this example to an one-dimensional space. Figure 4.2 shows the cost-function with respect to \( \alpha \) using the SLS method to determine the optimal \( \beta \) for every value of \( \alpha \). The function has its
minimum at 0.8824 as expected. Moreover, the function does not contain more local minima than the original two-dimensional cost-function, because SLS only projects out the search directions in which no local minima are present. This projection does not alter the remaining search directions.

4.5.2 Comparison between non-SLS and SLS based optimization

In this section we show a comparison between the optimization of state space models with and without the SLS technique. We used the output normal parameterization in both cases. The model we used is the second order model from example 4.1.

\[
x(k + 1) = \begin{bmatrix} 1.5000 & -0.7000 \\ 1.0000 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)
\]

(4.33)

\[
y(k) = \begin{bmatrix} 1.0000 & 0.5000 \end{bmatrix} x(k) + \begin{bmatrix} 0 \end{bmatrix} u(k)
\]

(4.34)

We used a Levenberg-Marquardt technique for the optimization. This method is identical to the Gauss-Newton method, with the exception that the approximate Hessian is regularized when it tends to become singular. This makes the method more robust. We optimized the model in a Monte Carlo simulation of 100 trials. At every trial a new set of input and output data and a new initial estimate was generated. The input
Figure 4.2: Optimization of the model with two free parameters using SLS.

was taken to be normally distributed white noise of unit variance. No disturbance was added.

At every trial we optimized a model using the SLS method and the non-SLS optimization. In both cases, the output normal parameterization was used. With this parameterization the SLS optimization used two parameters. The optimization without SLS required 5 parameters.

Of the 100 trials, the SLS optimization found the global optimum in all cases. The non-SLS optimization converged to a local optimum in 4 cases. This is surprising, since the cost-function for SLS has the same amount of local minima. However, it can be due to numerical problems in the implementation of the optimization routine. We did not investigate this further.

Figure 4.3 shows the number of cost-function evaluations. The left histogram shows the number of evaluations during the optimization without SLS. The right histogram shows this for the optimization with SLS. This shows the decrease of cost-function evaluations, that is caused by the lower dimension of the search space.

The lower number of cost-function evaluations with SLS is partly compensated by the more complex cost-function. This requires more computation time per evaluation. To investigate this effect we measured the number of floating point operations (FLOPS) and the CPU time that was spend during optimization at every trial. Figure 4.4 and 4.5 show the histograms for the CPU time and the number of FLOPS with both
4.6 Recursive optimization using SLS

Figure 4.3: Number of cost-function evaluations during the optimization with and without SLS.

optimization methods

Figure 4.4: CPU-time needed for the optimization with and without SLS.

From the example we can see that although the SLS optimization requires more computations per optimization step, the total time that is needed for the optimization is smaller than with the ordinary optimization technique without SLS. This is due to the fact that less iterations are necessary to arrive at the optimum.

4.6 Recursive optimization using SLS

In this section, we investigate how the proposed algorithm can be used in a recursive framework.

In the recursive problem setup we assume that at every time step the value of the
estimated parameters $\hat{\theta}_n(k - 1)$ and $\hat{\theta}_t(k - 1)$ which are estimated using the data sequences up to time instant $k - 1$ are available. We want to update our estimates to $\hat{\theta}_n(k)$ and $\hat{\theta}_t(k)$, when a new pair of input/output data $y(k)/u(k)$ arrives, without re-processing the whole data sequence up to time instant $k - 1$ again.

The cost-function to be optimized at every time instant is denoted by

$$ V_k(\theta_n) = \frac{1}{2} E_k(\theta_n)^T E_k(\theta_n) $$

The parameter updating function now becomes:

$$ \hat{\theta}_n(k) = \hat{\theta}_n(k - 1) - \mu H_k^{-1}(\hat{\theta}_n(k - 1)) V'_k(\hat{\theta}_n(k - 1)) $$

When we assume that the parameter vector $\theta_n(k - 1)$ minimizes $V_{k-1}(\theta)$, the gradient $V'_{k-1}(\theta(k - 1))$ is zero. This reduces $V'_k(\hat{\theta}_n(k - 1)$ to $\epsilon_k(\theta_n(k - 1))$ and the recursion becomes

$$ \hat{\theta}_n(k) = \hat{\theta}_n(k - 1) - \mu H_k^{-1}(\hat{\theta}_n(k - 1)) \psi(\hat{\theta}_n(k - 1)) \epsilon_k(\theta_n(k - 1)) $$

with

$$ \epsilon_k(\theta_n) = y(k) - \phi(\hat{\theta}_n(k - 1)) \hat{\theta}_t(k) $$

$$ \psi(\hat{\theta}_n(k - 1)) = \frac{\partial \epsilon_k(\hat{\theta}_n(k - 1))}{\partial \theta_n} $$

We introduce the following notation for convenience. We will use $\hat{H}(k)$ as a short notation for $H_k(\hat{\theta}_n(k - 1))$, $\hat{E}_k$ for $E_k(\hat{\theta}_n(k - 1))$ and $\epsilon_k$ instead of $\epsilon_k(\hat{\theta}_n(k - 1))$. Similarly we use $\hat{\Psi}_k$ to denote $\Psi_k(\hat{\theta}_n(k - 1))$, $\hat{\psi}_k$ to denote $\psi_k(\hat{\theta}_n(k - 1))$, $\hat{\Phi}_k$ for $\Phi_k(\hat{\theta}_n(k - 1))$ and $\phi_k$ for $\phi_k(\hat{\theta}_n(k - 1))$. The short-hand notation $\hat{A}_k$ denotes $A(\hat{\theta}_n(k - 1))$ and $\hat{C}_k$ is used for $C(\hat{\theta}_n(k - 1))$. 
4.6.1 Recursion of $H_k^{-1}(\hat{\theta}_n(k - 1))$

In this subsection we will first search for a recursive expression for the term $H_k^{-1}(\hat{\theta}_n(k - 1))$.

Define:

\[
R_n(k) = H_k^{-1}(\hat{\theta}_n(k - 1)) = \left(\Psi_k^T \hat{\Psi}_k\right)^{-1} = \left(\Psi_k(\hat{\theta}(k - 1))^T \Psi_k(\hat{\theta}(k - 1))\right)^{-1} = \left(\Psi_{k-1}(\hat{\theta}(k - 1))^T \Psi_{k-1}(\hat{\theta}(k - 1)) + \psi_k^T \hat{\psi}_k\right)^{-1}
\]

The matrix $\Psi_{k-1}(\hat{\theta}(k - 1))$ equals

\[
\frac{\partial E_{k-1}(\hat{\theta}_n(k - 1))}{\partial \theta_n^T}
\]

This is the derivative of the error with respect to parameters $\theta_n$ at the point $\hat{\theta}_n(k - 1)$, which is not available. However, during the recursion, at the previous time instance, $k - 1$, we have calculated the matrix

\[
\Psi_{k-1}(\hat{\theta}_n(k - 2)) = \frac{\partial E_{k-1}(\hat{\theta}_n(k - 2))}{\partial \theta_n^T}
\]

Therefore we use the approximation

\[
\Psi_{k-1}(\hat{\theta}_n(k - 1)) \approx \Psi_{k-1}(\hat{\theta}_n(k - 2)) = \hat{\Psi}_{k-1}
\]

This is a reasonable assumption when $\hat{\theta}_n(k)$ does not change rapidly between time steps.

Using the matrix inversion lemma from section A.12, we now get:

\[
R_n(k) = \left(\hat{\Psi}_{k-1}^T \hat{\Psi}_{k-1} + \hat{\psi}_k^T \hat{\psi}_k\right)^{-1} = \left(\hat{\Psi}_{k-1}^T \hat{\Psi}_{k-1}\right)^{-1}
\]

\[
- \left(\hat{\Psi}_{k-1}^T \hat{\Psi}_{k-1}\right)^{-1} \hat{\psi}_k^T \left(\hat{I}_p + \hat{\psi}_k^T \left(\hat{\Psi}_{k-1}^T \hat{\Psi}_{k-1}\right)^{-1} \hat{\psi}_k\right)^{-1} \hat{\psi}_k \left(\hat{\Psi}_{k-1}^T \hat{\Psi}_{k-1}\right)^{-1}
\]

\[
= R_n(k - 1) - R_n(k - 1) \hat{\psi}_k^T \left(\hat{I}_p + \hat{\psi}_k^T R_n(k - 1) \hat{\psi}_k\right)^{-1} \hat{\psi}_k R_n(k - 1)
\]
Since the parameter vector $\hat{\theta}_n(k)$ does change over time, it is common practice to introduce a constant forgetting factor $\nu_n$, valued between 0 and 1, that reduces the influence of the previous estimate $R_n(k-1)$ on the present estimate of $R_n(k)$.

\[
R_n(k) = \left( \nu_n(\hat{\Psi}^T_{k-1} \hat{\Psi}_k) + \hat{\psi}_k^T \hat{\psi}_k \right)^{-1} = \frac{1}{\nu_n} R_n(k-1) - \frac{1}{\nu_n} R_n(k-1) \hat{\psi}_k \left( I_{p_n} + \hat{\psi}_k^T \frac{1}{\nu_n} R_n(k-1) \hat{\psi}_k \right)^{-1} \hat{\psi}_k^T \frac{1}{\nu_n} R_n(k-1)
\]

In order to calculate the matrix $R_n(k)$ recursively we need estimates of $\hat{\psi}_k$. For the parameter updating function we also need $\epsilon(k)$. This in turn requires the estimate of $\hat{\phi}_k$ and $\theta_l(k)$. We will now derive recursive formulas for these four quantities.

### 4.6.2 Recursion of $\hat{\phi}_k$

Recalling equation (4.9), the matrix $\hat{\phi}_k$ is defined as

\[
\hat{\phi}_k = \left[ \hat{C}_k \hat{A}_k^{k-1} \Sigma_{\tau=0}^{k-1} u^T(\tau) \otimes \hat{C}_k \hat{A}_k^{k-1-\tau} u^T(k) \otimes I_l \right]
\]

This can be cast in a recursive form as follows:

\[
\hat{\phi}_k = \left[ \hat{C}_k \hat{A}_k^{k-1} \hat{A}_k \left( \Sigma_{\tau=0}^{k-2} u^T(\tau) \otimes \hat{C}_k \hat{A}_k^{k-1-\tau} \left( I_m \otimes \hat{A}_k \right) \right) + \left( u^T(k-1) \otimes \hat{C}_k \right) \right] u^T(k) \otimes I_l
\]

\[
\times \left[ \begin{array}{ccc} \hat{A}_k & 0 & 0 \\ 0 & I_m \otimes \hat{A}_k & 0 \\ 0 & 0 & 0 \end{array} \right] + \left[ \begin{array}{ccc} 0 & u^T(k-1) \otimes \hat{C}_k & u^T(k) \otimes I_l \end{array} \right]
\]

\[
= \phi_{k-1}(\theta_n(k-1)) P_1(k) + P_2(k)
\]

\[
\approx \phi_{k-1}(\theta_n(k-2)) P_1(k) + P_2(k)
\]

\[
= \hat{\phi}_{k-1} P_1(k) + P_2(k)
\]

where we used the approximation

\[
\phi_{k-1}(\hat{\theta}_n(k-1)) \approx \phi_{k-1}(\hat{\theta}_n(k-2))
\]

and the definitions

\[
P_1(k) = \left[ \begin{array}{ccc} \hat{A}_k & 0 & 0 \\ 0 & I_m \otimes \hat{A}_k & 0 \\ 0 & 0 & 0 \end{array} \right] \quad P_2(k) = \left[ \begin{array}{ccc} 0 & u^T(k-1) \otimes \hat{C}_k & u^T(k) \otimes I_l \end{array} \right]
\]

(4.39)
4.6.3 Recursion of $\hat{\theta}_l(k)$

The linear parameters in the iterative case were found using a least squares solution

$$\hat{\theta}_l(k) = \Phi_k^T Y_k$$  

(4.40)

$$= \left(\Phi_k^T \Phi_k\right)^{-1} \Phi_k^T Y_k$$  

(4.41)

$$= R_l(k) K_l(k)$$  

(4.42)

with $R_l(k)$ and $K_l(k)$ defined as follows

$$R_l(k) = \left(\Phi_k^T \Phi_k\right)^{-1}$$

$$K_l(k) = \Phi_k^T Y_k$$

Using the standard recursive least squares algorithm [59] we obtain:

$$L(k) = R_l(k-1)\hat{\phi}_l^T(k) \left(\nu_l I_l + \hat{\phi}_k R_l(k-1) \hat{\phi}_k^T\right)^{-1}$$  

(4.43)

$$R_l(k) = \frac{1}{\nu_l} \left( R_l(k-1) - L(k) \hat{\phi}_k R_l(k-1) \right)$$  

(4.44)

$$\hat{\theta}_l(k) = \hat{\theta}_l(k-1) + L(k) \left( y(k) - \hat{\phi}_k \hat{\theta}_l(k-1) \right)$$  

(4.45)

Where $\nu_l$ is a forgetting factor $\nu_l \in [0, 1]$.

4.6.4 Recursion of $\hat{\psi}_k$

Using the definition (4.37) we formulate

$$\hat{\psi}_k = \frac{\partial \hat{\psi}_k}{\partial \theta_n^T}$$  

(4.46)

$$= \frac{\partial (y(k) - \hat{\phi}_k \hat{\theta}_l(k))}{\partial \theta_n^T}$$  

(4.47)

$$= -\frac{\partial \hat{\phi}_k}{\partial \theta_n^T} (I_p \otimes \hat{\theta}_l(k)) - \hat{\phi}_k \frac{\partial \hat{\theta}_l(k)}{\partial \theta_n^T}$$  

(4.48)

The first term in this equation is easy to develop. The derivative of $\hat{\phi}_k$ can be written as:

$$\frac{\partial \hat{\phi}_k}{\partial \theta_n^T} = \frac{\partial \hat{\phi}_k}{\partial \theta_n^T} \left(I_p \otimes P_1(k)\right) + \hat{\phi}_k \frac{\partial P_1(k)}{\partial \theta_n^T} + \frac{\partial P_2(k)}{\partial \theta_n^T}$$  

(4.49)
The derivatives $\frac{\partial P_k(k)}{\partial \theta_n^T}$ and $\frac{\partial P_k(k)}{\partial \theta_n^T}$ have to be calculated at every step and involve the calculation of the derivatives of $\hat{A}_k$ and $\hat{C}_k$ with respect to $\theta_n$.

The second term in equation (4.46) is more difficult to calculate, since this requires the derivative of $\hat{r}_i(k)$. With equation (4.42) for $\hat{r}_i(k)$, we obtain:

$$\frac{\partial \hat{r}_i(k)}{\partial \theta_n^T} = \frac{\partial R_i(k)}{\partial \theta_n^T} \left( I_{p_n} \otimes K_i(k) \right) + R_i(k) \frac{\partial K_i(k)}{\partial \theta_n^T} \quad (4.50)$$

For the derivative of the $R_i(k)$ we find:

$$\frac{\partial R_i(k)}{\partial \theta_n^T} = \frac{\partial (\hat{\Phi}_i^T \hat{\Phi}_k)}{\partial \theta_n^T}^{-1} \quad (4.51)$$

$$= - (\hat{\Phi}_i^T \hat{\Phi}_k)^{-1} \left( \frac{\partial (\hat{\Phi}_i^T \hat{\Phi}_k)}{\partial \theta_n^T} \right) \left( I_{p_n} \otimes (\hat{\Phi}_i^T \hat{\Phi}_k)^{-1} \right) \quad (4.52)$$

With

$$\frac{\partial \hat{\Phi}_i^T \hat{\Phi}_k}{\partial \theta_n^T} = \frac{\partial \hat{\Phi}_i^T}{\partial \theta_n^T} \left( I_{p_n} \otimes \hat{\Phi}_k \right) + \hat{\Phi}_k^T \frac{\partial \hat{\Phi}_k}{\partial \theta_n^T} \quad (4.53)$$

this becomes

$$\frac{\partial R_i(k)}{\partial \theta_n^T} = - R_i(k) \left( \frac{\partial \hat{\Phi}_i^T}{\partial \theta_n^T} \left( I_{p_n} \otimes \hat{\Phi}_k \right) + \hat{\Phi}_k^T \frac{\partial \hat{\Phi}_k}{\partial \theta_n^T} \right) \left( I_{p_n} \otimes R_i(k) \right) \quad (4.54)$$

For the derivative of the $K_i(k)$ we find:

$$\frac{\partial K_i(k)}{\partial \theta_n^T} = \frac{\partial \Phi_k^T}{\partial \theta_n^T} (I_{p_n} \otimes Y_k) \quad (4.55)$$

Let us now define

$$C_i(k) = \frac{\partial \hat{\Phi}_k^T}{\partial \theta_n^T} (I_{p_n} \otimes \hat{\Phi}_k) + \hat{\Phi}_k^T \frac{\partial \hat{\Phi}_k}{\partial \theta_n^T} \quad (4.56)$$

$$D_i(k) = \frac{\partial \Phi_k^T}{\partial \theta_n^T} (I_{p_n} \otimes Y_k) \quad (4.57)$$

We now have

$$\frac{\partial \hat{r}_i(k)}{\partial \theta_n^T} = - R_i(k) C_i(k) \left( I_{p_n} \otimes R_i(k) \right) \left( I_{p_n} \otimes K_i(k) \right) + R_i(k) D_i(k) \quad (4.58)$$

$$= R_i(k) \left( D_i(k) - C_i(k) (I_{p_n} \otimes \hat{\theta}_k(k)) \right) \quad (4.59)$$
The term $R_t(k)$ has already been defined in equation (4.43) for the recursion of $\hat{\theta}_t(k)$. Rests us to find recursive expressions for $C_t(k)$ and $D_t(k)$.

\[
C_t(k) = \frac{\partial \hat{\Phi}_k^T}{\partial \theta_n^T} (I_{p_n} \otimes \hat{\Phi}_k) + \hat{\Phi}_k^T \frac{\partial \hat{\Phi}_k}{\partial \theta_n^T} \\
= \frac{\partial \hat{\Phi}_{k-1}^T}{\partial \theta_n^T} (I_{p_n} \otimes \hat{\Phi}_{k-1}) + \hat{\Phi}_{k-1}^T \frac{\partial \hat{\Phi}_{k-1}}{\partial \theta_n^T} + \frac{\partial \hat{\phi}_k^T}{\partial \theta_n^T} (I_{p_n} \otimes \hat{\phi}_k) + \hat{\phi}_k^T \frac{\partial \hat{\phi}_k}{\partial \theta_n^T} \\
= C_t(k-1) + \frac{\partial \hat{\phi}_k^T}{\partial \theta_n^T} (I_{p_n} \otimes \hat{\phi}_k) + \hat{\phi}_k^T \frac{\partial \hat{\phi}_k}{\partial \theta_n^T} \tag{4.60}
\]

\[
D_t(k) = \frac{\partial \hat{\phi}_k^T}{\partial \theta_n^T} (I_{p_n} \otimes y_k) \\
= \frac{\partial \hat{\phi}_{k-1}^T}{\partial \theta_n^T} (I_{p_n} \otimes y_{k-1}) + \frac{\partial \hat{\phi}_k^T}{\partial \theta_n^T} (I_{p_n} \otimes y_k) \\
= D_t(k-1) + \frac{\partial \hat{\phi}_k^T}{\partial \theta_n^T} (I_{p_n} \otimes y_k) \tag{4.61}
\]

The recursive SLS optimization for discrete time state space systems is summarized in algorithm 4.2.

### 4.7 Separable Least Squares and continuous time state space models

Also in the identification of continuous time systems it is possible to use the technique of Separable Least Squares. In recent works[38, 40, 41, 47, 49, 78] and chapter 3, methods for identifying continuous time systems from sampled data are proposed. Each of these methods can be used to obtain an initial estimate for a non-linear optimization, using the SLS technique proposed in this section.

In this section we investigate the use of Separable Least Squares in the identification of continuous time models. We again show how it is possible in this case to separate the parameter vector into a group of linear parameters and a group of non-linear parameters. For the continuous time case we have the following model:

\[
\dot{x}(t, \theta) = A(\theta)x(t, \theta) + B(\theta)u(t) \tag{4.62}
\]
\[
y(t, \theta) = C(\theta)x(t, \theta) + D(\theta)u(t) + v(t) \tag{4.63}
\]
\[
x(0, \theta) = x_0(\theta) \tag{4.64}
\]
Algorithm 4.2  Discrete time recursive optimization of a state space model with SLS

For $k = 0, ..., N_m - 1$ do:

(1). Calculate $\theta_l(k)$:

$$
\begin{align*}
P_1(k) &= \begin{bmatrix} \hat{A}_k & 0 & 0 \\ 0 & I_m \otimes \hat{A}_k & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
P_2(k) &= \begin{bmatrix} 0 & u^T(k-1) \otimes \hat{C}_k & u^T(k) \otimes I_l \\ 0 & 0 & 0 \end{bmatrix} \\
\dot{\phi}_k &= \dot{\phi}_{k-1} P_1(k) + P_2(k) \\
L(k) &= R_l(k-1) \dot{\phi}_T(k) \left( \nu_l I_I + \dot{\phi}_k R_l(k-1) \dot{\phi}_k^T \right)^{-1} \\
R_l(k) &= \frac{1}{\nu_l} \left( R_l(k-1) - L(k) \dot{\phi}_k R_l(k-1) \right) \\
\theta_l(k) &= \theta_l(k-1) + L(k) \left( y(k) - \dot{\phi}_k \theta_l(k-1) \right)
\end{align*}
$$

(2). Calculate $\frac{\partial \hat{A}_k}{\partial \theta_n}$ and $\frac{\partial \hat{C}_k}{\partial \theta_n}$

(3). Calculate $\frac{\partial P_1(k)}{\partial \theta_n}$ and $\frac{\partial P_2(k)}{\partial \theta_n}$

(4). Calculate $\theta_n(k)$:

$$
\begin{align*}
\epsilon(k) &= y(k) - \dot{\phi}_k \theta_l(k) \\
\frac{\partial \dot{\phi}_k}{\partial \theta_n^T} &= \frac{\partial \dot{\phi}_{k-1}}{\partial \theta_n^T} (I_{p_n} \otimes P_1(k)) + \dot{\phi}_k \frac{\partial P_1(k)}{\partial \theta_n^T} + \frac{\partial P_2(k)}{\partial \theta_n^T} \\
C_l(k) &= C_l(k-1) + \frac{\partial \dot{\phi}_k}{\partial \theta_n^T} (I_{p_n} \otimes \dot{\phi}_k) + \dot{\phi}_k \frac{\partial \dot{\phi}_k}{\partial \theta_n^T} \\
D_l(k) &= D_l(k-1) + \frac{\partial \dot{\phi}_k}{\partial \theta_n^T} (I_{p_n} \otimes y_k) \\
\frac{\partial \theta_l(k)}{\partial \theta_n^T} &= R_l(k) \left( D_l(k) - C_l(k) (I_{p_n} \otimes \theta_l(k)) \right) \\
R_n(k) &= \frac{1}{\nu_n} R_n(k-1) \\
&= \frac{1}{\nu_n} R_n(k-1) \dot{\psi}_k \left( I_{p_n} + \dot{\psi}_k^T \frac{1}{\nu_n} R_n(k-1) \dot{\psi}_k \right)^{-1} \dot{\psi}_k^T \frac{1}{\nu_n} R_n(k-1) \\
\dot{\theta}_n(k) &= \dot{\theta}_n(k-1) - \mu R_n(k) \dot{\psi}_k^T \epsilon_k
\end{align*}
$$
with state \( x(t, \theta) \in \mathbb{R}^n \), input \( u(t) \in \mathbb{R}^m \) and output \( y(t, \theta) \in \mathbb{R}^l \). \( v(t) \) is the measurement noise which is independent of \( u(t) \). Since we do not have to assume white noise properties here, we will not use the Wiener process description of chapter 3 for \( v(t) \). The system matrices \( A, B, C \) and \( D \) and the initial state \( x_0 \) are of appropriate dimensions. The system is assumed to be stable and minimal.

We assume that the signals are measured at discrete time instances \( t_k \) for \( k = 0, 1, \ldots, N \). Written in explicit form \( y(t_k) \) becomes:

\[
y(t_k) = C e^{A(t_k)} x(0) + C \int_0^t e^{A(t-\tau)} Bu(\tau) d\tau + Du(t_k)
\]

As in the discrete time case this can be rewritten to

\[
y(t_k) = C e^{At_k} x(0) + C \int_0^{t_k} u^T(\tau) \otimes e^{A(t_k-\tau)} d\tau \text{vec}(B) + u^T(\tau) \otimes I_l)\text{vec}(D)
\]

\[
= \phi_k(\theta_n) \theta_l
\]

with

\[
\phi_k(\theta_n) = \begin{bmatrix} C e^{At_k} & C \int_0^{t_k} u^T(\tau) \otimes e^{A(t_k-\tau)} d\tau & u^T(t) \otimes I_l \end{bmatrix}
\]

and

\[
\theta_l = \begin{bmatrix} x_0 \\
\text{vec}(B) \\
\text{vec}(D)
\end{bmatrix}
\]

This results in the model equation

\[ \hat{Y}_N = \Phi_N(\theta_n) \theta_l \]

As in the discrete time case the matrices \( A \) and \( C \) can be parameterized independently from \( B, D \) and \( x_0 \) and the model depends linearly on the elements of the latter.

### 4.7.1 Continuous time output normal parameterization

As in the discrete time case we search for a parameterization of \( A \) and \( C \) for the continuous time model with a minimal number of parameters.

The tri-diagonal parameterization, that was introduced in section 4.3 can be used in continuous time without modification.

Also the output normal form, as used in the discrete time case can be applied here again. The output normal form allows us to parameterize \( A \) and \( C \) independently
from $B$, $D$ and $x_0$ using only $n \times \ell$ parameters. The continuous time output normal parameterization is however different from the discrete time one.

The MIMO output normal form for continuous time systems is described in [16, 34]. The matrices $A$ and $C$ are first transformed to output normal form. This means that the observability Gramian $W_0$ is identity. The continuous time observability Gramian is given by the following Lyapunov equation

$$A^T W_0 + W_0 A = -C^T C$$

The minimal parameterization of an output normal state space model is given by the following matrices.

$$C = \begin{bmatrix}
    c_{11} & 0 & \cdots & \cdots & 0 \\
    \vdots & \ddots & \ddots & \vdots \\
    c_{1\ell} & \cdots & c_{\ell} & 0 & \cdots & 0
\end{bmatrix}$$

and

$$A_{ss} = \begin{bmatrix}
    0 & a_{11} & \cdots & a_{1\ell} \\
    -a_{11} & 0 & a_{21} & \cdots & a_{2\ell} \\
    \vdots & \ddots & \ddots & \ddots & \ddots \\
    -a_{1\ell} & \cdots & 0 & a_{(n-\ell)\ell} & \cdots & a_{(n-\ell)\ell} \\
    -a_{2\ell} & -a_{(n-\ell)\ell} & \ddots & \cdots & \ddots & \ddots \\
    \vdots & \ddots & \ddots & \ddots & \ddots & \cdots & a_{(n-1)\ell} \\
    -a_{(n-\ell)\ell} & \cdots & -a_{(n-1)\ell} & \cdots & \cdots & \cdots & 0
\end{bmatrix}$$

with

$$A = -\frac{1}{2} C^T C + A_{ss}$$

In order to make the parameterization unique the elements $a_{21}$ and $c_{11}$ are restricted to be positive valued.

This parameterization requires $n\ell$ parameters to describe $A$ and $C$. The vector $\theta_n$ contains the stacking of the elements $a_{ij}$ and $c_{ij}$.

A method to obtain the above parameterization from an ordinary set of state space matrices is described in algorithm 4.3.

### 4.7.2 Recursive optimization of continuous time model

Here we briefly show the optimization of continuous time models. Since the optimization is very similar to the discrete time case we only stress the differences with
Algorithm 4.3 Output normal parameterization of a continuous time state space system

(1). Compute the observability Gramian $W_0$ of $A$ and $C$. Perform a Cholesky factorization of this Gramian $W_0 = Q_1Q_1^T$ and apply the transformation $A_1 = Q_1AQ_1^{-1}$ and $C_1 = CQ_1^{-1}$. The observability Gramian of $(A_1,C_1)$ now equals identity.

(2). Compute a LQ factorization of this observability matrix $C = LQ_2$. Then perform the transformation $A_2 = Q_2AQ_2^T$ and $C_2 = CQ_2^T$. $C$ is now a lower triangular matrix.

(3). Divide $A$ as follows

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

where $A_{12} \in R^{p \times p}$ and, $A_{21} \in R^{p \times p}$. Compute a series of Givens-rotations to bring $A_{21}$ in upper triangular form and combine these in one orthogonal transformation $Q_3$. Make $A_3 = Q_3AQ_3^T$ and $C_3 = CQ_3^T$.

(4). Construct $Q_4$ as a diagonal matrix with elements $-1$ and $1$, according to the signs of the elements on the diagonal $a_{i1}$ and $c_{11}$ Apply the transformation $A_4 = Q_4A_3Q_4$ and $C_4 = C_3Q_4$. The matrices $A_4$ and $C_4$ are now in the desired form.

(5). Finally collect the appropriate entries of $A_4$ and $C_4$ in the vector $\theta_n$. 
discrete time optimization. The cost function we want to minimize is:

\[ V_k(\theta_n) = \frac{1}{2} E_k(\theta_n) E_k^T(\theta_n) \]

The Gauss-Newton parameter updating step is given by

\[ \theta_n(k + 1) = \theta_n(k) + \mu H_k^{-1}(k) V_k'(k) \]

with

\[ H_k(k) = \hat{\Psi}_k^T \hat{\Psi}_k \]
\[ V_k'(k) = \hat{\Psi}_k^T \hat{E}_k \]

Instead of equation (4.9) we have:

\[
\begin{align*}
\hat{\phi}_k(\theta_n) &= \left[ Ce^{\hat{A}_k t_n} \left| \hat{C}_k \int_0^{t_n} u^T(\tau) \otimes e^{\hat{A}_k (t_n - \tau)} d\tau \right| u^T(t_n) \otimes I_t \right] \\
&= \left[ \hat{C}_k \int_0^{t_{k-1}} u^T(\tau) \otimes e^{A(t_{k-1} - \tau)} d\tau (I_m \otimes e^{\hat{A}_k (t_{k-1} - t_n)}) \\
&\quad + \hat{C}_k \int_{t_{k-1}}^{t_k} u^T(\tau) \otimes e^{\hat{A}_k (t_k - \tau)} d\tau \\
&\quad \left| u^T(t_k) \otimes I_t \right] \\
&= \phi_k(\theta_n(k-1)) \left[ e^{\hat{A}_k (t_k - t_{k-1})} 0 0 \\
&0 (I_m \otimes e^{\hat{A}_k (t_k - t_{k-1})}) 0 \\
&0 0 0 \right] \\
&+ \left[ 0 \hat{C}_k \int_{t_{k-1}}^{t_k} u^T(\tau) \otimes e^{\hat{A}_k (t_k - \tau)} d\tau \right] \left[ u^T(t_k) \otimes I_t \right] \\
&\approx \phi_{k-1}(\theta_n(k-2)) P_1(k) + P_2(k) \\
&= \hat{\phi}_k P_1(k) + P_2(k)
\end{align*}
\]

with

\[
\begin{align*}
P_1(k) &= \left[ e^{\hat{A}_k (t_k - t_{k-1})} 0 0 \\
&0 (I_m \otimes e^{\hat{A}_k (t_k - t_{k-1})}) 0 \\
&0 0 0 \right] \\
P_2 &= \left[ 0 \hat{C}_k \int_{t_{k-1}}^{t_k} u^T(\tau) \otimes e^{\hat{A}_k (t_k - \tau)} d\tau \right] \left[ u^T(t_k) \otimes I_t \right]
\end{align*}
\]

The calculation of \( \theta_t \) proceeds in the same way as the discrete time counterpart, since the equation

\[ \hat{Y}_k = \hat{\Phi}_k \hat{\theta}_t \]
still holds.

Also the calculation of

\[ \frac{\partial \hat{P}_k}{\partial \hat{\theta}^T_n} = \frac{\partial \hat{\phi}_k}{\partial \hat{\theta}^T_n} (I_{p_n} \otimes P_1(k)) + \hat{\phi}_k \frac{\partial P_1(k)}{\partial \hat{\theta}^T_n} + \frac{\partial P_2(k)}{\partial \hat{\theta}^T_n} \]  \hspace{1cm} (4.65)

and

\[ \frac{\partial \hat{\theta}_l(k)}{\partial \hat{\theta}^T_n} = R_l(k) \left( D_l(k) - C_l(k)(I_{p_n} \otimes R_l(k)K_l(k)) \right) \] \hspace{1cm} (4.66)

remains the same. The matrices \( \frac{\partial \hat{\phi}_k}{\partial \hat{\theta}_n} \) and \( \frac{\partial \hat{\phi}_k}{\partial \hat{\theta}_n} \) however depend on the parameterization and need to be calculated in a different manner.

The calculation of \( \psi_k(\theta_n) \) is again identical to the discrete time case because of the equation

\[ \hat{\psi}_k(\theta_n) = \frac{\partial \hat{\psi}_k}{\partial \theta^T_n} = \frac{\partial \hat{\phi}_k}{\partial \theta^T_n} \theta_l(k) - (I_{p_n} \otimes \hat{\phi}_k) \frac{\partial \hat{\theta}_l(k)}{\partial \theta^T_n} \] \hspace{1cm} (4.67)

### 4.8 prediction-error optimization and SLS

In the previous sections we have assumed the state space model in output-error form. The criterion that was optimized was the output error. In this section we briefly show how the Separable Least Squares technique can be applied in the optimization of the innovations model form. With the innovations model form, the optimization criterion becomes the prediction-error.

For the prediction-error optimization the system is assumed to be in innovations form:

\[ x(k+1, \theta) = A(\theta)x(k, \theta) + B(\theta)u(k) + Ke(k) \] \hspace{1cm} (4.68a)
\[ y(k, \theta) = C(\theta)x(k, \theta) + D(\theta)u(k) + e(k) \] \hspace{1cm} (4.68b)
\[ x(0, \theta) = x_0(\theta) \] \hspace{1cm} (4.68c)

This description can be rewritten as (with implicit dependency on \( \theta \))

\[ x(k+1) = \left( A - KC \right)x(k) + \left( B - KD \right)u(k) + Ky(k) \]
\[ \hat{y}(k) = Cx(k) + Du(k) + e(k) \]
\[ x(0) = x_0 \]

Since the innovation \( e(k) \) is unknown, the estimated output \( \hat{y}(k) \) is given by

\[ \hat{y}(k) = Cx(k) + Du(k) \]
Using this description, the prediction-error optimization problem is given by

\[
\hat{\theta} = \arg\left(\min_\theta \frac{1}{2} \sum_{k=0}^{N-1} ||e(k, \theta)||_2^2\right) = \arg\left(\min_\theta \frac{1}{2} \sum_{k=0}^{N-1} ||y(k) - \hat{y}(k, \theta)||_2^2\right) = \arg\left(\min_\theta \frac{1}{2} ||Y_N - \hat{Y}_N(\theta)||_2^2\right)
\]

We can proceed similar as before, with the matrices \(\tilde{A} = A - KC\) and \(\tilde{B} = B - KD\). \(\Phi(\theta_n)\) now has to be extended with an extra block column to account for \(y(k)\). The elements of the matrix \(K\) are included in the parameter vector \(\theta_i\).

Once the matrices \(\tilde{A}, C, K, \tilde{B}, D\) and \(x_0\) are estimated we can find \(A\) and \(B\) back as

\[
A = \tilde{A} + KC \\
B = \tilde{B} + KD
\]

### 4.9 Wiener models and SLS

In the setting of Separable Least Squares it is possible to include certain types of non-linear models. In this section we will show how the previously described SLS method is applicable to non-linear Wiener models. The Wiener model was introduced in section 2.10. It consists of a linear time-invariant model, followed by a static non-linearity (see figure 2.6). The output of the linear time-invariant part is of dimension \(\ell\). The non-linear part can have a different number of outputs, denoted by \(q\).

As in section (2.10) we model the non-linear function as a weighted sum of Chebychev polynomials. But many other types of basis functions are possible. The output of the Wiener model, can be written as

\[
Z_{cheb} = M(\hat{Y}_{cheb})\Theta
\]

with

\[
Z_{cheb} = \begin{bmatrix} z(0)^T \\ z(1)^T \\ \vdots \\ z(N-1)^T \end{bmatrix} \quad \hat{Y}_{cheb} = \begin{bmatrix} \hat{y}(0)^T \\ \hat{y}(1)^T \\ \vdots \\ \hat{y}(N-1)^T \end{bmatrix}
\]

and

\[
M(x) = \begin{bmatrix} T_1(x) & T_2(x) & \cdots & T_{n_a}(x) \end{bmatrix}
\]
$T_i(x)$ denotes the i-th order Chebychev polynomial. $\Theta$ is a matrix of dimension $(n_n \times \ell)$ consisting of weightings of the individual Chebychev polynomials.

We observe that the model is linear in the coefficients of the non-linear function and non-linear in the elements of the state space matrices. Therefore we can apply the Separable Least Squares principle to optimize only the matrices $A$, $B$, $C$ and $D$. Notice that due to the non-linearity at the output, the elements of $B$ and $D$ now are not in the linear part of the parameter vector.

### 4.10 Numerical example of recursive MIMO identification

In this section we give a numerical example of the use of recursive Separable Least Squares. This algorithm is implemented in the function `drslslin`. For this example we use the following second order system.

$$x(k+1) = \begin{bmatrix} 0.8 & 0.1 \\ -1 & 0.9 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x(k)$$

As input we use a Gaussian distributed white noise signal with unit variance. We generated 1000 samples of data from this system.

First we use the algorithm on a noise-free data-set. The step size $\mu$, the non-linear forgetting factor $\nu_n$ and the linear forgetting factor $\nu_l$ were set to respectively 1, 0.95 and 0.95. For the linear parameters only $B$ and $D$ were estimated. The initial state $x_0$ was disregarded. The initial value of the non-linear parameters were taken by perturbing the original $A$ and $C$ matrix with a Gaussian distributed random value of variance 0.2. Figure 4.6 shows a typical set of input an output data.

The result from the noise-free recursive optimization is shown in figure 4.7 The top plot shows the optimization of the non-linear parameters $\theta_n$. The second figure shows the optimization of the linear parameters $\theta_l$. The solid lines are the true parameters and the dashed lines are the estimated parameters. The last plot shows the value of the cost-function $V_k(\theta_n)$.

Next we add a disturbance to the output. This disturbance is a Gaussian white noise signal. The variance is taken such that the resulting SNR is 10 (20 dB). We performed a Monte Carlo simulation of 20 trials. At every trial a new set of input/output data is generated. Figure 4.8 shows the outcome of a typical trial. Figure 4.9 shows the Variance Accounted For of the models obtained from the different trials. The average VAF was 96.21 for output 1 and 98.74 for output 2.
Figure 4.6: Typical input and output signals of the noise-free experiment.
Figure 4.7: Recursive parameter estimate on a noise-free dataset.
Figure 4.8: Recursive parameter estimate on a noisy dataset.
4.11 Example of the use of continuous time SLS on a real life data set.

In this section we demonstrate the use of continuous time SLS on a real life data set. The data set is the same as we used to demonstrate CPO-MOEESP in section 3.10.2: the measurements on a MPV.

The optimization took 3138 function evaluations. After the optimization, the VAF on the evaluation set increased on average 2.15%.

In table 4.11 we have summarized the results from modeling the MPV data. The first row shows the results of the discrete time analysis from [14]. The continuous time analysis was described in section 3.10.2. The last row shows the results from the SLS optimization described above. All VAFs are calculated on validation data.

<table>
<thead>
<tr>
<th>Method</th>
<th>Output 1</th>
<th>Output 2</th>
<th>Output 3</th>
<th>Output 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete time</td>
<td>88.87</td>
<td>85.93</td>
<td>87.56</td>
<td>87.47</td>
</tr>
<tr>
<td>Continuous time</td>
<td>87.90</td>
<td>84.28</td>
<td>86.29</td>
<td>83.82</td>
</tr>
<tr>
<td>Continuous time + SLS</td>
<td>89.25</td>
<td>86.52</td>
<td>87.63</td>
<td>87.48</td>
</tr>
</tbody>
</table>

Table 4.1: Resulting VAFs from models of the MPV.
4.12 Conclusion

In this chapter we have discussed the identification of state space models using optimization of a cost-function. We have shown different ways to parameterize a state space model. The tridiagonal parameterization allows for a simple and direct parameterization of the state space model. However, it requires more parameters than strictly necessary. The output normal parameterization is a minimal parameterization that is numerically robust.

We have introduced an optimization of the output-error criterion based on the SLS technique, using a suitable parameterization. With this technique the optimization is only concerned with the parameterization of the state space matrices $A$ and $C$. This significantly reduces the search space for the optimization routine. This has been visually demonstrated by an example.

For the specific case of Gauss-Newton optimization we have demonstrated how the optimization with SLS can be cast in a recursive form. This allows the tracking of time-varying systems and the on-line estimation of a state space model.

The optimization of continuous time models has been discussed. We have demonstrated how the optimization scheme that was initially designed for the discrete time case can be extended to the continuous time case.

Finally, we have shown how the SLS optimization of LTI models can be extended to the prediction-error framework and how SLS can be applied to Wiener models.
Chapter 5

Implementation issues

5.1 Introduction

The algorithms in the MOESP framework for estimating the $A$ and $C$ matrices are generally very straightforward in the implementation. The main numerical routines are the LQ or QR factorization (depending on which routine is available) and the SVD. The algorithm for the estimation of $B$, $D$ and the initial state consists of the construction of some data matrices and the solution of a linear least squares problem involving these data matrices.

However, by exploiting several properties of the data matrices we can construct algorithms that are faster than the straightforward implementation. The implementation of MOESP can be improved with the use of a Cholesky factorization. This implementation has a significant speed advantage over the ordinary implementation with a QR factorization. The estimation of $B$, $D$ and the initial state consists of the construction of several data matrices. By proper implementation of this algorithm, a significant increase in computation speed is possible.

A second topic of interest is the comparison between different SMI algorithms. Algorithms of the N4SID and CVA family have different properties with respect to computation time and the number of operations. A comparison of these properties will lead to a combination of the strong points of both algorithms into one improved algorithm.

In this chapter the speed of a numerical algorithm is measured in two ways. The first way is to measure the time in seconds in which the algorithm is executed. This is called the CPU-time. However, this is not only depending on the algorithm itself, but also on the type of computer, the processor speed and the operating system. A second way is to count the number of floating point operations (FLOPS) This is a hardware independent measure. In MATLAB the FLOPS are the number of multiplications, additions and subtractions used by an algorithm. This will be identical for different
types of hardware. However, in some cases arithmetic operations are not the most significant part of the algorithm. Copying a matrix or transposing a matrix are two operations that are not accounted for in the number of FLOPS. However, they do take time, which shows up in the measured CPU-time. In algorithms where these type of manipulations are a significant part of the work, the number of FLOPS can give a deceptive view of the cost of an implementation.

This chapter is structured as follows. In section 5.2 the implementation of the MOESP algorithm for the estimation of $A$ and $C$ is discussed. In section 5.3 the implementation issues regarding the estimation of $B$, $D$ and the initial state are discussed. In section 5.5 the advantages and disadvantages of three different SMI algorithms are examined. Finally, in section 5.6 the possibilities with of a combined algorithm using MOESP and N4SID is discussed.

5.2 Efficient implementation of MOESP by using the Cholesky factorization

In this section we discuss the implementation of the MOESP algorithm. In chapter 2 we described the different members of the MOESP family of algorithm as a number of steps using numerical algebra operations. However, in practice the implementations of OM-MOESP, PI-MOESP, PO-MOESP and EIV-MOESP are made in programming environments like MATLAB. The implementation can be done in many different ways. In this section we look for a fast and numerically stable implementation of the MOESP algorithms.

The implementation of MOESP according to algorithm 2.2 requires a LQ factorization and a SVD. As we have shown in section A.2 the LQ factorization and the QR factorization are related by a single transpose.

Two facts complicate the straight forward implementation of MOESP in MATLAB. Firstly, in MATLAB the input and output $u$ and $y$ are traditionally given as matrices of size $N \times m$ and $N \times \ell$ where every row is a time-sample and every column represents a different input or output. In order to build the matrices $U_{0,i,N}$ and $Y_{0,i,N}$ we are required to transpose the input and output matrices. Secondly, the LQ factorization is not available in MATLAB. Therefore we are forced to use the QR factorization instead. This would require a second transpose of the matrices $U_{0,i,N}$ and $Y_{0,i,N}$. A more efficient implementation is, however, the construction of the matrices $U_{0,i,N}^T$ and $Y_{0,i,N}^T$ instead of $U_{0,i,N}$ and $Y_{0,i,N}$. On these matrices we perform a QR factorization and the resulting $R$ factor is transposed to obtain $L$.

```
N=Nm-i+1;
Y=zeros(N,i*1);U=zeros(N,i*m);
for j=(1:i),
    U(:,(j-1)*m+1:j*m)=u(j:N+j-1,:);
    Y(:,(j-1)*l+1:j*l)=y(j:N+j-1,:);
```
end
L=triu(qr([U Y]));

Table 5.2 shows that the most time consuming step of MOESP is the QR factorization. The figure was made after a profile of PO-MOESP on test data. The data was generated using the following piece of code. In this case the QR factorization took 93% of the total time that was spent inside the algorithm.

\[
\begin{align*}
N &= 1000; L = 5; m = 5; n = 6; \\
[A, B, C, D] &= drmodel(n, L, m); \\
u &= randn(N, m); \\
y &= dlssim(A, B, C, D, u);
\end{align*}
\]

<table>
<thead>
<tr>
<th>Function</th>
<th>Time(sec.)</th>
<th>Percent(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix constr.</td>
<td>0.16</td>
<td>3</td>
</tr>
<tr>
<td>QR</td>
<td>5.28</td>
<td>93</td>
</tr>
<tr>
<td>SVD</td>
<td>0.20</td>
<td>4</td>
</tr>
<tr>
<td>Other</td>
<td>0.01</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>5.65</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 5.1: profile of dordpo

In [100] it was shown how the QR factorization can be replaced by a Cholesky factorization. This was demonstrated for the SISO case, but can be expanded to the MIMO case as well. The idea behind this approach is that the correlation matrix

\[
M = \begin{bmatrix} U_{0,i,N} & U_{0,i,N} \\ Y_{0,i,N} & Y_{0,i,N} \end{bmatrix}^T
\]

(5.1)
can be computed very fast, due to the special Hankel structure in \(U_{0,i,N}\) and \(Y_{0,i,N}\). The Cholesky factorization of a symmetric positive definite matrix \(M\) is given by

\[
M = R^T R
\]

where \(R\) is an upper triangular matrix. With the LQ factorization of \(\begin{bmatrix} U_{0,i,N} \\ Y_{0,i,N} \end{bmatrix}\) we have

\[
M = LL^T
\]

We see that the Cholesky factorization of \(M\) gives the same lower triangular matrix as the LQ factorization of \(\begin{bmatrix} U_{0,i,N} \\ Y_{0,i,N} \end{bmatrix}\).
For the construction of $M$ we can use the Hankel structure of the data matrices. Let $R_{UU} = U_{0,i,N}U_{0,i,N}^T$, then a block-element in this matrix, denoted by $R_{UU}(j,k)$, is given by

$$R_{UU}(j,k) = \sum_{q=1}^{N} u(q+j-1)u(q+k-1)^T \quad (5.2)$$

The next element down the diagonal, $R_{UU}(j+1,k+1)$, is given by

$$R_{UU}(j+1,k+1) = \sum_{k=1}^{N} u(q+j)u(q+k)^T \quad (5.3)$$

This can be rewritten as

$$R_{UU}(j+1,k+1) = R_{UU}(j,k) - u(j)u(k)^T + u(N+j)u(N+k)^T \quad (5.4)$$

In effect, only the top row of the correlation matrix has to be computed fully. After that, the rest of the entries follow with a single addition and subtraction. Moreover, because the matrix $M$, that is used in the Cholesky factorization, has to be symmetrical, only the upper triangular part of the matrix needs to be calculated. This saves even more calculations.

An example of the proposed algorithm is

```matlab
Ruu=zeros((2*m)*i,(2*m)*s);
%some ranges, to improve readability
rowu = [1-m:0]
coln = [0:N-1];
% Compute top row of the correlation
j=1;buj=j*m
for k = 1:2*i
    buk = k*m;
    Ruu(buj+rowu,buk+rowu)=u(coln+j,:)\'*u(coln+k,:);
end
% compute rest of correlations, using the hankel structure
for k = 2:2*i
    for j = 2:k
        buj = j * m;
        buk = k * m;
        Ruu(buj+rowu,buk+rowu)=Guu((buj-m)+...
                rowu,(buk-m)+rowu)-u(j-1,:)\'*u(k-1,:)+u(N+j-1,:)\'*u(N+k-1:);
    end
end
```

A further possible improvement, in certain circumstances, is to calculate the correlations by using the FFT. Especially for large values of the block matrix dimension
parameter $i$ and the number of inputs and outputs, this can further improve the performance. More improvement can be possible with the use of fast QR factorization methods to alternatively compute the R-factor [25, 52].

Table 5.2 shows the results of the improved PO-MOESP on the same test as used for 5.2 This time a Cholesky factorization was used instead of a QR factorization. We see that the matrix construction takes more time in this case, but the total time spend is reduced from 5.64 to 2.14 sec. This is a reduction of almost 40%.

<table>
<thead>
<tr>
<th>Function</th>
<th>Time (sec.)</th>
<th>Percent (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix constr.</td>
<td>1.64</td>
<td>77</td>
</tr>
<tr>
<td>Cholesky</td>
<td>0.24</td>
<td>11</td>
</tr>
<tr>
<td>SVD</td>
<td>0.15</td>
<td>7</td>
</tr>
<tr>
<td>Other</td>
<td>0.11</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>2.14</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 5.2: profile of dordpo

We finish this section with a more detailed comparison between the direct implementation of MOESP that uses a QR factorization and the proposed implementation using a Cholesky factorization. For this test we generated data from a randomly chosen model of third order with three inputs and three outputs. We varied the number of data points $N_s$ that was available for identification from 200 to 5000. The block matrix parameter $i$ was varied from 6 to 15. For every value of $N_s$ and $i$ we identified the matrices $A$ and $C$. Since no noise was added, all tests successfully identified these system matrices.

In figure 5.1 the number of FLOPS and in figure 5.2 the CPU-time are shown. From the figures we see that for $N_s = 5000$, $i = 15$ the Cholesky factorization improves the computation time from 20.5 to 8.2 seconds.

Figure 5.1: Number of FLOPS for MOESP with the QR factorization(left) and with the Cholesky factorization(right).
Figure 5.2: CPU-time for MOESP using the QR factorization (left) and with the Cholesky factorization (right).

5.3 Efficient implementation of the estimation of $B$ and $D$

In this section we discuss alternative implementations of the algorithm for the identification of the matrices $B$ and $D$ (and $x_0$). Section 2.9 described this algorithm at a conceptual level. However, straightforward implementation of algorithm 2.7 will result in a very time consuming program. Here we will look at the implementation issues of this algorithm. The main reason for this analysis is the fact that the estimation of $B$, $D$ and $x_0$ is the most time consuming part of the whole SMI identification procedure. Implementing this part in an efficient manner improves the total SMI method. We will examine several alternative approaches in this section. Afterward a comparative study shows the differences between the methods.

Let us recall formula 2.74.

\[ Y_{0,N,1} = \begin{bmatrix} \Gamma_N & \mathcal{Y} & \mathcal{U} \end{bmatrix} \begin{bmatrix} x_0 \\ B \\ D \end{bmatrix} + E \]  \hspace{1cm} (5.5)

with

\[ Y_{0,N,1} = \begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{bmatrix} \]  \hspace{1cm} (5.6)
\[ \Gamma_N = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{N-1} \end{bmatrix} \]  

(5.7)

\[ \mathcal{Y} = \begin{bmatrix} 0 \\ u(0)^T \otimes C \\ \vdots \\ \sum_{\tau=0}^{N-2} u(\tau)^T \otimes CA^{N-1-\tau} \end{bmatrix} \]  

(5.8)

\[ \mathcal{U} = \begin{bmatrix} u(0)^T \otimes I_f \\ u(1)^T \otimes I_f \\ \vdots \\ u(N-1)^T \otimes I_f \end{bmatrix} \]  

(5.9)

\[ E = \begin{bmatrix} v(0) \\ Cw(0) + v(1) \\ \vdots \\ \sum_{\tau=0}^{N-2} CA^{N-1-\tau}w(\tau) + v(N-1) \end{bmatrix} \]  

(5.10)

\[ B = \text{vec}(B) \]  

(5.11)

\[ D = \text{vec}(D) \]  

(5.12)

In the algorithm, the following matrices need to be constructed: \( Y_{0,N,1}, \Gamma_N, \mathcal{Y} \) and \( \mathcal{U} \). These are discussed in the following subsections.

### 5.3.1 Implementation of \( Y_{0,N,1} \)

The most straightforward implementation of \( Y_{0,N,1} \) is given by

\[
Y = \text{zeros}(1 \times N, 1);
\]

for \( i=1:N \)

\[
Y((i-1) \times 1 + 1:i \times 1) = y(i,:);
\]

end

In MATLAB, the use of a for-loop is highly time consuming and therefore undesirable in an efficient implementation. Especially a for-loop over the number of samples (which is often high) will cost a lot time. A better method to construct \( Y_{0,N,1} \) is the following:
ytrans=y';
Y=ytrans(:);

The colon-operator (:) in MATLAB is the fastest way to implement the vec(·) operation. However, since the matrix \( y \) commonly is given as a tall matrix of size \( N \times \ell \) where every row is a time-sample, we first need to transpose this matrix, to get the correct ordering of \( Y_{0,N,1} \).

The extra transpose is, however, not necessary. The ordering of the equations in equation (5.5) does not influence the least squares solution. We can re-order these as we see fit. Therefore we can remove the transpose operation, as long as the matrices \( \Gamma_N \), \( \mathcal{Y} \), \( \mathcal{U} \) and the least square solution are reordered accordingly. Instead of \( \text{vec}(Y_{0,N,1}) \) we therefore use the re-ordered form \( \text{vec}(y) \) or \( \text{vec}(Y_{0,1,N}^T) \). The reordered \( Y_{0,N,1} \) has the following form:

\[
\text{vec}(Y_{0,N,1}^T) = \begin{bmatrix}
y_1(0) \\
y_1(1) \\
\vdots \\
y_1(N-1) \\
y_2(0) \\
\vdots \\
y_2(N-1) \\
\vdots \\
y_t(N-1)
\end{bmatrix}
\]

where

\[
y(k) = \begin{bmatrix}
y_1(k) \\
y_2(k) \\
\vdots \\
y_{\ell}(k)
\end{bmatrix}
\]

5.3.2 Implementation of \( \Gamma_N \)

The straightforward implementation of \( \Gamma_N \) is given by:

```matlab
Gamma=zeros(N*1,n);
for i=1:N
    Gamma((i-1)*l+1:i*l,:)=C*A^-(i-1);
end
```

Again, we must try to remove or at least reduce the size of the for-loop. This can be done in the following manner:
Gamma = zeros(N*1,n);
Gamma(1:1,:) = C;
An = A;
for i = 1:floor(log(N)/log(2)),
    Gamma(2^(i-1)*1+1:2^(i)*1,:) = Gamma(1:2^(i-1)*1,:)*An;
    An = An*An;
end
Gamma(2^n*1+1:N*1,:) = Gamma(1:N*1-2^n*1,:)*An;

This implementation minimizes the number of times the for-loop has to be executed.
The matrix $\Gamma_N$ is calculated recursively as follows

$$
\begin{bmatrix}
    C & \\
    \vdots & \\
    C A^{2^i} & \\
    \vdots & \\
    C A^{2^{i+1}-1}
\end{bmatrix} = \begin{bmatrix}
    C A^{2^i} & \\
    \vdots & \\
    C A^{2^{i+1}-1}
\end{bmatrix}.
$$

(5.13)

The right-hand side of this multiplication can be concatenated to $\Gamma_N$ of the previous step. This way the length of $\Gamma_N$ increases exponentially. This implementation reduces the number of iterations drastically. For instance, for $N = 1000$, the for-loop is only entered 9 times, instead of 1000 times in the original implementation.

When we use the reordered version of $Y_{0,N,1}$, we need to reorder $\Gamma_N$ as well. This requires some extra computation:

```
temp = zeros(n*1,N);
temp(:) = Gamma';
for j = 1:n
    Gamma(N*(j-1)+1:N*j,:) = temp(n*(j-1)+1:n*j,:);'
end
```

However, this only involves a for-loop over the number of outputs, which is typically not a large number.

### 5.3.3 Implementation of $\mathcal{Y}$

If we implement $\mathcal{Y}$ directly from equation (5.8) we have

```
Yij=zeros(1*N,m*n);
for i=1:N
    for j=1:i-1
        Yij((i-1)*1+1:i*1,:) = Yij((i-1)*1+1:i*1,:)+kron(u(j,:),C*A^(-i-j-1));
    end
end
```
This implementation, although simple, is highly impractical and time consuming. Because of the two loops, the number of computations on $Y_{ij}$ is $N!$

The first improvement can be made by noting that the columns in $\mathcal{Y}$ are the output of the state space models

\begin{align}
x_{ij}(k+1) &= Ax_{ij}(k) + e_iu_j(k) \\
y_{ij}(k) &= Cx_{ij}(k)
\end{align}

(5.14) (5.15)

with $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$ and $x_{ij}(0) = 0$

Therefore we can use the following code to construct the matrix $\mathcal{Y}$:

```matlab
Yij=zeros(N*1,n*m);
e=eye(n);
for j=1:m
    for i=1:n
        x=ltitr(A,e(:,i),u(:,j));
        yij = C*x'*;
        Yij(:,(j-1)*n+i)=yij(:);
    end
end
```

This removes the long for-loops and replaces them by two smaller ones (over the number of inputs and outputs). The reordered version that is needed to work with $\text{vec}(Y_{0,N,1}^T)$ can be obtained very easily. For this we only need to transpose $y_{ij}$ before adding to $y_{ij}$. This can be done by replacing $y_{ij} = Cx'$ with $y_{ij} = x'C'$. This has the extra advantage that the transpose of $x$ has disappeared.

A second improvement can be made in the calculation of $\mathcal{Y}$. The most time consuming part in the previous implementation is the simulation of the state space models. By using a special form of the state space system, the order of the systems that are simulated can be reduced. This will save time, especially for higher order models.

The special form can be constructed by transforming the matrix $A$ into the real Schur form.

The Schur decomposition of $A \in \mathbb{R}^{n \times n}$ is given by

\[ A = T A' T^T \]

where $T \in \mathbb{R}^{n \times n}$ is orthogonal and

\[ A' = \begin{bmatrix}
A'_{11} & A'_{12} & \cdots & A'_{1m} \\
0 & A'_{22} & \cdots & A'_{2m} \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & A'_{mm}
\end{bmatrix} \]

(5.16)
where each $A'_{ij}$ is either a 1-by-1 matrix or a 2-by-2 matrix. The eigenvalues of these matrices are also eigenvalues of $A$.

Issai Schur (1875-1941) is mainly known for his fundamental work on the representation theory of groups, but he also worked in number theory and analysis. In 1894 he entered the University of Berlin to read mathematics and physics. Frobenius was one of his teachers and was to greatly influence Schur. They collaborated on many papers. Schur was also interested in reducibility, location of roots and the construction of the Galois group of classes of polynomials such as Laguerre and Hermite polynomials. In 1922 he was elected to the Prussian Academy. From 1933 events in Germany made his life increasingly difficult. He left Germany for Palestine in 1939, broken in mind and body. He died two years later.

The Schur transform is used to transform the original system matrices $A$ and $C$ into

$$A_2 = T^T A T \quad C_2 = C T$$  \hspace{1cm} (5.17)

We can then solve the estimation of $B_2$, $D_2$ and $x_{2_0}$ on the basis of these matrices. The result has to be transformed back afterwards to the original state space matrices with

$$B = B_2 T \quad D = D_2 \quad x_0 = x_{2_0} T$$  \hspace{1cm} (5.18)

With the transformed system we can reduce the order of the model (5.14) - (5.15). To see this, we observe that only the $i$-th element of $x_{ij}(k+1)$ is directly affected by the input $u_j(k)$, since the vector $e_i$ has only an one on the $i$-th position. This state will only propagate upward in the state vector, because $A'$ is block-upper triangular. The other states will remain zero and can be disregarded. Therefore we only need to simulate a $i$-th or $i+1$-th order model, depending on whether $A(i,i)$ is part of a $1 \times 1$ or a $2 \times 2$ block. The following program makes use of the Schur form of $A_2$.

```matlab
Yij=zeros(N*1,n*m);
e=eye(n);
for i=1:n
    if i==n
        s=i;
    elseif A(i+1,i)==0
        s=i;
    else
        s=i+1;
    end
A2=A(1:s,1:s);
E2=e(1:s,i);
```
C2=C(:,1:s);
for j=1:m
    x=titr(A2,E2,u(:,j),zeros(size(A2,1),1));
    yij=x*C2';
    Yij(:,(j-1)*n+i)=yij(:,);
end

5.3.4 Implementation of $\mathcal{U}$

The straightforward implementation of $\mathcal{U}$ using equation (5.9) is given by the following code:

$$
\text{Uij}=\text{zeros}(1*N,m*1);
\text{for } i=1:N
\quad \text{Uij}((i-1)*l+1:i*1,:)=\text{kron}(u(i,:),\text{eye}(1));
\end$$

The use of the \text{kron}-function product and the long for-loop make this a slow choice of implementation. A significantly faster method is the following implementation. First a single column of $U_{ij}$ is computed by adding zeros and reshaping the resulting matrix. Then the columns are joined together in the $U_{ij}$ matrix.

$$
\text{Uij}=\text{zeros}(N*1,m*1);
\text{uij}=\text{zeros}(N*1,1);
\text{for } i=1:m
\quad \text{for } j=1:l,
\quad \quad \text{uij}(:,)=\text{zeros}(N,j-1),u(:,i),\text{zeros}(N,1-j)];
\quad \text{Uij}(:,(i-1)*l+j)=\text{uij};
\end$$

When we look at the reordering of $\mathcal{U}$ to make it fit the reordered $\text{vec}(Y_{0,1,N}^T)$, we find that $\mathcal{U}'$ has the following structure:

$$
\mathcal{U}' = I_l \otimes \begin{bmatrix}
    u(0)^T \\
    u(1)^T \\
    \vdots \\
    u(N-1)^T
\end{bmatrix}
$$

This results in an even simpler implementation of $\mathcal{U}$:

$$
\text{Uij}=\text{zeros}(N*1,m*1);
$$
for j = 1:l,
    Uij((j-1)*N+1:j*N,(j-1)*m+1:j*m)=u;
end

5.3.5 Benchmark test

In this section we look at the efficiency of four implementations of the same algorithm: destbd. The first implementation is the straightforward implementation of equation (5.6) - (5.9). The second implementation uses the more efficient construction of these matrices, with the normal ordering of the rows and without the Schur form for computing \( \mathcal{Y} \). In the third implementation we change the ordering of the rows in the constructed matrices. In the fourth implementation we add the Schur transformation.

As we saw earlier, the reordering of the rows gives a simpler implementation of \( \mathcal{Y}, \mathcal{Y}_ij \) and \( \mathcal{U}_ij \) but a more complex implementation for \( \Gamma_N \). Which variant is more time-efficient will be shown in the difference between the results of implementation two and three. The improvement made by using the Schur form will be shown in the difference between implementation three and four.

The implementations given in this chapter are for MATLAB. We measured both the CPU-time and the floating point operations (FLOPS) for every matrix. The figures are measured for an example of order 6, with 5 inputs and 5 outputs. The input data \( u \) and the output data \( y \) were taken to be tall matrices of 1000 data points. Both the number of FLOPS and the CPU-time were measured. The measurements were taken from a 120MHz pentium running Linux 2.2.5 and MATLAB 5.3 called guppy. The data was generated using the following matlab program.

\[
N=1000;L=5;m=5;n=6;\]
\[
[A,B,C,D]=drmodel(n,L,m);\]
\[
u=randn(N,m);\]
\[
y=dlsim(A,B,C,D,u);\]

Table 5.3 and 5.4 show the results of the measurements for all four implementations. The best results in every row are indicated in bold.

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Straight forward</th>
<th>Efficient</th>
<th>Reordered</th>
<th>Schur form</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{Y}_{ij,N,1} )</td>
<td>0.6</td>
<td>0.11</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>( \Gamma_N )</td>
<td>1.50</td>
<td>0.11</td>
<td>0.17</td>
<td>0.17</td>
</tr>
<tr>
<td>( \mathcal{Y} )</td>
<td>3.040.40</td>
<td>1.01</td>
<td>0.93</td>
<td>0.74</td>
</tr>
<tr>
<td>( \mathcal{U} )</td>
<td>4.30</td>
<td>0.23</td>
<td>0.04</td>
<td>0.04</td>
</tr>
</tbody>
</table>

Table 5.3: Measured CPU-time for the four implementations of destbd.
From the test we can conclude that the straight forward implementation is indeed very time-consuming. Moreover, we find that the reordering of the matrices is disadvantageous for the construction of $\Gamma_N$, as expected, but that this is compensated for in the calculation of $\mathcal{U}$. The Schur form shows its advantage in the calculation of $\mathcal{Y}$. Although this decrease in calculation time is only small, a larger increase can be found when a higher order model is used for the test.

### 5.4 Solving for $B$ and $D$

In the previous section we discussed the construction of the matrices $Y_{0,N,1}$, $\Gamma_N$, $\mathcal{Y}$ and $\mathcal{U}$. These matrices are needed for the estimation of $B$ and $D$(and $x_0$). These system matrices are estimated by the least squares solution of equation (5.5).

In this section we compare four methods for solving the above equation.

- The Penrose-Moore pseudo-inverse.
- The MATLAB build-in $\backslash$ (left matrix divide) operator.
- The matlab build-in “pinv” function.
- An explicit QR factorization.

#### 5.4.1 The Penrose-Moore pseudo-inverse

The Penrose-Moore pseudo inverse of a matrix $\Phi$, denoted by $\Phi^\dagger$, is given by $(\Phi^T\Phi)^{-1}\Phi^T$.

Let $\Phi = \begin{bmatrix} \Gamma_N & \mathcal{Y} & \mathcal{U} \end{bmatrix}$ then

$$
\begin{bmatrix}
  x_0 \\
  B \\
  D
\end{bmatrix} = (\Phi^T\Phi)^{-1}\Phi^TY_{0,N,1}
$$

However, this approach has a serious drawback. If the matrix $\Phi$ is close to rank deficient, then the matrix $(\Phi^T\Phi)^{-1}$ will be badly conditioned. This will cause numerical
problems in the solution of the least squares problem. This can happen when \( N \) becomes large. In this case the influence of the initial state \( x_0 \) becomes small, because the bottom part of \( \Gamma_N \) becomes approximately zero. It can be advantageous in such circumstances to neglect the influence of \( x_0 \) in the above equation and only solve for \( B \) and \( D \).

\[
\begin{bmatrix}
B \\
D
\end{bmatrix} = \begin{bmatrix}
\mathcal{Y} & \mathcal{U}
\end{bmatrix}^\dagger Y_{0,N,1}
\]

However, this will lead to slightly biased estimates of \( B \) and \( D \), because we don’t take into account the effect of \( x_0 \) at all.

### 5.4.2 Left matrix divide

The second method for solving the least squares problem is with the build-in MATLAB command “\( \backslash \)”.

\[
\text{XBD} = [\text{Gamma} \ Yij \ Uij] \backslash Y;
\]

This command checks the rank of the matrix \( \begin{bmatrix} \Gamma_N & \mathcal{Y} & \mathcal{U} \end{bmatrix} \) using a rank-revealing QR factorization. Then it uses the (lower rank) matrix \( R \) to solve the least squares problem. This method does not suffer from the badly conditioned \( \Gamma_N \) matrix. In the rank detection, the matrix \( R \) will automatically be reduced in dimension and a lower dimensional problem (if necessary) is solved.

### 5.4.3 The pseudo inverse

The third method is based on the MATLAB function \texttt{pinv}.

\[
\text{XBD} = \text{pinv}([\text{Gamma} \ Yij \ Uij])^* Y;
\]

This method computes the SVD of \( \begin{bmatrix} \Gamma_N & \mathcal{Y} & \mathcal{U} \end{bmatrix} \) and decides the rank, based on the singular values. Then it calculates a pseudo inverse based on a low rank approximation of the matrix. This method is also robust against a badly conditioned least squares problem.
5.4.4 The QR factorization

The last method is based on the QR factorization. Consider the following QR factorization:

\[
\begin{bmatrix}
\Gamma_N & Y & U & Y_{0,N,1}
\end{bmatrix} =
\begin{bmatrix}
Q_1 & Q_2 & Q_3 & Q_4
\end{bmatrix}
\begin{bmatrix}
R_{11} & R_{12} & R_{13} & R_{14} \\
R_{22} & R_{23} & R_{24} & \\
R_{33} & R_{34} & \\
R_{44} &
\end{bmatrix}
\] (5.20)

The following lemma gives a method for consistently estimating \(x_0, B\) and \(D\) using the above QR factorization.

**Lemma 5.1** Given the state space system (2.19a) - (2.19b), let \(u(k)\) be PE of sufficient order. Let \(Y_{0,N,1}, \Gamma_N, Y, U\) be constructed according to equations (5.6)-(5.9). Perform the QR factorization (5.20). Then a consistent estimate for \(x_0, B\) and \(D\) is given by:

\[
\begin{bmatrix}
x_0 \\
B \\
D
\end{bmatrix} =
\begin{bmatrix}
R_{11} & R_{12} & R_{13} \\
R_{22} & R_{23} & \\
R_{33} &
\end{bmatrix}^{-1}
\begin{bmatrix}
R_{14} \\
R_{24} \\
R_{34}
\end{bmatrix}
\] (5.21)

**Proof:**

*Using equation (5.20) and (5.5) we have*

\[
\begin{bmatrix}
Q_1 & Q_2 & Q_3 & Q_4
\end{bmatrix}
\begin{bmatrix}
R_{14} \\
R_{24} \\
R_{34} \\
R_{44}
\end{bmatrix}
= \begin{bmatrix}
Q_1 & Q_2 & Q_3
\end{bmatrix}
\begin{bmatrix}
R_{11} & R_{12} & R_{13} \\
0 & R_{22} & R_{23} \\
0 & 0 & R_{33}
\end{bmatrix}
\begin{bmatrix}
x_0 \\
B \\
D
\end{bmatrix} + E
\] (5.22)

*Multiplying on both sides with \(\begin{bmatrix}
Q_1 & Q_2 & Q_3
\end{bmatrix}^T\) gives:*

\[
\begin{bmatrix}
R_{14} \\
R_{24} \\
R_{34}
\end{bmatrix} =
\begin{bmatrix}
R_{11} & R_{12} & R_{13} \\
R_{22} & R_{23} & \\
R_{33} &
\end{bmatrix}^{-1}
\begin{bmatrix}
x_0 \\
B \\
D
\end{bmatrix} + \begin{bmatrix}
Q_1 & Q_2 & Q_3
\end{bmatrix}^T E
\]

*In the limit*

\[
\lim_{N \to \infty} \frac{1}{\sqrt{N}} \begin{bmatrix}
Q_1 & Q_2 & Q_3
\end{bmatrix}^T E = 0
\]
because $E$ is independent of the matrices $\Gamma_N$, $Y$ and $U$. When $u(t)$ is persistently exciting

$$\lim_{N \to \infty} \frac{1}{\sqrt{N}} \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{bmatrix}$$

is invertible and we can estimated $x_0$, $B$ and $D$ as in equation (5.21).

When $N$ is large and the matrix $\Gamma_N$ will be badly conditioned. The above method will then fail to give consistent results, because $R$ is close to singular. This is because the influence of the initial state is very small in this case and $x_0$ will become hard to identify. In this case we can still find $B$ and $D$ consistently using the following method:

**Lemma 5.2** Given the state space system (2.19a) - (2.19b), let $u(k)$ be PE of sufficient order. Let $Y_{0,N,1}$, $\Gamma_N$, $Y$, $U$ be constructed according to equations (5.6)-(5.9). Perform the QR factorization (5.20). Then a consistent estimate for $B$ and $D$ is given by:

$$\begin{bmatrix} B \\ D \end{bmatrix} = \begin{bmatrix} R_{22} & R_{23} \\ 0 & R_{33} \end{bmatrix}^{-1} \begin{bmatrix} R_{24} \\ R_{34} \end{bmatrix}$$  (5.23)

$\blacksquare$

**Proof:**

The proof is very similar to the proof of lemma 5.1. Instead of multiplying equation (5.22) with $[Q_1 \quad Q_2 \quad Q_3]^T$ we multiply with $[Q_2 \quad Q_3]^T$. This results in

$$\begin{bmatrix} R_{24} \\ R_{34} \end{bmatrix} = \begin{bmatrix} R_{22} & R_{23} \\ 0 & R_{33} \end{bmatrix}^{-1} \begin{bmatrix} B \\ D \end{bmatrix} + \begin{bmatrix} Q_2 \\ Q_3 \end{bmatrix}^T E$$

For the same reasons as in lemma 5.1

$$\lim_{N \to \infty} \frac{1}{\sqrt{N}} \begin{bmatrix} Q_2 \\ Q_3 \end{bmatrix}^T E$$

is zero and the inverse of

$$\lim_{N \to \infty} \frac{1}{\sqrt{N}} \begin{bmatrix} R_{22} & R_{23} \\ 0 & R_{33} \end{bmatrix}$$

exists. Therefore we can estimate $B$ and $D$ consistently as in equation (5.23). $\blacksquare$

**Remark 5.1** The result of lemma 5.2 is also directly applicable in the continuous time case by substituting the proper matrix definitions (3.81) - (3.87).
5.4.5 Benchmark test

We used the same benchmark as before to test the four methods for solving the least squares problem. Table 5.5 shows the results of the benchmark test. The first row shows the number of FLOPS used by the four methods. The pinv method shows the highest number. This is due to the SVD involved in this calculation. The computation of a SVD is computational expensive. The left matrix divide method and the QR method show a similar number of FLOPS as well as the lowest CPU-time. This is according to expectations, since both methods are based on the QR factorization. From this benchmark test we can conclude that the QR factorization is the most suited algorithm for solving the LS problem in the estimation of $B$ and $D$.

<table>
<thead>
<tr>
<th>Method:</th>
<th>Penrose-Moore</th>
<th>left matrix divide</th>
<th>pinv</th>
<th>QR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flop-count</td>
<td>75,502,123</td>
<td>40,417,503</td>
<td>235,702,197</td>
<td>39,319,720</td>
</tr>
<tr>
<td>CPU-time</td>
<td>7.4100</td>
<td>5.5700</td>
<td>24.5800</td>
<td>5.3000</td>
</tr>
</tbody>
</table>

Table 5.5: Flop-count and CPU-time of four methods for solving the LS problem in the estimation of $B$ and $D$.

5.5 Comparing different SMI algorithms

In this section we investigate different implementations of the Subspace Model Identification algorithm. In the previous part of this thesis we have focused on only one type of SMI algorithm, the MOESP family of algorithms. However, many different implementations of SMI exist. These implementations differ significantly, theoretically as well as in practical implementation.

We shall examine three main types of SMI algorithms, namely CVA, N4SID and MOESP. The methods are compared in accuracy, on different industrial datasets. For this the prediction-error and simulation error are compared. We also compare the methods on numerical complexity. For this, the number of floating point operations and the CPU-time is used.

The comparative study was performed for the Niconet project. In [29] an initial comparative study between the CVA, N4SID and MOESP was made. The algorithms were compared on the aspects of number of floating point operations, prediction-error and simulation error on a number of industrial data-sets.

The main conclusion of this study was that the three algorithms differed much in computational complexity. The CVA method showed a significantly lower number of floating point operations, compared to the other two methods. A second study was then proposed to find the cause of this difference and to examine possible reductions in computational complexity of the MOESP and N4SID algorithms.
In this section the results from a more extensive comparative study are presented. Moreover, a possible combination of the MOESP and N4SID algorithm is explored, leading to a number of new algorithms. For this purpose the N4SID and MOESP algorithm were divided into modules. A module is a separate part of the algorithm. By combining the fastest modules from the two algorithms, we can improve the overall computational speed.

5.5.1 Modulizing MOESP and N4SID

In this section we decompose N4SID and MOESP into several separate modules. These modules are as independent as possible from each other and have separate functions within the algorithms. Using these modules, we then derive a faster algorithm by combining modules from N4SID and MOESP.

The MOESP algorithm contains the following modules:

Module 1: Construction of \( U_{i+1,N} \), \( Y_{i,j,N} \), \( U_{0,i,N} \) and \( Y_{0,i,N} \) and the LQ factorization is computed.

\[
\begin{bmatrix}
U_{i+1,N} \\
U_{0,i,N} \\
Y_{0,i,N} \\
Y_{i+1,j,N}
\end{bmatrix} =
\begin{bmatrix}
L_{11} & 0 & 0 & 0 \\
L_{21} & L_{22} & 0 & 0 \\
L_{31} & L_{32} & L_{33} & 0 \\
L_{41} & L_{42} & L_{43} & L_{44}
\end{bmatrix}
\begin{bmatrix}
Q_1 \\
Q_2 \\
Q_3 \\
Q_4
\end{bmatrix}
\]  

(5.24)

Module 2: The singular value decomposition

\[
\begin{bmatrix}
L_{42} \\
L_{43}
\end{bmatrix} = U \Sigma V^T
\]

The order \( n \) is decided from the singular values. The first \( n \) columns of \( U \) are denoted by \( U_n \).

Module 3: With \( U_n \) we compute \( \hat{A}_T \) and \( \hat{C}_T \). They are found from

\[
\hat{C}_T = \text{the upper } \ell \text{ rows of } U_n \\
\hat{A}_T = U_n^T U_2
\]

Module 4: \( \hat{B}_T \) and \( \hat{D}_T \) are calculated with algorithm 2.7

Module 5: The estimate Kalman gain \( \hat{K}_T \) is estimated using algorithm 2.5.

The N4SID algorithm[84] is decomposed into the following modules.
Module 1: Construction of $U_{i,j,N}$, $Y_{i,j,N}$, $U_{0,i,N}$ and $Y_{0,i,N}$ and the LQ factorization

$$
\begin{bmatrix}
U_{0,i,N} \\
U_{i,j,N} \\
Y_{0,i,N} \\
Y_{i,j,N}
\end{bmatrix} =
\begin{bmatrix}
L_{11} & 0 & 0 & 0 \\
L_{21} & L_{22} & 0 & 0 \\
L_{31} & L_{32} & L_{33} & 0 \\
L_{41} & L_{42} & L_{43} & L_{44}
\end{bmatrix}
\begin{bmatrix}
Q_1 \\
Q_2 \\
Q_3 \\
Q_4
\end{bmatrix}
$$

(5.25)

Module 2: Construct

$$
L_f = \begin{bmatrix}
L_{41} & L_{42} & L_{43} & L_{44}
\end{bmatrix}
$$

$$
L_p = \begin{bmatrix}
L_{11} & 0 & 0 & 0 \\
L_{31} & L_{32} & L_{33} & 0
\end{bmatrix}
$$

$$
L_u = \begin{bmatrix}
L_{21} & L_{22} & 0 & 0
\end{bmatrix}
$$

$$
L_{fp} = \begin{bmatrix}
L_f \\
\Pi_{L_u}^1
\end{bmatrix}
$$

$$
L_{pp} = \begin{bmatrix}
L_p \\
\Pi_{L_u}^1
\end{bmatrix}
$$

$$
O_b = L_{fp} \Pi_{L_p}
$$

$$
WO_bW = O_b \Pi_{L_u}^1
$$

The estimate of the extended observability matrix is calculated by performing the following SVD

$$
WO_bW = U \Sigma V^T
$$

Estimate $\hat{\Gamma}_i$ as $U \Sigma^\frac{1}{2}$. Construct $\hat{\Gamma}_{i-1}$ by removing rows from $\Gamma_i$.

Module 3: Repartition the $L$ matrix according to

$$
\begin{bmatrix}
U_{i,1,N+j} \\
U_{1,i-1,N} \\
Y_{i,1,N-1} \\
Y_{i,j+1,N}
\end{bmatrix} =
\begin{bmatrix}
L'_{11} & 0 & 0 & 0 \\
L'_{21} & L'_{22} & 0 & 0 \\
L'_{31} & L'_{32} & L'_{33} & 0 \\
L'_{41} & L'_{42} & L'_{43} & L'_{44}
\end{bmatrix}
\begin{bmatrix}
Q'_1 \\
Q'_2 \\
Q'_3 \\
Q'_4
\end{bmatrix}
$$

Take $L_u$ to be the row in $L$ corresponding to the $U_{i,1,N}$ and $L_y$ the row corresponding to $Y_{i,1,N}$. In MATLAB notation this becomes $L_u = L(1 : m, :)$, $L_y = L(m(i+j) + li + 1 : m(i+j) + li + l, :)$. Construct $L'_f$ from the rows in $L$ corresponding with $Y_{i,j,N}$. Compute

$$
\Xi_{i+1} = \hat{\Gamma}_{i-1}^i \left( \begin{bmatrix}
L_f \\
0
\end{bmatrix} \right)
$$

$$
\Xi_i = \hat{\Gamma}_i^i \left( \begin{bmatrix}
L'_f
\end{bmatrix} \right)
$$

Module 4: Compute $A_T$, $C_T$ and $K$ as the solution of

$$
\begin{bmatrix}
\hat{\Xi}_{i+1,N} \\
L_y
\end{bmatrix} =
\begin{bmatrix}
\hat{A}_T \\
\hat{C}_T
\end{bmatrix} \Xi_{i,N} + K L_u
$$

and recompute $\hat{\Gamma}_i$ based on $\hat{A}_T$ and $\hat{C}_T$. 


Module 5: The matrices $\hat{B}_T$ and $\hat{D}_T$ are the solution of

$$
\hat{B}_T, \hat{D}_T = \arg \left( \min_{\hat{B}, \hat{D}} \| \begin{bmatrix} \hat{\Xi}_{i+1:N} \\ \hat{L}_y \end{bmatrix} - \begin{bmatrix} \hat{A}_T \\ \hat{C}_T \end{bmatrix} \hat{\Xi}_{i:N} - \kappa(B, D) L_u \| \right)
$$

Module 6: The Kalman gain $\hat{K}_T$ is recovered from the residuals of the least squares solution of the previous step.

5.6 Combining MOESP and N4SID

By combining parts of N4SID and MOESP, it is possible to derive various different algorithms. By combining the faster parts of both algorithms, it is possible to find a variant that is computationally more efficient than the original algorithms.

The first obvious difference between N4SID and MOESP is the way in which $A$ and $C$ are calculated. In N4SID, the projected states have to be constructed. From these projected states the matrices $A$ and $C$ are calculated. In MOESP, $A$ and $C$ are found at an earlier stage. Here the matrices are computed directly from $U_u$. Since the latter method is faster, we will use this one.

A second difference is the calculation of $B$ and $D$. In MOESP, a large least squares problem in $B$, $D$ and, if necessary, the initial state is solved. This requires the construction of $Y_{0:N,1}$, $\Gamma_N$, $U$ and $\gamma$. This increases the computation time significantly. In N4SID, $B$ and $D$ are computed based on matrices that are obtained during the computation of $A$ and $C$. Since this is a more direct approach, we use this method for the mixed algorithm.

The mixture of N4SID and MOESP we have examined is the following: $A$ and $C$ are constructed using the MOESP method, directly from the extended observability matrix. Then $B$ and $D$ are found using the N4SID method. The module to estimate $K$ needs to use the MOESP variant, since $R$ is built using MOESP but this is essentially identical to N4SID.

5.6.1 Benchmark test

We tested in total six algorithms. First the three original algorithms MOESP, N4SID and CVA. Next to these, the mixed version of MOESP and N4SID, called MIX1 was tested. To examine the speed improvement from the alternative way of computing $L$, it was incorporated both in an alternative version of MOESP, and in the mixed version, resulting in two new algorithms, which we named respectively FM0ESP and FMIX1.

We tested the algorithms first on an artificial random model of order four, with two inputs and two outputs. Two sequences of 1000 samples of input/output data
were generated using this model. One set was used for identification, the other for validation. The identification output was polluted with 20% measurement noise. The VAF was calculated on the second dataset. To obtain a single number, the VAF was averaged over all outputs. The parameter $i$ was set to eight and the order of the identified system, $n$, was set to the true order of the system, which is four. The experiment was repeated ten times, after which the results were averaged. These results are shown in table 5.6.

Secondly, a larger random model of order ten, with four inputs and twelve outputs, was used to test the algorithms. Again, two sets of 5000 samples of input/output data were generated. The output was polluted with 20% measurement noise. The parameter $i$ was set to twenty and the order of the identified system, $n$, was set to ten. Because of the long time a single experiment takes, only one trial was calculated here. The results are shown in table 5.7. Also here, the reported VAF is the average over all twelve outputs.

Next, we used the 15 dataset from [29] that were taken of the DAISY database. Using these datasets, we identified a model with all six methods. The block matrix parameter $i$ was set to the maximum value that was used in [29]. The order of the identified model was set to half this value. For all algorithms, we measured the number of FLOPS and the amount of CPU-time in seconds. Because the order is chosen “arbitrarily”, the resulting models are not of particular quality and the VAF is not shown, (i.e. very bad in most cases). The main point of this test, however, is to show the different performances of the algorithms on real-life sized datasets. Therefore the quality of the models was irrelevant in this test. For every dataset, we estimated the model ten times. The CPU-time we report is the average over these ten trial. The dimensions of the datasets are summarized in table 5.8.

In table 5.9 and 5.10 the resulting number of FLOPS and CPU-time is given. Table 5.9 shows that CVX has significantly less FLOPS than MOESP and N4SID. This is in correspondence with the results from [29]. MOESP and N4SID are about equal in this sense.

In table 5.10 we see a different picture. Of the three original algorithms, CVX is clearly the slowest. Although the number of floating point operations is smaller than for MOESP and N4SID, the computation time is worse. This can be explained by the fact that the CVX method uses a constrained least squares method to solve for $A$ and $C$. This requires the construction of a large matrix, containing zeros and ones. This is not counted as FLOPS but takes a significant time nonetheless. Figure 5.6.1 shows the number of FLOPS and the CPU-time in a graphical way. This gives a clear indication of the relative speed and the number of operations of the algorithms.
Figure 5.3: Measured FLOPS(left) and CPU-time (right) of the tested algorithms, on the DAISY datasets.

<table>
<thead>
<tr>
<th></th>
<th>mix1</th>
<th>fmix1</th>
<th>fmoesp</th>
<th>moesp</th>
<th>n4sid</th>
<th>eva</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU-time</td>
<td>0.5400</td>
<td>0.4860</td>
<td>0.6450</td>
<td>0.7080</td>
<td>0.6480</td>
<td>0.7380</td>
</tr>
</tbody>
</table>

Table 5.6: Resulting million floating point operations (MFLOPS), computation time and variance accounted for (VAF) for model one, for all six algorithms

<table>
<thead>
<tr>
<th></th>
<th>mix1</th>
<th>fmix1</th>
<th>fmoesp</th>
<th>moesp</th>
<th>n4sid</th>
<th>eva</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mflops</td>
<td>4287.6</td>
<td>580.9</td>
<td>1768.7</td>
<td>5484.0</td>
<td>5130.7</td>
<td>911.1</td>
</tr>
<tr>
<td>CPU-time</td>
<td>198</td>
<td>29</td>
<td>165</td>
<td>341</td>
<td>236</td>
<td>59791</td>
</tr>
</tbody>
</table>

Table 5.7: Resulting million floating point operations (MFLOPS), computation time and variance accounted for (VAF) for model two, for all six algorithms
<table>
<thead>
<tr>
<th>dataset</th>
<th>m</th>
<th>l</th>
<th>n</th>
<th>i</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>ball &amp; beam</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>24</td>
<td>1000</td>
</tr>
<tr>
<td>cd player arm</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>14</td>
<td>1500</td>
</tr>
<tr>
<td>dryer 1</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>24</td>
<td>750</td>
</tr>
<tr>
<td>dryer 2</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>14</td>
<td>600</td>
</tr>
<tr>
<td>evaporator</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>10</td>
<td>5000</td>
</tr>
<tr>
<td>heat exchanger</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>24</td>
<td>3000</td>
</tr>
<tr>
<td>flexible structure</td>
<td>2</td>
<td>1</td>
<td>7</td>
<td>14</td>
<td>6000</td>
</tr>
<tr>
<td>glass furnace</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>10</td>
<td>1000</td>
</tr>
<tr>
<td>pH data</td>
<td>2</td>
<td>1</td>
<td>7</td>
<td>14</td>
<td>1500</td>
</tr>
<tr>
<td>powerplant</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>200</td>
</tr>
<tr>
<td>robot arm</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>24</td>
<td>800</td>
</tr>
<tr>
<td>steam generator</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>10</td>
<td>7000</td>
</tr>
<tr>
<td>thermic wall</td>
<td>2</td>
<td>1</td>
<td>10</td>
<td>20</td>
<td>1000</td>
</tr>
<tr>
<td>winding process</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>1500</td>
</tr>
</tbody>
</table>

Table 5.8: The dimensions of the DAISY datasets used. \( m \) is the number of inputs, \( l \) the number of outputs, \( i \) the used block matrix dimension parameter, \( n \) the order of the identified model and \( N \) the number of samples in the dataset.

<table>
<thead>
<tr>
<th>dataset</th>
<th>mix1</th>
<th>fnmix1</th>
<th>fmoesp</th>
<th>moesp</th>
<th>ndsid</th>
<th>cva</th>
</tr>
</thead>
<tbody>
<tr>
<td>ball &amp; beam</td>
<td>22.9173</td>
<td>6.7698</td>
<td>9.4587</td>
<td>25.8039</td>
<td>24.8189</td>
<td>2.7857</td>
</tr>
<tr>
<td>cd player arm</td>
<td>39.3372</td>
<td>5.0125</td>
<td>10.6355</td>
<td>45.2030</td>
<td>43.4158</td>
<td>4.7362</td>
</tr>
<tr>
<td>dryer 1</td>
<td>18.0562</td>
<td>6.3979</td>
<td>8.3028</td>
<td>20.0618</td>
<td>20.0312</td>
<td>2.6546</td>
</tr>
<tr>
<td>dryer 2</td>
<td>36.4539</td>
<td>9.9721</td>
<td>14.0574</td>
<td>40.8338</td>
<td>50.2641</td>
<td>12.0438</td>
</tr>
<tr>
<td>evaporator</td>
<td>146.7591</td>
<td>11.2746</td>
<td>44.0044</td>
<td>180.6030</td>
<td>151.8486</td>
<td>11.5370</td>
</tr>
<tr>
<td>heat exchanger</td>
<td>60.2864</td>
<td>8.2384</td>
<td>18.6799</td>
<td>71.1663</td>
<td>62.1980</td>
<td>3.7204</td>
</tr>
<tr>
<td>flexible structure</td>
<td>87.2851</td>
<td>6.4728</td>
<td>20.1506</td>
<td>101.8250</td>
<td>88.9316</td>
<td>4.6604</td>
</tr>
<tr>
<td>glass furnace</td>
<td>68.4128</td>
<td>13.5945</td>
<td>31.6528</td>
<td>86.8306</td>
<td>86.2001</td>
<td>18.9537</td>
</tr>
<tr>
<td>pH data</td>
<td>69.0229</td>
<td>13.0650</td>
<td>20.3254</td>
<td>76.7501</td>
<td>76.7783</td>
<td>7.7484</td>
</tr>
<tr>
<td>powerplant</td>
<td>2.0040</td>
<td>0.8732</td>
<td>1.5681</td>
<td>2.7210</td>
<td>2.7979</td>
<td>0.7698</td>
</tr>
<tr>
<td>steam generator</td>
<td>364.0110</td>
<td>26.4220</td>
<td>137.1440</td>
<td>476.0641</td>
<td>375.9107</td>
<td>27.2481</td>
</tr>
<tr>
<td>thermic wall</td>
<td>32.7055</td>
<td>7.5004</td>
<td>10.4479</td>
<td>35.8716</td>
<td>37.3519</td>
<td>4.4906</td>
</tr>
<tr>
<td>winding process</td>
<td>61.2757</td>
<td>8.5658</td>
<td>18.6928</td>
<td>71.7759</td>
<td>69.3506</td>
<td>8.3406</td>
</tr>
</tbody>
</table>

Table 5.9: Millions of floating point operations (MFLOPS), over each method, for fixed value of \( i \) and \( n \), for each dataset.
Table 5.10: Average CPU-time, over 10 trials for each method, for fixed value of $i$ and $n$, for each dataset.

## 5.7 Conclusions

In this chapter we have looked at the implementation issues of SMI algorithms. First we have examined the implementation of MOESP. Significant improvements can be obtained in calculation time by using an efficient implementation and by replacing the QR factorization by a Cholesky factorization.

Then we have examined the estimation of $B$, $D$ and $x_0$. This algorithm is computationally slow due to the large least squares problem that has to be solved. We have shown that, by using an efficient implementation, much computing time can be saved.

Thirdly, we have studied the computational complexity of three different SMI algorithms, namely MOESP, N4SID and CVA. It was shown that although CVA has a significantly lower number of floating point operations, the CPU-time needed for the calculations in this implementation is much higher than for N4SID and MOESP. This is mainly due to the fact that in this algorithm a large matrix must be constructed. This has no impact on the number of floating point operations, but increases the calculation time.

Finally, we have looked at the combination of two types of subspace model identification algorithms, namely MOESP and N4SID. This resulted in a significant increase in the calculation speed, compared with the original algorithms. Further improvements are possible, by improving the speed of the QR factorization. This is a topic for further research.
Chapter 6

The SMI-toolbox

6.1 Introduction

The Subspace Model Identification (SMI) toolbox is a collection of MATLAB functions for the identification of dynamical systems. The routines were developed at the Systems and Control laboratory of Delft University of Technology. Together they form a powerful toolkit for the estimation of a state space model from measured data. The first version of SMI (version 1.0) was introduced in 1997 [37]. In version 2.0, that is presented in this thesis many of the functions have been rewritten, extended or improved. Moreover, several new powerful identification tools are added to the toolbox. Main extension is the addition of the following algorithms: continuous time MOESP, non-causal MOESP and recursive MOESP[62].

The construction of this toolbox is motivated by the need for identification tools that can be used by non-experts. System identification is used in many industrial, medical and academic fields. Often, a model needs to be estimated by someone who is trained in a very different field than system identification. In order for such a person to arrive at a good model, an identification tool needs to guide him or her through the necessary steps. Also the number of tuning parameters has to be kept to a minimum, in order to make the procedure as simple as possible. The SMI-toolbox contains the MOESP family of system identification routines that can deal with those requirements. Moreover, the toolbox contains a number of functions that help the user determine the quality of the model as well as make a choice for the optimal system order. This gives the user the possibility to identify a model without deep knowledge of system identification.

This chapter contains a set of examples to clarify the use of the toolbox. The first example introduces the basic functions and structure of the toolbox with the use of a simple identification problem. Example two shows how multiple data-sets can
be combined for the identification of a model identify a model. In example 3 the identification of a non-linear Wiener system is shown. Finally, in example 4 we use MOESP for the identification of a system from data that is obtained in closed loop.

In appendix B.1 a list of all functions and their description is given.

6.2 Example 1: Using the SMI-toolbox

This section gives an example of the possibilities which the SMI toolbox offers. We will show the identification capabilities of the PO-MOESP scheme using the functions dordpo and destac. This example is also available in the toolbox as smidemo1.

Figure 6.1 shows the structure of the system in this example. First we generate data for a second order system with a pseudo random binary sequence as input $u(k)$. For a realistic example the output signal is contaminated with process noise $w(k)$, that enters the system with the input. The signal to noise ratio (SNR) between the input and the process noise is 20dB. Also Gaussian white measurement noise, $v(k)$, with 20dB SNR is added to the output.

![Figure 6.1: Schematic representation of system used in smidemo1 and smidemo2.](image)

The data is generated with the following MATLAB commands.

```matlab
den=[1 -1.69 0.96];
num=[1 0.5];
u=prbn(300,0.1);
y=dlsim(num,den,u+0.1*std(u)*randn(300,1));
y=y+0.1*std(y)*randn(300,1);
```

In figure 6.2 the generated input and output (with noise) are shown.

We will now try to identify the system from the input $u$ and output $y$. We assume the following state space model:

$x(k + 1) = Ax(k) + Bu(k) + w(k)$

$y(k) = Cx(k) + Du(k) + v(k)$

$x(0) = x_0$
As this model falls within the class of innovation model identification problems, we use the functions `dordpo` and `destac` in this example.

The SMI-toolbox identifies this model in three stages:

1. First, the order \( n \) of the system, is estimated. This is the dimension of the \( A \) matrix.

2. Then the \( A \) and \( C \) matrices are estimated. We can also calculate the Kalman gain if we wish to.

3. In the final step, \( B, D \) and the initial state \( x_0 \) are estimated.

**Step 1**

In this step, we supply `dordpo` with the input/output data and a dimension parameter \( i \). The only restriction on the value of \( i \) is that it needs to be bigger than the order \( n \) of the system. Since the order of the system is generally unknown, in practice we make a rough guess for the order of the system. Then we take the value for \( i \) twice as big as this expected order of the system. The function `dordpo` returns a data matrix \( R \), which contains all the information that could be extracted from the data. This matrix is used in the next step. Also a singular value vector \( S \) is returned. This
vector can be plotted, to visually find the order $n$ of the system. Before starting the identification it is useful to remove the mean of the data, with the function \texttt{detrend}. When this is not done, a bad estimate of the system might be obtained.

\begin{verbatim}
qu=detrend(u,0);y=detrend(y,0);
i=10;
[S,R]=ordpo(u,y,i);
n=ordselect(S)
\end{verbatim}

For this demo, we used a very high value for $i$. This gives a good picture of the relation between $i$ and the order of the system. The singular values are shown in figure 6.3. We clearly see the difference between the second and the third singular value, the gap. In the noise-free case only $n$ singular values are non-zero. When noise is present, the singular values that used to be zero will be perturbed by the noise, but still a gap is visible. In this case we choose the order $n$ to be 2. The gap between SV 2 and 3 is much bigger than between SV 3 and 4. When a non-linear phenomenon is present in the data, no clear gap may be visible. In that case the singular value plot can be used to select an order of the linear approximation on the basis of the largest singular values.

![Figure 6.3: Singular value plot for estimation of the order of the system.](image)
Step 2

In this step, we use the function destac. As input we give it the data matrix \( R \), that we got from dordpo, and the value of \( n \), that we found from the singular value plot. As output, we obtain \( A \) and \( C \).

\[
n=2; \quad [A,C]=\text{destac}(R,n);
\]

Step 3

Once \( A \) and \( C \) are calculated, we can use them to estimate \( B \) and \( D \) and the initial state. This is done using the function destbd.

\[
[B,D,x0]=\text{destbd}(u,y,A,C,[1 1 1]);
\]

Verification

Now that we have found a model from the data, we can see how good this model is. We use the model to estimate the output of the system. This estimated output is then compared with the real output of system without noise.

\[
ye=\text{dlsim}(A,B,C,D,u,x0);
\]
\[
ynf=\text{dlsim}(\text{num},\text{den},u);
\]

As a figure of merit, we take the Variance-Accounted-For (VAF). The VAF is calculated as:

\[
1 - \frac{\text{variance}(y - y_{\text{est}})}{\text{variance}(y)} \times 100\%
\]

This value is an indication of how close the original signal and its estimate resemble each other. If they are completely equal, the VAF is 100%. In other cases, the VAF is lower, down to minus infinity if the error is bigger than the original signal itself.

\[
\text{VAF}=\text{vaf}(y,ye)
\]

The result in this example was 99.8%.
6.3 Example 2: Using the SMI-toolbox with multiple data-sets

In this example we show how the toolbox can be used to combine multiple data-sets for the identification of a single model and how a Kalman filter can be estimated. This example is available in the toolbox as smidemo2.

We consider the identification of a three degree-of-freedom mass spring system given in [50]. This system is schematically represented in figure 6.5 and is a one-input, two-output system. The input is the force $f$ at the top of the structure. The states $x_1$, $x_2$ and $x_3$ are the positions of the masses $m_1$, $m_2$ and $m_3$. The output of the system is taken equal to a combination of the position measurements:

$$ y = \begin{bmatrix} x_1 + x_2 + x_3 \\ x_1 - x_3 \end{bmatrix} $$

In addition to the measurable force $f$, depicted in the figure, the system is excited by unmeasurable white noise sequence $w_k$ entering at the same point as $f$. The output is perturbed by another white noise sequence $v_k$. 
As such the defined problem falls within the class of innovation model problems as shown in figure 6.1. Therefore, this example highlights the use of the functions dordpo and destac. The system, without the disturbances is shown in figure 6.5.

![Diagram of 3 degree-of-freedom mass spring system](image)

Figure 6.5: Schematic view of the 3 degree-of-freedom mass spring system.

The system is represented using the state space matrices $A$, $B$ and $C$. $D$ equals zero in our example. The noise signal $w_k$ is process noise with covariance matrix $Q$. The measurement noise signal $v_k$ has covariance $R$.

\[
A = \begin{bmatrix}
[[.9856 \ .1628;-.1628 \ .9856] \& \ zeros(2,4); \\
zeros(2,2) \& \ [.8976 \ .4305;-.4305 \ .8976] \& \ zeros(2,2); \\
zeros(2,4) \& \ [.8127 \ .5690;-.5690 \ .8127]];
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
[.0011;\ .0134;-.0016;-.0072;\ .0011;\ .0034];
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
[1.5119 \ 0 \ 2 \ 0 \ 1.5119 \ 0; \\
1.3093 \ 0 \ 0 \ -1.3093 \ 0];
\end{bmatrix}
\]

\[
D = zeros(2,1);
\]

\[
Q = 10^{-(-4)}*diag([.0242 \ .5920 \ .0534 \ .1034 \ .0226 \ .2279]);
\]

\[
R = 10^{-(-2)}*diag([2.785 \ 2.785]);
\]

Using the above system matrices we generate three sets of input/output data. The first set is generated with a pink input noise input $u_1$. It is generated by filtering random sequence by a 4th order butterworth filter. The second set is generated with a multi-sine input $u_2$ and the third with a pseudo random binary sequence $u_3$. A fourth data-set with pink noise input is reserved for validation.

Figure 6.6 shows an example of the three inputs that are used for identification.

We will now use the three data batches to estimate back the system matrices $A$, $B$, $C$ and $D$ without knowledge of the system itself. We will do this in three steps. First we will build a data matrix $R$ from the available data batches and estimate the order of the system using dordpo. We specify $i$ to be 12, which is bigger than the system.
Figure 6.6: Three different input signals.
order we expect. The function dordpogathers the information from the first data batch \([u_1, y_1]\) in the matrix \(R_1\). In a second call to dordpo the information from the second data batch \([u_2, y_2]\) is added to \(R_1\) and stored in \(R_2\). Finally the third data batch is added and the final result is stored in \(R_3\). This is done with the following matlab commands.

\[
\begin{align*}
    u1 &= \text{detrend}(u1); y1 = \text{detrend}(y1); \\
    u2 &= \text{detrend}(u2); y2 = \text{detrend}(y2); \\
    u3 &= \text{detrend}(u3); y3 = \text{detrend}(y3); \\
    [Sn1, R1] &= \text{dordpo}(u1, y1, 12); \\
    [Sn2, R2] &= \text{dordpo}(u2, y2, 12, R1); \\
    [Sn3, R3] &= \text{dordpo}(u3, y3, 12, R2);
\end{align*}
\]

We can take a look at the singular values of the intermediate steps, to see the improvement of our knowledge about the system. Figure 6.7 shows the singular values of \(Sn_1\), \(Sn_2\) and \(Sn_3\). After every added data batch, the singular values that belong to the system are higher and the gap becomes more clear. From the third plot we estimate the system order to be 6.

Next we estimate \(A\) and \(C\) from the final data matrix \(R_3\). For this we use the function destac.

\[
[\text{Ae}, \text{Ce}] = \text{destac}(R3, 6);
\]

With this estimate of \(A\) and \(C\), we find \(B\) and \(D\). For this, we use the function destbd Again we pass all three datasets one by one to destbd and gather the information in a data matrix. At every step we get intermediate results for the estimated \(B\) and \(D\).

\[
\begin{align*}
    [\text{Be}, \text{De}, R1bd] &= \text{destbd}(u1, y1, \text{Ae}, \text{Ce}, [1 \ 1 \ 0]); \\
    [\text{Be}, \text{De}, R2bd] &= \text{destbd}(u2, y2, \text{Ae}, \text{Ce}, [1 \ 1 \ 0], R1bd); \\
    [\text{Be}, \text{De}] &= \text{destbd}(u3, y3, \text{Ae}, \text{Ce}, [1 \ 1 \ 0], R2bd);
\end{align*}
\]

Now we would like to verify the accuracy of our model. Because the system is marginally stable, our model can become unstable due to a small error on the estimated poles. Therefore comparing the output of our model with the real output is, in this case, not a fair comparison. We compare the position of the estimated poles of the model with respect to the true system. This is shown in figure 6.8.

Because it is not possible to predict the output on the basis of the input alone when the model is unstable, we will now estimate a Kalman-filter, to build an one-step ahead predictor on the basis of our model. The one-step ahead predictor is given by

\[
\begin{align*}
    \dot{x}(k + 1) &= Ax(k) + Bu(k) + K(y(k) - \hat{y}(k)) \\
    \hat{y}(k) &= Cx(k) + Du(k)
\end{align*}
\]
Figure 6.7: Singular values after processing batch 1, 2 and 3.
Figure 6.8: Estimated (+) and true (×) pole positions.
The function `destk` can be used to estimate the Kalman gain $K$.

$$K = \text{destk}(\text{Ae, Be, Ce, De, R3});$$

Also we calculate the initial state for the validation data batch:

$$x04e = \text{destx}(u4, y4, \text{Ae, Be, Ce, De});$$

With these estimates, we calculate the one-step ahead prediction of the output:

$$\text{yek} = \text{dlsim}(\text{Ae-Ke*Ce, [Be-Ke*De Ke], Ce, [De zeros(2,2)], [u4 y4], x04e});$$

To evaluate our estimate, we compare it with the true one-step ahead prediction, where the Kalman gain is calculated from the real system matrices $A$, $B$, $C$, $D$, $Q$, $R$ and $S$.

$$L = \text{dlqe}(\text{A, eye(6)}, C, Q, R);$$

$$K = A \cdot L;$$

$$\text{yk} = \text{dlsim}(A-K*C, [B-K*D K], C, [D zeros(2,2)], [u4 y4]);$$

$$\text{vaf(yek, yk)}$$

The VAF between the output of the estimated one-step ahead predictor $y_e(k)$ and the output of the optimal one-step ahead predictor $y(k)$ was for the first output 95%. For the second output we found a VAF of 93%.

### 6.4 Example 3: Identifying a Wiener model

In this example we show the capabilities of the SMI-toolbox to identify non-linear systems of the Wiener type. The example is taken from [102] and is available in the toolbox as smidemo3.

![Figure 6.9: Schematic representation of the Wiener model.](image)
The LTI part of the system under consideration is a second order system with the following transfer function.

\[ H(s) = \frac{0.1578s + 0.1379}{s^2 - 1.3746s + 0.6703} \]

The non-linear part is formed by a saturation, with a cut-off from below at 0.5 and from above at 1.5. This is shown in figure 6.10.

\[ f(y) = \begin{cases} 
0.5, & \text{if } y < 0.5 \\
y, & \text{if } 0.5 \leq y \leq 1.5 \\
1.5, & \text{if } 1.5 < y 
\end{cases} \]

![Non-Linear function, formed by a saturation.](image)

Figure 6.10: Non-Linear function, formed by a saturation.

First we demonstrate the use of the pi-moESP routine to estimate the LTI part of such a system. The pi-moESP algorithm is able to estimate the LTI part unbiased if the input has a Gaussian distributions.

The system is simulated using the following code

```matlab
% system transfer function
num = [.1578 .1379];
den = [1 -1.3746 .6703];

% input and output signals
N = 1000;
t = [0:N-1]';
unf=2*randn(N,1);
ynf = dlsim(num,den,unf);
```
The non-linear element
\[ znf = (ynf<.5)*.5+((ynf>-.5)&(ynf<-1.5)).*ynf+(ynf>1.5)*1.5; \]

5 percent process noise and 5 percent measurement noise are added:
\[ u = unf; \]
\[ usim = unf + .05*std(unf)*randn(size(unf)); \]
\[ y = dlsim(NUM,DEN,u); \]
\[ zsim = (y<-.5)*.5+((y>-.5)&(y<1.5)).*y+(y>=1.5)*1.5; \]
\[ z = zsim+.05*std(zsim)*randn(size(zsim)); \]

Figure 6.11 shows the Gaussian input, the output of the linear part and the output of the non-linear part of the system.

![Figure 6.11: Gaussian input, linear output and non-linear output of example system.](image)

We will estimate the Wiener model in two steps. First the linear part is estimated using dordpi, destac and destbdi. Then we use the function chebest to estimate the non-linear function.

For the estimation of the linear dynamic part, we use the following commands.

\[ zd = \text{detrend}(z); ud = \text{detrend}(u); \]
\[ [S_n,R] = \text{dordpi}(ud,zd,5); \]
\[ [Ae, Ce] = \text{destac}(R, 2); \]
\[ [Be, De] = \text{destbd}(ud, zd, Ae, Ce); \]

Once we know the linear dynamic part, we can estimate the static non-linear function, between the estimated output \( y_e \) of the linear model and the measured output \( z \) of the Wiener system. We can do this for instance with an estimation based on Chebyshev polynomials[99].

\[ ye = \text{dlsim}(Ae, Be, Ce, De, u); \]
\[ [thl, zel] = \text{chebest}(ye, z, 7); \]

Figure 6.12 shows the relation between the estimated linear output \( y \) and the non-linear output \( z \). The \( \times \)'s are the relation between two signals. The solid line is the approximation with the Chebyshev polynomials up to order 7. The VAF for of this estimate was 99%. The figure shows that the Chebyshev polynomials provide a good fit of the function within the region where data is available. However, one has to be careful with the use of the model for extrapolation outside this region.

![Figure 6.12: The true and the estimated non-linear function.](image)

The PI-MOESP method will fail to give an unbiased estimate of the linear part [89] when the input is not Gaussian. We then need to use a non-linear optimization technique to find a consistent estimate. The SMI Toolbox contains Gauss-Newton
optimization routines for linear models as well as Wiener models. These routines use the SLS technique to minimize the number of parameters to be optimized\cite{16}. These functions are called\texttt{ds1slin} and\texttt{ds1swie}. In the next example we have used the same system as before, but now a random binary sequence is used as input. The signals are shown in figure 6.13.

![Graph showing non-linear output, linear output, and noise-free input](image)

Figure 6.13: Random binary input, linear output and non-linear output of example system.

As an initial estimate of the system, we use the value \texttt{pi-MOESP} provides us. Although we know this value will not be correct, it is a good approximation of the true system, which increases the chance of finding the global optimum during the Gauss-Newton optimization.

```
[Sn,R] = dordpi(u,z,5);
[Ad,Cd] = destac(R,2);
[Bd,Dd] = destbd(u,z,Ad,Cd);
```

With this initial estimate, we proceed with the Gauss-Newton optimization function\texttt{ds1swie}. This function takes as input the estimated matrices and the input and output of the system. The next input argument is a vector containing some information about the model structure. This vector contains the order\texttt{m} of Chebyshev polynomials, whether we want to estimate the\texttt{D} matrix and if we need an estimate of the initial state. In the example we only estimate\texttt{D}. The last argument contains options for the optimization routine.\texttt{ds1swie} uses the function\texttt{leastsq} internally. This function is part of the\texttt{MATLAB} optimization toolbox.
6.5 Example 4: Identifying in a closed loop setting.

After the optimization we simulate the model. As before we plot the relation between the estimated linear output and the non-linear output in one plot with $\times$'s and the estimated non-linear function with a solid line. The result is shown in figure 6.14.

![Figure 6.14: Estimated non-linear function between $y$ and $z$](image)

6.5 Example 4: Identifying in a closed loop setting.

In this example we show the capabilities of the SMI-toolbox to identify a system, when the data is measured in closed loop, using the errors-in-variables (EIV) algorithm. The example is available in the toolbox as smidemo4.

The model structure for the EIV problem is an innovations model with noisy input
and noisy output.

\[
\begin{align*}
x(k+1) &= Ax(k) + B\tilde{u}(k) + w(k) \\
y(k) &= Cx(k) + D\tilde{u}(k) + v(k) \\
u(k) &= \tilde{u}(k) + w(k)
\end{align*}
\] (6.1) (6.2) (6.3)

Only \(u(k)\) and \(y(k)\) are available for identification. \(w(k), v(k)\) and \(f(k)\) are zero-mean white noise sequences independent of the noise-free input \(\tilde{u}(k)\).

![Schematic representation of the errors-in-variables identification problem.](image)

Figure 6.15: Schematic representation of the errors-in-variables identification problem.

The plant is a mechanical system consisting of 2 rotating discs and an integrator. Its transfer function (order = 5) is

\[
P(z) = 10^{-3} \frac{0.98z^4 + 12.99z^3 + 18.59z^2 + 3.30z - 0.02}{z^5 - 4.44z^4 + 8.09z^3 - 7.83z^2 + 4z - 0.86}
\]

A controller has been designed based on \(H_\infty\) method

\[
C(z) = \frac{0.61z^4 - 2.03z^3 + 2.76z^2 - 1.83z + 0.49}{z^4 - 2.65z^3 + 3.11z^2 - 1.75z + 0.39}
\]

The process noise \(w(t)\) is shaped by the following filter

\[
F(z) = 10^{-2} \frac{2.89z^2 + 11.13z + 2.74}{z^3 - 2.7z^2 + 2.61z - 0.9}
\]

We first build a combined deterministic/stochastic system with as input respectively the plant input, process noise, input measurement noise and output measurement noise. The two outputs of the combined system are the noisy plant output and the noisy plant input. This system is then put in closed loop with the \(H_\infty\) controller as shown in figure 6.16.

```matlab
plant=tf(Pnum,Pden);
controller=tf(Cnum,Cden);
oisefilter=tf(Pnum,Fden);
```
Example 4: Identifying in a closed loop setting.

\[
\text{interconnection}=\text{ss}([],[],[],[1 0 1; 0 1 0]);
\text{combinedsystem}=\text{parallel}(\text{plant}, \text{noisefilter}, [], [], 1, 1);
\text{combinedsystem2}=\text{parallel}(\text{combinedsystem1}, \text{interconnection}, 1, 1, 1, 2);
\text{closedloopsystem}=\text{feedback}(\text{combinedsystem2}, \text{controller}, 2, 1, -1);
\]

![Schematic representation of the closed loop model.](image)

Figure 6.16: Schematic representation of the closed loop model.

With this closed loop system we simulate 1200 samples. The external input, \(r(k)\), a white noise of standard deviation (std) 1, is used to excite the system. The white noise, \(w(k)\), driving the process noise filter \(F(z)\) has a std 0.11. The input and output noise, \(f(k)\) and \(v(k)\) are both of std 0.01. This results in SNRs of approximately 20dB at the plant output and 5dB at the plant input. Figure 6.17 shows the simulated input and output.

Now we have the data needed for identification. The data available for identification is plant input \(u(k)\), plant output \(y(k)\) and the external excitation \(r(k)\).

The first step in identification is to find out about the order of the system. The function `dordeiv` estimates the singular values and returns them in the vector \(S\) from which the order can be found.

\[
[S,R] = \text{dordeiv}(u,y,r,20);
\]

The order can then be visually inspected with the function `orderselect`.

\[
n=\text{orderselect}(S);
\]

The singular value plot indicates that the order of the system is 7. The expected order is 8 which is the sum of the orders of the plant and the process noise shaping filter. One pole cannot be recovered from identification and this pole belongs to the shaping filter. The information contents on the process noise shaping filter is usually very little in closed-loop as the controller is designed to reject this signal.

We proceed to estimate the system matrices. First the \(A\) and \(C\) matrices are estimated using the function `destac`.
Figure 6.17: Input and output signal of the closed-loop system.

Figure 6.18: Singular values shown in the orderselect GUI.
\[ [A,C] = destac(R,7); \]

For the estimation of \( B \) and \( D \) we use the reference input as instrumental variable, in order to arrive at a consistent estimate.

\[ [B,D] = destbd(u,y,A,C,[],[],r); \]

Figure 6.19 compares the frequency responses of the estimated model to the one of the true systems.

![Frequency response comparison](image)

Figure 6.19: Frequency response of the estimated model (dashed) and the true system (solid).

### 6.6 Conclusion

This chapter has given a number of examples of the use of the SMI toolbox. The examples showed how a simple model can be identified using measured data. The data can be obtained from different experiments and concatenated in order to arrive at a single high quality model. The examples also showed how a model can be optimized using the separable least square functions of the toolbox and how a non-linear Wiener
model can be estimated. Finally the estimation of a model in closed loop has been shown.
Chapter 7

Conclusions

In this final chapter we present general conclusions of the research described in this thesis. More detailed conclusions are given at the end of each chapter. Also some remarks about further research are made.

This thesis is dedicated to the topic of system identification, which is a wide and (to my personal opinion) a fascinating research area. Although there were many directions in this area to explore, we had to restrict ourself within the four years period in which the Ph.D. thesis had to be completed.

The goals to be set for this thesis therefore had to be concrete and limited to only a selected part of the whole field. We focused on one of the most promising and recent developments in system identification, namely the Subspace Model Identification methods. To this topic our research group has made major contributions in the past.

At the start of this thesis we defined the goals.

- The extension of MOESP to new model classes and identification problems, to include new application areas.
- The combination of PEM and SMI, to benefit from the advantages of both methods.
- The construction of a subspace identification toolbox, to present the methods in an user-friendly and unified way.

These tasks have been addressed in the previous chapters. The MOESP methods have been extended with the continuous time identification of state space models. This allows the identification of a state space model in the continuous time from sampled data. Several algorithms have been proposed and analyzed. For the combination of PEM and SMI we have introduced the Separable Least Squares technique. This
allows the use of SMI to obtain an initial estimate of the model and the use of PEM for a parametric optimization of this model. The thesis contains algorithms for the optimization of parameterized state space models in batch form as well as algorithms for the recursive identification on a sample by sample basis. The third goal, the construction of a system identification toolbox has been achieved by the completion of the SMI toolbox. This toolbox contains amongst others, the algorithms that were described in this thesis. An attempt is made to provide a user-friendly interface for choosing the right method and the most adequate parameters.

In chapter 2 we have presented the state of the art of the MOESP type of SMI algorithms, which is the background needed for further research in this direction. The chapter has given a detailed overview of the existing discrete time MOESP algorithms. Such an overview has not been presented before, to the author's knowledge. The different types of algorithms have been described on the basis of the identification problem that they solve: the noise free identification problem, the output-error problem, the innovations model problem and the errors-in-variables problem. We have shown how all these problems can be solved by one basic algorithm with few modifications. Also the identification of non-linear Wiener systems has been described. Furthermore, the algorithms for the identification of discrete time linear systems have been investigated for the requirements on the input that they pose, in order to obtain a consistent estimate of the system.

The algorithms that existed thus far in the SMI framework were almost all developed for the identification of discrete time systems. In chapter 3 we have introduced a set of new identification algorithms, based on the MOESP method: the continuous time identification of state space models from sampled data. The described algorithms are able to solve a number of difficult identification problems, such as the identification of models in the case of non-equidistantly sampled data or data from stiff systems. In these cases discrete methods often fail due to numerical problems. The method is able to deal with data that is contaminated process and measurement noise. We have presented an error analysis of the new method and we have shown that in this case the method is consistent in the limit. By extending the identification scheme for the Kalman filter from discrete time to the continuous time setting, the algorithm is able to estimate an one-step ahead predictor directly from data. Finally the extension of the continuous time method to the identification of non-linear Wiener systems has been derived.

In chapter 4 a new method for the combination of PEM and SMI has been described. In this combination SMI is used to obtain an initial estimate of the system parameters. The model that is obtained with SMI is then parameterized. The parameterized model is optimized using the PEM optimization procedure. We have introduced the Separable Least Squares (SLS) technique as a means to minimize the number of parameters in the optimization. We have shown that when we use a suitable parameterization, like the output normal parameterization, the MOESP method can be restricted to estimating only the matrices $A$ and $C$ of the state space model. Because in the SLS optimization, only the parameters describing $A$ and $C$ are used. This reduces the number of parameters to the same number as would be used in
a transfer function based optimization. Practical evidence has shown that MOESP delivers accurate initial estimates such that in the optimization only a few iterations are necessary. Examples in this thesis have shown the advantages of this method over straightforward optimization of the cost-function with a random initial estimate. Next we have developed a recursive SLS optimization. In this recursive algorithm, the optimization takes place on a sample basis rather than on the entire data batch. This allows the use of the method in an on-line identification process, for instance for process parameter tuning. Finally the SLS method has been extended from the discrete time case to the identification of continuous time models and non-linear Wiener models.

Three different Subspace Model Identification methods have been investigated in chapter 5, namely MOESP, N4SID and CVA. Measurements have shown that the three methods differ significantly in terms of computational requirements. A number of improvements have been proposed to obtain a faster SMI algorithm. Firstly we have optimized the MOESP algorithm by replacing the LQ factorization by a multiplication and a Cholesky factorization. The estimation of the matrices $B, D$ and the initial state $x_0$ have been optimized by a number of modifications of the algorithm. Secondly we have investigated the combination of the MOESP algorithm with the N4SID algorithm. This has resulted in a novel algorithm that is significantly faster than the original two.

Finally, in chapter 6 a toolbox of Subspace Model Identification functions for the popular MATLAB toolbox has been introduced. This toolbox is called the SMI-toolbox and provides a set of identification routines that are reliable, fast and user-friendly. The toolbox implements, amongst others, the algorithms that have been described in this thesis. Together these functions provide a powerful set of tools that can tackle a wide range of system identification problems. The chapter has shown a number of examples that can help the user in setting up a successful identification run.

In this thesis we have presented both a theoretical treatment of Subspace Model Identification and a practical approach for performing identification experiments using the SMI-toolbox. Although many topics have been treated in this thesis, this does not imply that the work has been finished. Many ideas were left untouched due to time limitations. Many questions still need to be answered and many tasks still need to be done. On the topic of continuous time identification, the use of higher order approximations of the Laguerre filters could deliver more accurate estimates[23]. The discrete approximation of the Laguerre filter can be implemented using the delta notation of Goodwin and Middleton [65]. This would allow the use of this algorithm at higher sampling rates, where now the approximation of the filters is the limiting factor. The SLS approach, which combines SMI with PEM identification leaves a lot of things to be explored. Which parameterization is most suitable for this type of identification and what is the influence of a non-minimal parameterization, are two of the questions that need to be answered. The implementation and maintenance of the SMI-toolbox functions are a continuous process. The goal of building a consistent tool for system identification requires a thoughtful design of the user interface. In future versions of the toolbox more identification routines could be included and tools could
be provided to identify a wider range of models.

The last decades have shown a steady growth in knowledge of identification methods and a solid basis of linear system identification techniques has been build. However, the field of system identification is still in progress. The theory for the identification of non-linear systems is not very much developed and is difficult to investigate. This field has many difficulties. Partial solutions have been found in the identification of special types of non-linear systems. This thesis contains one of those systems: the Wiener model. Other types are Hammerstein models[94] and bilinear [28, 85] models. The identification of more general types of non-linear systems is a much more difficult task. This is the real challenge for the future.
Appendix A

Important basic concepts

A.1 Projection matrices

Orthogonal projection matrices have an important role in linear algebra. In this thesis we make regularly use of the properties of projection matrices. Therefore we give a brief overview of projection matrices in this section.

Given a matrix $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) whose column-space equals $S_A \subseteq \mathbb{R}^n$. Then the orthogonal projection of a matrix $B \in \mathbb{R}^{p \times n}$ onto $S_A$ constitutes of left multiplying of $B$ with a projection matrix, denoted by $P_A$. Provided that $A$ has full column rank, $P_A$ can be calculated as follows.

$$ P_A = A(A^T A)^{-1} A^T \quad (A.1) $$

The matrix $P_A$ projects the columns of the matrix $B \in \mathbb{R}^{p \times n}$ orthogonally onto the subspace $S_A$.

$$ \text{column-space}(P_A B) \subseteq S_A $$

The orthogonal complement of the column-space of $A$ is given by $S_A^\perp = \mathbb{R}^m / S_A$. The projection matrix

$$ P_A^\perp = I_m - P_A \quad (A.2) $$

projects the columns of $B$ orthogonally onto $S_A^\perp$ such that

$$ P_A^\perp B \subseteq S_A^\perp $$

Equivalently, the orthogonal projection of the rows of a matrix $B \in \mathbb{R}^{m \times p}$ on the row-space of $A \in \mathbb{R}^{m \times n}$, $(m \leq n)$ is a right multiplication with the matrix $\Pi_A$. If $A$
has full row rank, the projection matrix is calculated as:

\[ \Pi_A = A^T (A A^T)^{-1} A \]  \hspace{1cm} (A.3)

and has the property

\[ \text{row-space}(B \Pi_A) \subseteq \text{row-space}(A) \]

The projection matrix

\[ \Pi_A \subsetneq I_n - \Pi_A \]  \hspace{1cm} (A.4)

projects the rows of the matrix \( B \) onto orthogonal complement of the row-space of \( A \).

\[ \text{row-space}(B \Pi_A) \subseteq \mathbb{R}^n / \text{row-space}(A) \]

**Example A.1** As a simple example of the orthogonal projection we take \( A \) and \( B \) two dimensional vectors, \( A = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \) and \( B = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \). The orthogonal projection of \( B \) onto the column spanned by \( A \), can be found easily. Using formula A.1 for \( P_A \) we have

\[
P_A = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \left( \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix}^T = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \]

The projection \( P_A B \) becomes

\[
P_A B = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 2/3 \\ 2/3 \end{bmatrix}
\]

In the case of two two-dimensional vectors we can show the projection in a graphical way. In figure A.1 the orthogonal projection of \( B \) onto the line spanned by \( A \) is shown. For higher dimensional cases, the projection can be thought of in the same way. ■

**A.2 The QR factorization**

A basic matrix factorization in linear algebra, that we will use regularly, is the QR-factorization. The QR factorization of a matrix \( A \in \mathbb{R}^{m \times n} \) is given by

\[ A = QR \]  \hspace{1cm} (A.5)

where \( Q \in \mathbb{R}^{m \times m} \) is a matrix with orthonormal columns such that \( Q^T Q = I_m \) and \( R \in \mathbb{R}^{m \times n} \) is an upper triangular matrix.

When \( m > n \) the last \( m - n \) rows of \( R \) will contain only zeros. Therefore we can reduce the sizes of \( Q \) and \( R \) to an “economy size” by removing the last \( m - n \) columns of \( Q \) and the last \( m - n \) rows of \( R \). In this case \( Q \in \mathbb{R}^{m \times n} \) and \( R \in \mathbb{R}^{n \times n} \).
There are many interesting applications of the QR factorization, but here we shall limit ourselves to one of them, namely the orthogonal projection. We defined the orthogonal projection in the section A.1.

**Lemma A.1** Let the orthogonal projection of the columns of a matrix $B \in \mathbb{R}^{m \times p}$ onto the column-space spanned by $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) be denoted by $P_{AB}$ where $P_A$ is given by equation A.1. $P_{AB}$ can be calculated using the QR factorization in the following way:

Calculate the QR factorization equation A.6.

\[
\begin{bmatrix}
A & B
\end{bmatrix} =
\begin{bmatrix}
Q_1 & Q_2
\end{bmatrix}
\begin{bmatrix}
R_{11} & R_{12} \\
0 & R_{22}
\end{bmatrix} \quad (A.6)
\]

and divide it into blocks, such that $Q_1 \in \mathbb{R}^{m \times n}$, $Q_2 \in \mathbb{R}^{m \times m-n}$, $R_{11} \in \mathbb{R}^{n \times n}$, $R_{12} \in \mathbb{R}^{n \times p}$ and $R_{22} \in \mathbb{R}^{m-n \times p}$.

Then the orthogonal projection $P_{AB}$ is given by $Q_1R_{12}$.

**Proof:**

Using the QR factorization of $\begin{bmatrix} A & B \end{bmatrix}$, we find that

\[
A = Q_1R_{11}
\]

\[
B = Q_1R_{12} + Q_2R_{22}
\]

Note that $Q_1^TQ_1 = I_n$, $Q_2^TQ_2 = I_{m-n}$ and $Q_1^TQ_2 = 0$. 

When we use this in the equation for the projection we obtain:

\[
P_AB = A(A^TA)^{-1}A^TB \\
= Q_1R_{11}(R_{11}^{-1}Q_1^TQ_1R_{11})^{-1}R_{11}^TQ_1^T(Q_1R_{12} + Q_2R_{22}) \\
= Q_1R_{11}(R_{11}^{-1}R_{11})^{-1}R_{11}^TR_{11}R_{12} \\
= Q_1R_{11}^{-1}R_{11}^{-1}R_{11}^TR_{11}R_{12} \\
= Q_1R_{12}
\]

![Box](image)

**Example A.2** As a simple example we of the orthogonal projection using the QR factorization we revisit the example of the previous section. \(A = \begin{bmatrix} 1 \\ 1 \end{bmatrix}\) and \(B = \begin{bmatrix} 2 \\ 3 \end{bmatrix}\). The QR factorization of \([ A \ B ]\) becomes:

\[
\begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{2} & 2\frac{1}{2}\sqrt{2} \\ 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}
\]

The projection is then given by

\[
\Pi_A B = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 2\frac{1}{2}\sqrt{2} \end{bmatrix} = \begin{bmatrix} 2\frac{1}{2} \\ 2\frac{1}{2} \end{bmatrix}
\]

which is equal to the projection found with formula A.1, as expected.  

![Box](image)

The LQ factorization is very similar to the QR factorization. In the LQ factorization, a matrix \(A \in \mathbb{R}^{m \times n}\) is factorized into a matrix \(Q\), with orthonormal rows such that \(QQ^T = I\), and a lower triangular matrix \(L\) such that:

\( A = LQ \)

The LQ factorization of a matrix can be easily found using the QR factorization. If the QR-factorization of \(A\) is given by \(QR\), the LQ factorization of \(A^T\) is given by \(R^TQ^T\). As the QR factorization, the LQ factorization can be used to do orthogonal projections, but in this case the projection is done on the row-space and the projection is a right multiplication instead of a left multiplication. In subspace identification we will see this use of the LQ factorization back frequently.

### A.3 The Singular Value Decomposition

A second important matrix factorization that we treat is the Singular Value Decomposition, or short the SVD. The SVD is is used for solving an almost infinite number of numerical problems, ranging from image processing to system identification.
A.3 The Singular Value Decomposition

The SVD of a matrix $A \in \mathbb{R}^{m \times n}$ is given by [31]

$$A = U S V^T$$  \hspace{1cm} (A.7)

where $U \in \mathbb{R}^{m \times m}$ is an orthogonal matrix such that $U^T U = U U^T = I$. The columns of $U$ are called the left singular vectors. $S \in \mathbb{R}^{m \times n}$ is a diagonal matrix, with positive entries on the diagonal, which are called the singular values of the matrix $A$. The singular values in $S$ are sorted, such that they decrease in magnitude when going down the diagonal. $V \in \mathbb{R}^{n \times n}$ is a second orthogonal matrix. The columns in $V$ are the right singular vectors.

When $m > n$ the last $m-n$ rows of $S$ will contain only zeros. Therefore we can reduce the sizes of $U$, $S$ and $V$, by removing the last columns and rows, to be respectively $m \times n$, $n \times n$ and $n \times n$.

For symmetric positive definite matrices the factorization is identical to the eigenvalue decomposition and $U$ is equal to $V$. In other cases $U$ is unequal to $V$, but still the matrix $S$ contains only positive elements.

The first $r$ columns of $U$ are a basis for the column-space of $A$. This means the matrix $A$ can be constructed by taking linear combinations of these columns in $U$. The value of $r$ can be found from the number of nonzero singular values in $S$. The row-space of $A$ is spanned by the first $n$ rows of $V^T$. In this case we have:

$$A = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} S_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} = U_1 S_1 V_1^T$$

An important use of the SVD is for rank reduction or low rank approximation. Denote the $i$-th column of $U$ with $u_i$ and use $v_i$ for the $i$-th column of $V$. The $i$-th diagonal element of $S$ we denote by $\sigma_i$. Then the matrix $A$ can be written as

$$A = u_1 \sigma_1 v_1^T + u_2 \sigma_2 v_2^T + \cdots = \sum_{i=1}^{\min(m,n)} u_i \sigma_i v_i^T$$  \hspace{1cm} (A.8)

Note that all $u_i$ and $v_i$ have unit norm, because they come from orthogonal matrices and $\sigma_i$ decreases in magnitude as $i$ increases. Therefore the influence of the terms in the summation on the resulting sum decrease, with increasing $i$. Depending on the magnitude of $\sigma_i$ we can decide to reduce the number of terms in the sum A.8, because we add less and less information. The following lemma states this in a formal way.

**Lemma A.2 (Golub and van Loan)** Let the SVD of $A \in \mathbb{R}^{m \times n}$ be given by $USV^T$. If $k < \text{rank}(A)$ then the difference between $A$ and the best approximation of $A$ by a matrix $B$ with rank $k$ is bounded by the following equation(with the notation from equation A.8).

$$\min_{\{B \in \mathbb{R}^{m \times n} \mid \text{rank}(B) = k\}} \|A - B\|_2 = \sigma_{k+1}$$
and the approximation itself by

$$\arg \min_{\{B \in \mathbb{R}^{m \times n} \mid \text{rank}(B) = k\}} \|A - B\|_2 = \sum_{i=1}^{k} u_i \sigma_i v_i^T$$

Example A.3 We take the matrix $A$ a very simple square $3 \times 3$ matrix

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Note that this matrix is of rank 2. We can form the last column of this matrix by taking twice column 2 minus column 1. The SVD of this matrix is given by

$$\begin{bmatrix} 0.4797 & 0.7767 & -0.4082 \\ 0.5724 & 0.0757 & 0.8165 \\ 0.6651 & -0.6253 & -0.4082 \end{bmatrix} \begin{bmatrix} 16.8481 & 0 & 0 \\ 0 & 1.0684 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.2148 & -0.8872 & 0.4082 \\ 0.5206 & -0.2496 & -0.8165 \\ 0.8263 & 0.3879 & 0.4082 \end{bmatrix}^T$$

We see that there are only two non-zero singular values. This confirms that the rank of $A$ is 2. Therefore we only need two singular vectors and singular values to exactly describe $A$. Moreover, the first singular value (16.8481) is much larger than the second one (1.0684). The best rank 1 approximation of $A$ (in the two norm sense) is obtained by using only the first singular value and singular vectors. We get

$$u_1 \sigma_1 v_1^T = \begin{bmatrix} 0.4797 \\ 0.5724 \\ 0.6651 \end{bmatrix} \begin{bmatrix} 16.8481 \\ 0.2148 \\ 0.8263 \end{bmatrix}^T = \begin{bmatrix} 1.7362 & 4.2072 & 6.6781 \\ 2.0717 & 5.0202 & 7.9686 \\ 2.4073 & 5.8332 & 9.2592 \end{bmatrix}$$

Low rank approximations of matrices are used in many area’s. We will use it in subspace identification to obtain a low order model. This will be discussed in detail in chapter 2. Another possible application is image compression. If we are dealing with a $1000 \times 1000$ image matrix, we can reduce the number of elements we need to deal with dramatically. By doing a rank two approximation we don’t need to save all $1000 \times 1000$ entries. Instead we need only $4 \times 1000 + 2$ entries!

A.4 The Kronecker product

The Kronecker product is not a very common operation in matrix calculus, although it has several nice properties, which can be taken advantage of in many matrix problems. Because many readers might be unfamiliar with Kronecker products, its definition and a small overview of it’s properties will be given here. A more elaborate overview is
given in [15]. The Kronecker product of a matrix \( A \in \mathbb{R}^{p \times q} \) and \( B \in \mathbb{R}^{m \times n} \) is denoted by \( A \otimes B \in \mathbb{R}^{pm \times qn} \) and is given by

\[
A \otimes B = \begin{bmatrix}
a_{11}B & a_{12}B & \cdots & a_{1q}B \\
a_{21}B & \ddots & & \\
\vdots & & \ddots & \\
a_{p1}B & \cdots & & a_{pq}B
\end{bmatrix}
\]  

(A.9)

We also define the vector valued function \( \text{vec} \) of a matrix \( A \in \mathbb{R}^{p \times q} \). It gives back a vector \( \text{vec}(A) \in \mathbb{R}^{1 \times pq} \) that consists of the stacking of the columns of \( A \).

Some important properties of the Kronecker product are the following:

\[
(A \otimes B) \otimes C = A \otimes (B \otimes C) 
\]

(A.10)

\[
(A + B) \otimes (C + D) = A \otimes C + A \otimes D + B \otimes C + B \otimes D 
\]

(A.11)

\[
(A \otimes B)(C \otimes D) = AC \otimes BD 
\]

(A.12)

\[
(A \otimes B)^T = A^T \otimes B^T 
\]

(A.13)

\[
(A \otimes B)^{-1} = A^{-1} \otimes B^{-1} 
\]

(A.14)

\[
\text{vec}(ABC) = (C^T \otimes A)\text{vec}(B) 
\]

(A.15)

Especially the last property will be used frequently to our advantage in this thesis.

## A.5 Matrix derivations

In this thesis we use derivatives of matrices with respect to other matrices. Normally this is denoted as a tensor, but here we prefer to use a two dimensional approach. This approach is explained in [15] and makes use of Kronecker products to give generalizations for many derivation formulas, such as the chain rule. Within this framework for matrix calculus, the derivative of a matrix with respect to a second matrix is the stacking of the derivative of the matrix with respect to every element of the second matrix. For instance

\[
\frac{\partial A}{\partial B} = \begin{bmatrix}
\frac{\partial A}{\partial b_{11}} & \frac{\partial A}{\partial b_{12}} & \cdots \\
\frac{\partial A}{\partial b_{21}} & \frac{\partial A}{\partial b_{22}} \\
\vdots & & \ddots
\end{bmatrix}
\]

With this notation, a generalization of the chain rule for differentiation for matrices becomes:

\[
\frac{\partial AC}{\partial B} = \frac{\partial A}{\partial B}(I_p \otimes C) + (I_q \otimes A) \frac{\partial C}{\partial B}
\]

where \( B \in \mathbb{R}^{q \times r} \).
A.6 The State-space equation

The general discrete time linear state space description of a system (without disturbance) is given by the following two equations:

\begin{align*}
x(k + 1) &= Ax(k) + Bu(k) \\
y(k) &= Cx(k) + Du(k)
\end{align*}

where \( x(k) \in \mathbb{R}^n \), \( u(k) \in \mathbb{R}^m \), \( y(k) \in \mathbb{R}^\ell \), \( A \in \mathbb{R}^{n \times n} \), \( B \in \mathbb{R}^{n \times m} \), \( C \in \mathbb{R}^{\ell \times n} \) and \( D \in \mathbb{R}^{\ell \times m} \). At time instant 1, the state is set to be \( x_1 \). From there on it evolves through time, using the previous state, and the input at every time instance.

![Schematic representation of the state space equations.](image)

In figure A.2 the evolution of the state is shown graphically. In every block, the state \( x(k + 1) \) and the output \( y(k) \) is calculated as function of \( x(k) \) and \( u(k) \). The state represents the internal memory of the system. It accumulates the effects of all the past inputs, that are necessary to calculate the output at the present and future time instances.

The state space system A.16 - A.17 is not unique. This means that there are many state space realizations for the same system. The following lemma shows how these realizations relate to each other.

**Lemma A.3** Given a system described by the state space equations A.16- A.17 with state \( x(k) \). Then a linear transformation of the state \( z(k) = T x(k) \), with \( T \in \mathbb{R}^{n \times n} \) invertible, results in the following realization.

\begin{align*}
z(k + 1) &= A_T z(k) + B_T u(k) \\
y(k) &= C_T z(k) + D_T u(k)
\end{align*}
with the new state space matrices given by

\[
A_T = TAT^{-1} \quad B_T = TB \\
C_T = CT^{-1} \quad D_T = D
\]

and the initial state \( z_0 = T x_0 \).

Both realizations are equal in the sense that they have identical input/output behavior. From measurements of the input and output, we can therefore never distinguish between them and both are equally valid state space realizations of our system.

### A.7 Controllability

**Definition A.1** [73] A system, described by the discrete time linear state space equations A.16 - A.17 is called controllable if there exists a finite discrete time interval \([k_0, k_f]\) such that given any initial state \( x_1 \) and any final state \( x_f \), there exists an input signal \( u(k) \) defined on \([k_1, k_f]\) that satisfies \( x(k_1 = x_1 \) and \( x(k_f) = x_f \).

In other words, we need to be able to steer all the states independently with our input. This way we can steer from any initial state \( x_1 \) to any other state within finite time.

**Example A.4** An example of a non-controllable system is

\[
\begin{bmatrix}
x_1(k+1) \\
x_2(k+1)
\end{bmatrix} = \begin{bmatrix}
0.3 & 0.5 \\
0 & 0.4
\end{bmatrix} \begin{bmatrix}
x_1(k) \\
x_2(k)
\end{bmatrix} + \begin{bmatrix}
1 \\
0
\end{bmatrix} u(k)
\]

We see that we can never influence the second element of the new state, \( x_2(k+1) \), with our input. We can't change it directly, because the last element in \( B \) is zero. Also trying to influence it indirectly, through the first element of the state \( x_1(k) \) is impossible because the cross coupling element \( A_{2,1} \) is zero.

When we choose \( B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \) however, the system becomes controllable. We can influence \( x_2(k+1) \) directly and \( x_1(k+1) \) indirectly through \( x_2(k) \).

To test whether a given state space description is controllable is given in the following theorem.

**Theorem A.1** ([73]) A discrete time linear state space realization is controllable if and only if

\[
\text{rank} \left( \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix} \right) = n
\]
A.8 Observability

The concept of observability is dual to the concept of controllability. While in the later case we want full control over the state by the input, here we need that the state is fully observable from the output over a certain time span.

Definition A.2 ([73]) A System, described by the discrete time linear state space equations A.16 - A.17 is called observable if there exists a finite time span \([k_1, k_f]\) such that any initial state \(x_1\) at time \(k_1\) is uniquely determined by the corresponding response \(y(k)\) for \(k \in [k_1, k_f]\).

The duality between observability and controllability continues with the tests for them. As in the previous section, given a system, we can test if a system is observable.

Theorem A.2 [73] A linear discrete time state space equation is observable if and only if

\[
\text{rank} \left( \begin{bmatrix}
C \\
CA \\
\vdots \\
CA^{n-1}
\end{bmatrix} \right) = n
\]

A.9 Minimality

To finish the list of properties of state space system descriptions we define the concept of minimality. As we have seen in section A.6 there are many realizations of the same system. We would like to know if the state space realization we have at hand is the smallest one, i.e. has the smallest dimension of the state vector. This is called minimality.

Definition A.3 ([73]) A system, described by the discrete time linear state space equations A.16 - A.17, with state dimension \(n\) is called a minimal realization of that system if there exists no other realization of the state space system, that has the same input-output behavior, with state dimension smaller than \(n\).

Theorem A.3 ([73, 51]) A system, described by the discrete time linear state space equations A.16- A.17 is minimal if and only if it is controllable and observable.

A minimal state space realization is both controllable and observable. If there is a state that is not controllable, this state can be disregarded, because it is independent
from the input. On the other hand, if there exists a state that is unobservable it can as well be removed, because it has no influence on the output.

## A.10 Relation to impulse response functions

The impulse response function (IRF) of a system is given by the Markov parameters \( L(i) \in \mathbb{R}^{t \times m}, i = 0, 1, \cdots, \infty \), such that

\[
y(k) = \sum_{i=0}^{\infty} L(i)u(k - i)
\]

(A.18)

The parameters of the IRF relate to the matrices in the state space description as follows.

\[
L(i) = \begin{cases} 
D & i = 0 \\
CA^{i-1}B & i \neq 0 
\end{cases}
\]

(A.19)

**Remark A.1** In chapter 2 we will give a description of the Ho-Kalman algorithm, to construct a state space description of a system from the IRF.

## A.11 Relation to transfer functions

The transfer function description of a system is given by \( H(z) \in \mathbb{H}^{t \times m} \). Each element in the matrix \( H(z) \) is a polynomial of the form

\[
H_{ij} = \frac{b_{0}^{ij}z^{m} + b_{1}^{ij}z^{m-1} + \cdots + b_{m}^{ij}}{a_{0}z^{n} + a_{1}z^{n-1} + \cdots + a_{n}} \quad (m \leq n)
\]

(A.20)

The relation between the transfer function matrix description \( H(z) \) and the state space description of a system is

\[
H(z) = CZ(zI - A)^{-1}B + D
\]

(A.21)

## A.12 Matrix inversion lemma

The following lemma is used a number of times in this thesis. We include it in this chapter for completeness.
Lemma A.4 Let $A$, $B$, $C$ and $D$ be matrices of compatible dimensions, so that the product $BCD$ and the sum $A + BCD$ exist. Then, if the inverse of $A$ exists the following holds:

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B[D A^{-1} B + C^{-1}]^{-1}DA^{-1} \quad \text{(A.22)}$$
Appendix B

User guide to SMI 2.0

This appendix gives a function description of every function that is part of the SMI-toolbox.

B.1 List of functions in SMI 2.0

The following table gives an overview of the functions in the SMI toolbox. MATLAB functions for the identification of dynamical systems. The routines were developed at the Systems and Control laboratory of Delft University of Technology. Together they form a powerful toolkit for the estimation of a state space model from measured data. The first version of SMI (version 1.0) was introduced in 1997.

During the course of writing this thesis more functions were added and others were improved. This resulted in version 2.0 of the SMI-toolbox. Some functions appeared in the previous version of the toolbox under another name. These names have been changed to prevent conflicts with other toolboxes and to provide a more consistent naming. This makes the functions easier to recognize and remember. An example of this is the function dmodpo that is changed to destac (discrete time estimate of $A$ and $C$) and dac2bd that is renamed to destbd (discrete time estimate of $B$ and $D$). Functions that were added are the continuous time MOESP functions and the recursive MOESP functions. The following table shows an overview of all the functions that are contained in version 2.0. The last column in the table shows the old name from version 1.0 when applicable.
<table>
<thead>
<tr>
<th>function</th>
<th>description</th>
<th>old name</th>
</tr>
</thead>
<tbody>
<tr>
<td>dordom</td>
<td>Ordinary moesp preprocessor</td>
<td>dordom</td>
</tr>
<tr>
<td>dordpi</td>
<td>Past input moesp preprocessor</td>
<td>dordpi</td>
</tr>
<tr>
<td>dordpo</td>
<td>Past output moesp preprocessor</td>
<td>dordpo</td>
</tr>
<tr>
<td>dordeiv</td>
<td>Eiv moesp preprocessor</td>
<td>-</td>
</tr>
<tr>
<td>dordrs</td>
<td>Reconstructed state moesp preprocessor</td>
<td>dordrs</td>
</tr>
<tr>
<td>destac</td>
<td>Estimate A,C</td>
<td>dmodpi,dmodpo</td>
</tr>
<tr>
<td>destbd</td>
<td>Estimate B,D</td>
<td>dac2bd,destb</td>
</tr>
<tr>
<td>destk</td>
<td>Estimate Kalman gain</td>
<td>dmodpo</td>
</tr>
<tr>
<td>destx</td>
<td>Estimate initial state</td>
<td>dinit</td>
</tr>
<tr>
<td>dmoesp</td>
<td>Frontend for DT moesp</td>
<td>-</td>
</tr>
</tbody>
</table>
### Continuous time moesp

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cordom</td>
<td>Ordinary moesp preprocessor</td>
</tr>
<tr>
<td>cordpi</td>
<td>Past input moesp preprocessor</td>
</tr>
<tr>
<td>cordpo</td>
<td>Past output moesp preprocessor</td>
</tr>
<tr>
<td>cestac</td>
<td>Estimate A,C</td>
</tr>
<tr>
<td>cestbd</td>
<td>Estimate B,D</td>
</tr>
<tr>
<td>cestx</td>
<td>Estimate initial state</td>
</tr>
</tbody>
</table>

### Recursive moesp

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>drpi</td>
<td>Recursive PI moesp</td>
</tr>
<tr>
<td>drpo</td>
<td>Recursive PO moesp</td>
</tr>
</tbody>
</table>

### SLS optimization

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dss2th</td>
<td>Parameterization of state space system</td>
</tr>
<tr>
<td>dth2ss</td>
<td>Reconstruction of state space system</td>
</tr>
<tr>
<td>dsllsln</td>
<td>Optimize DT linear model using SLS</td>
</tr>
<tr>
<td>dsllswie</td>
<td>Optimize DT wiener model using SLS</td>
</tr>
<tr>
<td>dfunlin</td>
<td>Cost-function for dsllsln</td>
</tr>
<tr>
<td>dfunwie</td>
<td>Cost-function for dsllswie</td>
</tr>
<tr>
<td>drlsllin</td>
<td>Recursive optimization of DT model using SLS</td>
</tr>
<tr>
<td>cllsln</td>
<td>Optimize CT linear model using SLS</td>
</tr>
<tr>
<td>cfunlin</td>
<td>Cost-function for dsllsln</td>
</tr>
<tr>
<td>crllsllin</td>
<td>Recursive optimization of CT model using SLS</td>
</tr>
</tbody>
</table>

### Non-causal models

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ncddsim</td>
<td>Simulate non-causal model</td>
</tr>
<tr>
<td>ncestac</td>
<td>Estimate A and C for non-causal model</td>
</tr>
<tr>
<td>ncestbd</td>
<td>Estimate B and D for non-causal model</td>
</tr>
<tr>
<td>kroneckf</td>
<td>Calculate Kronecker canonical form</td>
</tr>
</tbody>
</table>

### Non-Linear models

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chebest</td>
<td>Estimate MIMO non-linear model</td>
</tr>
<tr>
<td>chebsim</td>
<td>Simulate MIMO non-linear model</td>
</tr>
</tbody>
</table>

### Miscellaneous

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prbn</td>
<td>Pseudo random binary sequence</td>
</tr>
<tr>
<td>vaf</td>
<td>Variance accounted for</td>
</tr>
<tr>
<td>shave</td>
<td>Remove peaks and outliers</td>
</tr>
</tbody>
</table>
Purpose
Estimates the matrices $A$ and $C$ of a LTI state space model using the result of the preprocessor routines cordxx (cordom, cordpo, etc.). General model structure:

\[
\begin{align*}
    \dot{x}(t) &= Ax(t) + Bu(t) + w(t) \\
    y(t) &= Cx(t) + Du(t) + v(t)
\end{align*}
\]

For more information about the disturbance properties see the help pages for the preprocessor cordxx functions.

Syntax
\[
[A,C]=cestac(R,n);
\]

Inputs
- $R$ Data structure obtained from cordxx, containing the triangular factor and additional information (such as i/o dimension etc.).
- $n$ Order of system to be estimated.

Outputs
- $A,C$ Estimated system matrices.

See Also
cordom, cordpi, cordpo, cestbd
Purpose
Estimates the matrices $B$ and $D$ of the state space model

\[
\begin{align*}
\dot{x}(t) &= Ax(t) + Bu(t) + w(t) \\
y(t) &= Cx(t) + Du(t) + v(t) \\
x(0) &= x_0
\end{align*}
\]

using the knowledge of the pair $A$, $C$. This function can concatenate different input-output data batches, through the matrix $R$ and $\text{Rold}$). $B$ and $D$ and $x_0$ are calculated by solving a linear least squares problem.

Syntax
\[
\begin{align*}
[B,D,x0,R]=\text{cestbd}(u,y,A,C,[fB fD fx],\text{Rold}); \\
[B,D]=\text{cestbd}(u,y,A,C);
\end{align*}
\]

Inputs
\[
\begin{align*}
u, y & \quad \text{The input and output data of the system to be identified.} \\
A, C & \quad \text{The estimated system matrices $A$ and $C$.} \\
model & \quad \text{Three element flag vector $[fB \ fD \ fx]$ indicating whether $B$, $D$ and $x_0$ should be estimated. The default value is $[1 \ 1 \ 0]$. The matrix $B$ or $D$ can be assumed zero by setting $fB$ or $fD$ to zero. The calculation of $x_0$ can be omitted by setting $fx$ to zero. However, $x_0$ will not be assumed zero then. It's influence will still be taken into account for the computation of $B$ and $D$.} \\
\text{Rold} & \quad \text{$R$ matrix obtained from previous data batch. This variable can be used to process data in batches, or to combine data from different experiments.}
\end{align*}
\]

Outputs
\[
\begin{align*}
B, D & \quad \text{The estimated system matrices $B$ and $D$.} \\
x0 & \quad \text{The estimated initial state of the system.} \\
\text{R} & \quad \text{Compressed data matrix, storing information on the calculation of the matrices $B$ and $D$ in following } \text{cestbd}. \text{ Used when analyzing multiple input-output data sequences.}
\end{align*}
\]

See Also
\[
\begin{align*}
\text{cordxx, cestac, cestx}
\end{align*}
\]
Purpose
Estimate the initial state, given the estimated system matrices and a set of input/output data.

Syntax
\[ x_0 = \text{cestx}(u, y, Ts, A, B, C, D); \]

Inputs
- \( u, y \)  The input and output data of the system to be identified.
- \( Ts \)    Sampling period of the measured data.
- \( A, B, C, D \) System matrices.

Outputs
- \( x_0 \) Estimated initial state.

See Also
- cestbd
Purpose
This function estimates a MIMO static non-linear function between the signals \( y \) and \( z \). The function is estimated on the basis of Chebychev polynomials. Before estimating the coefficients of the polynomials the input signal is shifted and scaled to fall within the region \([-1,1]\). The shifting and scaling factors are included in the parameter vector.

Syntax
\[
[\text{thl,ze,Phi}]=\text{chebest}(y,z,\text{nn});
\]

Inputs
- \( y,z \) Input and output of the non-linearity.
- \( \text{nn} \) Order of the Chebychev polynomials in the non-linear function.

Outputs
- \( \text{thl} \) Vector with the parameters of the static non-linearity.
- \( \text{ze} \) Estimated output, on basis of the model that is obtained.
- \( \Phi \) matrix with the Chebychev functions of \( y \), such that \( \text{ze} = \Phi \times \text{thl} \).

See Also
- chebsim, dslswie
Purpose
Simulates a static non-linear function on the basis of Chebychev polynomials with input \( y \). The coefficients of the Chebychev polynomials are given with the vector \( thl \) and commonly estimated with either \texttt{dlswie} or \texttt{chebest}.

Syntax
\[
[ze,Phi] = \text{chebsim}(y,thl)
\]

Inputs
- \( y \) The input to the non-linearity.
- \( thl \) Parameter matrix, with coefficients of the non-linear function.

Outputs
- \( ze \) Estimated output.
- \( Phi \) matrix with the Chebychev functions of \( y \), such that \( ze = Phi \times thl \).

See Also
- \texttt{chebest}, \texttt{dlswie}
**Purpose**

This function is a preprocessor function that extracts the column-space of the extended observability matrix of a continuous time LTI system from input/output data. The estimated column-space is used in the function cestac to extract the matrices A and C. Model structure:

\[
\dot{x}(t) = Ax(t) + Bu(t) \\
y(t) = Cx(t) + Du(t)
\]

This function does not use an instrumental variable and is comparable to dordom for the discrete time case.

**Syntax**

\[
[S_n,R]=\text{cordpo}(u,y,t,a,i);
\]

\[
[S_n,R]=\text{cordpo}(u,y,t,a,i,Rold);
\]

**Inputs**

- **u, y**
  - The input and output data of the system to be identified.
- **t**
  - Sampling time or time-vector for the sampled data.
- **a**
  - Filter coefficient of the Laguerre filters that are used in the algorithm. When a is negative, the anti-causal instrumental variable method is used. Otherwise the causal instrumental variable method is chosen.
- **i**
  - The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.
- **Rold**
  - Data structure obtained from processing a previous data-batch with cordom containing the same items as R.

**Outputs**

- **S_n**
  - Singular values bearing information on the order of the system.
- **R**
  - Data structure used by cestac for the estimation of A and C or by a next call to cordom. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).

**See Also**

- cordpi, cordpo, cestac, cestbd.
**Purpose**

This function is a preprocessor function that extracts the column-space of the extended observability matrix of a continuous time LTI system from input/output data. The estimated column-space is used in the function cestac to extract the matrices A and C. Model structure:

\[
\dot{x}(t) = Ax(t) + Bu(t) \\
y(t) = Cx(t) + Du(t) + v(t)
\]

where \( v(t) \) is a finite variance disturbance on the output and is independent of the noise-free input \( u(t) \). This function uses the past input instrumental variable and is comparable to dordpi for the discrete time case.

**Syntax**

\[ [S_n,R] = \text{cordpi}(u,y,t,a,i); \]

\[ [S_n,R] = \text{cordpi}(u,y,t,a,i,Rold); \]

**Inputs**

- **u, y**
  - The input and output data of the system to be identified.
- **t**
  - Sampling time or time-vector for the sampled data.
- **a**
  - Filter coefficient of the Laguerre filters that are used in the algorithm. When \( a \) is negative, the anti-causal instrumental variable method is used. Otherwise the causal instrumental variable method is chosen.
- **i**
  - The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.
- **Rold**
  - Data structure obtained from processing a previous data-batch with cordpi containing the same items as \( R \).

**Outputs**

- **Sn**
  - Singular values bearing information on the order of the system.
- **R**
  - Data structure used by cestac for the estimation of \( A \) and \( C \) or by a next call to cordpi. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).

**See Also**

cordom, cordpo, cestac, cestbd.
Purpose
This function is a preprocessor function that extracts the column-space of the extended observability matrix of a continuous time LTI system from input/output data. The estimated column-space is used in the function cestac to extract the matrices A and C. Model structure:

\[
\begin{align*}
\dot{x}(t) &= Ax(t) + Bu(t) + w(t) \\
y(t) &= Cx(t) + Du(t) + v(t)
\end{align*}
\]

where \( w(t) \) is a Wiener noise signal and \( v(t) \) a finite variance disturbance on the output, independent of the noise-free input \( u(t) \).

This function uses the past output instrumental variable and is comparable to dordpo for the discrete time case.

Syntax

\[
[S_n,R]=\text{cordpo}(u,y,t,a,i);
\]

\[
[S_n,R]=\text{cordpo}(u,y,t,a,i,Rold);
\]

Inputs

- \( u, y \) The input and output data of the system to be identified.
- \( t \) Sampling time or time-vector for the sampled data.
- \( a \) Filter coefficient of the Laguerre filters that are used in the algorithm. When \( a \) is negative, the anti-causal instrumental variable method is used. Otherwise the causal instrumental variable method is chosen.
- \( i \) The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.
- \( Rold \) Data structure obtained from processing a previous data-batch with \text{cordpo} containing the same items as \( R \).

Outputs

- \( S_n \) Singular values bearing information on the order of the system.
- \( R \) Data structure used by \text{cestac} for the estimation of \( A \) and \( C \) or by a next call to \text{cordpo}. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).

See Also

\text{cordom}, \text{cordpi}, \text{cestac}, \text{cestbd}. 
Purpose

This function converts a continuous time state space model to a parameter vector that describes the model.

Model structure:

\[
\dot{x}(t) = Ax(t) + Bu(t) + Ke(t) \\
y(t) = Cx(t) + Du(t) + e(t) \\
x(0) = x_0
\]

Syntax

\[
[\theta,\text{params}] = \text{css2th}(A,B,C,D,x0,K,\text{partype}); \\
[\theta,\text{params}] = \text{css2th}(A,B,C,D); \\
[\theta,\text{params}] = \text{css2th}(A,C);
\]

Inputs

- \(A,B,C,D\): System matrices describing the state space system. The \(B\) and \(D\) matrices are optional. An optional element can be left out or given as an empty matrix to indicate it is not part of the model.
- \(x0\): Initial condition, This is optional.
- \(K\): Kalman gain. Also this matrix is optional.
- \(\text{partype}\): This parameter specifies the type of parameterization that is used to parameterize the state space model. Two types of parameterization are supported: 'on' = Output Normal and 'tr'=TRidiagonal.

Outputs

- \(\theta\): Parameters vector describing the system.
- \(\text{params}\): A structure that contains the dimension parameters of the system, such as the order, the number of inputs, whether \(B, D, x_0\) or \(K\) are present, etc.
- \(T\): Transformation matrix between the input state space system and the state space system in the form described by \(\theta\) (output normal or tridiagonal).

See Also

- \(\text{cth2ss, cs1slin}\)
**Purpose**

This function converts a parameter vector that describes a continuous time state space model in output normal form to the state space matrices of that model.

Model structure:

\[
\dot{x}(t) = Ax(t) + Bu(t) + Kc(t) \\
y(t) = Cx(t) + Du(t) + e(t) \\
x(0) = x_0
\]

**Syntax**

\[ [A,B,C,D,x0,K] = cth2ss(theta, params) \]
\[ [A,C] = cth2ss(theta, params) \]

**Inputs**

- theta: Parameter vector describing the system.
- params: A structure that contains the dimension parameters of the system, such as the order, the number of inputs, whether D, x0 or K are present, etc.

**Outputs**

- A, B, C, D: System matrices describing the state space system in output normal form. If theta does not contain parameters for D, this matrix will be returned as an empty matrix.
- x0: Initial condition. If theta does not contain parameters for x0 this vector will be returned as an empty matrix.
- K: Kalman gain. Same here.

**See Also**

cth2ss, cslslin
Purpose
Estimates the A and C matrices of a LTI state space model form using the result of the preprocessor routines dordxx. (dordom, dordpo, etc.)

General model structure:

\[
\begin{align*}
x(k+1) &= Ax(k) + Bu(k) \\
y(k) &= Cx(k) + Du(k)
\end{align*}
\]

For information about the possible disturbance signals see the help pages for the preprocessor dordxx functions.

Syntax
\[ [A,C]=destac(R,n); \]

Inputs
\[
\begin{align*}
R & \quad \text{Data structure obtained from a dordxx function, containing the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).} \\
n & \quad \text{Order of system to be estimated.}
\end{align*}
\]

Outputs
\[
\begin{align*}
A,C & \quad \text{Estimated system matrices.}
\end{align*}
\]

See Also
dordom, dordpi, dordpo, dordeiv, dordrs, destbd
Purpose

Estimates the matrices $B$ and $D$ of the state space model

$$\begin{align*}
x(k+1) &= Ax(k) + Bu(k) + w(k) \\
y(k) &= Cx(k) + Du(k) + v(k)
\end{align*}$$

using the knowledge of the pair $A, C$. This function can concatenate different input-output data batches, through the matrix $R$ and $R_{old}$. $B$ and $D$ are calculated by solving a linear least squares problem. The function can handle errors-in-variables models by using an instrumental variable.

Syntax

$$[B,D]=destbd(u,y,A,C);$$
$$[B,D,x0,R,Phi]=destbd(u,y,A,C,[fB fD fx],Rold,iv,niv);$$

Inputs

| u, y | The input and output data of the system to be identified. |
| A, C | The state space matrices $A$ and $C$. |
| model | Three element flag vector $[fB \ fD \ fx]$ indicating whether $B$, $D$ and $x_0$ should be estimated. The default value is $[1 \ 1 \ 0]$. The matrix $B$ or $D$ can be assumed zero by setting $fB$ or $fD$ to zero. The calculation of $x_0$ can be omitted by setting $fx$ to zero. However, $x_0$ will not be assumed zero. It's influence will still be taken into account for the computation of $B$ and $D$. Variables that are not asked for will be returned as empty matrices. |
| Rold | $R$ matrix obtained from previous data batch. This variable can be used to process data in batches, or to combine data from different experiments. |
| iv | Instrumental variable. For instance, for the errors-in-variables case the iv can be the past output or for closed loop data, the reference input. |
| niv | Three element vector describing the number of lagged IVs used. niv(1) is the number of lagged IVs, niv(2) the number of lagged past outputs and niv(3) the number of lagged past inputs. If only one element is present, no past data is used. If not given or empty appropriate default values are used. When past data is used as instrumental variable, about half of the input/output data is used for the instrument. |

Outputs

| B, D | The estimated system matrices $B$ and $D$. |
| x0 | The estimated initial state of the system. |
\textbf{R} \hspace{1cm} \text{Compressed data matrix, storing information on the calculation of the matrices} \ B \ \text{and} \ D \ \text{in following} \ \text{destbd. Used when analyzing multiple input-output data sequences.}

\textbf{Phi} \hspace{1cm} \text{Regressors matrix that was used for the Least squares estimate.}

\textbf{See Also} \hspace{1cm} \text{destac destx}
**Purpose**
Estimates the Kalman gain of an innovations model, when the matrices $A$ and $C$ are known. They can be estimated with the functions `dordpo` and `destac`. Model structure:

$$\begin{align*}
x(k+1) &= Ax(k) + Bu(k) + w(k) \\
y(k) &= Cx(k) + Du(k) + v(k)
\end{align*}$$

where $w(k)$, $v(k)$ are zero-mean white noise sequences, independent of the noise-free input $u(k)$.

**Syntax**

```
K=destk(A,B,C,D,R)
K=destk(A,C,R)
```

**Inputs**

- **A,B,C,D**  
  State space matrices (B and D are optional).
- **R**  
  Data structure that is obtained from `dordpo`. This matrix contains compressed data needed to estimate the Kalman gain. This includes the lower triangular factor and additional information (such as i/o dimension etc.).

**Outputs**

- **K**  
  Estimated Kalman gain.

**See Also**

See also: `dordpo`, `destac`, `destbd`
Purpose
Estimates the initial state of a state space system on the basis of the system matrices and a set of input/output data.

Syntax
\[ x_0 = \text{destx}(u, y, A, B, C, D); \]

Inputs
\[ u, y \quad \text{The input and output data of the system to be identified.} \]
\[ A, B, C, D \quad \text{System matrices.} \]

Outputs
\[ x_0 \quad \text{Estimated initial state.} \]

See Also
\[ \text{destbd} \]
Purpose
This function implements the cost-function for dslslin. It is not meant for stand-alone use.

Syntax
`epsilon=dfunlin(thn,u,y,params)`

Inputs
- `thn` Parameter vector describing the system matrices $A$ and $C$.
- `u,y` The input and output data of the system to be optimized.
- `params` A structure that contains the dimension parameters of the system, such as the order, the number of inputs, whether $D$, $r_0$ or $K$ is present in the model, etc.

Outputs
- `epsilon` Output of the cost-function, which is the square of the error between the output and the estimated output, divided by the number of samples.

See Also
dslslin, dslswie, dfunwie
Purpose
This function implements the cost-function for dslswie. It is not meant for stand-alone use.

Syntax
\[
\text{epsilon} = \text{dfunwie}(\text{thn}, u, y, \text{params})
\]

Inputs
- \text{thn} \quad \text{Parameter vector describing the system matrices} \ A, \ B, \ C \text{ and } D.
- \text{u, z} \quad \text{The input and output data of the system to be optimized.}
- \text{params} \quad \text{A structure that contains the dimension parameters of the system, such as the order, the number of inputs, whether } D \text{ or } x_0 \text{ is present in the model, etc.}

Outputs
- \text{epsilon} \quad \text{Output of the cost-function, which is the square of the error between the output and the estimated output, divided by the number of samples.}

See Also
dslswie, dslslin, dfunlin
**Purpose**

High level function for the discrete time moesp that gives the user a simple interface to the lower level MOESP functions such as dordxx and destac. This function calculates the system matrices $A$, $B$, $C$ and $D$ from the given input and output data sequences $u$ and $y$.

Model structure:

$$
\begin{align*}
    x(k+1) &= Ax(k) + Bu(k) \\
    y(k)   &= Cx(k) + Du(k)
\end{align*}
$$

For information about the possible disturbance signals see the help pages for the preprocessor dordxx functions.

The user only needs to specify the input and output data. If the order is not given, an order selection dialog is shown where the user can select the order by examining the singular value plot.

**Syntax**

$$
\begin{align*}
[A,B,C,D]=\text{dmoesp}(u,y,\text{maxorder}) \\
[A,B,C,D,K]=\text{dmoesp}(u,y,\text{maxorder,order,method})
\end{align*}
$$

**Inputs**

- $u,y$ : Input and Output data sequence.
- $\text{maxorder}$ : Maximum expected order of the system, this value is used as the Hankel dimension parameter in the underlying identification routines.
- $\text{order}$ : Order of the system to be estimated, if not specified a dialog will be presented which allows the user to choose the order.
- $\text{method}$ : Method used for identification. possible options:
  - "om" : ordinary moesp
  - "pi" : past input moesp
  - "po" : past output moesp

The default value is past output moesp.

**Outputs**

- $A,B,C,D$ : State space matrices describing the estimated model.
- $K$ : Estimated Kalman gain, only available for past output moesp.

**See Also**

destac, dordpo, etc
**Purpose**

dordeiv is a preprocessor function that extracts the column-space of the extended observability matrix from input/output data. Data from different experiments can be concatenated using the extra input argument $Z$. The estimated column-space is used in the function destac to extract the matrices $A$ and $C$. This function implements the errors-in-variables MOESP algorithm which can be used for the errors-in-variables identification problem.

Model structure:

\[
\begin{align*}
x(k+1) &= Ax(k) + B\ddot{u}(k) + w(k) \\
y(k) &= Cx(k) + D\ddot{u}(k) + v(k)
\end{align*}
\]

with measurements

\[u(k) = \ddot{u}(k) + f(k) \quad \text{and} \quad y(k)\]

where $f(k), w(k)$ and $v(k)$ are zero-mean white noise sequences independent of the input $\ddot{u}(j)$ for $k \geq j$.

The system can be operated under either open-loop or closed-loop. For closed-loop operation, $r(k)$ is an external reference input. For open-loop operation, where $\ddot{u}(k)$ is white noise, dordpo will give better results.

**Syntax**

\[
\begin{align*}
[Sn,R,Z] &= \text{dordeiv}(u,y,r,i,Zold) \\
[Sn,R,Z] &= \text{dordeiv}(u,y,[],i,Zold)
\end{align*}
\]

**Inputs**

- $u,y$ The input and output data of the system to be identified.
- $r$ Closed-loop reference input.
- $i$ The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.
- $Zold$ Matrix obtained from processing a previous data-batch with dordrs.

**Outputs**

- $Sn$ Singular values bearing information on the order of the system
Data structure used by \texttt{destac} for the estimation of $A$ and $C$. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).

Matrix containing information on system which can be used by future calls of \texttt{dordeiv}.

\textbf{See Also}

\texttt{dordom, dordpi, dordpo, dordrs, destac, destbd}.  

**Purpose**

dordom is a preprocessor function that extracts the column-space of the extended observability matrix from input/output data. Data from different experiments can be concatenated using the extra input argument \( R \). The estimated column-space is used in the function destac to extract the matrices \( A \) and \( C \).

This function implements the ordinary MOESP algorithm which can only be used for the noise-free identification problem.

Model structure:

\[
\begin{align*}
x(k+1) &= Ax(k) + Bu(k) \\
y(k) &= Cx(k) + Du(k)
\end{align*}
\]

**Syntax**

\[
\begin{align*}
[Sn,R] &= \text{dordom}(u,y,i); \\
[Sn,R] &= \text{dordom}(u,y,i,Rold);
\end{align*}
\]

**Inputs**

- \( u,y \) The input and output data of the system to be identified.
- \( i \) The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.
- \( Rold \) Data structure obtained from processing a previous data-batch with dordom containing the same items as \( R \).

**Outputs**

- \( Sn \) Singular values bearing information on the order of the system.
- \( R \) Data structure used by destac for the estimation of \( A \) and \( C \) or by a next call to dordom. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).

**See Also**

dordpi, dordpo, dordrs, dordeiv, destac, destbd.
Purpose

dordpi is a preprocessor function that extracts the column-space of the extended observability matrix from input/output data. Data from different experiments can be concatenated using the extra input argument \( R \). The estimated column-space is used in the function destac to extract the matrices \( A \) and \( C \). This function implements the past input MOESP algorithm which can be used for the output-error identification problem. Model structure:

\[
\begin{align*}
    x(k+1) &= Ax(k) + Bu(k) \\
    y(k) &= Cx(k) + Du(k) + v(k)
\end{align*}
\]

where \( v(k) \) is zero-mean noise of arbitrary color, independent of the noise-free input \( u(k) \).

Syntax

\[
[\text{Sn},R]=\text{dordpi}(u,y,i);
\]

\[
[\text{Sn},R]=\text{dordpi}(u,y,i,Rold);
\]

Inputs

\( u, y \) The input and output data of the system to be identified.

\( i \) The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.

\( Rold \) Data structure obtained from processing a previous data-batch with dordpi containing the same items as \( R \).

Outputs

\( \text{Sn} \) Singular values bearing information on the order of the system.

\( R \) Data structure used by destac for the estimation of \( A \) and \( C \) or by a next call to dordpi. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).

See Also

dordom, dordpo, dordrs, dordev, destac, destbd.
**Purpose**

dordpo is a preprocessor function that extracts the column-space of the extended observability matrix from input/output data. Data from different experiments can be concatenated using the extra input argument $R$. The estimated column-space is used in the function destac to extract the matrices $A$ and $C$. This function implements the past output moesp algorithm which can be used for the innovations model identification problem.

Model structure:

$$
x(k + 1) = Ax(k) + Bu(k) + w(k)$$
$$y(k) = Cx(k) + Du(k) + v(k)$$

where $w(k)$, $v(k)$ are zero-mean white noise sequences, independent of the noise-free input $u(k)$.

**Syntax**

```matlab
[Sn,R]=dordpo(u,y,i);
[Sn,R]=dordpo(u,y,i,ROLS);
```

**Inputs**

$u$, $y$  
The input and output data of the system to be identified.

$i$  
The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.

$ROLS$  
Data structure obtained from processing a previous data-batch with dordpo containing the same items as $R$.

**Outputs**

$Sn$  
Singular values bearing information on the order of the system.

$R$  
Data structure used by destac for the estimation of $A$ and $C$ or by a next call to dordpo. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).

**See Also**

dordom, dordpi, dordrs, dordeiv, destac, destbd.
**Purpose**

`dordrs` is a preprocessor function that extracts the column-space of the extended observability matrix from input/output data. Data from different experiments can be concatenated using the extra input argument `R`. The estimated column-space is used in the function `destac` to extract the matrices `A` and `C`. This function implements the reconstructed state `MOESP` algorithm which can be used for the output-error identification problem.

Model structure:

\[
\begin{align*}
    x(k+1) &= Ax(k) + Bu(k) \\
    y(k)    &= C r(k) + Du(k) + v(k)
\end{align*}
\]

where \(v(k)\) is zero-mean noise of arbitrary color, independent of the noise-free input \(u(k)\).

**Syntax**

\[
[S_n, R] = dordrs(u, y, x, i); \\
[S_n, R] = dordrs(u, y, x, i, Rold);
\]

**Inputs**

- `u`, `y`  The input and output data of the system to be identified.
- `x`  Reconstructed state.
- `i`  The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.
- `Rold`  Data structure obtained from processing a previous data-batch with `dordrs` containing the same items as `R`.

**Outputs**

- `S_n`  Singular values bearing information on the order of the system.
- `R`  Data structure used by `destac` for the estimation of `A` and `C` or by a next call to `dordrs`. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).

**See Also**

`dordom`, `dordpi`, `dordpo`, `dordeiv`, `destac`, `destbd`
**Purpose**

Estimates in a recursive way the matrices of an LTI state space model in innovation form using the output of the dordpi, destac and destbd routines as initial guess.

Model structure:

\[
\begin{align*}
    x(k+1) &=Ax(k) + Bu(k) \\
    y(k) &=Cx(k) + Du(k) + v(k)
\end{align*}
\]

where \(v(k)\) is zero-mean white noise sequence, independent of the noise-free input \(u(k)\).

**Syntax**

\([A,B,C,D]= \text{drpi}(u,y,i,n,\text{beta},A_i,B_i,C_i,D_i,R_i)\)

**Inputs**

- \(u,y\): The input and output data of the system to be identified.
- \(i\): The dimension parameter that determines the number of block rows in the processed Hankel matrices such as was used in dordpi for the creation of \(R_i\).
- \(n\): Order of system to be estimated.
- \(\text{beta}\): Forgetting parameter for recursive algorithm (\(0 < \beta < 1\)).
- \(A_i,B_i,C_i,D_i\): Initial estimate of state space matrices. These matrices can for instance be obtained by using dordpi/destac/destbd on the first few samples.
- \(R_i\): Triangular factor from dordpi.

**Outputs**

- \(A,B,C,D\): The estimated system matrices.

**See Also**

- drpo, dordpi, destac, destbd
Purpose
Estimates in a recursive way the matrices of an LTI state space model in
innovation form using the output of the dordpo and dmodpo routines as
initial guess. Model structure:

\[
\begin{align*}
x(k + 1) &= A_r(k) + Bu(k) + w(k) \\
y(k) &= Cx(k) + Du(k) + v(k)
\end{align*}
\]

where \(w(k), v(k)\) is zero-mean white noise sequences, independent of the
noise-free input \(u(k)\).

Syntax
\[[A,B,C,D] = \text{drpo}(u,y,i,n,beta,Ai,Bi,Ci,Di,Ri)\]

Inputs
- \(u, y\) The input and output data of the system to be identified.
- \(i\) The dimension parameter that determines the number of
  block rows in the processed Hankel matrices such as was
  used in \text{dordpo} for the creation of \(R_i\).
- \(n\) Order of system to be estimated.
- \(beta\) Forgetting parameter for recursive algorithm \((0 < \beta < 1)\).
- \(Ai, Bi, Ci, Di\) Initial estimate of state space matrices. These matrices
  can for instance be obtained by using \text{dordpo/destac/destbd}
  on the first few samples.
- \(Ri\) Triangular factor from \text{dordpo}.

Outputs
- \(A, B, C, D\) The estimated system matrices

See Also
- \text{drpi, dordpo, destac, destbd}
Purpose
This function performs a recursive update of a discrete time state space system,

\[ x(k+1) = Ax(k) + Bu(k) \]
\[ y(k) = Cx(k) + Du(k) + v(k) \]

using the Separable Least Squares technique. For this, only the initial estimates of \( A \) and \( C \) are needed. The same function can also be used to initialize the matrices needed to start up the recursion.

Syntax

\[ [A,B,C,D,hist] = drslslin(u,y,A,C) \]
\[ [A,B,C,D,hist] = drslslin(u,y,A,C,K,model,partype,options,slsstat) \]
\[ [slsstate] = drslslin(u,y,A,C,K,model,partype,'init') \]

Inputs

\( A,C \)  
Initial estimate of the state space matrices \( A \) and \( C \)

\( u,y \)  
The input and output data of the system to be optimized.

\( \text{model} \)  
Vector \([B,fD,fx,fK]\) specifying whether the matrix \( B, D, \) the initial state \( x_0 \) and the Kalman filter gain \( K \) should be estimated. Default is \([1 1 0 0]\). It is recommended only to estimate the initial state if the data length is short compared to the largest time constant of the system.

\( \text{options} \)  
This vector consists of parameters that can be used to influence the optimization process. \text{options(1)} Display parameter (Default:0). 1 displays some results. \text{options(2)} is the step size parameter. \text{options(3)} is a forgetting factor of the non-linear param. \text{options(4)} is a forgetting factor of the linear param.

Outputs

\( A,B,C,D \)  
Estimated state space matrices.

\( x0 \)  
Initial state. If it is not estimated, this vector will be returned as an empty matrix.

\( K \)  
Kalman gain. Same here.

\( \text{hist} \)  
History of the recursion the first column is the value of the cost-function at every step. the other columns are the estimated parameters at every step.

See Also

drslslin
Purpose
Performs a Least Squares optimization of a discrete time linear state space system with model structure:

\[
\begin{align*}
x(k+1) &= Ax(k) + Bu(k) + Ke(k) \\
y(k) &= Cx(k) + Du(k) + e(k)
\end{align*}
\]

First, the state space matrices are parameterized. In order to minimize the number of parameters, the Separable Least Squares technique is used. This allows the cancellation of the parameters of B and D, such that only the parameters that describe A and C need to be optimized. The parameterized model is optimized with the leastsq function from the MATLAB optimization toolbox. If needed also the initial state and a Kalman gain can be optimized.

Syntax

\[
\begin{align*}
[A,B,C,D]=\text{dslslin}(u,y,A,C) \\
[A,B,C,D,x0,K,options] = \text{dslslin}(u,y,A,C,K,\text{model,partytype,options})
\end{align*}
\]

Inputs

\begin{itemize}
\item **u, y** The input and output data of the system to be optimized.
\item **A, C** Initial estimates of the system matrices A and C.
\item **model** Vector with flags that specify the kind of model to use. model(1) specifies if matrix B should be estimated. model(2) specifies if matrix D should be estimated. model(3) specifies if the initial state \( x_0 \) should be estimated. model(4) specifies if the Kalman filter gain K is estimated. Default is \( \left[ 1 \ 1 \ 0 \ 0 \right] \). It is recommended only to estimate the initial state if the data length is short compared to the largest time constant of the system.
\item **partytype** This parameter specifies the type of parameterization that is used to parameterize the state space model. Two types of parameterization are supported: 'on'= Output Normal and 'tr'=TRidiagonal.
\item **options** Input parameters that are passed on directly to the optimization function from the Optimization Toolbox. See options for more information.
\end{itemize}

Outputs

\begin{itemize}
\item **A, B, C, D** System matrices of the optimized linear model. If B or D is not estimated, it will be returned as an empty matrix.
\item **x0** Estimate of the initial state. If the \( x_0 \) matrix is not estimated, it will be returned empty.
\item **K** Estimate of the initial state. Same here.
\item **options** Output parameters from the Optimization Toolbox. See options.
\end{itemize}
See Also

See also:  leastsq, foptions
**Purpose**

Separable Least Squares optimization of a wiener model. Model structure:

\[
\begin{align*}
x(k+1) &= Ax(k) + Bu(k) \\
y(k) &= Cx(k) + Du(k) + v(k) \\
z(k) &= f(y(k))
\end{align*}
\]

First, the state space matrices are parameterized. In order to minimize the number of parameters, the Separable Least Squares technique is used. This allows the cancellation of the parameters of the non-linearity, such that only the parameters that describe the linear part need to be optimized. The parameterized model is optimized with the `leastsq` function from the MATLAB optimization toolbox.

**Syntax**

\[
[A,B,C,D,x0,thl,options] = dslswie(u,z,A,B,C,D,x0,nn,model,options)
\]

**Inputs**

- `u, z`: The input and output data of the system to be optimized.
- `A, B, C, D`: Initial estimates of the linear part of the wiener model.
- `x0`: Initial condition. This is optional.
- `nn`: The order of the Chebychev polynomials in the static non-linearity.
- `model`: Vector with flags that specify the kind of model to use. `model(1)` specifies if `B` should be estimated. `model(2)` specifies if `D` should be estimated. `model(3)` specifies if the initial state `x0` should be estimated. Default is \([1, 1, 0]\). It is recommended only to estimate the initial state if the data length is short compared to the largest time constant of the system.
- `partype`: This parameter specifies the type of parameterization that is used to parameterize the state space model. Two types of parameterization are supported: 'on' = Output Normal and 'tr' = TRidiagonal.
- `options`: Input parameters that are passed on directly to the optimization function from the Optimization Toolbox. See `foptions` for more information.

**Outputs**

- `A, B, C, D`: System matrices of the optimized linear model. If the `B` or `D` matrix is not estimated, it will be zero.
- `x0`: Estimated initial state. If `x0` is not estimated it will be returned as an empty matrix.
- `options`: Output parameters from the Optimization Toolbox. See `foptions`.
See Also
foptions, dsllin
Purpose
This function converts a discrete time state space model to a parameter vector that describes the model.

Model structure:

\[
\begin{align*}
x(k + 1) &= Ax(k) + Bu(k) + Ke(k) \\
y(k) &= Cx(k) + Du(k) + e(k)
\end{align*}
\]

Syntax

\[
\begin{align*}
[\text{theta},T,\text{params}] &= \text{dss2th}(A,C,\text{partype}) \\
[\text{theta},T,\text{params}] &= \text{dss2th}(A,B,C,D,\text{partype}) \\
[\text{theta},T,\text{params}] &= \text{dss2th}(A,B,C,D,x0,K,\text{partype})
\end{align*}
\]

Inputs

- \(A,B,C,D\): System matrices describing the state space system. The \(B\) and \(D\) matrices are optional and can be left out or given as an empty matrix to indicate it is not part of the model.
- \(x0\): Initial condition. This is optional.
- \(K\): Kalman gain. Also this matrix is optional.
- \(\text{partype}\): This parameter specifies the type of parameterization that is used to parameterize the state space model. Two types of parameterization are supported: 'on' = Output Normal and 'tr' = TRidiagonal.

Outputs

- \(\text{theta}\): Parameters vector describing the system.
- \(\text{params}\): A structure that contains the dimension of the system, such as the order, the number of inputs, whether \(D\), \(x0\) or \(K\) is present, etc.
- \(T\): Transformation matrix between the input state space system and the state space system in the form described by theta. (the one that is constructed by \text{dth2ss}.

See Also

- \text{dth2ss}
**Purpose**

This function converts a parameter vector that describes a discrete time state space model in output normal form to the state space matrix of that model. Model structure:

\[
\begin{align*}
x(k + 1) &= Ax(k) + Bu(k) + Ke(k) \\
y(k) &= Cx(k) + Du(k) + e(k)
\end{align*}
\]

**Syntax**

\[
[A,B,C,D,x0,K] = \text{dth2ss}(\text{theta},\text{params})
\]

\[
[A,C] = \text{dth2ss}(\text{theta},\text{params})
\]

**Inputs**

- **theta**: Parameter vector describing the system.
- **params**: A structure that contains the dimension parameters of the system, such as the order, the number of inputs, whether D, x0 or K is present in the model, etc.
- **T**: Transformation matrix to be applied to the state space system that is constructed from theta. This transformation might come from the function dss2th and can be used to reconstruct the original state space matrices that were given to dss2th.

**Outputs**

- **A,B,C,D**: System matrices describing the state space system in output normal form. If theta does not contain the parameters for a matrix, this matrix will be returned as an empty matrix.
- **x0**: Initial state. If theta does not contain parameters for x0, this vector will be returned as an empty matrix.
- **K**: Kalman gain. Same here.

**See Also**

dss2th
Purpose

Calculates the Kronecker canonical form from a regular pencil $A,B$ such that

$$A_a = QAZ = \begin{bmatrix} A_k & 0 \\ 0 & I \end{bmatrix}$$

$$B_b = QBZ = \begin{bmatrix} I & 0 \\ 0 & B_k \end{bmatrix}$$

where the eigenvalues of $A$s and $B$s are split according to alpha and option.

Syntax

$[AA,BB,Q,Z,na,nb]=\text{kronekf}(A,B,alpha,option)$

Inputs

- $A,B$ The given matrix pencil.
- $alpha$ The value used to split the eigenvalues of the pencil $A$ and $B$. Default value is 1.
- $option$ This parameter can be either 'circle' or 'halfplane'. 'circle' sorts the eigenvalues in $A_k$ to be absolutely smaller than alpha and those in $B_k$ to be larger than alpha. 'halfplane' sorts the eigenvalues in $A_k$ to have smaller real part than alpha, and those in $B_k$ to have larger real part. Default is 'circle'.

Outputs

- $AA,BB$ The pencil in canonical form.
- $Q$ The required left transformations.
- $Z$ The required right transformations.
- $na$ Dimension of the matrix $A_k$.
- $nb$ Dimension of the matrix $B_k$.

See Also

ncdestac
Purpose
Estimates the system matrices $A_c$, $A_a$, $C_c$ and $C_a$ of a non-causal LTI state space model using the output of one of the dordxx preprocessor routines. The model is divided in a causal part and an anti-causal part. The causal part is stable and can be simulated forward in time. The anti-causal part can only be simulated stably backward in time. The other matrices are estimated with ncdestbd. The model can be simulated with ncdlsim.

Model structure:
\[
\begin{align*}
x_c(k + 1) & = A_c x_c(k) + B_c u(k) \quad \text{(causal part)} \\
x_a(k - 1) & = A_a x_a(k) + B_a u(k) \quad \text{(anti-causal part)} \\
y(k) & = C_c x_a(k) + C_a x_a(k) + D u(k)
\end{align*}
\]

All preprocessor functions can be used for the causal case as well as the non-causal case. For information about the possible disturbance signals see the help pages for the preprocessor dordxx functions.

Syntax
\[
[A_c, A_a, C_c, C_a] = ncdestac(R, n)
\]

Inputs
- $R$  
  Data structure obtained from a dordxx function, containing the triangular factor and additional information (such as i/o dimension etc.).
- $n$  
  Order of system to be estimated.

Outputs
- $A_c, A_a, C_c, C_a$  
  Estimated system matrices.

See Also
- ncdestbd, ncdlsim
Purpose
Estimates the matrices $B_c$, $B_a$ and $D$ of a non-causal state space model, using the knowledge of the matrices $A_c$, $A_a$, $C_c$ and $C_a$. This function is able to concatenate different data batches, through the matrices $R$, $R_{old}$.
Model structure:

$$x_c(k + 1) = A_c x_c(k) + B_c u(k) \quad \text{(causal part)}$$

$$x_a(k - 1) = A_a x_a(k) + B_a u(k) \quad \text{(anti-causal part)}$$

$$y(k) = C_c x_c(k) + C_a x_a(k) + D u(k)$$

$B_c$, $B_a$ and $D$ are calculated by solving a linear least squares problem. The influence of the initial state $x_0$ in the data batches is compensated for in the solution. $x_{0_c}$ and $x_{0_a}$ can also be estimated.

Syntax

$$[B_c,B_a,D,x0c,x0a,R]=ncdestbd(u,y,Ac,Aa,Cc,Ca,model,Rold,iv,niv);$$

$$[B_c,B_a,D]=ncdestbd(u,y,Ac,Aa,Cc,Ca);$$

Inputs

$u,y$  
The input and output data of the system to be identified.

$Ac,Ae$  
The estimated system matrices $A_c$, $A_r$, $C_c$ and $C_a$ of the non-causal state space system matrices.

$Cc,Ca$  
Two element vector $[fB \ fD \ fX]$ indicating whether $B_c$, $B_a$, $D$ and $x_{0_c}$, $x_{0_a}$ should be estimated. Default value is $[1 \ 1 \ 0]$.

$R$  
$R$ matrix obtained from previous data batch. This variable can be used to process data in batches, or to combine data from different experiments.

$iv$  
Instrumental variable. For instance, for the errors-in-variables case the iv can be the past output or for closed loop data, the reference input.

$niv$  
Three element vector describing the number of lagged IV's used. $niv(1)$ is the number of lagged IVs, $niv(2)$ the number of lagged past outputs and $niv(3)$ the number of lagged past inputs. If only one element is present, no past data is used. If not given or empty appropriate default values are used. When past data is used as instrumental variable, about half of the input/output data is used for the instrument.

Outputs

$B_c,B_a,D$  
The estimated system matrices $B_c$, $B_a$ and $D$.

$x0c,x0a$  
The estimated initial state of the system.

$R$  
Compressed data matrix, storing information on the calculation of the matrices $B_c$, $B_a$ and $D$ in a following
ncdestbd. Used when analyzing multiple input-output data sequences.

See Also
ncdlsim, ncdestac
Purpose
This function simulates a non-causal, unstable or improper system. The model is divided in a causal part and an anti-causal part. The causal part is stable and can be simulated forward in time. The anti-causal part can only be simulated stably backward in time. The system can be given in the following forms Non-causal state space form:

\[
\begin{align*}
x_c(k + 1) &= A_c x_c(k) + B_c u(k) \\
x_a(k + 1) &= A_a x_a(k) + B_a u(k) \\
y(k) &= C_c x_c(k) + C_a x_a(k) + D u(k)
\end{align*}
\]

Unstable state space form:

\[
\begin{align*}
x(k + 1) &= A x(k) + B u(k) + w(k) \\
y(k) &= C x(k) + D u(k) + v(k)
\end{align*}
\]

Non-proper transfer function:

\[
a_0 y(k + n) + ... + a_0 y(k) = b_0 u(k + m) + ... + b_1 u(k + 1) + b_0 u(k)
\]

Syntax

\[
[y,x] = ncdlsim(Ac,Aa,Bc,Ba,Cc,Ca,D,u,xc0,xa0)
\]

\[
[y,x] = ncdlsim(A,B,C,D,u,x0)
\]

\[
[y,x] = ncdlsim(num,den,u)
\]

Inputs

- **Ac, Aa, Bc, Ba** State space matrices representing a non-causal state space model.
- **Cc, Ca, D** State space matrices representing a possibly unstable state space model.
- **A, B, C, D** State space matrices representing a possibly unstable state space model.
- **num, den** Numerator and denominator of possibly non-proper transfer function.
- **u** Input to the system.
- **xc0, xa0** Initial states of causal and anti-causal part of the non-causal state space model.
- **x0** Initial state of the state space model.

Outputs

- **y** Simulated output.
- **x** Simulated state of the state space model. The first \( n_c \) columns are the states of the causal part, the rest are the states of the anti-causal part.

See Also

- ncestac, ncestbd, dlsim
Purpose
Gives an estimate of the order of a system based on the singular values. Two methods are available: "manual" and "largest-gap". The manual method shows a semi-logarithmic plot of the singular values and lets the user manually choose the order by inspection. The largest-gap method simply selects the order to be the one after which the largest gap occurs in the semi-logarithmic plot.

Syntax
order=orderselect(S,method)

Inputs
S Vector with singular values, obtained from one of the preprocessor functions dordxx(dordom, dordpi, etc).
method Method for selecting the order. Possible methods are: "manual" and "largest-gap". Default is "manual".

Outputs
order Selected model order.

See Also
moesp, dordpo, dordpi
Purpose
Produces a binary sequence, with values 0 and 1. The chance of switching from level is given by the parameter 'rate'. Rate=0 gives a constant value 0. Rate=1 gives a signal that changes constantly between 0 and 1. Any value in between results in a random binary sequence. This kind of test signal has been described in [81].

Syntax
\[ y = \text{prbn}(N, \text{rate}); \]

Inputs
- \text{N} \quad \text{Number of points.}
- \text{rate} \quad \text{Chance of the signal changing level at every sample. Default is 0.5.}

Outputs
- \text{y} \quad \text{Random binary noise.}

See Also
Shave

**Purpose**

This function is used for reducing spikes from a measured signal. The spikes of the signal are 'shaved' as follows:

- From the signal a trend is computed using a fourth-order Butterworth filter.
- The standard deviation of the trend-corrected, clipped signal is computed.
- Detection band is defined by the trend plus and minus a certain factor times the standard deviation. All samples that are outside this band are replaced using linear interpolation. This 'shaving' method has been described in [5].

**Syntax**

```plaintext
shave(Signal);
Shaved_signal = shave(Signal);
Shaved_signal = shave(Signal, factor, Wn, lo_lim, up_lim);
```

**Inputs**

<table>
<thead>
<tr>
<th>Signal</th>
<th>factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal to be 'shaved' (column vector)</td>
<td>Multiplication factor which determines the width of the detection band. When the detection is poor, you should change this factor. This argument is optional. Its default value is 2.</td>
</tr>
<tr>
<td>Wn</td>
<td>Cut-off frequency of the low-pass filter used for trend determination. It must be in the range 0.0 &lt; Wn &lt; 1.0, with 1.0 corresponding to half the sample rate. This argument is optional. Its default value is 0.01.</td>
</tr>
<tr>
<td>lo_lim, up_lim</td>
<td>If these arguments are present, the signal is clipped to a minimum value of lo_lim and a maximum value of up_lim before the 'shaving' starts.</td>
</tr>
</tbody>
</table>

**Outputs**

- **Shaved_signal** 'Shaved' signal.

**See Also**
Purpose
Simulation of the time response of LTI continuous time systems to arbitrary inputs. The input and simulated output are represented by sampled signals. Unlike lsim, where the sampling is assumed to be zero order or first order hold, the sampling in tlsim is assumed to be trapezoidal or Tustin’s (hence the t in tlsim).

Syntax
\[ [y,x] = 
\]

Inputs
- A, B, C, D: State space matrices of the LTI system.
- u: Input signal.
- Ts: Sampling time.
- x0: Initial state of the system.

Outputs
- y: Output of the LTI system.
- x: State of the LTI system.

See Also
lsim, dlsim
Purpose
Compute the percentage Variance Accounted For (VAF) between two signals. The VAF is calculated as:

\[ 1 - \frac{\text{variance}(y - y_{est})}{\text{variance}(y)} \times 100\% \]

The VAF of two signals that are the same is 100%. If they differ, the VAF will be lower. When \( y \) and \( y_{est} \) are matrices, the VAF is calculated for every column in \( y \) and \( y_{est} \). The VAF is often used to verify the correctness of a model, by comparing the real output with the estimated output of the model.

Syntax
\[ v = \text{vaf}(y, ye) \]

Inputs
- \( y \) Reference signal, often the real output
- \( ye \) Second signal, often the estimated output of a model

Outputs
- \( v \) VAF, computed for the two signals

See Also
Index

Bilinear model, 54
Chebyshev polynomials, 63
column-space, 229
COM-MOESP, 101
Controllability, 237

Data equation, 17, 95
EIV, 40
Errors-in-variables, 40
Extended controllability matrix, 12
Extended observability matrix, 12

FLOPS, 177

guppy, 189
Hammerstein model, 54
Hankel matrix, 17
Ho-Kalman algorithm, 10

Impulse response function, 239
incremental covariance, 76
incremental mean, 76
independent increments, 76
innovation model, 106
Innovation model problem, 30
Instrumental variable, 26
IRF, 239
IV, 26

Kalman filter, 36
Kung’s algorithm, 10

left matrix divide, 190
Linux, 189
LQ factorization, 232

Minimality, 238
MoESP, 9
noise-free, 97
noise-free problem, 14

Observability, 238
Om-moESP, 16
orthogonal stationary increments, 75
orthogonal increments, 76
Orthogonal projection, 229, 231
output-error model, 25
output-error problem, 26

Past input MOESP, 26
Past output MOESP, 31
PE, 18, 45
Pem, 2
Penrose-Moore pseudo inverse, 190
Persistence of excitation, 18, 45
Pi-moESP, 26
pinv, 190
Po-moESP, 31
power spectral density, 73
Prediction-Error Method, 2
Projection, 229, 231
PSD, 73
pseudo inverse, 190

QR-factorization, 230

Realization problem, 10
row-space, 229

Selena, vii
Separable Least Squares, 141
Singular Value Decomposition, 232
SLS, 141
SMI, 9
stationary increments, 76
Stochastic model, 36
Subspace model identification, 9
SVD, 232

Transfer function, 239

White output-error problem, 20
Wiener model, 54, 128
Wiener process, 76
Biographies of scientists

Andrei Andreyevich Markov (1856-1922) ........................................ 11
Hermann Hankel (1839-1873) .................................................. 12
Leopold Kronecker (1823-1891) ................................................. 51
Norbert Wiener (1894-1964) ..................................................... 54
Johann Carl Friedrich Gauss (1777-1855) .................................... 55
Pafnuty Lvovich Chebyshev (1821-1894) ...................................... 63
Paul Dirac (1902-1984) ............................................................. 74
Robert Brown (1773-1858) ....................................................... 75
Aleksandr Mikhailovich Lyapunov (1857-1918) .............................. 83
Edmond Laguerre (1834-1886) ................................................... 86
Jorgen Pedersen Gram (1850-1916) ........................................... 147
Andre-Louis Cholesky (1875-1918) .......................................... 149
Otto Hesse (1811-1874) ........................................................... 151
Sir Isaac Newton (1643-1727) ............................................... 153
Issai Schur (1875-1941) ......................................................... 187
Bibliography


Summary

The subject of this thesis is the extension of the Subspace Model Identification (SMI) method MOESP in both theory and practice. In recent years, SMI methods have become increasingly popular and different families of Subspace Model Identification techniques have arisen in the past decade, such as N4SID, MOESP and CVA. While these methods are very successful in the identification of discrete time state space models, a lot of research still has to be done in the field of Subspace Model Identification.

Chapter 2 gives an overview of the existing discrete time MOESP algorithms as found in literature. First, an introduction into Subspace Model Identification is given with the use of Kung's realization algorithm. This algorithm shows the underlying principles of SMI in a very clear way. It uses a singular value decomposition (SVD) to make a state space realization of a system from its impulse response parameters. The chapter then continues with the ordinary MOESP algorithm which does the same, but now based on the input and output signals of the system instead of the impulse response. The chapter then describes more sophisticated algorithms like PI-MOESP, PO-MOESP and EIV-MOESP that deal with different cases of perturbed input and output signals.

Next the consistency requirements for MOESP are discussed. In order for the MOESP algorithms to arrive at a consistent estimate of a system from input and output measurements, certain requirements have to be made to the input signal. These requirements deal with the persistence of excitation of the signal. We have given an overview of the results on this topic that can be found in literature, for every one of the described MOESP algorithms. For the OM-MOESP and PO-MOESP algorithm explicit requirements can be given for the input, in order to guarantee a consistent estimate of the system. For the PI-MOESP and EIV-MOESP algorithms no such results were found, but implicit requirements can be given. The MOESP algorithms estimate only the A and C matrices of the state space model. Therefore the matrices B and D and the initial state have to be estimated in a second step that is discussed in section 2.9. The chapter finishes with a discussion of how a non-linear Wiener model can be estimated using PI-MOESP under the assumption that the input is Gaussian distributed.

The next chapter, chapter 3, deals with the extension of SMI to the estimation of continuous time models from sampled input and output data. The chapter starts with
a discussion on the description of white noise in a continuous time setting. Ideally, a continuous time white noise description should be similar to the discrete time case. In the discrete time case, a white noise signal has samples that are uncorrelated with each other. However, in order to translate this principle to a continuous time setting, a signal should have infinite energy. Therefore a different approach, based on the Wiener process, is proposed. In this approach, the signal itself is not uncorrelated, but it’s increments. The Wiener process corresponds to the observations of the botanist Robert Brown (Brownian motion).

Next we introduced the Laguerre filters. The Laguerre filters are a set of orthogonal filters. As shown in chapter 3, filtering a Wiener process with a Laguerre filter has similar properties as using the shift-operator on discrete time white noise. When the Wiener process is filtered by two Laguerre filters of different order, the resulting filtered signals are uncorrelated. This is identical to applying two shift operators of different order to a discrete time white noise signal. Then the Laguerre filter is applied in the transformation of the normal continuous time state space equations to an alternative form. This alternative form has the all-pass operator as basis, instead of the differential operator that is used in the normal state space equations. It is shown how matrices of the one description relate to those of the other, and how the disturbance signals are transformed. The newly introduced state space domain is called the all-pass domain.

This newly introduced notation and all-pass domain is then used for a continuous time MOESP algorithm. This algorithm uses a bank of Laguerre filters to construct a data matrix of the input and output signals. It is shown that these data matrices relate to each other through a data equation that is almost identical to the discrete time case. The difference with the discrete time case is that the state space matrices that are used in the data equation are now from the all-pass domain state space equation. After solving for these matrices, using the MOESP technique, they can be translated to the normal continuous time state space matrices using a simple matrix equation. This procedure can be used to construct a continuous time variant of the OM-MOESP, PI-MOESP and the PO-MOESP algorithms. In the thesis we have focused on the OM-MOESP and PO-MOESP algorithms.

The formulation of the algorithm has been done entirely in the continuous time. No sampling was involved. Therefore it is necessary to look at the implications of the sampling procedure, that happens in practice. It has been shown that in both the noise-free case and the noisy case, the error that is due to the sampling and the discrete time approximation of the Laguerre filters is bounded. In the noise-free case the error is proportional to the square of the sampling time. In the case where measurement noise is present, this error is proportional to the sampling time itself. In both cases the error goes to zero as the sampling time approaches zero. Finally the chapter concludes with the use of continuous time MOESP for the identification of continuous time Wiener models.

One of the open questions in SMI is the relation between Prediction-Error method (PEM) identification and SMI. PEM approaches system identification as the optimization of
a cost-function to the parameters of the model. This optimization is often non-linear and the initial estimates of the parameters are therefore important for the final result of the optimization. The sensitivity for the initial estimate and the time consuming optimization process are two problems of the PEM approach. But on the other hand the PEM approach has the advantage of finding the optimal model with respect to the prediction-error criterion. The SMI approach does not have the disadvantages of the PEM approach. It is a non-iterative method that does not require an initial estimate. However, the model that is delivered by MOESP is not optimal with respect to any known cost-function. In chapter 4 a combination of MOESP and an optimization based method is proposed. This combination makes use of the Separable Least Squares (SLS) technique. This technique separates the parameter set that describe the model class in two groups: those on which the output of the model depends linearly and those on which the output depends non-linearly. The optimization with SLS then only needs to optimize the parameters on which the cost-function depends non-linearly.

It is shown that the state space matrices $A$ and $C$ can be parameterized independent from the matrices $B$ and $D$, and that the output is linearly dependent on the parameters that describe $B$ and $D$. Therefore we can use the SLS technique to reduce the number of parameters in the optimization step to those that describe the matrices $A$ and $C$. Initial estimates for these matrices can be obtained from MOESP, since this method estimates only $A$ and $C$ in the first step. In examples it has been shown that the optimization with SLS can require less iterations and less time than an equivalent optimization without SLS. The chapter also deals with the recursive optimization problem. In this case a new sample becomes available at every iteration. An algorithm is derived that updates the model recursively, using as much information as possible from the previous step in the calculation of the updated model parameters.

Finally the optimization of a non-linear model of the Wiener form is discussed. For this type of model, where the non-linearity is modeled as a weighted sum of Chebychev polynomials, the parameters that describe the non-linearity are in the group of parameters on which the output depends linearly. Therefore only the parameters that describe the linear state space model need to be optimized.

To make the Subspace Model Identification algorithms available to a broader public, the SMI-toolbox was developed. This toolbox contains the MOESP functions that were developed over the years at the Control laboratory and during the research for this thesis. For a high quality toolbox it is necessary to have a good implementation of the routines that are included. This is why in chapter 5 the implementation of MOESP is examined at a detailed level. Two different methods of implementing MOESP are compared. The first is the ordinary way, using a QR factorization. The second way uses the Cholesky factorization. It has been shown that the Cholesky factorization results in a faster routine. The second problem that has been studied in chapter 5 is the implementation of the estimation of $B$ and $D$. This is a computational intensive algorithm. Therefore, a detailed analysis of different implementations has been made. The resulting routine that is proposed, is more computational efficient than the straight forward implementation. The last topic in this chapter deals with
the combination of MOESP and N4SID. The study shows that a combination of the
two different Subspace Model Identification algorithms allows for a faster and more
computational-efficient algorithm without a significant loss of accuracy.

Chapter 6 introduces the SMI-toolbox itself. It gives four examples of the use of
the functions that are included in the toolbox. The first example introduces the ba-
sic functions and the structure of the toolbox at the hand of a simple identification
problem. Example two shows how multiple data-sets can be combined for the identi-
fication of a single model. The third example shows the identification of a non-linear
Wiener system. The last example shows the use of EIV-MOESP for the identification
of a system from data that is obtained in a closed loop system.

Summarizing, this thesis contains a number of extensions to the MOESP family of
Subspace Model Identification methods. It gives a new set of algorithms for the iden-
tification of continuous time systems and shows how the models that are identified
with MOESP can be used as an initial estimate in a subsequent optimization. This
optimization can be done with the use of SLS to make full use of the combination
between MOESP and the optimization method. Finally, the results of this thesis are
implemented for MATLAB in the SMI-toolbox, that contains a collection of the algo-
rithms that are discussed in the different chapters.
Samenvatting

Het onderwerp van dit proefschrift is de uitbreiding van Deelruimte-Model-Identificatie (Subspace Model Identification, SMI) in zowel theorie als praktijk. In de afgelopen jaren zijn SMI methoden steeds populairder geworden. In de laatste tien jaar zijn verschillende families van Deelruimte-Model-Identificatie technieken ontstaan, zoals N4SID, MOESP en CVA. Hoewel deze methoden erg succesvol zijn in de identificatie van discrete tijd toestandsmodellen, is er nog veel onderzoek nodig op het gebied van Deelruimte-Model-Identificatie.


Vervolgens worden de consistentie eisen voor MOESP behandeld. Om de MOESP-algoritmen een consistentie schatting van een systeem op basis van ingangs- en uitgangssignalen te kunnen laten maken, moeten een aantal eisen aan het ingangssignaal worden gesteld. Deze eisen hebben te maken met de consistentie van de signalen. We hebben een overzicht gegeven van de resultaten op dit gebied die in de literatuur kunnen worden gevonden, voor iedere van de beschreven MOESP-algoritmen. Voor het OM-MOESP en PO-MOESP algoritme kunnen expliciete eisen worden gegeven voor het ingangssignaal, zodat een consistentie schatting van het systeem wordt gegarandeerd. Voor de PI-MOESP en EIV-MOESP algoritme zijn dergelijke resultaten niet gevonden, echter impliciet resultaten kunnen wel worden gegeven. De MOESP algoritmen schatten alleen de $A$ en $C$ matrices van het toestandsmodel. Om deze reden moeten de matrices $B$ en $D$ en de initiële toestand worden geschat in een tweede stap die wordt besproken in paragraaf 2.9. Het hoofdstuk sluit af met een bespreking van het schatten van een niet-lineair Wiener model met behulp van PI-MOESP, onder de aanname dat de ingang Gaussisch verdeeld is.


De nieuwe notatie en het al-doorlaat domein worden vervolgens gebruikt voor een continue-tijd-MOESP algoritme. Dit algoritme gebruikt een bank van Laguerre-filters om data matrices te construeren uit de ingangs- en uitgangssignalen. We laten zien dat deze data matrices aan elkaar worden gerelateerd door middel van een data-vergelijking die identiek is aan het discrete tijd geval. Het verschil met het discrete tijd geval is dat de toestandsmatrices die voorkomen in de data-vergelijking nu uit het al-doorlaat-domein-toestandsmodel komen. Nadat deze matrices zijn opgelost met gebruik van de MOESP techniek, kunnen ze worden vertaald naar de normale continue tijd toestandsmatrices met een simpele matrix vergelijking. Deze procedure kan worden gebruikt om een continue tijd variant van de OM-MOESP, PI-MOESP en de PO-MOESP algoritmen te construeren. In het proefschrift hebben we ons geconcentreerd op de OM-MOESP en PO-MOESP algoritmen.

De formulering van de algoritmen is volledig gedaan in continue tijd. Er is geen gebruik gemaakt van bemonstering. Daarom is het nodig om te kijken naar de gevolgen van de bemonsteringsprocedure die in de praktijk wordt gebruikt. We hebben laten zien dat in zowel het ruisvrije geval als de situatie met ruis, de fout, die ontstaat door het bemonsteren en de discrete tijd benadering van de Laguerre filters, begrenst is. In het ruisvrije geval is de fout proportioneel met het kwadraat van de bemonsteringsstijd. In de situatie waar meet-ruis aanwezig is, is de fout proportioneel met
Samenvatting

De bemonsteringstijd zelf. In beide gevallen benaderd de fout nul als de bemonsteringstijd nul benadert. Tot slot besluit het hoofdstuk met het gebruik van continue tijd MOESP voor de identificatie van continue tijd Wiener modellen.

Een van de open vragen in SMI is de relatie tussen Predictie-fout methoden (Prediction-Error Method, PEM) en SMI. PEM benadert systeem identificatie als de optimalisatie van een kostenfunctie naar de parameters van het model. Deze optimalisatie is vaak niet-lineair en de initiële schatting van de parameters zijn dus belangrijk voor het uiteindelijke resultaat van de optimalisatie. De gevoeligheid voor initiële schatting en het tijdrovende optimalisatie-proces zijn twee problemen van de PEM-benadering. Maar aan de andere kant heeft de PEM-aanpak het voordeel van het vinden van een optimaal model, op basis van het predictie-fout criterium. De SMI benadering heeft de nadelen van de PEM aanpak niet. Het is een niet-iteratieve methode die geen initiële schatting nodig heeft. Maar het model dat een SMI-methode leverd is niet optimaal met betrekking tot enige bekende kostenfunctie. In hoofdstuk 4 wordt een combinatie van MOESP en een op optimalisatie gebaseerde methode voorgesteld. Deze maakt gebruik van de Scheidbaar Kleinste Kwadraten (Separable Least Squares, SLS) techniek. Deze techniek scheidt de parameter-set die de model-klasse beschrijft in twee groepen. Diegene waar de uitgang van het model lineair van afhankt en diegene waar de uitgang niet-lineair van afhangt. De optimalisatie met SLS hoeft dan alleen die parameters te optimaliseren waar de uitgang niet-lineair van afhangt.

We hebben in hoofdstuk 4 aangetoond dat de toestandsmatrices $A$ en $C$ onafhankelijk van $B$ en $D$ kunnen worden geparametriserend en dat de uitgang lineair afhankt van de parameters in $B$ en $D$. Daarom kunnen we de SLS-techniek gebruiken voor het reduceren van het aantal parameters in de optimalisatie-stap. Alleen diegene die de matrizen $A$ en $C$ beschrijven hoeven dan te worden geoptimaliseerd. Initiële schattingen voor deze matrizen kunnen worden verkregen met MOESP, daar deze methode in de eerste stap alleen $A$ en $C$ schat. In voorbeelden is aangetoond dat de optimalisatie met SLS in veel gevallen minder iteraties nodig zal hebben, en minder tijd nodig heeft, dan een equivalente optimalisatie zonder SLS. Het hoofdstuk gaat ook in op het recursieve optimalisatie-probleem. In dit geval komt bij iedere iteratie een nieuwe meting beschikbaar. Er is een algoritme afgeleid dat het model recursief verbetert. Het algoritme gebruikt bij het berekenen van de nieuwe modelparameters zoveel mogelijk informatie uit de vorige stap.

Tot slot wordt in hoofdstuk 4 de optimalisatie van een niet-lineair model van de Wiener-vorm besproken. Voor dit type model, waar de niet-lineairiteit gemoduleerd wordt als een gewogen som van Chebychev polynomen, zitten de parameters die de niet-lineairiteit beschrijven in de groep met parameters waar de uitgang lineair van afhangt. Daarom hoeven met de SLS-techniek alleen de parameters die het lineaire toestandsmodel beschrijven te worden geoptimaliseerd.

Om de Deelruimte-Model-Identificatie algoritmen beschikbaar te maken voor een breder publiek, is de 'SMI-toolbox' ontwikkeld. Deze 'toolbox' bevat de MOESP-functies die door de jaren heen in de vakgroep regeltechniek en tijdens het onderzoek voor dit proefschrift zijn ontwikkeld. Voor een 'toolbox' van hoge kwaliteit is het
nodig om een goede implementatie van de functies te hebben. Om deze reden wordt in hoofdstuk 5 de implementatie van MOESP bestudeerd. Twee methodes voor het implementeren van MOESP worden vergeleken. De eerste is de normale manier, met behulp van de QR-factorisatie. De tweede manier gebruikt de Cholesky-factorisatie. We hebben laten zien dat de Cholesky-factorisatie leidt tot een snellere functie. De tweede probleem dat bestudeerd is in hoofdstuk 5 is de implementatie van de schatting van $B$ en $D$. Dit is een rekenkundig zwaar en algoritme. Daarom is een gedetailleerde analyse van verschillende implementaties gemaakt. Het resulterende algoritme dat wordt voorgesteld, is rekenkundig meer efficiënt dan de rechttoe-rechtlaan implementatie. Het laatste onderwerp in dit hoofdstuk is de combinatie van MOESP en N4SID. Deze studie toont aan dat de combinatie van de twee verschillende Deelruimte-Model-Identificatie algoritmen een sneller en rekenkundig efficiënter algoritme oplevert, zonder significant verlies van nauwkeurigheid.

Hoofdstuk 6 introduceert de 'SMI-toolbox' zelf. Het geeft vier voorbeelden van het gebruik van de functies die in de 'toolbox' zijn opgenomen. Het eerste voorbeeld introduceert de basis-functies en de structuur van de 'toolbox' aan de hand van een simpel identificatie probleem. Voorbeeld twee toont hoe meerdere datasets kunnen worden gecombineerd om één model te identificeren. Het derde voorbeeld toont de identificatie van een niet-lineair Wiener systeem. Het laatste voorbeeld toont het gebruik van EIV-MOESP voor de identificatie van een systeem vanuit data die is verkregen in een gesloten-lus-systeem.

Samenvattend, bevat dit proefschrift een aantal uitbreidingen van de MOESP familie van Deelruimte-Model-Identificatie methodes. Het geeft een nieuwe set algoritmen voor de identificatie van continue-tijd-systemen en toont hoe de modellen die met MOESP zijn geïdentificeerd kunnen worden gebruikt als een initiële schatting in een daaropvolgende optimalisatie. Deze optimalisatie kan worden uitgevoerd met behulp van de SLS-techniek om volledig gebruik te maken van de combinatie van MOESP en de optimalisatie-methode. Tot slot zijn de resultaten van dit proefschrift geïmplementeerd voor MATLAB in de 'SMI-toolbox', die een collectie bevat van de algoritmen die in de verschillende hoofdstukken zijn besproken.
Publications


Curriculum Vitae

Bert Haverkamp was born on February 18th, 1971 in Den Helder and lived his childhood on Wieringen. After having received secondary education at the MAVO in Hippolytushoef, he continued with an intermediate technical study at the MTS in Alkmaar. This study he finished in 1990. Next he visited the college of technology (HTS) in Alkmaar, where he passed the propaedeutic exam in 1991.

That same year, he started his study of Electrical Engineering at Delft University of Technology. He graduated at the Control laboratory. The subject of his graduation thesis was the identification of the human ankle dynamics. This project was a collaboration with the neuromuscular control laboratory at McGill University in Montréal, Canada, where he spent five months for research. He received his Master of Science degree (cum laude) at Delft University of Technology in September 1995.

He started as a Ph.D. student at the Systems and Control Engineering group in November that same year. The research topic was the extension of the Subspace model identification method MOESP, that was developed in the group, in both theory and practice. The results of this research are described in this thesis.

After finishing his Ph.D period at Delft University of Technology he moved in February 2000 to the beautiful Limburg country where he and his girlfriend Selena now live in Maastricht. He started working at Libertel NV. Here he is concerned with strategic radio planning issues in the mobile telecommunication network.