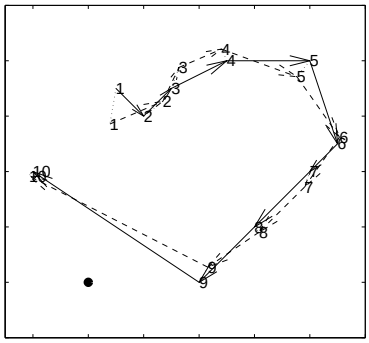
Exact and Approximate Modeling of Linear Systems:

A Behavioral Approach



Ivan Markovsky Jan C. Willems

Sabine Van Huffel Bart De Moor

Leuven, December 29, 2005

Preface

The behavioral approach, put forward in the three part paper by J. C. Willems [Wil87], includes a rigorous framework for deriving mathematical models, a field called system identification. By the mid 80's there was a well developed stochastic theory for linear time-invariant system identification—the prediction error approach of L. Ljung—which has numerous "success stories". Nevertheless, the rationale for using the stochastic framework, the question of what is meant by an optimal (approximate) model, and even more basically what is meant by a mathematical model remained to some extent unclear.

A synergy of the classical stochastic framework (linear system driven by white noise) and a key result of [Wil87] that shows how a state sequence of the system can be obtained directly from observed data led to the very successful subspace identification methods [VD96]. Now the subspace methods together with the prediction error methods are the classical approaches for system identification.

Another follow-up of [Wil87] is the global total least squares approach due to Roorda and Heij. In a remarkable paper [RH95], Roorda and Heij address an approximate identification problem truly in the behavioral framework, i.e., in a representation free setting. Their results lead to practical algorithms that are similar in structure to the prediction error methods: double minimization problems, of which the inner minimization is a smoothing problem and the outer minimization is a nonlinear least squares problem. Unfortunately, the global total least squares method has gained little attention in the system identification community and the algorithms of [RH95, Roo95] did not find their way to robust numerical implementation and consequently to practical applications.

The aim of this book is to present and popularize the behavioral approach to mathematical modeling among theoreticians and practitioners. The framework we adopt applies to static as well as dynamic and to linear as well as nonlinear problems. In the linear static case, the approximate modeling problem considered specializes to the total least squares method, which is classically viewed as a generalization of the least squares method to fitting problems $Ax \approx b$, in which there are errors in both the vector b and the matrix A. In the quadratic static case, the behavioral approach leads to the orthogonal regression method for fitting data to ellipses. In the first part of the book we examine static approximation problems: weighted and structured total least squares problems and estimation of bilinear and quadratic models, and in the second part of the book we examine dynamic approximation problems: exact and approximate system identification. The exact identification problem falls in the field of subspace identification and the approximate identification problem is the global total least squares problem of Roorda and Heij.

Most of the problems in the book are presented in a deterministic setting, although one can give a stochastic interpretation to the methods derived. The appropriate stochastic model for this aim is the errors-in-variables model, where all observed variables are assumed inexact due to measurement errors added on "true data" generated by a "true model". The assumption of the existence of a true model and the additional stochastic ones about the measurement errors, however, are rarely verifiable in practice.

Except for the chapters on estimation of bilinear and quadratic models, we consider total least squares-type problems. The unifying framework for approximate modeling put forward in the book is called *misfit approach*. In philosophy it differs essentially from the classical approach, called *latency approach*, where the model is augmented with unobserved latent variables. A topic of current research is to clarify how the misfit and latency approaches compare and complement each other.

We do not treat in the book advanced topics like statistical and numerical robustness of the methods and algorithms. On the one hand, these topics are currently less developed in the misfit setting than in the latency setting and, on the another hand, they go beyond the scope of a short monograph. Our hope is that robustness as well as recursivity, further applications, and connections with other methods will be explored and presented elsewhere in the literature.

The prerequisites for reading the book are modest. We assume an undergraduate level linear algebra and systems theory knowledge. Familiarity with system identification is helpful but is not necessary. Sections with more specialized or technical material are marked with $*$. They can be skipped without loss of continuity on a first reading.

This book is accompanied by a software implementation of the described algorithms. The software is callable from MATLAB and most of it is written in MATLAB® code. This allows readers who have access to and knowledge of MATLAB to try out the examples, modify the simulation setting, and apply the methods on their own data.

The book is based on the first author's Ph.D. thesis at the Department of Electrical Engineering of the Katholieke Universiteit Leuven, Belgium. This work would be impossible without the help of sponsoring organizations and individuals. We acknowledge the financial support received from the Research Council of K.U. Leuven and the Belgian Programme on Interuniversity Attraction Poles, projects IUAP IV–02 (1996–2001) and IUAP V–22 (2002–2006). The work presented in the first part of the book is done in collaboration with Alexander Kukush from the National Taras Shevchenko University, Kiev, Ukraine, and the work presented in the second part is done in collaboration with Paolo Rapisarda from the University of Maastricht, The Netherlands. We would like to thank Diana Sima and Rik Pintelon for useful discussions and proofreading the drafts of the manuscript.

Ivan Markovsky
Jan C. Willems
Sabine Van Huffel
Bart De Moor

Leuven, Belgium
December 29, 2005

Contents

Chapter 1

Introduction

The topic of this book is fitting models to data. We would like the model to fit the data exactly; however, in practice often the best that can be achieved is only an approximate fit. A fundamental question in approximate modeling is how to quantify the lack of fit between the data and the model. In this chapter, we explain and illustrate two different approaches for answering this question.

The first one, called *latency*, augments the model with additional unobserved variables that allow the augmented model to fit the data exactly. Many classical approximate modeling techniques such as the least squares and autoregressive moving average exogenous (ARMAX) system identification methods are latency oriented methods. The statistical tool corresponding to the latency approach is *regression*.

An alternative approach, called *misfit*, resolves the data–model mismatch by correcting the data, so that it fits the model exactly. The main example of the misfit approach is the total least squares method and the corresponding statistical tool is *errors-in-variables regression*.

1.1 Latency and Misfit

Classically a model is defined as a set of equations involving the data variables, and the lack of fit between the data and the model is defined as a norm of the *equation error*, or residual, obtained when the data is substituted in the equations. Consider, for example, the familiar linear static model, represented by an overdetermined system of equations $AX \approx B$, where A, B are given measurements, and the classical least squares (LS) method, which minimizes the Frobenius norm of the residual $E := AX - B$, i.e.,

$$\min_{E,X} \|E\|_{\mathrm{F}} \quad \text{subject to} \quad AX = B + E.$$

The residual E in the LS problem formulation can be viewed as an unobserved, latent variable that allows us to resolve the data–model mismatch. An approximate model for the data is obtained by minimizing some norm (e.g., the Frobenius norm) of E. This cost function is called *latency*, and equation error based methods are called latency oriented.

A fundamentally different approach is to find the smallest correction on the data that makes the corrected data compatible with the model (i.e., resulting in a zero equation error).

Then the quantitative measure, called *misfit*, for the lack of fit between the data and the model is taken to be a norm of the correction. Applied to the linear static model, represented by the equation $AX \approx B$, the misfit approach leads to the classical total least squares (TLS) method [GV80, VV91]:

$$\min_{\Delta A, \Delta B, X} \left\| \begin{bmatrix} \Delta A & \Delta B \end{bmatrix} \right\|_{\mathrm{F}} \quad \text{subject to} \quad (A + \Delta A)X = B + \Delta B.$$

Here ΔA, ΔB are corrections on the data A, B; and X is a model parameter.

> The latency approach corrects the model in order to make it match the data. The misfit approach corrects the data in order to make it match the model. Both approaches reduce the approximate modeling problem to exact modeling problems.

When the model fits the data exactly, both the misfit and the latency are zero, but when the model does not fit the data exactly, in general, the misfit and the latency differ.

Optimal approximate modeling aims to minimize some measure of the data–model mismatch over all models in a given model class. The latency and the misfit are two candidate measures for approximate modeling. The classical LS and TLS approximation methods minimize, respectively, the latency and the misfit for a linear static model class, represented by the equation $AX \approx B$. Similarly, the algebraic and geometric methods for ellipsoid fitting minimize the latency and the misfit for a quadratic static model class. For the linear time-invariant (LTI) dynamic model class, the latency and the misfit approaches lead to, respectively, the ARMAX and errors-in-variables (EIV) identification methods.

In the next section we illustrate via examples the misfit and latency approaches for data fitting by linear static, quadratic static, and LTI dynamic models.

1.2 Data Fitting Examples

Consider a data set $\mathscr{D} = \{ d_1, \ldots, d_N \}$ consisting of 2 real variables, denoted by a and b, i.e.,

$$d_i = \begin{bmatrix} a_i \\ b_i \end{bmatrix} =: \mathrm{col}(a_i, b_i) \in \mathbb{R}^2,$$

and $N = 10$ data points. This data is visualized in the plane; see Figure 1.1. The order of the data points is irrelevant for fitting by a static model. For fitting by a dynamic model, however, the data is viewed as a *time series*, and therefore the order of the data points is important.

Line Fitting

First, we consider the problem of fitting the data by a line passing through the origin $(0, 0)$. This problem is a special case of modeling the data by a linear static model. The classical LS and TLS methods are linear static approximation methods and are applied next to the line fitting problem in the example.

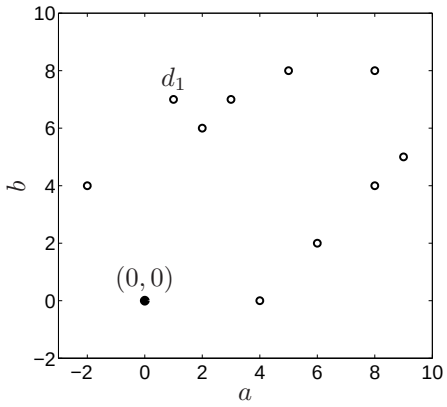


Figure 1.1. *The data $\mathscr{D}$ consists of 2 variables and 10 data points (∘). (•—point $(0,0)$.)*

Least Squares Method

If the data points $d_1, \ldots, d_{10}$ were on a line, then they would satisfy a linear equation

$$a_i x = b_i, \quad \text{for } i = 1, \ldots, 10 \quad \text{and} \quad \text{for some } x \in \mathbb{R}.$$

The unknown x is a *parameter* of the fitting line (which from the modeling point of view is the linear static model). In the example, the parameter x has a simple geometric meaning: it is the tangent of the angle between the fitting line and the horizontal axis. Therefore, exact fitting of a (nonvertical) line through the data boils down to choosing $x \in \mathbb{R}$.

However, unless the data points were on a line to begin with, exact fit would not be possible. For example, when the data is obtained from a complicated phenomenon or is measured with additive noise, an exact fit is not possible. In practice most probably both the complexity of the data generating phenomenon and the measurement errors contribute to the fact that the data is not exact.

The latency approach introduces an equation error $e = \mathrm{col}(e_1, \ldots, e_{10})$, so that there exists a corresponding parameter $\hat{x} \in \mathbb{R}$, satisfying the modified equation

$$a_i \hat{x} = b_i + e_i, \qquad \text{for } i = 1, \ldots, 10.$$

For any given data set $\mathscr{D}$ and a parameter $\hat{x} \in \mathbb{R}$, there is a corresponding e, defined by the above equation, so that indeed the latency term e allows us to resolve the data–model discrepancy.

The LS solution $\hat{x}_{\text{ls}} := \big(\sum_{i=1}^{10} b_i a_i\big) / \big(\sum_{i=1}^{10} a_i^2\big)$ minimizes the latency,

$$\text{latency} := \|e\|,$$

over all $x \in \mathbb{R}$. The line corresponding to the parameter $\hat{x}_{\text{ls}}$ is the optimal fitting line according to the latency criterion. It is plotted in the left plot of Figure 1.2.

The LS method can also be given an interpretation of correcting the data in order to make it match the model. The equation error e can be viewed as a correction on the second

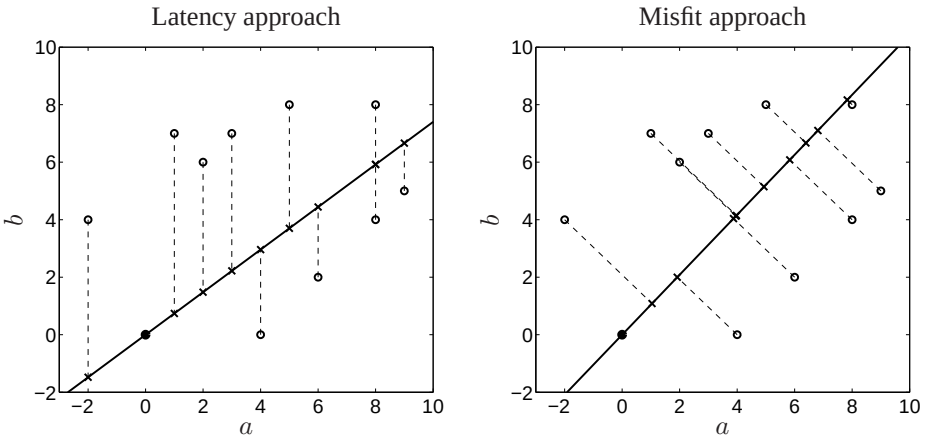


Figure 1.2. *Optimal fitting lines (—) and data corrections (- - -).*

coordinate b. The first coordinate a, however, is not corrected, so that the LS corrected data is

$$\hat{a}_{\text{ls},i} := a_i \quad \text{and} \quad \hat{b}_{\text{ls},i} := b_i + e_i, \quad \text{for } i = 1, \ldots, 10.$$

By construction the corrected data lies on the line given by the parameter $\hat{x}_{\text{ls}}$, i.e.,

$$\hat{a}_{\text{ls},i}\hat{x}_{\text{ls}} = \hat{b}_{\text{ls},i}, \quad \text{for } i = 1, \ldots, 10.$$

The LS corrections $\Delta d_{\text{ls},i} := \text{col}(0, e_i)$ are vertical lines in the data space (see the dashed lines in Figure 1.2, left).

> Geometrically, the latency is the sum of the squared vertical distances from the data points to the fitting line.

Total Least Squares Method

The misfit approach corrects both coordinates a and b in order to make the corrected data exact. It seeks corrections $\Delta d_1, \ldots, \Delta d_{10}$, such that the corrected data

$$\hat{d}_i := d_i + \Delta d_i$$

lies on a line; i.e., with $\text{col}(\hat{a}_i, \hat{b}_i) := \hat{d}$, there is an $\hat{x} \in \mathbb{R}$, such that

$$\hat{a}_i\hat{x} = \hat{b}_i, \quad \text{for } i = 1, \ldots, 10.$$

For a given parameter $\hat{x} \in \mathbb{R}$, let $\Delta\mathscr{D} = \{\, \Delta d_1, \ldots, \Delta d_{10} \,\}$ be the smallest in the Frobenius norm correction of the data that achieves an exact fit. The misfit between the line corresponding to $\hat{x}$ and the data is defined as

$$\text{misfit} := \left\| \begin{bmatrix} \Delta d_1 & \cdots & \Delta d_{10} \end{bmatrix} \right\|_{\text{F}}.$$

Geometrically, the misfit is the sum of the squared orthogonal distances from the data points to the fitting line.

The optimal fitting line according to the misfit criterion and the corresponding data corrections are shown in the right plot of Figure 1.2.

Ellipsoid Fitting

Next, we consider fitting an ellipse to the data. This problem is a special case of modeling the data by a quadratic static model. We show the latency and misfit optimal fitting ellipses. The misfit has the geometric interpretation of finding the orthogonal projections of the data points on the ellipse. The latency, however, has no meaningful geometric interpretation in the ellipsoid fitting case.

Algebraic Fitting Method

If the data points $d_1, \ldots, d_{10}$ were on an ellipse, then they would satisfy a quadratic equation

$$d_i^\top A d_i + \beta^\top d_i + c = 0, \quad \text{for } i = 1, \ldots, 10 \quad \text{and}$$
$$\text{for some } A \in \mathbb{R}^{2\times 2},\ A = A^\top,\ A > 0,\ \beta \in \mathbb{R}^2,\ c \in \mathbb{R}.$$

The symmetric matrix A, the vector β, and the scalar c are *parameters* of the ellipse (which from the modeling point of view is the quadratic static model). As in the line fitting example, generically the data does not lie on an ellipse.

The latency approach leads to what is called the *algebraic fitting method*. It looks for equation errors $e_1, \ldots, e_{10}$ and parameters $\hat{A} \in \mathbb{R}^{2\times 2}$, $\hat{\beta} \in \mathbb{R}^2$, $\hat{c} \in \mathbb{R}$, such that

$$d_i^\top \hat{A} d_i + \hat{\beta}^\top d_i + \hat{c} = e_i, \qquad \text{for } i = 1, \ldots, 10.$$

Clearly, for any $\hat{A} \in \mathbb{R}^{2\times 2}$, $\hat{\beta} \in \mathbb{R}^2$, $\hat{c} \in \mathbb{R}$, i.e., for any chosen second order surface (in particular an ellipse), there is a corresponding equation error $e := \mathrm{col}(e_1, \ldots, e_{10})$ defined by the above equation. Therefore, the latency term e again allows us to resolve the data–model discrepancy. The 2-norm of e is by definition the latency of the surface corresponding to the parameters $\hat{A}$, $\hat{\beta}$, $\hat{c}$ and the data. The left plot of Figure 1.3 shows the latency optimal ellipse for the data in the example.

Geometric Fitting Method

The misfit approach leads to what is called the geometric fitting method. In this case, the aim is to find the minimal corrections in a Frobenius norm sense $\Delta d_1, \ldots, \Delta d_{10}$, such that the corrected data $\hat{d}_1, \ldots, \hat{d}_{10}$ lies on a second order surface; i.e., there exist $\hat{A} \in \mathbb{R}^{2\times 2}$, $\hat{\beta} \in \mathbb{R}^2$, $\hat{c} \in \mathbb{R}$, for which

$$\hat{d}_i^\top \hat{A} \hat{d}_i + \hat{\beta}^\top \hat{d}_i + \hat{c} = 0, \qquad \text{for } i = 1, \ldots, 10.$$

For a given ellipse, the Frobenius norm of the smallest data corrections that make the data exact for that ellipse is by definition the misfit between the ellipse and the data. The norm of the correction Δd_i is the orthogonal distance from the data point d_i to the ellipse. The misfit optimal ellipse is shown in the right plot of Figure 1.3.

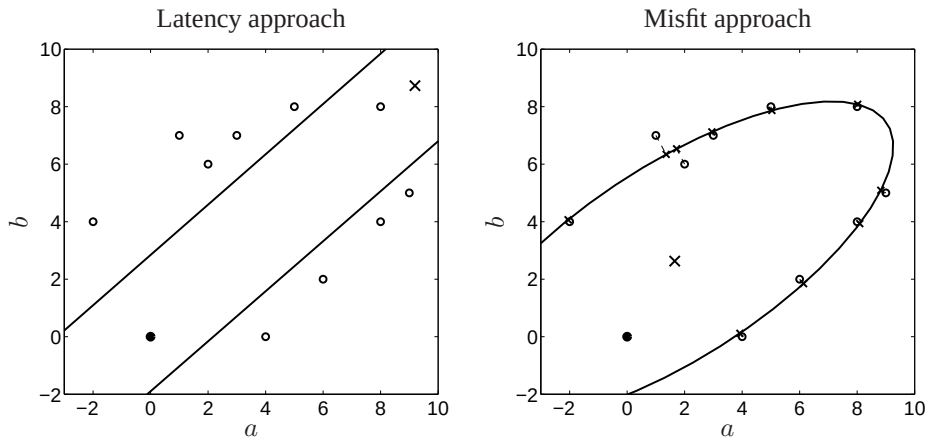


Figure 1.3. *Optimal fitting ellipses (—) and data corrections (- - -) for the misfit approach. (×—centers of the ellipses.)*

Linear Time-Invariant System Identification

Next, we consider fitting the data by a dynamic model. In this case the data $\mathscr{D}$ is viewed as a vector time series. Figure 1.4 shows the data in the plane (as in the static case) but with numbers indicating the data point index, viewed now as a time index. The dynamics is expressed in a motion (see the arrow lines in the figure) starting from data point 1, going to data point 2, then to data point 3 (for the same period of time), and so on, until the last data point 10.

The considered model class consists of LTI systems with one input and one time lag.

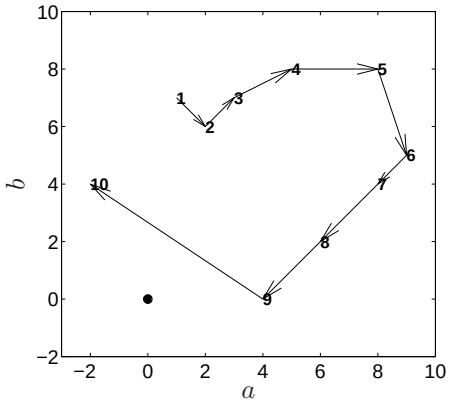


Figure 1.4. *The data $\mathscr{D}$ viewed as a time series. The numbers show the data point index, or, equivalently, the time index. The arrow lines show the dynamics of the model: motion through the consecutive data points.*

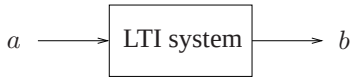


Figure 1.5. *Signal processor interpretation of an LTI system.*

Models of this type admit a difference equation representation

$$R_0 d_i + R_1 d_{i+1} = 0, \quad \text{where } R_0, R_1 \in \mathbb{R}^{1\times 2}.$$

The vectors R_0 and R_1 are parameters of the model.

Let $R_i =: \begin{bmatrix} Q_i & -P_i \end{bmatrix}$, $i = 1, 2$, and suppose that $P_1 \neq 0$. Then the variable a acts as an input (free variable) and the variable b acts as an output (bound variable). This gives an input/output separation of the variables

$$Q_0 a_i + Q_1 a_{i+1} = P_0 b_i + P_1 b_{i+1}$$

and corresponds to the classical notion of a dynamical system as a signal processor, accepting inputs and producing outputs; see Figure 1.5.

Autoregressive Moving Average Exogenous and Output Error Identification

If the data $\mathscr{D}$ were an exact trajectory of an LTI model in the considered model class, then there would exist vectors $R_0, R_1 \in \mathbb{R}^{1\times 2}$ (parameters of the model) and $d_{11} \in \mathbb{R}^2$ (initial condition), such that

$$R_0 d_i + R_1 d_{i+1} = 0, \quad \text{for } i = 1, \ldots, 10.$$

However, generically this is not the case, so that an approximation is needed. The latency approach modifies the model equation by adding an equation error e

$$\hat{R}_0 d_i + \hat{R}_1 d_{i+1} = e_i, \quad \text{for } i = 1, \ldots, 10.$$

The residual e can be considered to be an unobserved (latent) variable; see Figure 1.6.

From this point of view it is natural to further modify the system equation by allowing for a time lag in the latent variable (as in the other variables)

$$Q_0 a_i + Q_1 a_{i+1} - P_0 b_i - P_1 b_{i+1} = M_0 e_i + M_1 e_{i+1}. \qquad (*)$$

The real numbers M_0 and M_1 are additional parameters of the model.

An interesting special case of the latent variable equation $(*)$, called *output error identification* model, is obtained when $M_0 = P_0$ and $M_1 = P_1$. Then the latent variable e

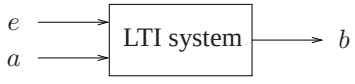


Figure 1.6. *LTI system with a latent variable e.*

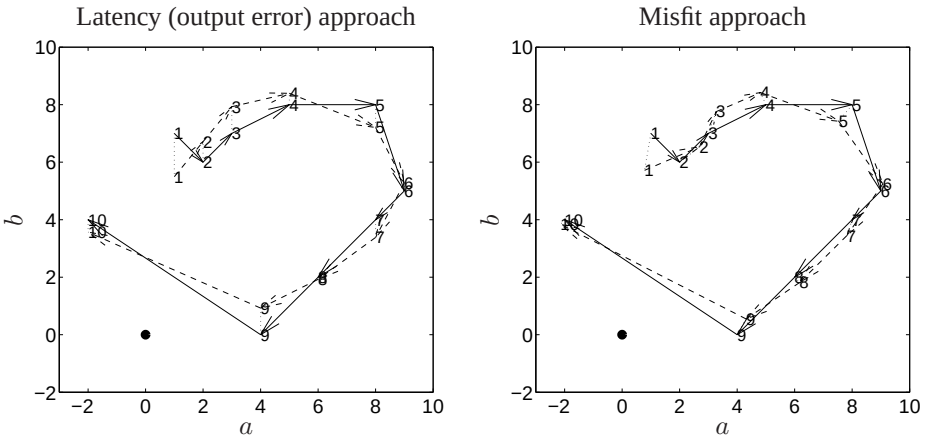


Figure 1.7. *Data $\mathscr{D}$ (—), optimal fitting trajectory $\hat{\mathscr{D}}_{\text{oe}}$ (---), and data corrections ($\cdots$).*

acts like a correction on the output. The input, however, is not corrected, so that the corrected data by the output error model is

$$\hat{a}_{\text{oe},i} := a_i, \quad \text{and} \quad \hat{b}_{\text{oe},i} := b_i + e_i, \quad \text{for } i = 1, \ldots, 10.$$

By construction the corrected time series $\hat{d}_{\text{oe}} := \text{col}(\hat{a}_{\text{oe}}, \hat{b}_{\text{oe}})$ satisfies the equation

$$Q_0 \hat{a}_{\text{oe},i} + Q_1 \hat{a}_{\text{oe},i+1} = -P_0 \hat{b}_{\text{oe},i} - P_1 \hat{b}_{\text{oe},i+1}.$$

The optimal output error fitting data $\hat{\mathscr{D}}_{\text{oe}} := \{ \hat{d}_{\text{oe},1}, \ldots, \hat{d}_{\text{oe},10} \}$ over the parameters P_i, Q_i (i.e., over all models with one input and one time lag) is visualized in the left plot of Figure 1.7.

Note the similarity between the output error identification method and the classical LS method. Indeed,

> output error identification can be viewed as a "dynamic LS method".

Errors-in-Variables Identification

The misfit approach leads to what is called the global total least squares method. It is a generalization of the TLS method for approximate modeling by an LTI dynamic model. In this case the given time series is modified by the smallest corrections $\Delta d_1, \ldots, \Delta d_{10}$, in a Frobenius norm sense, such that the corrected time series $\hat{d}_i := d_i + \Delta d_i$, $i = 1, \ldots, 10$ is a trajectory of a model in the model class. Therefore, there are parameters of the model $\hat{R}_0, \hat{R}_1 \in \mathbb{R}^{1\times 2}$ and an initial condition $\hat{d}_{11} \in \mathbb{R}^2$, such that

$$\hat{R}_0 \hat{d}_i + \hat{R}_1 \hat{d}_{i+1} = 0, \qquad \text{for } i = 1, \ldots, 10.$$

The right plot of Figure 1.7 shows the misfit optimal fitting data $\hat{\mathscr{D}}$.

1.3 Classical vs. Behavioral and Stochastic vs. Deterministic Modeling

In what sense can the examples of Section 1.2 be viewed as *data modeling*? In other words, what are the *models* in these examples? In the line fitting case, clearly the model is a line. The data is a collection of points in $\mathbb{R}^2$ and the model is a subset of the same space. In the ellipsoid fitting case, the model is an ellipse, which is again a subset of the data space $\mathbb{R}^2$. A line and an ellipse are static models in the sense that they describe the data points without relations among them. In particular, their order is not important for static modeling.

In the system identification examples, the data set $\mathscr{D}$ is viewed as an entity—a finite vector time series. A dynamical model is again a subset, however, consisting of time series. The geometric interpretation of the dynamic models is more subtle than the one of the static models due to the time series structure of the data space. In the static examples of Section 1.2 the data space is 2-dimensional while in the dynamic examples it is 20-dimensional.

> The point of view of the model as a subset of the data space is inspired by the behavioral approach to system theory.

This point of view has a number of important advantages over the classical point of view of a model as a set of equations. In the behavioral approach an equation is a *representation* of its solution set (which is the model itself). A model has infinitely many representations, so that a particular representation is not an intrinsic characteristic of the model.

Consider, for example, a linear static model $\mathscr{B}$ that is a one-dimensional subspace of $\mathbb{R}^2$. Perhaps the most commonly used way to define $\mathscr{B}$ is via the representation

$$\mathscr{B} = \{\, d := \mathrm{col}(a, b) \mid ax = b \,\}.$$

However, the same model can be represented as the kernel of a 1×2 matrix R, i.e.,

$$\mathscr{B} = \ker(R) := \{\, d \mid Rd = 0 \,\},$$

or as the image of a 2×1 matrix P, i.e.,

$$\mathscr{B} = \mathrm{col\,span}(P) := \{\, d \mid \text{there is } l, \text{such that } d = Pl \,\}.$$

Moreover, the parameters R and P of a kernel and an image representation are not unique. Which particular representation one is going to choose is a matter of convenience. Therefore, an approximate modeling problem formulation in terms of a particular representation is unnecessarily restrictive. Note that the representation $ax = b$ does not exist for all one-dimensional subspaces of $\mathbb{R}^2$. (Consider the vertical line $\mathrm{col\,span}(\mathrm{col}(0, 1))$.)

Another feature in which the presentation in this book differs from most of the existing literature on approximate modeling is the use of deterministic instead of stochastic assumptions and techniques. It is well known that the classical LS method has deterministic as well as stochastic interpretations. The same duality exists (and is very much part of the literature) for other modeling methods. For example, the TLS method, introduced by Golub and Van Loan [GV80] in the numerical linear algebra literature as a tool for approximate solution of an overdetermined linear system of equations, can be viewed as a consistent estimator in the linear EIV model, under suitable statistical assumptions.

> One and the same modeling method can be derived and "justified" in deterministic as well as stochastic setting.

Both approaches are useful and contribute to a deeper understanding of the methods. In our opinion, however, the stochastic paradigm is overused and sometimes misused. Often the conceptual simplicity of the deterministic approach is an important advantage (certainly so from the pedagogical point of view). Unlike the stochastic approach, the deterministic one makes no unverifiable assumptions about the data generating phenomenon. As a consequence, however, fewer properties can be proven in the deterministic setting than in the stochastic one.

Most of the problems in the book are posed in the behavioral setting and use the misfit approach. This new paradigm and related theory are still under development and are currently far less mature than the classical stochastic latency oriented approach. Our aim is to popularize and stimulate interest in the presented alternative approaches for approximate modeling.

1.4 Chapter-by-Chapter Overview*

The introduction in Sections 1.1–1.3 is informal. Chapter 2 gives an in-depth introduction to the particular problems considered in the book. The main themes—exact and misfit optimal approximate modeling—are introduced in Sections 2.1 and 2.2 . Then we elaborate on the model representation issue. An important observation is that the misfit optimal model is independent of the particular representation chosen, but the latency optimal model in general depends on the type of representation. In Sections 2.4–2.6 we specify the misfit approximation problem for the linear static, bilinear and quadratic static, and LTI dynamic model classes. An approximate modeling problem, called structured total least squares (STLS), which can treat various static and dynamic linear misfit approximation problems, is introduced in Section 2.7. Chapter 2 ends with an overview of the adopted solution methods.

The book is divided into two parts:

Part I deals with static models and

Part II deals with dynamic models.

Optional sections (like this section) are marked with $*$. The material in the optimal sections is more technical and is not essential for the understanding of what follows.

Chapter 3: Weighted Total Least Squares The weighted total least squares (WTLS) problem is a misfit based approximate modeling problem for linear static models. The WTLS misfit is defined as a weighted projection of the data $\mathscr{D}$ on a model $\mathscr{B}$. The choice of the weight matrices for the projection is discussed in Section 3.1, where two possibilities are described. The first one leads to a problem, called relative error total least squares, and the second one leads to the problem of maximum likelihood estimation in the EIV model.

The kernel, image, and input/output representations of a linear static model are presented in Section 3.2. We believe that these representations and the links among them are prerequisites for the proper understanding of all static approximation problems.

In Section 3.3, we solve the TLS and the generalized TLS problems, which are special cases of the WTLS problem. They are treated separately because a closed form solution in terms of the singular value decomposition (SVD) exists. The ingredients for the solution are

1. the equivalence between data consistent with a linear static model and a low-rank matrix, and
2. the Eckart–Young–Mirsky low-rank approximation lemma, which shows how an optimal (in the sense of the Frobenius norm) low-rank approximation of a given matrix can be computed via SVD.

The solution of the TLS problem is given in terms of the SVD of the data matrix and the solution of the GTLS problem is given in a similar way in terms of the SVD of a modified data matrix.

The WTLS problem is a double minimization problem. In Section 3.4, we solve in closed form the inner minimization, which is the misfit computation subproblem. The results are given in terms of kernel and image representations, which lead to, respectively, least norm and least squares problems.

In the optional Section 3.5, we consider the remaining subproblem—minimization with respect to the model parameters. It is a nonconvex optimization problem that in general has no closed form solution. For this reason, numerical optimization methods are employed. We present three heuristic algorithms: alternating least squares, an algorithm due to Premoli and Rastello, and an algorithm based on standard local optimization methods.

Chapter 4: Structured Total Least Squares The STLS problem is a flexible tool that covers various misfit minimization problems for linear models. We review its origin and development in Section 4.1. There are numerous (equivalent) formulations that differ in the representation of the model and the optimization algorithm used for the numerical solution of the problem. The proposed methods, however, have high computational complexity and/or assume a special type of structure that limit their applicability in real-life applications. Our motivation is to overcome as much as possible these limitations and propose a practically useful solution.

In Section 4.2, we define the considered STLS problem. The data matrix is partitioned into blocks and each of the blocks is block-Toeplitz/Hankel structured, unstructured, or exact. As shown in Section 4.6, this formulation is general enough to cover many structured approximation problems and at the same time allows efficient solution methods. Our solution approach is based on the derivation of a closed form expression for an equivalent unconstrained problem, in which a large number of decision variables are eliminated. This step corresponds to the misfit computation in the misfit approximation problems.

The remaining problem is a nonlinear least squares problem and is solved numerically via local optimization methods. The cost function and its first derivative evaluation, however, are performed efficiently by exploiting the structure in the problem. In the optional Section 4.3, we prove that as a consequence of the structure in the data matrix, the equivalent optimization problem has block-Toeplitz and block-banded structure. In Section 4.4, a stochastic interpretation of the Toeplitz and banded structure of the equivalent problem is given.

A numerical algorithm for solving the STLS problem is described in Section 4.5. It is implemented in the software package described in Appendix B.2. In Section 4.6, we show simulation examples that demonstrate the performance of the proposed STLS solution method on standard approximation problems. The performance of the STLS package is compared with that of alternative methods on LS, TLS, mixed LS-TLS, Hankel low-rank approximation, deconvolution, and system identification problems.

Chapter 5: Bilinear Errors-in-Variables Model In Chapter 5, we consider approximations by a bilinear model. The presentation is motivated from the statistical point of view of deriving a consistent estimator for the parameters of the true model in the EIV setup. The misfit approach yields an inconsistent estimator in this case, so that an alternative approach based on the adjustment of the LS approximation is adapted.

An adjusted least squares (ALS) estimator, which is in principle a latency oriented method, is derived in Section 5.2, and its statistical properties are stated in the optional Section 5.3. Under suitable conditions, it is strongly consistent and asymptotically normal. In Section 5.4, we show simulation examples illustrating the consistency of the ALS estimator.

In Section 5.5, we consider a different approximation problem by a static bilinear model. It is motivated from an application in computer vision, called fundamental matrix estimation. The approach is closely related to the one of Section 5.2.

Chapter 6: Ellipsoid Fitting The ALS approach of Chapter 5 is further applied for approximation by a quadratic model. The motivation for considering the quadratic model is the ellipsoid fitting problem. In Section 6.1, we introduce the ellipsoid fitting problem and review the literature. As in Chapter 5, we consider the EIV model and note that the misfit approach, although intuitively attractive and geometrically meaningful, yields a statistically inconsistent estimator. This motivates the application of the ALS approach.

In Section 6.2, we define the quadratic EIV model. The LS and the ALS estimators are presented, respectively, in Sections 6.3 and 6.4. The ALS estimator is derived from the LS estimator by properly adjusting its cost function. Under suitable conditions the ALS estimator yields a consistent estimate of the parameters of the true model. In the optional Section 6.6, we present an algorithm for the computation of the ALS estimator. Simulation examples comparing the ALS and alternative estimators on benchmark problems from the literature are shown in Section 6.7.

Chapter 7: Introduction to Dynamical Models Chapter 7 is an introduction to Part II of the book. The main emphasis is on the representation of an LTI system. Different representations are suitable for different problems, so that familiarity with a large number of alternative representations is instrumental for solving the problems. First, we give a high level characterization of an LTI system: its behavior is linear, shift-invariant, and closed in the topology of pointwise convergence. Then we consider a kernel representation of an LTI system, i.e., difference equation representation. However, we use polynomial matrix notation. A sequence of equivalence operations on the difference equations is represented by premultiplication of a polynomial operator by a unimodular matrix. Also, certain properties of the representation such as minimality of the number of equations is translated to equivalent properties of polynomial matrices. Special forms of the polynomial matrix display important

invariants of the system such as the number of inputs and the minimal state dimension.

We discuss the question of what inputs and outputs of the system are and show representations that display the input/output structure. The classical input/state/output representation of an LTI system is obtained by introducing, in addition, latent variables with special properties. The controllability property of a system is introduced and a test for it is shown in terms of a kernel representation. Any system allows a decomposition into an autonomous subsystem and a controllable subsystem. A controllable system can be represented by a transfer function or a convolution operator or as the image of a polynomial operator. Finally, the latent variable and driving input state space representation are presented.

The introduction of the various system representations is summarized by a representation theorem that states their equivalence. The chapter continues with the related question of parameterizing a trajectory of the system. The most convenient representation for this purpose is the input/state/output representation that displays explicitly both the input and the initial conditions.

Chapter 8: Exact Identification The simplest and most basic system identification problem is considered first: given a trajectory of an LTI system, find a representation of that system. The data is an exact trajectory and the system has to be recovered exactly. The problem can be viewed as a representation question: pass from a sufficiently informative trajectory to a desirable representation of the system.

We answer the question of when a trajectory is sufficiently informative in order to allow exact identification. This key result is repeatedly used and is called the *fundamental lemma*.

The exact identification problem is closely related to the construction of what is called the most powerful unfalsified model (MPUM). Under the condition of the fundamental lemma, the MPUM is equal to the data generating system, so that one can look for algorithms that obtain specific representations of that system from the data. We review algorithms for passing from a trajectory to kernel, convolution, and input/state/output representations. Relationships to classical deterministic subspace identification algorithms are given.

Our results show alternative system theoretic derivations of the classical subspace identification methods. In particular, the orthogonal and oblique projections from the MOESP and N4SID subspace identification methods are interpreted. It is shown that the orthogonal projection computes free responses and the oblique projection computes sequential free responses, i.e., free responses of which the initial conditions form a state sequence. From this perspective, we answer the long-standing question in subspace identification of how to partition the data into "past" and "future". The "past" is used to set the initial condition for a response computed in the "future".

The system theoretic interpretation of the orthogonal and oblique projections reveals their inefficiency for the purpose of exact identification. We present alternative algorithms that correct this deficiency and show simulation results that illustrate the performance of various algorithms for exact identification.

Chapter 9: Balanced Model Identification Balancing is often used as a tool for model reduction. In Chapter 9, we consider algorithms for obtaining a balanced representation of the MPUM directly from data. This is a special exact identification problem.

Two algorithms were previously proposed in the setting of the deterministic subspace identification methods. We analyze their similarity and differences and show that they fall under the same basic outline, where the impulse response and sequential zero input responses are obtained from data. We propose alternative algorithms that need weaker assumptions on the available data. In addition, the proposed algorithms are computationally more efficient since the block-Hankel structure of certain matrices appearing in the computations is explicitly taken into account.

Chapter 10: Errors-in-Variables Smoothing and Filtering The approximate system identification problem, based on the misfit approach, has as a subproblem the computation of the closest trajectory in the behavior of a given model to a given time series. This is a smoothing problem whose solution is available in closed form. However, efficient recursive algorithms are of interest. Moreover, the filtering problem, in which the approximation is performed in real time, is of independent interest.

Deterministic smoothing and filtering in the behavioral setting are closely related to smoothing and filtering in the EIV setting. We solve the latter problems for systems given in an input/state/output representation. The optimal filter is shown to be equivalent to the classical Kalman filter derived for a related stochastic system. The result shows that smoothing and filtering in the EIV setting are not fundamentally different from the classical smoothing and Kalman filtering for systems driven by white noise input and with measurement noise on the output.

Chapter 11: Approximate System Identification The approximate identification problem, treated in Chapter 11, is the global total least squares (GlTLS) problem, i.e., the misfit minimization problem for an LTI model class of bounded complexity. This problem is a natural generalization of the exact identification problem of Chapter 8 for the case when the MPUM does not exist.

Because of the close connection with the STLS problem and because in Part I of the book numerical solution methods are developed for the STLS problem, our goal in this chapter is to link the GlTLS problem to the STLS problem. This is done in Section 11.2, where conditions under which the equivalence holds are given. The most restrictive of these conditions is the condition on the order of the identified system: it should be a multiple of the number of outputs. Another condition is that the optimal approximation allows a fixed input/output partition, which is conjectured to hold generically.

In Section 11.3, we discuss several extensions of the GlTLS problem: treating exact and latent variables and using multiple time series for the approximation. In Section 11.4, the problem is specialized to what is called the approximate realization problem, where the given data is considered to be a perturbed version of an impulse response, the related problem of autonomous system identification, and the problem of finite time ℓ_2 model reduction.

In Section 11.5, we present simulation examples with data sets from the data base for system identification DAISY. The results show that the proposed solution method is effective and efficient for a variety of identification problems.

Chapter 2

Approximate Modeling via Misfit Minimization

This chapter gives a more in-depth introduction to the problems considered in the book: data fitting by linear, bilinear, and quadratic *static* as well as linear time-invariant *dynamic* models. In the linear case, the discrepancy between the data and the approximate model is measured by the *misfit*. In the nonlinear case, the approximation is defined as a quadratically constrained least squares problem, called adjusted least squares.

The main notions are data, model, and misfit. Optimal exact modeling aims to fit the data and as little else as possible by a model in a given model class. The model obtained is called the most powerful unfalsified model (MPUM). The MPUM may not exist in a specified model class. In this case we accept a falsified model that fits optimally the data according to the misfit approximation criterion. The total least squares (TLS) problem and its variations, generalized total least squares (GTLS) and weighted total least squares (WTLS), are special cases of the general misfit minimization problem for the linear static model. In the dynamic case, the misfit minimization problem is called the global total least squares (GlTLS) problem.

An overview of the solution methods that are used is given. The misfit minimization problem has a quadratic cost function and a bilinear equality constraint. This is a nonconvex optimization problem, for whose solution we employ local optimization methods. The bilinear structure of the constraint, however, allows us to solve the optimization problem partially. This turns the constrained optimization problem into an equivalent nonlinear least squares problem. The adjusted least squares method, on the other hand, leads to a generalized eigenvalue problem.

2.1 Data, Model, Model Class, and Exact Modeling

Consider a phenomenon to be described by a mathematical model. Certain variables, related to the phenomenon, are observable, and the observed data from one or more experiments is recorded. Using prior knowledge about the phenomenon, a model class of candidate models is selected. Then the model is chosen from the model class that in a certain specified sense most adequately describes the available data.

We now formalize this modeling procedure. Call a data point recorded from an experiment an *outcome* and let $\mathscr{U}$ be the universum of possible outcomes from an experiment. The observed data $\mathscr{D}$ is collected from experiments, so that it is a subset $\mathscr{D} \subset \mathscr{U}$ of the universum.

Following the behavioral approach to system theory [PW98],

> we define a model $\mathscr{B}$ to be a set of outcomes, i.e., $\mathscr{B} \subseteq \mathscr{U}$.

Actually, for the purpose, of modeling this definition is a bit restrictive. Often the outcomes are functions of the to-be-modeled variables, i.e., the variables that we aim to describe by the model. By postulating the model to be a subset of the universum of outcomes, we implicitly assume that the observed variables are the to-be-modeled variables.

If for a particular experiment an observed outcome $d \in \mathscr{U}$ is such that $d \in \mathscr{B}$, then we say that "$\mathscr{B}$ explains d" or "$\mathscr{B}$ is unfalsified by d". In this case the model fits the data *exactly*. If $d \notin \mathscr{B}$, we say that the outcome d *falsifies* $\mathscr{B}$. In this case the model may fit the data only *approximately*.

Let $\mathscr{B}_1$ and $\mathscr{B}_2$ be two models such that $\mathscr{B}_1 \subseteq \mathscr{B}_2$. We say that $\mathscr{B}_1$ is simpler (less complex) than $\mathscr{B}_2$. "Simpler" means allowing fewer outcomes. If $\mathscr{U}$ is a vector space and we consider models that are (finite dimensional) subspaces, "simpler" means a lower dimensional subspace. Note that our notion of simplicity does not refer to a simplicity of a representation of $\mathscr{B}$.

Simpler models are to be preferred over more complicated ones. Consider the two statements $d \in \mathscr{B}_1$ and $d \in \mathscr{B}_2$ with $\mathscr{B}_1 \subseteq \mathscr{B}_2$. The first one is stronger and therefore more useful than the second one. In this sense, $\mathscr{B}_1$ is a more *powerful* model than $\mathscr{B}_2$.

On the other hand, the a priori probability that a given outcome $d \in \mathscr{U}$ falsifies the model $\mathscr{B}_1$ is higher than it is for the model $\mathscr{B}_2$. This shows a trade-off in choosing an exact model. The extreme cases are the model $\mathscr{U}$ that explains every outcome but "says" nothing about an outcome and the model $\{ d \}$ that explains only one outcome but completely describes the outcome.

Next, we introduce the notion of a model class. The set of all subsets of $\mathscr{U}$ is denoted by $2^{\mathscr{U}}$. In our setting, $2^{\mathscr{U}}$ is the set of all models. A model class $\mathscr{M} \subseteq 2^{\mathscr{U}}$ is a set of candidate models for a solution of the modeling problem. In theory, an arbitrary model class can be chosen. In practice, however, the choice of the model class is crucial in order to be able to obtain a meaningful solution. The choice of the model class is dictated by the prior knowledge about the modeled phenomenon and by the difficulty of solving the resulting approximation problem. We aim at general model classes that still lead to tractable problems.

> The most reasonable exact modeling problem is to find the model $\mathscr{B}_{\text{mpum}} \in \mathscr{M}$ that explains the data $\mathscr{D}$ and as little else as possible. The model $\mathscr{B}_{\text{mpum}}$ is called the most powerful unfalsified model (MPUM) for the data $\mathscr{D}$ in the model class $\mathscr{M}$.

The MPUM need not exist, but if it exists, it is unique.

Suppose that the data $\mathscr{D}$ is actually generated by a model $\mathscr{B} \in \mathscr{M}$; i.e., $d \in \mathscr{B}$ for all $d \in \mathscr{D}$. A fundamental question that we address is, Under what conditions can the unknown model $\mathscr{B}$ be recovered exactly from the data? Without any other a priori knowledge (apart from the given data $\mathscr{D}$ and model class $\mathscr{M}$), this question is equivalent to the question, Under what conditions does $\mathscr{B}_{\text{mpum}} = \mathscr{B}$?

2.2 Misfit and Approximate Modeling

The MPUM may not exist for a given data and model class. In fact, for "rough data", e.g., data collected from a real-life experiment, if the MPUM exists, it tends to be $\mathscr{B}_{\text{mpum}} = \mathscr{U}$. Therefore, the exact modeling problem has either no solution or a trivial one. Although the concept of the MPUM is an important theoretical tool, the computation of the MPUM is not a practical modeling algorithm. What enables the modeling procedure to "work with rough data" is approximation.

In an approximate modeling problem, the model is required to explain the data only approximately; i.e., it could be falsified by the data. Next, we define an approximation criterion called misfit. The misfit between an outcome $d \in \mathscr{U}$ and a model $\mathscr{B} \subseteq \mathscr{U}$ is a measure for the distance from the point d to the set $\mathscr{B}$. As usual, this is defined as the distance from d to the point $\hat{d}^*$ in $\mathscr{B}$ that is closest to d. (The "hat" notation, as in $\hat{d}$, means "an approximation of".) For example, if $\mathscr{U}$ is an inner product space and $\mathscr{B}$ is a closed subspace, then $\hat{d}^*$ is the projection of d on $\mathscr{B}$.

Underlying the definition of the misfit is a distance on $\mathscr{U}$. Let $\mathscr{U}$ be a normed vector space with a norm $\|\cdot\|_{\mathscr{U}}$ and define the distance (induced by the norm $\|\cdot\|_{\mathscr{U}}$) between two outcomes $d, \hat{d} \in \mathscr{U}$ as $\|d - \hat{d}\|_{\mathscr{U}}$.

> The misfit between an outcome d and a model $\mathscr{B}$ (with respect to the norm $\|\cdot\|_{\mathscr{U}}$) is defined as
>
> $$M(d, \mathscr{B}) := \inf_{\hat{d} \in \mathscr{B}} \|d - \hat{d}\|_{\mathscr{U}}.$$
>
> It measures the extent to which the model $\mathscr{B}$ fails to explain the outcome d.

A global minimum point $\hat{d}^*$ is the best (according to the distance measure $\|d - \hat{d}\|_{\mathscr{U}}$) approximation of d in $\mathscr{B}$. Alternatively, $M(d, \mathscr{B})$ is the minimal distance between d and an approximation $\hat{d}$ compatible with the model $\mathscr{B}$.

For data consisting of multiple outcomes $\mathscr{D} = \{ d_1 \ldots, d_N \}$, we choose N norms $\|\cdot\|_i$ in $\mathscr{U}$ and define $M_i(d_i, \mathscr{B})$ to be the misfit with respect to the norm $\|\cdot\|_i$. Then the misfit between the data $\mathscr{D}$ and the model $\mathscr{B}$ is defined as

$$M\big(\{ d_1 \ldots, d_N \}, \mathscr{B}\big) := \big\| \operatorname{col}\big(M_1(d_1, \mathscr{B}), \ldots, M_N(d_N, \mathscr{B})\big) \big\|. \tag{M}$$

In the context of exact modeling, there is a fundamental trade-off between the power and complexity of the model. A similar issue occurs in approximate modeling: an arbitrary small misfit can be achieved by selecting a complicated model. The trade-off now is between the worst achievable misfit and the complexity of the model. The issue can be resolved, for example, by fixing a maximal allowed complexity. With a constraint on the complexity (incorporated in the definition of the model class), the aim is to minimize the misfit.

> For a chosen misfit M and model class $\mathscr{M}$, the misfit minimization problem aims to find a model $\hat{\mathscr{B}}$ in the model class that is least falsified by the data, i.e.,
>
> $$\hat{\mathscr{B}} := \arg \min_{\mathscr{B} \in \mathscr{M}} M(\mathscr{D}, \mathscr{B}). \tag{APR}$$

The approximation problem (APR) can be interpreted in terms of the MPUM as follows:

> Modify the data as little as possible, so that the MPUM $\hat{\mathscr{B}}$ for the modified data $\hat{\mathscr{D}}$ is in a specified model class $\mathscr{M}$.

Next, we describe the important issue of a representation of a model and specify misfit minimization problems for particular model classes in terms of particular representations.

2.3 Model Representation and Parameterization

The definition of the model as a set of outcomes is general and powerful. It allows us to consider linear and nonlinear, and static and dynamic, stationary and nonstationary models in the same conceptual setting. For analysis, however, it is too abstract. It is often more convenient to work with particular representations of the model in terms of equations that capture the essential properties of the model.

> For a given model $\mathscr{B} \subseteq \mathscr{U}$, an equation $f(d) = 0$ with solution set equal to $\mathscr{B}$, i.e.,
>
> $$\mathscr{B} = \{\, d \in \mathscr{U} \mid f(d) = 0 \,\}, \qquad \text{(REPR)}$$
>
> is called a representation of $\mathscr{B}$.

The function $f : \mathscr{U} \to \mathbb{R}^g$ that describes the model $\mathscr{B}$ is defined in terms of parameters. Consider, for example, a real vector space $\mathscr{U} = \mathbb{R}^{n_\theta}$ and a linear function $f_\theta(d) = \theta^\top d$. The vector $\theta \in \mathbb{R}^{n_\theta}$ parameterizes f_θ and via (REPR) also $\mathscr{B}$.

Let $f_\theta(d) = 0$ be a representation with a parameter vector $\theta \in \mathbb{R}^{n_\theta}$. Different values of θ result in different models $\mathscr{B}(\theta)$. We can view the representation by f_θ as a mapping $\mathscr{B} : \mathbb{R}^{n_\theta} \to 2^{\mathscr{U}}$ from the parameter space to the set of models. A given set of parameters $\Theta \subseteq \mathbb{R}^{n_\theta}$ corresponds to the set of models $\mathscr{B}(\Theta) \subseteq 2^{\mathscr{U}}$, i.e., to a model class. Assume that for a given representation f_θ and a given model class $\mathscr{M}$, there is a corresponding parameter set $\Theta \subseteq \mathbb{R}^{n_\theta}$, such that $\mathscr{M} = \mathscr{B}(\Theta)$.

> In terms of the representation f_θ, the misfit minimization problem (APR) becomes the following parameter optimization problem:
>
> $$\hat{\theta} := \arg\min_{\theta \in \Theta} M\big(\mathscr{D}, \mathscr{B}(\theta)\big). \qquad \text{(APR')}$$

The numerical implementation of the algorithms depend on the particular representation chosen. From the point of view of the abstract formulation (APR), however, the representation issue is not essential. This is in contrast with approximation methods that minimize an equation error criterion.

Consider a model $\mathscr{B} \subseteq \mathscr{U}$ with representation (REPR). An outcome $d \in \mathscr{U}$ that is not consistent with the model $\mathscr{B}$ may not satisfy the equation, yielding $e(\theta) := f_\theta(d)$, called *equation error*. The equation error for a given d is a function $e : \mathbb{R}^{n_\theta} \to \mathbb{R}^g$ of the parameter θ and therefore it depends on the model $\mathscr{B}(\theta)$. Since

$$f_\theta(d) = e(\theta) = 0 \quad \Longleftrightarrow \quad d \in \mathscr{B}(\theta),$$

we define "equation misfit" (lack of fit in terms of equations representing the model)

$$M_{\text{eqn}}(d, \theta) := \|f_\theta(d)\|_{\text{eqn}},$$

where $\|\cdot\|_{\text{eqn}}$ is a norm defined in $\mathbb{R}^g$. The equation misfit depends on the representation. In contrast, the "behavioral misfit" M is representation independent.

Note 2.1 (Latency) The equation error e can be viewed as an unobserved, latent variable. From this alternative point of view the equation misfit M_{eqn} is the latency of Chapter 1.

As before, for multiple observed outcomes $\mathscr{D} = \{ d_1 \dots, d_N \}$, we define the equation misfits $M_{\text{eqn},i}(d_i, \theta)$ in terms of the norms $\|\cdot\|_i$ in $\mathbb{R}^g$, and

$$M_{\text{eqn}}\big(\{ d_1 \dots, d_N \}, \theta\big) := \big\| \operatorname{col}\big(M_{\text{eqn},1}(d_1, \theta), \dots, M_{\text{eqn},N}(d_N, \theta)\big) \big\|. \tag{Meqn}$$

Given a model class $\mathscr{M}$, represented in the parameter space by the parameter set Θ, an approximation problem that minimizes the equation misfit is

$$\hat{\theta}_{\text{eqn}} := \arg\min_{\theta \in \Theta} M_{\text{eqn}}(\mathscr{D}, \theta). \tag{APReqn}$$

Solving (APReqn) is often easier than solving (APR'), but the main disadvantage is that the obtained approximation is representation dependent.

2.4 Linear Static Models and Total Least Squares

In the rest of this chapter we consider real valued data. For static problems, the universum set $\mathscr{U}$ is defined to be $\mathbb{R}^{\mathtt{d}}$. The available data $\mathscr{D}$ consists of N outcomes $d_1, \dots, d_N \in \mathbb{R}^{\mathtt{d}}$. We define the data matrix $D := \begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix} \in \mathbb{R}^{\mathtt{d} \times N}$ and the shorthand notation

$$\begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix} \in \mathscr{B} \subseteq \mathscr{U} \quad :\Longleftrightarrow \quad d_i \in \mathscr{B}, \quad \text{for } i = 1, \dots, N.$$

A linear static model $\mathscr{B}$ is a linear subspace of $\mathscr{U} = \mathbb{R}^{\mathtt{d}}$.

Let $\mathtt{m} := \dim(\mathscr{B})$ be the dimension of the model $\mathscr{B}$ and let $\mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$ be the set of all linear static models with $\mathtt{d}$ variables of dimension *at most* $\mathtt{m}$. (The 0 in the notation $\mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$ indicates that the models in this model class are static.) The complexity of the model $\mathscr{B}$ is related to its dimension $\mathtt{m}$: the model is simpler, and therefore more powerful, when it has smaller dimension.

The model $\mathscr{B}$ imposes linear laws $r_i^\top d = 0$, $r_i \in \mathbb{R}^{\mathtt{d}}$ on the outcomes. If $\mathscr{B}$ is defined by g linear laws $r_1, \dots, r_g$, then $d \in \mathscr{B}$ if and only if $Rd = 0$, where $R := \begin{bmatrix} r_1 & \cdots & r_g \end{bmatrix}^\top$. Therefore, $\mathscr{B} = \ker(R)$. The representation of $\mathscr{B} := \ker(R)$ by the equation $Rd = 0$ is called a kernel representation of $\mathscr{B}$. Any linear model $\mathscr{B}$ admits a kernel representation with a parameter R of full row rank.

The MPUM for the data $\mathscr{D}$ in the model class $\mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$ exists if and only if $\operatorname{rank}(D) \leq \mathtt{m}$. If the MPUM exists, it is unique and is given by $\mathscr{B}_{\text{mpum}} = \operatorname{col}\operatorname{span}(D)$. For "rough" data and with $N > \mathtt{m}$, typically $\operatorname{rank}(D) = \mathtt{d}$, so that the MPUM either does not exist or is the trivial model $\mathscr{B}_{\text{mpum}} = \mathbb{R}^{\mathtt{d}}$. In such cases an approximation is needed.

The misfit minimization problem (APR) with model class $\mathscr{M} = \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$ and 2-norms $\|\cdot\|_i$,

$$\hat{\mathscr{B}}_{\text{tls}} = \arg\min_{\mathscr{B}\in\mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}} \left(\min_{\hat{D}\in\mathscr{B}} \|D - \hat{D}\|_{\text{F}} \right), \tag{TLS}$$

is called the total least squares (TLS) problem.

The squared TLS misfit

$$M^2_{\text{tls}}(D, \mathscr{B}) := \min_{\hat{D}\in\mathscr{B}} \|D - \hat{D}\|^2_{\text{F}}$$

is equal to the sum of the squared orthogonal distances from the outcomes $d_1, \ldots, d_N$ to the subspace $\mathscr{B}$. For this reason, the TLS problem is also known as orthogonal regression. In terms of a kernel representation, the TLS problem is equivalent to

$$\hat{R}_{\text{tls}} = \arg\min_{RR^\top = I} \Big(\min_{\hat{D}} \|D - \hat{D}\|_{\text{F}} \quad \text{subject to} \quad R\hat{D} = 0 \Big). \tag{TLS$_R$}$$

Note 2.2 (Equation labels) (TLS) is the abstract, representation-free definition of the TLS problem. Equivalent formulations such as (TLS$_R$) are obtained when a particular representation is chosen. We label frequently used equations with acronyms. Approximation problems, derived from an abstract one, are labeled with the acronym of the abstract problem with the standard variable used for the parameter in a subscript.

The variations of the TLS problem, called generalized total least squares (GTLS) and weighted total least squares (WTLS), are misfit minimization problems (APR) for the model class $\mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$ and weighted norms $\|\cdot\|_i$: in the GTLS case, $\|d\|_i := \|\sqrt{W}d\|$, and in the WTLS case, $\|d\|_i := \|\sqrt{W_i}d\|$, for certain positive definite weight matrices W and W_i. Clearly, the TLS problem is a special case of the GTLS problem and the GTLS problem is a special case of the WTLS problem.

The motivation for the weighted norms in the GTLS and WTLS problems comes from statistics. Assume that the data $\mathscr{D}$ is generated according to the EIV model:

$$D = \bar{D} + \tilde{D}, \qquad \text{where} \quad \bar{D} \in \bar{\mathscr{B}} \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}. \tag{EIV}$$

The model $\bar{\mathscr{B}}$ is called the true model and $\tilde{D} =: \begin{bmatrix} \tilde{d}_1 & \cdots & \tilde{d}_N \end{bmatrix}$ is called the measurement error. The measurement error is modeled statistically as a zero mean random matrix. Assuming in addition that the noise $\tilde{d}_i$ on the ith outcome is independent of the noise on the other outcomes and is normally distributed with covariances $\operatorname{cov}(\tilde{d}_i) = \sigma^2 W_i^{-1}$, the maximum likelihood estimation principle leads to the WTLS problem. Therefore, the weight matrices W_i in the WTLS problem formulation correspond (up to the scaling factor σ^2) to the inverse of the measurement error covariance matrices in the EIV setup.

Note 2.3 (About the notation) We follow the system theoretic notation and terminology that are adopted in the behavioral setting [PW98]. Translation of the ideas and the formulas to other equivalent forms is straightforward. For example, the system of linear equations $AX = B$, which is often the starting point for parameter estimation problems in the numerical linear algebra literature, can be viewed as a special kernel representation

$$AX = B \quad \Longleftrightarrow \quad \begin{bmatrix} X^\top & -I \end{bmatrix} \begin{bmatrix} A^\top \\ B^\top \end{bmatrix} = 0 \quad \Longrightarrow : \quad RD = 0.$$

Therefore, the model represented by the equation $AX = B$ is $\mathscr{B}(X) := \ker(\begin{bmatrix} X^\top & -I \end{bmatrix})$, so that $\mathscr{B}(X) \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$, with $\mathtt{d} = \operatorname{col\,dim}(A) + \operatorname{col\,dim}(B)$ and $\mathtt{m} = \operatorname{col\,dim}(A)$. The representation $\mathscr{B}(X)$ is what is called an input/output representation of a linear static model.

In terms of the representation $AX = B$, the TLS problem with a data matrix $D = \begin{bmatrix} A & B \end{bmatrix}^\top$ is the following parameter optimization problem:

$$\hat{X}_{\text{tls}} = \arg\min_X \left(\min_{\hat{A},\hat{B}} \left\| \begin{bmatrix} A - \hat{A} & B - \hat{B} \end{bmatrix} \right\|_{\text{F}} \quad \text{subject to} \quad \hat{A}X = \hat{B} \right). \tag{TLS$_X$}$$

It is not equivalent to (TLS), but generically $\mathscr{B}(\hat{X}_{\text{tls}}) = \ker(\hat{R}_{\text{tls}})$, where $\hat{R}_{\text{tls}}$ is the solution of (TLS$_R$). The nongeneric cases when $\hat{X}_{\text{tls}}$ does not exist occur as a consequence of the used fixed input/output partitioning of the variables in the representation $\mathscr{B}(X)$.

Note 2.4 (Quadratic cost function) Whenever $\|\cdot\|_i$ are weighted 2-norms, the *squared* misfit M^2 is a quadratic function of the decision variable $\hat{D}$. Squaring the cost function results in an equivalent optimization problem (the optimum point is not changed), so that the misfit minimization problem can equivalently be solved by minimizing the squared misfit.

The equation error minimization problem (APReqn) for the linear static model class $\mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$ with a kernel representation $\mathscr{B} = \ker(R)$ and 2-norms $\|\cdot\|_i$ is the quadratically constrained least squares problem

$$\hat{R}_{\text{ls}} = \arg \min_{RR^\top = I} \|RD\|_{\text{F}}, \tag{LS$_R$}$$

which happens to be equivalent to the TLS problem.

The classical least squares problem

$$\hat{X}_{\text{ls}} = \arg\min_X \left(\min_{\hat{B}} \|B - \hat{B}\|_{\text{F}} \quad \text{subject to} \quad AX = \hat{B} \right) \tag{LS$_X$}$$

is an equation error minimization problem (APReqn) for the representation $AX = B$ and for 2-norms $\|\cdot\|_i$. In general, $\mathscr{B}(\hat{X}_{\text{ls}}) \neq \ker(\hat{R}_{\text{ls}})$, where $\hat{R}_{\text{ls}}$ is the solution of (LS$_R$). It is well known that the solution of (LS$_X$) can be computed in a finite number of operations by solving the system of normal equations. In contrast, the solution of (LS$_R$) is given in terms of the eigenvalue decomposition of $DD^\top$ (or the singular value decomposition of D), of which the computation theoretically requires an infinite number of operations.

2.5 Nonlinear Static Models and Ellipsoid Fitting

An outcome $d \in \mathscr{U} = \mathbb{R}^{\mathtt{d}}$, consistent with a linear static model, satisfies linear relations $Rd = 0$. An outcome $d \in \mathscr{U} = \mathbb{R}^{\mathtt{d}}$, consistent with a nonlinear static model, satisfies nonlinear relations $f(d) = 0$, where $f : \mathbb{R}^{\mathtt{d}} \to \mathbb{R}^g$. We consider nonlinear models with representations that are defined by a single bilinear or quadratic function.

The function $f : \mathbb{R}^{\mathtt{d}} \to \mathbb{R}$ is bilinear if $f(d) = d_1^\top X d_2 - d_3$, for all $d \in \mathbb{R}^{\mathtt{d}}$ and for a certain $X \in \mathbb{R}^{\mathtt{d}_1 \times \mathtt{d}_2}$, where $d =: \operatorname{col}(d_1, d_2, d_3)$ (with $d_1 \in \mathbb{R}^{\mathtt{d}_1}$, $d_2 \in \mathbb{R}^{\mathtt{d}_2}$, and $d_3 \in \mathbb{R}$). For given $\mathtt{d}_1$ and $\mathtt{d}_2$, such that $\mathtt{d} = \mathtt{d}_1 + \mathtt{d}_2 + 1$, a bilinear model with a parameter $X \in \mathbb{R}^{\mathtt{d}_1 \times \mathtt{d}_2}$ is defined as follows:

$$\mathscr{B}_{\text{bln}}(X) := \left\{ \operatorname{col}(d_1, d_2, d_3) \in \mathbb{R}^{\mathtt{d}} \mid d_1^\top X d_2 = d_3 \right\}; \tag{BLN}$$

i.e., a bilinear model is a nonlinear model that allows the representation $f(d) = 0$, with f a bilinear function. Let $\mathscr{M}_{\text{bln}}$ be the set of all bilinear models of the form (BLN),

$$\mathscr{M}_{\text{bln}} := \left\{ \mathscr{B}_{\text{bln}}(X) \mid X \in \mathbb{R}^{\mathtt{d}_1 \times \mathtt{d}_2} \right\}.$$

In terms of the parameterization (BLN), the misfit minimization problem (APR) for the bilinear model class $\mathscr{M}_{\text{bln}}$ with 2-norms $\|\cdot\|_i$ is

$$\min_X \Big(\underbrace{\min_{\hat{D}} \|D - \hat{D}\|_{\text{F}} \quad \text{subject to} \quad \hat{d}_{i,1}^\top X \hat{d}_{i,2} = \hat{d}_{i,3}, \quad \text{for } i = 1, \ldots, N}_{M(D, \mathscr{B}_{\text{bln}}(X))} \Big). \tag{BLN TLS}$$

The function $f : \mathbb{R}^{\mathtt{d}} \to \mathbb{R}$ is quadratic if $f(d) = d^\top A d + d^\top b + c$ for all $d \in \mathbb{R}^{\mathtt{d}}$ and for certain $A \in \mathbb{R}^{\mathtt{d} \times \mathtt{d}}$, $b \in \mathbb{R}^{\mathtt{d}}$, and $c \in \mathbb{R}$. A quadratic model with parameters A, b, c is defined as follows:

$$\mathscr{B}_{\text{qd}}(A, b, c) := \left\{ d \in \mathbb{R}^{\mathtt{d}} \mid d^\top A d + d^\top b + c = 0 \right\}. \tag{QD}$$

The set of outcomes consistent with the quadratic model are ellipsoids, paraboloids, hyperboloids, etc., in $\mathbb{R}^{\mathtt{d}}$. Let $\mathscr{M}_{\text{qd}}$ be the set of all quadratic models,

$$\mathscr{M}_{\text{qd}} := \left\{ \mathscr{B}_{\text{qd}}(A, b, c) \;\middle|\; \begin{bmatrix} A & b \\ b^\top & c \end{bmatrix} \text{ is a symmetric } (\mathtt{d}+1) \times (\mathtt{d}+1) \text{ matrix} \right\}.$$

In terms of the parameterization (QD), the misfit minimization problem (APR) for the model class $\mathscr{M}_{\text{qd}}$ and 2-norms $\|\cdot\|_i$ is

$$\min_{\substack{A,b,c \\ A \neq 0}} \Big(\underbrace{\min_{\hat{D}} \|D - \hat{D}\|_{\text{F}} \text{ subject to } \begin{bmatrix} \hat{d}_i \\ 1 \end{bmatrix} \begin{bmatrix} A & b/2 \\ b^\top/2 & c \end{bmatrix} \begin{bmatrix} \hat{d}_i \\ 1 \end{bmatrix}^\top = 0, \text{ for } i = 1, \ldots, N}_{M(D, \mathscr{B}_{\text{qd}}(A,b,c))} \Big). \tag{QD TLS}$$

Problems (BLN TLS) and (QD TLS) have the same geometric interpretation as the TLS problem—minimize the sum of squared orthogonal distances from the data points to the estimated model. In the special case when $A > 0$ and $4c < b^\top A^{-1} b$, $\mathscr{B}_{\text{qd}}(A, b, c)$ is an ellipsoid and the approximation problem becomes an ellipsoid fitting problem. Because of the geometrically appealing cost function, the misfit minimization problem for ellipsoid fitting attracted much attention in the literature. Nevertheless, in the nonlinear case, we do not solve the misfit minimization problems (BLN TLS) and (QD TLS) but alternative modeling problems, called adjusted least squares (ALS). The reasons are

1. the minimization problems (BLN TLS) and (QD TLS) are expensive to solve, and
2. the solutions of these problems do not define consistent estimators.

In the EIV setting, i.e., assuming that the outcomes come from a true model with stochastic measurement error, the aim is to find consistent estimators. An estimator is consistent when it converges asymptotically to the true model as the number N of observed outcomes increases. The estimators defined by the orthogonal regression problems (BLN TLS) and (QD TLS) are not consistent, but the estimator defined by the ALS method is consistent. In addition, the computation of the ALS estimator reduces to a generalized eigenvalue computation and does not require expensive optimization methods.

2.6 Dynamic Models and Global Total Least Squares

In dynamic problems, the data consists of one or more *time series* $w_{\mathrm{d}} = \big(w_{\mathrm{d}}(1), \ldots, w_{\mathrm{d}}(T)\big)$.

Note 2.5 (Notation w_{d}) The letter "d" in subscript stands for "data". It is used to distinguish a general time series w from a particular given one w_{d}.

In the context of dynamic problems, we associate $\mathscr{U}$ with the set of sequences $(\mathbb{R}^{\mathtt{w}})^T$. The dynamic nature of a model $\mathscr{B}$ is expressed in the existence of relations among the values of a time series $w \in \mathscr{B}$ at consecutive moments of time. Restricting ourselves to linear constant coefficient relations, this yields the following difference equation:

$$R_0 w(t) + R_1 w(t+1) + \cdots + R_l w(t+l) = 0, \quad \text{for } t = 1, \ldots, T-l. \tag{DE}$$

For $l = 0$ (no time shifts in the linear relations), (DE) describes a linear static model. As in the static case, (DE) is called a kernel representation of the system.[1]The system induced by (DE) is denoted as follows:

$$\mathscr{B} = \ker\big(R(\sigma)\big) := \big\{\, w \in (\mathbb{R}^{\mathtt{w}})^T \mid \text{(DE) holds} \,\big\}, \quad \text{where } R(z) := \textstyle\sum_{i=0}^{l} R_i z^i, \tag{KR}$$

and σ is the shift operator: $(\sigma w)(t) = w(t+1)$.

Let $\mathscr{B} = \ker\big(R(\sigma)\big)$ with a row proper polynomial matrix $R(z) \in \mathbb{R}^{\mathtt{p}\times\mathtt{w}}[z]$ and define $\mathtt{l} := \deg(R)$, $\mathtt{m} := \mathtt{w} - \mathtt{p}$. It can be shown that for T sufficiently large, $\dim(\mathscr{B}) \leq T\mathtt{m} + \mathtt{lp}$. Thus the complexity of the system, which is related to $\dim(\mathscr{B})$, is specified by the maximum lag $\mathtt{l}$ and the integer $\mathtt{m}$. Under the above assumption, $\mathtt{m}$ is equal to the input cardinality of the system, i.e., the number of inputs in an input/output representation of the system. We denote by $\mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}}$ the class of all linear time-invariant (LTI) systems with $\mathtt{w}$ variables, maximum input cardinality $\mathtt{m}$, and maximum lag $\mathtt{l}$. Note that the class of systems $\mathscr{L}^{\mathtt{w}}_{\mathtt{m},0}$, described by zero lag difference equations, is the set of linear static systems of dimension at most $\mathtt{m}$ as defined before.

Modeling a dynamic system from data is called system identification. We consider the identification problem for the LTI model class $\mathscr{M} = \mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}}$ and treat first the exact identification problem: given data w_{d}, such that $w_{\mathrm{d}} \in \mathscr{B} \in \mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}}$, find a representation of $\mathscr{B}$. Under certain identifiability conditions on the data and the model class, the MPUM $\mathscr{B}_{\mathrm{mpum}}$ of w_{d} in the model class $\mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}}$ exists and is equal to $\mathscr{B}$.

We consider algorithms for passing from w_{d} to a kernel or input/state/output representation of $\mathscr{B}_{\mathrm{mpum}}$. The algorithms are in the setting of what are called subspace identification methods; i.e., the parameters of the system are retrieved from certain subspaces computed from the given data. We do not emphasize the geometric interpretation and derivation of the subspace algorithms and give instead more system theory oriented derivations.

In their pure form, exact identification algorithms are mainly of theoretical interest. Most system identification problems start from rough data, so that the approximation element is critical. The exact identification algorithms can be modified so that they can "work" with rough data. We do not pursue this approach but consider instead the misfit approximation problem (APR), which is optimization based.

[1]We do not distinguish between model and system but preferably use model in the static context or in general discussions and system in the dynamic context.

The misfit minimization problem (APR) with model class $\mathscr{M} = \mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}}$ and 2-norm $\|\cdot\|_{\mathscr{U}}$ is called the global total least squares problem (GlTLS). In terms of the kernel representation (KR), the GlTLS problem is

$$\min_{R(z)} \Big(\underbrace{\min_{\hat{w}} \|w_{\mathtt{d}} - \hat{w}\| \ \text{ s.t. } \ \hat{w} \in \mathscr{B} := \ker\big(R(\sigma)\big)}_{M(w_{\mathtt{d}}, \ker(R(\sigma)))} \Big) \ \text{ s.t. } \ R \text{ full row rank.} \tag{TLS$_{R(z)}$}$$

The constraint $R(z)$ full row rank, $\deg(R) = l$ is equivalent to $\mathscr{B} := \ker\big(R(\sigma)\big) \in \mathscr{L}^{\mathtt{w}}_{\mathtt{m},l}$ and the constraint $w \in \mathscr{B}$ is equivalent to (DE). In turn, (DE) can be written as the structured system of equations

$$\begin{bmatrix} R_0 & R_1 & \cdots & R_l \end{bmatrix} \begin{bmatrix} w(1) & w(2) & \cdots & w(T-l) \\ w(2) & w(3) & \cdots & w(T-l+1) \\ \vdots & \vdots & & \vdots \\ w(l+1) & w(l+2) & \cdots & w(T) \end{bmatrix} = 0,$$

which makes a link with the structured total least squares problem.

2.7 Structured Total Least Squares

The GlTLS problem (TLS$_{R(z)}$) is similar to the TLS problem, the main difference being that the generally unstructured matrix $\hat{D}$ in the TLS problem is replaced by a block-Hankel structured matrix in the GlTLS problem. In this section, we define a general approximation problem with a constraint expressed as rank deficiency of a structured matrix.

Let $\mathscr{S} : \mathbb{R}^{n_p} \to \mathbb{R}^{m \times (n+d)}$ be an injective function. A matrix $C \in \mathbb{R}^{m \times (n+d)}$ is said to be $\mathscr{S}$-structured if $C \in \operatorname{image}(\mathscr{S})$. The vector p for which $C = \mathscr{S}(p)$ is called the parameter vector of the structured matrix C. Respectively, $\mathbb{R}^{n_p}$ is called the parameter space of the structure $\mathscr{S}$.

The structured total least squares (STLS) problem aims to find an optimal structured low-rank approximation $\mathscr{S}(\hat{p})$ of a given structured matrix $\mathscr{S}(p)$; i.e., given a structure specification $\mathscr{S}$, a parameter vector p, and a desired rank n, find

$$\hat{p}_{\text{stls}} = \arg\min_{\hat{p}} \|p - \hat{p}\| \quad \text{subject to} \quad \operatorname{rank}\big(\mathscr{S}(\hat{p})\big) \leq n. \tag{STLS}$$

By representing the rank constraint in (STLS) as "there is a full row rank matrix $R \in \mathbb{R}^{d \times (n+d)}$, such that $R\mathscr{S}^\top(\hat{p}) = 0$", the STLS problem can be written equivalently as

$$\hat{R}_{\text{stls}} = \arg \min_{RR^\top = I_d} \ \min_{\hat{p}} \|p - \hat{p}\| \quad \text{subject to} \quad R\mathscr{S}^\top(\hat{p}) = 0, \tag{STLS$_R$}$$

which is a double minimization problem, similar to the general misfit minimization problem (APR). The STLS formulation, however, is not linked with a particular model class: it is viewed as a flexible tool that can match different misfit minimization problems.

Table 2.1 gives a summary of the misfit minimization problems described up to now.

Table 2.1. *Misfit minimization problems.*

Name	$\mathscr{U}$	$\mathscr{M}$	Problem
TLS	$\mathbb{R}^{\mathtt{d}}$	$\mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}$	$\min_{\hat{D}\in\hat{\mathscr{B}}\in\mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}} \|D-\hat{D}\|_{\mathrm{F}}$
GTLS	$\mathbb{R}^{\mathtt{d}}$	$\mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}$	$\min_{\hat{D}\in\hat{\mathscr{B}}\in\mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}} \sqrt{\sum_i \|\sqrt{W}(d_i-\hat{d}_i)\|^2}$
WTLS	$\mathbb{R}^{\mathtt{d}}$	$\mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}$	$\min_{\hat{D}\in\hat{\mathscr{B}}\in\mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}} \sqrt{\sum_i \|\sqrt{W_i}(d_i-\hat{d}_i)\|^2}$
Bilinear	$\mathbb{R}^{\mathtt{d}}$	$\mathscr{M}_{\mathrm{bln}}$	$\min_{\hat{D}\in\hat{\mathscr{B}}\in\mathscr{M}_{\mathrm{bln}}} \|D-\hat{D}\|_{\mathrm{F}}$
Quadratic	$\mathbb{R}^{\mathtt{d}}$	$\mathscr{M}_{\mathrm{qd}}$	$\min_{\hat{D}\in\hat{\mathscr{B}}\in\mathscr{M}_{\mathrm{qd}}} \|D-\hat{D}\|_{\mathrm{F}}$
GlTLS	$(\mathbb{R}^{\mathtt{w}})^T$	$\mathscr{L}_{\mathtt{m},1}^{\mathtt{w}}$	$\min_{\hat{d}\in\hat{\mathscr{B}}\in\mathscr{L}_{\mathtt{m},1}^{\mathtt{w}}} \|w_{\mathtt{d}}-\hat{w}\|_{\ell_2}$

2.8 Algorithms

Optimization Methods

The approximate modeling problem (APR) is a double minimization problem: on the inner level is the misfit computation and on the outer level is the search for the optimal model. In the linear case, the model $\mathscr{B}$ is a subspace of $\mathscr{U}$, so that the misfit computation is equivalent to projection of the data $\mathscr{D}$ on $\mathscr{B}$. In this case, it is possible to express the misfit $M(\mathscr{D},\mathscr{B})$ in a closed form. The outer minimization problem $\min_{\mathscr{B}\in\mathscr{M}} M(\mathscr{D},\mathscr{B})$, however, is a nonlinear least squares problem. We employ *local* optimization methods for its numerical solution. The local optimization methods require initial approximation and find only one locally optimal model.

Important issues we deal with are finding good and computationally inexpensive initial approximations and making the misfit function and its first derivative evaluation numerically efficient. By solving these issues, we obtain an "engineering solution" of the problem, i.e., a solution that is effective for real-life applications.

Caveat: We aim at efficient evaluation of the misfit function, which ensures efficiency only with respect to the amount of given data: in the static case, the number N of observed outcomes and in the dynamic case the length T of the observed time series. In this book, we do not address the related question of achieving efficiency on the level of the outer minimization problem, i.e., with respect to the number of model parameters. Thus an implicit assumption throughout this work is that a simple approximate model we aim for. Dealing with large scale nonlinear least squares problems, however, is a well developed topic (see, e.g., [BHN99]) so that general purpose solutions can be used.

Adjusted Least Squares Method

In the nonlinear case not only the outer minimization but also the misfit computation is a nonconvex problem and requires iterative solution methods. This makes the misfit minimization problem numerically rather expensive. In addition, from a statistical point of view

Table 2.2. *Problems, algorithms, and application fields.*

Problem	Algorithm	Application field
WTLS	optimization	chemometrics
STLS	optimization	system identification
bilinear model approximation	ALS	motion analysis
quadratic model approximation	ALS	ellipsoid estimation

the obtained solution is not attractive, because in the EIV setting, it is inconsistent. For these reasons, we adopt an alternative approach.

The ALS method is a quadratically constrained least squares method. Its solution is obtained from a generalized eigenvalue decomposition. The problem is motivated and derived from the consideration of obtaining a consistent estimator in the EIV setting. Knowing the noise variance, the bias of the ordinary least squares method is removed. This involves adding a correction to the sample covariance matrix. If the noise variance is unknown, it can be estimated together with the model parameters.

Benchmark examples show that the ALS estimator gives good fits that are comparable with those obtained from the orthogonal regression methods. The advantage over the misfit approximation problem, however, is that the ALS approximation does not depend on a user-supplied initial approximation and is computationally less expensive.

Table 2.2 gives a summary of the problems, algorithms, and applications considered in the book.

Software Implementation

The algorithms in the book have documented software implementation. Each algorithm is realized by one or more functions and the functions are separated in the following packages:

- MATLAB software for weighted total least squares approximation,
- C software for structured total least squares approximation,
- MATLAB software for balanced model identification, and
- MATLAB software for approximate system identification.

Many of the simulation examples presented in the book are included in the packages as demo files. Thus the reader can try out these examples and modify the simulation settings. Appendix B gives the necessary background information for starting to use the software.

Part I

Static Problems

Chapter 3

Weighted Total Least Squares

We start this chapter with the simplest of the approximate modeling problems—the ones for the linear static model. The kernel, image, and input/output representations of a linear static model are reviewed in Section 3.2. The misfit criterion is defined as a weighted projection and the corresponding misfit approximation problem is called weighted total least squares (WTLS). Two interpretations of the weight matrices in the WTLS formulation are described in Section 3.1. The TLS and GTLS problems are special cases of the WTLS problem and are considered in Section 3.3.

In Section 3.4, we start to discuss the solution of the general WTLS problem. First, the misfit computation is explained. It is a quadratic minimization problem, so that its solution reduces to solving a linear system of equations. The remaining part of the WTLS problem is the minimization with respect to the model parameters. This problem is treated in Section 3.5, where three different algorithms are presented. In Section 3.6, we show simulation results that compare the performance of the algorithms.

3.1 Introduction

In this chapter, we consider approximate modeling by a linear static model. Therefore, the universum of possible outcomes is $\mathscr{U} = \mathbb{R}^{\mathtt{d}}$, the available data is the set of N outcomes $\mathscr{D} = \{ d_1, \ldots, d_N \} \subset \mathscr{U}$, and the model class is $\mathscr{M} = \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$. The parameter $\mathtt{m}$ specifies the maximum allowed complexity for the approximating model.

The matrix $D := \begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix}$ is called the data matrix. We use the shorthand notation

$$\begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix} \in \mathscr{B} \subseteq \mathscr{U} \quad :\Longleftrightarrow \quad d_i \in \mathscr{B}, \quad \text{for } i = 1, \ldots, N.$$

The WTLS misfit between the data $\mathscr{D}$ and a model $\mathscr{B} \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$ is defined as follows:

$$M_{\text{wtls}}\big(\begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix}, \mathscr{B} \big) := \min_{\hat{d}_1, \ldots, \hat{d}_N \in \mathscr{B}} \sqrt{\sum_{i=1}^{N} (d_i - \hat{d}_i)^\top W_i (d_i - \hat{d}_i)}, \qquad \text{(Mwtls)}$$

where $W_1, \ldots, W_N$ are given positive definite matrices.

Problem 3.1 (WTLS). *Given the data matrix $D = \begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix} \in \mathbb{R}^{\mathtt{d}\times N}$, a complexity bound $\mathtt{m}$, and positive definite weight matrices $W_1, \ldots, W_N$, find an approximate model*

$$\hat{\mathscr{B}}_{\text{wtls}} := \arg \min_{\hat{\mathscr{B}} \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}} M_{\text{wtls}}(D, \hat{\mathscr{B}}). \tag{WTLS}$$

Note 3.2 (Element-wise weighted total least squares) The special case when all weight matrices are diagonal is called element-wise weighted total least squares (EWTLS). Let $W_i = \operatorname{diag}(w_{1,i}, \ldots, w_{\mathtt{d},i})$ and define the $\mathtt{d} \times N$ matrix Σ by $\Sigma_{ji} := \sqrt{w_{j,i}}$ for all j, i. Denote by $\odot$ the element-wise product $A \odot B = \begin{bmatrix} a_{ij} b_{ij} \end{bmatrix}$. Then

$$\sum_{i=1}^{N} \Delta d_i^\top W_i \Delta d_i = \|\Sigma \odot \Delta D\|_{\mathrm{F}}^2,$$

where $\Delta D := \begin{bmatrix} \Delta d_1 & \cdots & \Delta d_N \end{bmatrix}$ is the correction matrix $D - \hat{D}$. In Note 3.7, we comment on a statistical interpretation of the EWTLS problem and in Note 3.17 on a relation with a GTLS problem.

Note 3.3 (TLS as an unweighted WTLS) The extreme special case when $W_i = I$ for all i is called unweighted. Then the WTLS problem reduces to the TLS problem. The TLS misfit M_{tls} weights equally all elements of the correction matrix ΔD. It is a natural choice when there is no prior knowledge about the data. In addition, the unweighted case is computationally easier to solve than the general weighted case.

In the unweighted case, $\hat{D}$ tends to approximate better the large elements of D than the small ones. This effect can be reduced by introducing proper weights, for example the reciprocal of the entries of the data matrix. The resulting relative error TLS problem is a special case of the WTLS problem with $W_i := \operatorname{diag}(1/d_{1i}^2, \ldots, 1/d_{\mathtt{d}i}^2)$ or, equivalently, an EWTLS problem with $\Sigma_{ji} = 1/d_{ji}$.

Problem 3.4 (Relative error TLS). *Given the data matrix $D \in \mathbb{R}^{\mathtt{d}\times N}$ and a complexity bound $\mathtt{m}$, find an approximate model*

$$\hat{\mathscr{B}}_{\text{rtls}} := \arg \min_{\mathscr{B} \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}} \min_{\hat{D} \in \mathscr{B}} \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{\mathtt{d}} \frac{(D_{ji} - \hat{D}_{ji})^2}{D_{ji}^2}}. \tag{RTLS}$$

The misfit function of the relative error TLS problem is

$$M_{\text{rtls}}(D, \mathscr{B}) = \min_{\hat{D} \in \mathscr{B}} \|\Sigma \odot (D - \hat{D})\|_{\mathrm{F}}, \qquad \textit{where } \Sigma := \begin{bmatrix} 1/d_{ji} \end{bmatrix}.$$

Example 3.5 (Relative errors) Consider the data matrix

$$D = \begin{bmatrix} 5.4710 & 0.2028 & 0.5796 & 0.6665 & 0.6768 \\ 0.9425 & 0.7701 & 0.7374 & 0.8663 & 0.9909 \end{bmatrix},$$

obtained from an experiment with $\mathtt{d} = 2$ variables and $N = 5$ observed outcomes. We aim to model this data, using the model class $\mathscr{L}^2_{1,0}$. Note that the elements of D, except for D_{11}, are approximately five times smaller than D_{11}. The matrices of the element-wise relative errors

$$\Delta D_{\text{rel}} := \left[\frac{|d_{ji} - \hat{d}_{ji}|}{|d_{ji}|}\right]$$

for the TLS and relative error TLS solutions are, respectively,

$$\Delta D_{\text{rel,tls}} = \begin{bmatrix} 0.0153 & 0.8072 & 0.2342 & 0.2404 & 0.2777 \\ 0.3711 & 0.8859 & 0.7673 & 0.7711 & 0.7907 \end{bmatrix}$$

and

$$\Delta D_{\text{rel,rtls}} = \begin{bmatrix} 0.8711 & 0.2064 & 0.0781 & 0.0661 & 0.0030 \\ 0.1019 & 0.5322 & 0.0674 & 0.0584 & 0.0030 \end{bmatrix}.$$

Note that $\Delta D_{\text{rel,tls},11} = 0.0153$ is small but the other elements of $\Delta D_{\text{rel,tls}}$ are larger. This is a numerical illustration of the above-mentioned undesirable effect of using the TLS method for approximation of data with elements of very different magnitude. The corresponding "total" relative errors $\|\Delta D_{\text{rel}}\|_{\text{F}}$, i.e., the misfits $M_{\text{rtls}}(D, \hat{\mathscr{B}})$, are $\|\Delta D_{\text{rel,tls}}\|_{\text{F}} = 1.89$ and $\|\Delta D_{\text{rel,rtls}}\|_{\text{F}} = 1.06$. The example illustrates the advantage of introducing element-wise scaling in the approximation criterion, in order to achieve adequate approximation. ∎

Another situation in which weights are needed is when the data matrix is a noisy measurement of a true matrix that satisfies a true model $\bar{\mathscr{B}} \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$ in the model class. Intuition suggests that the elements perturbed by noise with larger variance should be weighted less in the cost function. The precise formulation of this intuitive reasoning leads to the maximum likelihood criterion for the EIV model. The EIV model for the WTLS problem is defined as follows.

Definition 3.6 (WTLS EIV model). *The data is a noisy measurement $D = \bar{D} + \tilde{D}$ of true data $\bar{D} \in \bar{\mathscr{B}} \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$, where $\bar{\mathscr{B}}$ is a true model in the model class, and $\tilde{D}$ is the measurement error. In addition, the vector of measurement errors $\operatorname{vec}(\tilde{D})$ is zero mean and Gaussian, with covariance matrix $\bar{\sigma}^2 \cdot \operatorname{diag}(V_1, \ldots, V_N)$, i.e., $\operatorname{vec}(\tilde{D}) \sim \mathrm{N}\left(0, \bar{\sigma}^2 \cdot \operatorname{diag}(V_1, \ldots, V_N)\right)$. In the estimation problem, the covariance matrices $V_1, \ldots, V_N$ are assumed known but $\bar{\sigma}^2$ need not be known.*

Note 3.7 (Statistical interpretation of the EWTLS problem) From a statistical point of view, the EWTLS problem formulation corresponds to a WTLS EIV setup in which all measurement errors are uncorrelated. We refer to this EIV model as the EWTLS EIV model.

By choosing $W_i = V_i^{-1}$, the approximation $\hat{\mathscr{B}}_{\text{wtls}}$ is the maximum likelihood estimate of $\bar{\mathscr{B}}$ in the WTLS EIV model. Under additional assumptions (see [KV04])s it is a *consistent* estimator of $\bar{\mathscr{B}}$.

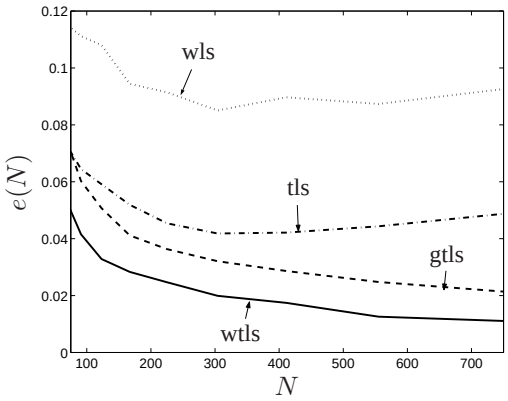


Figure 3.1. *Relative error of estimation e as a function of N for four estimators.*

Note 3.8 (Noise variance estimation) The optimal solution $\hat{\mathscr{B}}_{\text{wtls}}$ does not depend on a scaling of the weight matrices by the same scaling factor; i.e., the optimal solution with weights $\sigma^{-2}W_i$ does not depend on σ^2. It turns out that the normalized optimal misfit $M(D, \hat{\mathscr{B}}_{\text{wtls}})/N$ is an estimate of the noise variance $\bar{\sigma}^2$ in the EIV model.

Example 3.9 (Consistency) We set up a simulation example corresponding to the WTLS EIV model with $\mathtt{d} = 3$, $\mathtt{m} = 2$, and N ranging from 75 to 750. Let $U(\underline{u}, \overline{u})$ be a matrix of independent and uniformly distributed elements in the interval $[\underline{u}, \overline{u}]$. The true data matrix is $\bar{D} = U(0, 1)$ and the measurement error covariance matrices are $\bar{\sigma}^2 \cdot V_i = \operatorname{diag}(\sigma_{i1}^2, \sigma_{i2}^2, \sigma_{i3}^2)$, where $\sigma_{i1} = \sigma_{i2} = U(0.01, 0.26)$, and $\sigma_{i3} = U(0.01, 0.035)$.

For a fixed $N \in [75, 750]$, 500 noise realizations are generated and the estimates are computed with the following methods:

TLS total least squares ($W_i = W = I$),

GTLS generalized total least squares ($W_i = W = V_{\text{avr}}^{-1}$, where $V_{\text{avr}} := (\sum_{i=1}^{N} \sqrt{V_i}/N)^2$),

WTLS weighted total least squares ($W_i = V_i^{-1}$), and

WLS weighted least squares (that minimizes a weighted norm of the equation error of an input/output representation; see Section 3.2).

A relative error of estimation e that measures the distance from the estimated model $\hat{\mathscr{B}}$ to the true one $\bar{\mathscr{B}}$ (in terms of the parameter X in an input/output representation; see Section 3.2) is averaged for the 500 noise realizations and plotted as a function of N in Figure 3.1. Convergence of the relative error of estimation to zero as N increases indicates consistency of the corresponding estimator.

The stochastic framework gives a convincing interpretation of the weight matrices W_i. Also, it suggests possible ways to choose them. For example, they can be selected by noise variance estimation from repeated measurements or from prior knowledge about the accuracy of the measurement devices.

A practical application of the WTLS problem occurs in chemometrics, where the aim is to estimate the concentrations of certain chemical substances in a mixture from spectral measurements on the mixture. For details see [WAH+97, SMWV05].

3.2 Kernel, Image, and Input/Output Representations

In this section we review three common representations of a linear static model: kernel, image, and input/output. They give different parameterizations of the model and are important in setting up algorithms for approximate modeling with the model class $\mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}$.

Kernel Representation

Let $\mathscr{B} \in \mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}$; i.e., $\mathscr{B}$ is a subspace of $\mathbb{R}^{\mathtt{d}}$ with dimension at most $\mathtt{m}$. A kernel representation of $\mathscr{B}$ is given by a system of equations $Rd = 0$, such that $\mathscr{B} = \{\, d \in \mathbb{R}^{\mathtt{d}} \mid Rd = 0 \,\}$ or, equivalently, by $\mathscr{B} = \ker(R)$. The matrix $R \in \mathbb{R}^{g \times \mathtt{d}}$ is a parameter of the model $\mathscr{B}$.

The parameter R is not unique. There are two sources for the nonuniqueness:

1. R might have redundant rows, and
2. for a full-rank matrix U, $\ker(R) = \ker(UR)$.

The parameter R having redundant rows is related to the minimality of the representation. For a given linear static model $\mathscr{B}$, the representation $Rd = 0$ of $\mathscr{B}$ is minimal if R has the minimal number of rows among all parameters R that define a kernel representation of $\mathscr{B}$. The kernel representation, defined by R, is minimal if and only if R is full row rank.

Because of item 2, a minimal kernel representation is still not unique. All minimal representations, however, are related to a given one via a premultiplication of the parameter R with a nonsingular matrix U. In a minimal kernel representation, the rows of R are a basis for $\mathscr{B}^\perp$, the orthogonal complement of $\mathscr{B}$, i.e., $\mathscr{B}^\perp = \operatorname{row\,span}(R)$. The choice of R is nonunique due to the nonuniqueness in the choice of basis of $\mathscr{B}^\perp$.

Assuming that $\mathscr{B} \in \mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}$ and $\dim(\mathscr{B}) = \mathtt{m}$, the minimal number of laws necessary to define $\mathscr{B}$ is $\mathtt{p} := \mathtt{d} - \mathtt{m}$; i.e., in a minimal representation, $\mathscr{B} = \ker(R)$ with $\operatorname{row\,dim}(R) = \mathtt{p}$.

Image Representation

The dual of the kernel representation $\mathscr{B} = \ker(R)$ is the image representation

$$\mathscr{B} = \{\, d \in \mathbb{R}^{\mathtt{d}} \mid d = Pl,\ l \in \mathbb{R}^{\mathtt{l}} \,\}$$

or, equivalently, $\mathscr{B} = \operatorname{col\,span}(P)$. Again, for a given $\mathscr{B} \in \mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}$, an image representation $\mathscr{B} = \operatorname{col\,span}(P)$ is not unique because of the possible nonminimality of P and the choice of basis. The representation is minimal if and only if P is a full column rank matrix. In a minimal image representation, $\operatorname{col\,dim}(P) = \dim(\mathscr{B})$ and the columns of P form a basis for $\mathscr{B}$. Clearly, $\operatorname{col\,span}(P) = \operatorname{col\,span}(PU)$, for any nonsingular matrix $U \in \mathbb{R}^{\mathtt{l} \times \mathtt{l}}$. Note that

$$\ker(R) = \operatorname{col\,span}(P) = \mathscr{B} \in \mathscr{L}_{\mathtt{m},0}^{\mathtt{d}} \implies RP = 0,$$

which gives a link between the parameters P and R.

Input/Output Representation

Both the kernel and the image representations treat all variables on an equal footing. In contrast, the more classical input/output representation

$$\mathscr{B}_{\text{i/o}}(X) := \{\, d =: \operatorname{col}(d_{\text{i}}, d_{\text{o}}) \in \mathbb{R}^{\mathtt{d}} \mid X^\top d_{\text{i}} = d_{\text{o}} \,\} \qquad \text{(I/O repr)}$$

distinguishes free variables $d_{\text{i}} \in \mathbb{R}^{\mathtt{m}}$, called inputs, and dependent variables $d_{\text{o}} \in \mathbb{R}^{\mathtt{p}}$, called outputs. In an input/output representation, d_{i} can be chosen freely, while d_{o} is fixed by d_{i} and the model.

The partitioning $d = \operatorname{col}(d_{\text{i}}, d_{\text{o}})$ gives an input/output partitioning of the variables: the first $\mathtt{m} := \dim(d_{\text{i}})$ variables are inputs and the remaining $\mathtt{p} := \dim(d_{\text{o}}) = \mathtt{d} - \mathtt{m}$ variables are outputs. An input/output partitioning is not unique. Given a kernel or image representation, finding an input/output partitioning is equivalent to selecting a $\mathtt{p} \times \mathtt{p}$ full-rank submatrix of R or an $\mathtt{m} \times \mathtt{m}$ full-rank submatrix of P. In fact, generically, any splitting of the variables into a group of $\mathtt{p}$ variables (outputs) and a group of remaining variables (inputs) defines a valid input/output partitioning. In nongeneric cases, certain partitionings of the variables into inputs and outputs are not possible.

Note that in (I/O repr), the first $\mathtt{m}$ variables are fixed to be inputs, so that given X, the input/output representation $\mathscr{B}_{\text{i/o}}(X)$ is fixed and vice versa; given $\mathscr{B} \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$, the parameter X (if it exists) is unique. Thus, as opposed to the parameters R and P in the kernel and the image representations, the parameter X in the input/output representation (I/O repr) is unique.

Consider the input/output representation $\mathscr{B}_{\text{i/o}}(X)$ of $\mathscr{L}^{\mathtt{d}}_{\mathtt{m},0} \in \mathscr{B}$. The matrices

$$R = \begin{bmatrix} X^\top & -I \end{bmatrix} \quad \text{and} \quad P = \begin{bmatrix} I \\ X^\top \end{bmatrix}$$

are parameters of, respectively, kernel and image representations of $\mathscr{B}$, i.e.,

$$\mathscr{B}_{\text{i/o}}(X) = \ker\left(\begin{bmatrix} X^\top & -I \end{bmatrix}\right) = \operatorname{col\,span}\left(\begin{bmatrix} I \\ X^\top \end{bmatrix}\right).$$

Conversely, given the parameters

$$R =: \begin{bmatrix} R_{\text{i}} & R_{\text{o}} \end{bmatrix}, \quad R_{\text{o}} \in \mathbb{R}^{\mathtt{p}\times\mathtt{p}} \quad \text{and} \quad P =: \begin{bmatrix} P_{\text{i}} \\ P_{\text{o}} \end{bmatrix}, \quad P_{\text{i}} \in \mathbb{R}^{\mathtt{m}\times\mathtt{m}},$$

of, respectively, kernel and image representations of $\mathscr{B} \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$, and assuming that R_{o} and P_{i} are nonsingular,

$$X^\top = -R_{\text{o}}^{-1} R_{\text{i}} = P_{\text{o}} P_{\text{i}}^{-1}$$

is the parameter of the input/output representation (I/O repr) of $\mathscr{B}$, i.e.,

$$\ker\left(\begin{bmatrix} \overset{\mathtt{m}}{R_{\text{i}}} & \overset{\mathtt{p}}{R_{\text{o}}} \end{bmatrix}\right) = \operatorname{col\,span}\left(\begin{bmatrix} P_{\text{i}} \\ P_{\text{o}} \end{bmatrix}\begin{matrix} \mathtt{m} \\ \mathtt{p} \end{matrix}\right) = \mathscr{B}_{\text{i/o}}\big((-R_{\text{o}}^{-1}R_{\text{i}})^\top\big) = \mathscr{B}_{\text{i/o}}\big((P_{\text{o}}P_{\text{i}}^{-1})^\top\big).$$

Figure 3.2 shows the links among kernel, image, and input/output representations.

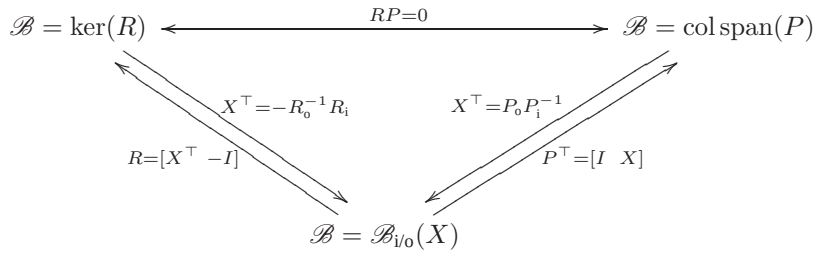


Figure 3.2. *Links among kernel, image, and input/output representations of* $\mathscr{B} \in \mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}$.

Note 3.10 (Weighted least squares) The input/output latency minimization problem

$$\hat{X}_{\mathrm{wls}} = \arg\min_X \sqrt{\sum_{i=1}^{N} (X^\top d_{\mathrm{i},i} - d_{\mathrm{o},i})^\top W_i (X^\top d_{\mathrm{i},i} - d_{\mathrm{o},i})}, \qquad (\mathrm{WLS}_X)$$

corresponding to problem (APReqn) with $\|e\|_i = \|\sqrt{W_i} e\|$, is the weighted LS problem.

Note 3.11 ($AX = B$ **notation)** A standard notation adopted in the numerical linear algebra literature for the input/output linear static model representation (I/O repr) is $a^\top X = b^\top$, i.e., $a = d_{\mathrm{i}}$ and $b = d_{\mathrm{o}}$. For repeated observations $D = \begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix}$, the statement $D \in \mathscr{B}_{\mathrm{i/o}}(X)$ is equivalent to the linear system of equations $AX = B$, where $\begin{bmatrix} A & B \end{bmatrix} := D^\top$ with $A \in \mathbb{R}^{N \times \mathtt{m}}$ and $B \in \mathbb{R}^{N \times \mathtt{p}}$.

3.3 Special Cases with Closed Form Solutions

The special cases

- $W_i = I$, i.e., the total least squares problem, and
- $W_i = W$, i.e., the generalized total least squares problem,

allow closed form solution in terms of the singular value decomposition (SVD) of the data matrix $D = \begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix}$. For general weight matrices W_i, however, the WTLS problem has no closed form solution and its solution is based on numerical optimization methods that are less robust and efficient. For this reason, recognizing the special cases and applying the special methods is important.

The following lemmas are instrumental for the solution of the TLS problem.

Lemma 3.12. *For* $D \in \mathbb{R}^{\mathtt{d} \times N}$ *and* $\mathtt{m} \in \mathbb{N}$, $D \in \mathscr{B} \in \mathscr{L}_{\mathtt{m},0}^{\mathtt{d}} \iff \operatorname{rank}(D) \leq \mathtt{m}$.

Proof. Let $D \in \mathscr{B} \in \mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}$ and consider a minimal kernel representation of $\mathscr{B} = \ker(R)$, where $R \in \mathbb{R}^{\mathtt{p} \times \mathtt{d}}$ is full row rank. Then $D \in \mathscr{B} \in \mathscr{L}_{\mathtt{m},0}^{\mathtt{d}} \implies RD = 0 \implies \operatorname{rank}(D) \leq \mathtt{m}$.

Now let D be rank deficient with $\operatorname{rank}(D) \leq \mathtt{m}$. Then there is a full row rank matrix $R \in \mathbb{R}^{\mathtt{p} \times \mathtt{d}}$, $\mathtt{p} := \mathtt{d} - \mathtt{m}$, that annihilates D, i.e., $RD = 0$. The matrix R defines a model in the class $\mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}$ via $\mathscr{B} := \ker(R)$. Then $RD = 0 \implies D \in \mathscr{B} \in \mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}$. ∎

Lemma 3.12 shows that the approximation of the data matrix D with a model in the class $\mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$ is equivalent to finding a matrix $\hat{D} \in \mathbb{R}^{\mathtt{d}\times N}$ with rank at most $\mathtt{m}$. In the case when the approximation criterion is $\|D - \hat{D}\|_{\mathrm{F}}$ (TLS problem) or $\|D - \hat{D}\|_2$, the problem has a solution in terms of the SVD of D. The result is known as the Eckart–Young–Mirsky low-rank matrix approximation theorem [EY36]. We state it in the next lemma.

Lemma 3.13 (Matrix approximation lemma). *Let $D = U\Sigma V^\top$ be the SVD of $D \in \mathbb{R}^{\mathtt{d}\times N}$ and partition the matrices U, $\Sigma =: \operatorname{diag}(\sigma_1, \ldots, \sigma_{\mathtt{d}})$, and V as follows:*

$$U =: \overset{\mathtt{m}}{\underset{\mathtt{d}}{\begin{bmatrix} U_1 \end{bmatrix}}} \overset{\mathtt{p}}{\begin{bmatrix} U_2 \end{bmatrix}}, \quad \Sigma =: \overset{\mathtt{m}}{\underset{\mathtt{p}}{\begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}}} \quad \text{and} \quad V =: \overset{\mathtt{m}}{\underset{N}{\begin{bmatrix} V_1 \end{bmatrix}}} \overset{\mathtt{p}}{\begin{bmatrix} V_2 \end{bmatrix}}, \qquad \text{(SVD PRT)}$$

where $\mathtt{m} \in \mathbb{N}$ is such that $0 \leq \mathtt{m} \leq \min(\mathtt{d}, N)$ and $\mathtt{p} := \mathtt{d} - \mathtt{m}$. Then the rank-$\mathtt{m}$ matrix

$$\hat{D}^* = U_1 \Sigma_1 V_1^\top$$

is such that

$$\|D - \hat{D}^*\|_{\mathrm{F}} = \min_{\operatorname{rank}(\hat{D}) \leq \mathtt{m}} \|D - \hat{D}\|_{\mathrm{F}} = \sqrt{\sigma_{\mathtt{m}+1}^2 + \cdots + \sigma_{\mathtt{d}}^2}.$$

The solution $\hat{D}^*$ is unique if and only if $\sigma_{\mathtt{m}+1} \neq \sigma_{\mathtt{m}}$.

The solution of the TLS problem (TLS) trivially follows from Lemmas 3.12 and 3.13.

Theorem 3.14 (Solution of the TLS problem). *Let $D = U\Sigma V^\top$ be the SVD of D and partition the matrices U, Σ, and V as in (SVD PRT). Then a TLS approximation of D in $\mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$ is*

$$\hat{D}_{\text{tls}} = U_1 \Sigma_1 V_1^\top, \qquad \hat{\mathscr{B}}_{\text{tls}} = \ker(U_2^\top) = \operatorname{col\,span}(U_1),$$

and the corresponding TLS misfit is

$$\|D - \hat{D}_{\text{tls}}\|_{\mathrm{F}} = \sqrt{\sigma_{\mathtt{m}+1}^2 + \cdots + \sigma_{\mathtt{d}}^2}, \quad \textit{where} \quad \Sigma_2 =: \operatorname{diag}(\sigma_{\mathtt{m}+1}, \ldots, \sigma_{\mathtt{d}}).$$

A TLS approximation always exists. It is unique if and only if $\sigma_{\mathtt{m}} \neq \sigma_{\mathtt{m}+1}$.

Note 3.15 (Efficient computation of $\hat{\mathscr{B}}_{\text{tls}}$) If one is interested in an approximate model $\hat{\mathscr{B}}_{\text{tls}}$ and not in an approximated data $\hat{D}_{\text{tls}}$, the computation can be done more efficiently. A TLS model $\hat{\mathscr{B}}_{\text{tls}}$ depends only on the left singular vectors of D. Therefore, for any orthogonal matrix Q, a TLS approximation computed for the data matrix DQ is still $\hat{\mathscr{B}}_{\text{tls}}$ (the left singular vectors are not affected by the multiplication with Q). Let $D = \begin{bmatrix} R_1 & 0 \end{bmatrix} Q^\top$ be the QR factorization of D. A TLS approximation of R_1 is the same as a TLS approximation of D. For $N \gg \mathtt{d}$, computing the QR factorization $D = \begin{bmatrix} R_1 & 0 \end{bmatrix} Q^\top$ and the SVD of R_1 is a more efficient alternative for finding $\hat{\mathscr{B}}_{\text{tls}}$ than computing the SVD of D.

Note 3.16 (Nongeneric TLS problems) The TLS problem formulation (TLS_X) suffers from the drawback that the optimal approximating model $\hat{\mathscr{B}}_{\text{tls}}$ might have no input/output

representation (I/O repr). In this case (known as nongeneric TLS problem), the optimization problem (TLS_X) has no solution. By suitable permutation of the variables, however, (TLS_X) can be made solvable, so that $\hat{X}_{\mathrm{tls}}$ exists and $\hat{\mathscr{B}}_{\mathrm{tls}} = \mathscr{B}_{\mathrm{i/o}}(\hat{X}_{\mathrm{tls}})$.

> The issue of whether the TLS problem is generic or not is not related to the approximation of the data *per se* but to the possibility of representing the optimal model $\hat{\mathscr{B}}_{\mathrm{tls}}$ in the form (I/O repr), i.e., to the possibility of imposing a *given* input/output partition on $\hat{\mathscr{B}}_{\mathrm{tls}}$.

The solution of the GTLS problem can be obtained from the solution of a TLS problem for a modified data matrix. In fact, with the same transformation technique, we can solve a more general WTLS problem than the previously defined GTLS problem. Define the following misfit criterion:

$$M_{\mathrm{gtls2}}(D, \mathscr{B}) = \min_{\hat{D} \in \mathscr{B}} \left\| \sqrt{W_{\mathrm{l}}}(D - \hat{D})\sqrt{W_{\mathrm{r}}} \right\|_{\mathrm{F}}. \tag{Mgtls2}$$

With $W_{\mathrm{l}} = W$ and $W_{\mathrm{r}} = I$ the misfit minimization problem

$$\hat{\mathscr{B}}_{\mathrm{gtls2}} = \arg \min_{\hat{\mathscr{B}} \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}} M_{\mathrm{gtls2}}(D, \hat{\mathscr{B}}) \tag{GTLS2}$$

reduces to the previously defined GTLS problem. The right weight matrix W_{r}, however, gives additional degrees of freedom for choosing an appropriate weighting pattern.

Note 3.17 (EWTLS with rank-one weight matrix Σ) Let $W_{\mathrm{l}} = \mathrm{diag}(w_{\mathrm{l}})$ and $W_{\mathrm{r}} = \mathrm{diag}(w_{\mathrm{r}})$, where $w_{\mathrm{l}} \in \mathbb{R}^{\mathtt{d}}_{+}$ and $w_{\mathrm{r}} \in \mathbb{R}^{N}_{+}$ are given vectors with positive elements. Then

$$M_{\mathrm{gtls2}}(D, \mathscr{B}) = \min_{\hat{D} \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}} \|\Sigma_{\mathrm{gtls2}} \odot (D - \hat{D})\|_{\mathrm{F}}, \qquad \text{where} \quad \Sigma_{\mathrm{gtls2}} = w_{\mathrm{l}} w_{\mathrm{r}}^{\top};$$

> i.e., the GTLS problem (GTLS2) with diagonal weight matrices W_{l} and W_{r} corresponds to an EWTLS problem with rank-one weight matrix Σ.

Theorem 3.18 (Solution of the GTLS problem). *Define the modified data matrix*

$$D_{\mathrm{m}} := \sqrt{W_{\mathrm{l}}} D \sqrt{W_{\mathrm{r}}},$$

and let $\hat{D}_{\mathrm{m,tls}}$, $\hat{\mathscr{B}}_{\mathrm{m,tls}} = \ker(R_{\mathrm{m,tls}}) = \mathrm{col\,span}(P_{\mathrm{m,tls}})$ be a TLS approximation of D_{m} in $\mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$. Then a solution of the GTLS problem (GTLS2) is

$$\hat{D}_{\mathrm{gtls2}} = \left(\sqrt{W_{\mathrm{l}}}\right)^{-1} \hat{D}_{\mathrm{m,tls}} \left(\sqrt{W_{\mathrm{r}}}\right)^{-1},$$

$$\hat{\mathscr{B}}_{\mathrm{gtls2}} = \ker\left(R_{\mathrm{m,tls}} \sqrt{W_{\mathrm{l}}}\right) = \mathrm{col\,span}\left(\left(\sqrt{W_{\mathrm{l}}}\right)^{-1} P_{\mathrm{m,tls}}\right),$$

and the corresponding GTLS misfit is

$$\left\| D - \hat{D}_{\mathrm{gtls2}} \right\|_{\mathrm{F}} = \left\| D_{\mathrm{m}} - \hat{D}_{\mathrm{m,tls}} \right\|_{\mathrm{F}}.$$

A GTLS solution always exists. It is unique if and only if $\hat{\mathscr{B}}_{\text{m,tls}}$ is unique.

Proof. The cost function of the GTLS problem (GTLS2) can be written as

$$\big\|\sqrt{W_{\text{l}}}(D-\hat{D})\sqrt{W_{\text{r}}}\big\|_{\text{F}} = \big\|D_{\text{m}} - \underbrace{\sqrt{W_{\text{l}}}\hat{D}\sqrt{W_{\text{r}}}}_{\hat{D}_{\text{m}}}\big\|_{\text{F}} =: \big\|D_{\text{m}}-\hat{D}_{\text{m}}\big\|_{\text{F}},$$

which is the cost function of a TLS problem for the modified data matrix D_{m}. Because the mapping $\hat{D}_{\text{m}} \mapsto \hat{D}$, defined by $\hat{D}_{\text{m}} = \sqrt{W_{\text{l}}}\hat{D}\sqrt{W_{\text{r}}}$, is one-to-one, the above transformation shows that the GTLS problem for D is equivalent to the TLS problem for D_{m}. The GTLS solution $\hat{D}_{\text{gtls2}}$ is recovered from the TLS solution $\hat{D}_{\text{m,tls}}$ by the inverse transformation $\hat{D}_{\text{gtls2}} = (\sqrt{W_{\text{l}}})^{-1}\hat{D}_{\text{m,tls}}(\sqrt{W_{\text{r}}})^{-1}$. We have

$$\mathscr{B}_{\text{gtls2}} = \operatorname{col\,span}\big(\hat{D}_{\text{gtls2}}\big) = \operatorname{col\,span}\Big(\big(\sqrt{W_{\text{l}}}\big)^{-1}\hat{D}_{\text{m,tls}}\Big) = \operatorname{col\,span}\Big(\big(\sqrt{W_{\text{l}}}\big)^{-1}P_{\text{m,tls}}\Big),$$

and it follows that $\mathscr{B}_{\text{gtls2}} = \ker\big(R_{\text{m,tls}}\sqrt{W_{\text{l}}}\big)$. □

3.4 Misfit Computation

The WTLS problem is a double minimization problem with an inner minimization, the search for the best approximation of the data in a given model, and an outer minimization, the search for the model. First, we solve the inner minimization problem: the misfit computation (Mwtls).

Since the model is linear, (Mwtls) is a convex quadratic optimization problem with a linear constraint. Therefore, it has an analytic solution. In order to give explicit formulas for the optimal approximation $\hat{D}_{\text{wtls}}$ and $M_{\text{wtls}}(D,\mathscr{B})$, however, we need to choose a particular parameterization of the given model $\mathscr{B}$. Three parameterizations—kernel, image, and input/output—are described in Section 3.2. We state the results for the kernel and the image representations. The results for the input/output representation follow from the given ones by the substitutions $R \mapsto \begin{bmatrix} X^\top & -I \end{bmatrix}$ and $P \mapsto \left[\begin{smallmatrix} I \\ X^\top \end{smallmatrix}\right]$.

Theorem 3.19 (WTLS misfit computation, kernel representation version). *Let $\ker(R)$ be a minimal kernel representation of $\mathscr{B} \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$. The best WTLS approximation of D in $\mathscr{B}$, i.e., the solution of (Mwtls), is*

$$\hat{d}_{\text{wtls},i} = \big(I - W_i^{-1}R^\top(RW_i^{-1}R^\top)^{-1}R\big)d_i, \qquad \text{for } i = 1,\ldots,N,$$

with the corresponding misfit

$$M_{\text{wtls}}\big(D,\ker(R)\big) = \sqrt{\sum_{i=1}^{N} d_i^\top R^\top (RW_i^{-1}R^\top)^{-1}Rd_i}. \qquad (\text{Mwtls}_R)$$

Proof. Define the correction $\Delta D := D - \hat{D}$. The misfit computation problem (Mwtls) is equivalent to

$$\min_{\Delta d_1,\ldots,\Delta d_N} \sum_{i=1}^{N} \Delta d_i^\top W_i \Delta d_i \quad \text{subject to} \quad R(d_i - \Delta d_i) = 0, \quad \text{for } i = 1,\ldots,N.$$

Observe that this is a separable weighted least norm problem; i.e., it involves N independent weighted least norm subproblems. Define $E := RD$ and let $E =: \begin{bmatrix} e_1 & \cdots & e_N \end{bmatrix}$. Consider the ith subproblem

$$\min_{\Delta d_i} \Delta d_i^\top W_i \Delta d_i \quad \text{subject to} \quad R\Delta d_i = e_i.$$

Its solution is

$$\Delta d_i^* = W_i^{-1} R^\top (R W_i^{-1} R^\top)^{-1} R d_i,$$

so that the squared minimum misfit is

$$M_{\text{wtls}}^2(D, \mathscr{B}) = \sum_{i=1}^N \Delta d_i^{*\top} W_i \Delta d_i^* = \sum_{i=1}^N d_i^\top R^\top (R W_i^{-1} R^\top)^{-1} R d_i. \quad \square$$

Next, we state the result in the special case of a GTLS problem.

Corollary 3.20 (GTLS misfit computation, kernel representation version). *Let* $\ker(R)$ *be a minimal kernel representation of* $\mathscr{B} \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$*. The best GTLS approximation of* D *in* $\mathscr{B}$ *is*

$$\hat{D}_{\text{gtls}} = \big(I - W^{-1} R^\top (R W^{-1} R^\top)^{-1} R\big) D,$$

with the corresponding misfit

$$M_{\text{gtls}}\big(D, \ker(R)\big) = \sqrt{\operatorname{trace}\,(D^\top R^\top (R W^{-1} R^\top)^{-1} R D)}. \qquad \text{(Mgtls}_R\text{)}$$

The image representation is dual to the kernel representation. Correspondingly, the misfit computation with kernel and with image representations of the model are dual problems. The kernel representation leads to a weighted least norm problem and the image representation leads to an WLS problem.

Theorem 3.21 (WTLS misfit computation, image representation version). *Let* $\operatorname{col\,span}(P)$ *be a minimal image representation of* $\mathscr{B} \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$*. The best WTLS approximation of* D *in* $\mathscr{B}$ *is*

$$\hat{d}_{\text{wtls},i} = P(P^\top W_i P)^{-1} P^\top W_i d_i, \quad \text{for } i = 1, \ldots, N,$$

with the corresponding misfit

$$M_{\text{wtls}}\big(D, \operatorname{col\,span}(P)\big) = \sqrt{\sum_{i=1}^N d_i^\top W_i \big(I - P(P^\top W_i P)^{-1} P^\top W_i\big) d_i}. \qquad \text{(Mwtls}_P\text{)}$$

Proof. In terms of the image representation $\hat{D} = PL$, with $L =: \begin{bmatrix} l_1 & \cdots & l_N \end{bmatrix}$, problem (Mwtls) is equivalent to

$$\min_{l_1, \ldots, l_N} \sum_{i=1}^N (d_i - \hat{d}_i)^\top W_i (d - \hat{d}_i) \quad \text{subject to} \quad \hat{d}_i = P l_i, \quad \text{for } i = 1, \ldots, N,$$

which is a separable WLS problem. The solution of the ith subproblem

$$\min_{l_i}(d_i - \hat{d}_i)^\top W_i(d - \hat{d}_i) \quad \text{subject to} \quad \hat{d}_i = Pl_i$$

is $l_i^* = (P^\top W_i P)^{-1} P^\top W_i d_i$, so that $\hat{d}_i^* = P(P^\top W_i P)^{-1} P^\top W_i d_i$. □

Corollary 3.22 (GTLS misfit computation, image representation version). *Let* $\operatorname{col\,span}(P)$ *be a minimal image representation of* $\mathscr{B} \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$. *The best GTLS approximation of* D *in* $\mathscr{B}$ *is*

$$\hat{D}_{\text{gtls}} = P(P^\top W P)^{-1} P^\top W D,$$

with the corresponding minimum value of the misfit function

$$M_{\text{gtls}}\big(D, \operatorname{col\,span}(P)\big) = \sqrt{\operatorname{trace}\big(D^\top W\big(I - P(P^\top W P)^{-1} P^\top W\big) D\big)}. \qquad (\text{Mgtls}_P)$$

3.5 Misfit Minimization*

In Section 3.4, we solved the inner minimization problem of the WTLS problem—misfit computation. Now we consider the remaining problem—the minimization with respect to the model parameters. This is a nonconvex optimization problem that in general has no closed form solution. For this reason, numerical optimization methods are employed for its solution. First, we review the methods proposed in the literature. Then we present in detail three algorithms. In Section 3.6, we compare their performance on test examples.

Algorithms Proposed in the Literature

Special optimization methods for the WTLS problem are proposed in [DM93, WAH+97, PR02, MMH03]. The Riemannian singular value decomposition (RiSVD) framework of De Moor [DM93] is derived for the STLS problem and includes the EWTLS problem with complexity specification $\mathtt{m} = \mathtt{d} - 1$ as a special case. The restriction to more general WTLS problems comes from the fact that the RiSVD framework is derived for matrix approximation problems with rank reduction by one and with diagonal weight matrices. In [DM93], an algorithm resembling the inverse power iteration algorithm is proposed for computing the RiSVD. The method, however, has no proven convergence properties.

The maximum likelihood principle component analysis (MLPCA) method of Wentzell et al. [WAH+97] is an alternating least squares algorithm. It applies to the general WTLS problems and is globally convergent. The convergence rate, however, is linear and the method can be rather slow in practice.

The method of Premoli and Rastello [PR02] is a heuristic for solving the first order optimality condition of (WTLS). A solution of a nonlinear equation is searched for instead of a minimum point of the original optimization problem. The method is locally convergent with superlinear convergence rate. The method is not globally convergent and the region of convergence around a minimum point can be rather small in practice.

The weighted low-rank approximation (WLRA) framework of Manton, Mahony, and Hua et al. [MMH03] proposes specialized optimization methods on a Grassman manifold. The least squares nature of the problem is not exploited by the algorithms proposed in [MMH03].

Table 3.1. *Model representations and optimization algorithms used in the methods of [DM93, WAH+97, PR02, MMH03].*

Method	Representation	Algorithm
RiSVD	kernel	inverse power iteration
MLPCA	image	alternating projections
PR	input/output	iteration based on heuristic linearization
WLRA	kernel	Newton method

The RiSVD, MLPCA, Premoli–Rastello (PR), and WLRA methods differ in the parameterization of the model and the optimization algorithm that are used.

Table 3.1 summarizes the model parameterizations and optimization algorithms for the different methods.

Alternating Least Squares Algorithm

The alternating least squares method is based on an image representation of the model, i.e., $\mathscr{B} = \operatorname{col}\operatorname{span}(P)$, where $P \in \mathbb{R}^{\mathtt{d}\times\mathtt{m}}$. First we rewrite (WTLS) in the form

$$\min_{\hat{D}\in\mathscr{B}\in\mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}} \sqrt{\operatorname{vec}^\top(D-\hat{D})W\operatorname{vec}(D-\hat{D})}, \quad \text{where} \quad W := \operatorname{diag}(W_1,\ldots,W_N).$$

The constraint $\hat{D} \in \mathscr{B}$ is equivalent to $\hat{D} = PL$, where $L \in \mathbb{R}^{\mathtt{m}\times N}$, so that (WTLS) can further be written as

$$\min_{P\in\mathbb{R}^{\mathtt{d}\times\mathtt{m}}} \min_{L\in\mathbb{R}^{\mathtt{m}\times N}} \operatorname{vec}^\top(D-PL)W\operatorname{vec}(D-PL). \tag{WTLS$_P$}$$

The two problems

$$\min_{P\in\mathbb{R}^{\mathtt{d}\times\mathtt{m}}} \operatorname{vec}^\top(D-PL)W\operatorname{vec}(D-PL), \tag{RLX1}$$

$$\min_{L\in\mathbb{R}^{\mathtt{m}\times N}} \operatorname{vec}^\top(D-PL)W\operatorname{vec}(D-PL), \tag{RLX2}$$

derived from (WTLS$_P$) by fixing, respectively, L and P to given values, are WLS problems and therefore can be solved in closed form. They can be viewed as relaxations of the non-convex problem (WTLS$_P$) to convex problems. Note that (RLX2) is the misfit computation of (WTLS$_P$) and the solution for the case, where W is block-diagonal is (Mwtls$_P$).

The alternating least squares method is an iterative method that alternatively solves (RLX1) and (RLX2) with, respectively, L and P fixed to the solution of the previously solved relaxation problem. The resulting algorithm is Algorithm 3.1.

Algorithm 3.1 is given for an arbitrary positive definite weight matrix W. When W is block-diagonal (WTLS problem) or diagonal (EWTLS problem), Algorithm 3.1 can be implemented more efficiently, taking into account the structure of W. For example, in the WTLS case, the solution of problem (RLX1) can be computed efficiently as follows:

$$l_i = (P^\top W_i P)^{-1} P^\top W_i d_i, \qquad \text{for } i = 1,\ldots,N,$$

Algorithm 3.1 Alternating least squares algorithm for WTLS problem `wtlsap`

Input: data matrix $D \in \mathbb{R}^{\mathtt{d}\times N}$, weight matrix $W \in \mathbb{R}^{N\mathtt{d}\times N\mathtt{d}}$, complexity specification $\mathtt{m}$ for the WTLS approximation, and relative convergence tolerance ε.

1: Initial approximation: compute a TLS approximation of D in $\mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$ and let $P^{(0)} := \hat{P}_{\text{tls}}$, $D^{(0)} := \hat{D}_{\text{tls}}$, $L^{(0)} := \hat{L}_{\text{tls}}$, where $\hat{L}_{\text{tls}}$ is the matrix, such that $\hat{D}_{\text{tls}} = \hat{P}_{\text{tls}}\hat{L}_{\text{tls}}$.

2: $k := 0$.

3: **repeat**

4: Compute the solution of (RLX1)

$$\operatorname{vec}(L^{(k+1)}) = \left(\mathbf{P}^{(k)\top} W \mathbf{P}^{(k)}\right)^{-1} \mathbf{P}^{(k)\top} W \operatorname{vec}(D),$$

where $\mathbf{P}^{(k)} = I_N \otimes P^{(k)}$.

5: $M^{(k)}_{\text{wtls}} = \sqrt{\operatorname{vec}^\top (D - P^{(k)}L^{(k+1)}) W \operatorname{vec}(D - P^{(k)}L^{(k+1)})}$.

6: $k = k + 1$.

7: Compute the solution of (RLX2)

$$\operatorname{vec}(P^{(k)}) = \left(\mathbf{L}^{(k)\top} W \mathbf{L}^{(k)}\right)^{-1} \mathbf{L}^{(k)\top} W \operatorname{vec}(D),$$

where $\mathbf{L}^{(k)} = L^{(k)\top} \otimes I_{\mathtt{d}}$.

8: $M^{(k)}_{\text{wtls}} = \sqrt{\operatorname{vec}^\top (D - P^{(k)}L^{(k)}) W \operatorname{vec}(D - P^{(k)}L^{(k)})}$.

9: **until** $|M^{(k)}_{\text{wtls}} - M^{(k-1)}_{\text{wtls}}|/M^{(k)}_{\text{wtls}} < \varepsilon$.

Output: $\hat{P}_{\text{wtls}} := P^{(k)}$ and $\hat{D}_{\text{wtls}} = P^{(k)}L^{(k)}$.

where l_i is the ith column of L. Similarly, (Mwtls$_P$) takes into account the block-diagonal structure of W for efficient solution of problem (RLX2).

The alternating least squares algorithm monotonically decreases the cost function value, so that it is globally convergent. The convergence rate, however, is linear and depends on the distribution of the singular values of D [MMH03, IV.A]. With a pair of singular values that are close to each other the convergence rate can be rather low.

Note 3.23 (Initial approximation) For simplicity the initial approximation is chosen to be a TLS approximation of D in $\mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$. The weight matrix W, however, can partially be taken into account in the computation of the initial approximation. Let the vector $w \in \mathbb{R}^{\mathtt{d}N}_{+}$ be the diagonal of W and let $\hat{W} \in \mathbb{R}^{\mathtt{d}\times N}$ be defined as $\hat{W} := \operatorname{vec}^{-1}(w)$, where the mapping $w \mapsto \hat{W}$ is denoted by vec^{-1}. The EWTLS problem with weights $\Sigma_{ij} := \sqrt{\hat{W}_{ij}}$ can be viewed as an approximation of the WTLS problem that takes into account only the diagonal elements of W and ignores all off-diagonal elements. In the EIV setting, this is equivalent to ignoring the cross covariance information.

The solution of the EWTLS problem, however, still requires local optimization methods that have to be initialized from a given initial approximation. An additional simplification that results in a GTLS problem and therefore in an exact solution method is to approximate the matrix Σ by a rank-one matrix; see Note 3.17.

Algorithm of Premoli and Rastello

The algorithm of Premoli and Rastello uses the input/output representation $\mathscr{B}_{\text{i/o}}(X)$ of the model. We have

$$D \in \mathscr{B} \in \mathscr{L}_{\mathtt{m},0}^{\mathtt{d}} \quad \Longleftrightarrow \quad AX = B, \quad \text{where } D^\top =: \begin{bmatrix} A & B \end{bmatrix},$$

where $\operatorname{col\,dim}(A) = \mathtt{m}$, $\operatorname{col\,dim}(B) = \mathtt{p}$, and $\mathtt{d} = \mathtt{m} + \mathtt{p}$. The WTLS misfit as a function of the parameter X is (see (Mwtls$_R$))

$$M_{\text{wtls}}(X) = \sqrt{\sum_{i=1}^{N} d_i^\top R^\top (R W_i^{-1} R^\top)^{-1} R d_i}, \qquad \text{where } R := \begin{bmatrix} X^\top & -I \end{bmatrix}. \quad \text{(Mwtls}_X\text{)}$$

Then (WTLS) becomes the unconstrained optimization problem

$$\hat{X}_{\text{wtls}} = \arg\min_X M_{\text{wtls}}(X). \qquad \text{(WTLS}_X\text{)}$$

Define $V_i := W_i^{-1}$ and the residual matrix

$$E(X) := AX - B, \qquad E^\top(X) =: \begin{bmatrix} e_1(X) & \cdots & e_N(X) \end{bmatrix},$$

and partition d_i and V_i as follows:

$$d_i =: \begin{bmatrix} a_i \\ b_i \end{bmatrix} \begin{matrix} \mathtt{m} \\ \mathtt{p} \end{matrix}, \qquad V_i =: \begin{bmatrix} V_{a,i} & V_{ab,i} \\ V_{ba,i} & V_{b,i} \end{bmatrix} \begin{matrix} \mathtt{m} \\ \mathtt{p} \end{matrix}.$$

The first order optimality condition $M'_{\text{wtls}}(X) = 0$ of (WTLS$_X$) is (see Appendix A.1)

$$2 \sum_{i=1}^{N} \Big(a_i e_i^\top(X) \Gamma_i^{-1}(X) - (V_{a,i} X - V_{ab,i}) \Gamma_i^{-1}(X) e_i(X) e_i^\top(X) \Gamma_i^{-1}(X) \Big) = 0,$$

where

$$\Gamma_i(X) := R W_i^{-1} R^\top.$$

We aim to find a solution of $M'_{\text{wtls}}(X) = 0$ that corresponds to a solution of the WTLS problem, i.e., to a global minimum point of M_{wtls}.

The algorithm proposed in [PR02] uses an iterative procedure starting from an initial approximation $X^{(0)}$ and generating a sequence of approximations $X^{(k)}$, $k = 0, 1, 2, \ldots$, that approaches a solution of $M'_{\text{wtls}}(X) = 0$. The iteration is implicitly defined by the equation

$$F(X^{(k+1)}, X^{(k)}) = 0, \qquad \text{(LIN RLX)}$$

where

$$\begin{aligned} F(X^{(k+1)}, X^{(k)}) := 2 \sum_{i=1}^{N} \Big(& a_i \big(X^{(k+1)\top} a_i - b_i \big)^\top \Gamma_i^{-1}(X^{(k)}) \\ & - (V_{a,i} X^{(k+1)} - V_{ab,i}) \Gamma_i^{-1}(X^{(k)}) e_i(X^{(k)}) e_i^\top(X^{(k)}) \Gamma_i^{-1}(X^{(k)}) \Big). \end{aligned}$$

Note that $F(X^{(k+1)}, X^{(k)})$ is linear in $X^{(k+1)}$, so that $X^{(k+1)}$ can be computed in a closed form as a function of $X^{(k)}$. Equation (LIN RLX) with $X^{(k)}$ fixed can be viewed as a linear relaxation of the first order optimality condition of (WTLS$_X$), which is a highly nonlinear equation.

An outline of the PR algorithm is given in Algorithm 3.2. In general, solving the equation (LIN RLX) for $X^{(k+1)}$ requires vectorization. The identity $\operatorname{vec}(AXB) = (B^\top \otimes A)\operatorname{vec}(X)$ is used in order to transform (LIN RLX) to the classical system of equations $G(X^{(k)})\operatorname{vec}(X^{(k+1)}) = h(X^{(k)})$, where G and h are given in the algorithm.

Algorithm 3.2 Algorithm of Premoli and Rastello for the WTLS problem `wtlspr`

Input: the data matrix $D \in \mathbb{R}^{\mathtt{d}\times N}$, the weight matrices $\{W_i\}_{i=1}^N$, a complexity specification $\mathtt{m}$ for the WTLS approximation, and a convergence tolerance ε.

1: Initial approximation: compute a TLS approximation $\mathscr{B}_{\text{i/o}}(\hat{X}_{\text{tls}})$ of D in $\mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}$, and let $X^{(0)} := \hat{X}_{\text{tls}}$. (See Note 3.23.)

2: Define: $D =: \begin{bmatrix} a_1 & \cdots & a_N \\ b_1 & \cdots & b_N \end{bmatrix} \begin{matrix} \mathtt{m} \\ \mathtt{p} \end{matrix}$, where $\mathtt{p} := \mathtt{d} - \mathtt{m}$.

3: $k := 0$.

4: **repeat**

5: Let $G = 0_{\mathtt{mp}\times\mathtt{mp}}$ and $h = 0_{\mathtt{mp}\times 1}$.

6: **for** $i = 1, \ldots, N$ **do**

7: $e_i := X^{(k)\top} a_i - b_i$.

8: $V_1 := W_i^{-1}$.

9: $M_i := \left(\begin{bmatrix} X^{(k)} \\ -I \end{bmatrix}^\top V_i \begin{bmatrix} X^{(k)} \\ -I \end{bmatrix} \right)^{-1}$.

10: $y_i := M_i e_i$.

11: $G = G + M_i \otimes (a_i a_i^\top) - (y_i y_i^\top) \otimes V_{a,i}$.

12: $h = h + \operatorname{vec}(a_i b_i^\top M_i - V_{ab,i} y_i y_i^\top)$.

13: **end for**

14: Solve the system $Gx = h$ and let $X^{(k+1)} := \operatorname{vec}^{-1}(x)$.

15: $k := k + 1$.

16: **until** $\|X^{(k)} - X^{(k-1)}\|_{\text{F}} / \|X^{(k)}\|_{\text{F}} < \varepsilon$

Output: $\hat{X}_{\text{wtls}} := X^{(k)}$.

Note 3.24 (Relation to Gauss–Newton-type algorithms) Algorithm 3.2 is *not* a Gauss–Newton-type algorithm for solving the first order optimality condition because the approximation F is not the first order truncated Taylor series of M'_{wtls}; it is a different linear approximation. The choice makes the derivation of the algorithm simpler but complicates the convergence analysis.

Note 3.25 (Convergence properties) Algorithm 3.2 is proven to be locally convergent with a superlinear convergence rate; see [MRP$^+$05, Section 5.3]. Moreover, the convergence rate tends to quadratic as the approximation gets closer to a minimum point. The algorithm, however, is not globally convergent, and simulation results suggest that the region of convergence to a minimum point could be rather small. This requires a good initial approximation for convergence.

An Algorithm Based on Classical Local Optimization Methods

Both the alternating least squares and the PR algorithms are heuristic optimization methods. Next, we describe an algorithm for the WTLS problem based on classical local optimization methods. The classical local optimization methods have by now reached a high level of maturity. In particular, their convergence properties are well understood, while the convergence properties of the RiSVD, MLPCA, and PR methods are still not.

In order to apply a classical optimization algorithm for the solution of (WTLS), first we have to choose a parameterization of the model. Possible parameterizations are given by the kernel, image, and input/output representations. For reasons to be discussed later (see Note 3.26), we choose the input/output representation (I/O repr), defined on page 34, so that the considered problem is (WTLS$_X$).

A quasi-Newton-type method requires an evaluation of the cost function $M_{\text{wtls}}(X)$ and its first derivative $M'_{\text{wtls}}(X)$. Both the misfit and its first derivative are available in closed form, so that their evaluation is a matter of numerical implementation of the involved operations. The computational steps are summarized in Algorithm 3.3. The proposed algorithm, based on a classical optimization method, is outlined in Algorithm 3.4.

Algorithm 3.3 WTLS cost function and first derivative evaluation `qncostderiv`

Input: $D \in \mathbb{R}^{\mathtt{d}\times N}$, $\{W_i\}_{i=1}^N$, $\mathtt{m}$, and X.

1: Define: $D =: \begin{bmatrix} a_1 & \cdots & a_N \\ b_1 & \cdots & b_N \end{bmatrix} \begin{matrix} \mathtt{m} \\ \mathtt{p} \end{matrix}$, where $\mathtt{p} := \mathtt{d} - \mathtt{m}$.
2: Let $f = 0_{1\times 1}$ and $f' = 0_{\mathtt{m}\times\mathtt{p}}$.
3: **for** $i = 1, \ldots, N$ **do**
4: $\quad e_i := X^\top a_i - b_i$.
5: $\quad V_1 := W_i^{-1}$.
6: $\quad$ Solve the system $\left(\left[\begin{smallmatrix} X \\ -I \end{smallmatrix} \right]^\top V_i \left[\begin{smallmatrix} X \\ -I \end{smallmatrix} \right] \right) y_i = e_i$.
7: $\quad f = f + e_i^\top y_i$.
8: $\quad f' = f' + a_i y_i^\top - (V_{a,i} X - V_{ab,i}) y_i y_i^\top$.
9: **end for**

Output: $M_{\text{wtls}}(X) = f$, $M'_{\text{wtls}}(X) = 2f'$.

Algorithm 3.4 Algorithm for WTLS based on classical local optimization `wtlsopt`

Input: the data matrix $D \in \mathbb{R}^{\mathtt{d}\times N}$, the weight matrices $\{W_i\}_{i=1}^N$, a complexity specification $\mathtt{m}$ for the WTLS model, and a convergence tolerance ε.

1: Initial approximation: compute a TLS approximation $\mathscr{B}_{\text{i/o}}(\hat{X}_{\text{tls}})$ of D in $\mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$, and let $X^{(0)} := \hat{X}_{\text{tls}}$. (See Note 3.23.)
2: Execute a standard optimization algorithm, e.g., the Broyden, Fletcher, Goldfarb, and Shanno (BFGS) quasi-Newton method, for the minimization of M_{wtls} over X with initial approximation $X^{(0)}$ and with cost function and first derivative evaluation performed via Algorithm 3.3. Let $\hat{X}$ be the approximation found by the optimization algorithm upon convergence.

Output: $\hat{X}_{\text{wtls}} = \hat{X}$.

The optimization problem (WTLS$_X$) is a nonlinear least squares problem; i.e.,

$$M_{\text{wtls}}(X) = F^\top(X)F(X)$$

for certain $F : \mathbb{R}^{\mathtt{m}\times\mathtt{p}} \to \mathbb{R}^{N\mathtt{p}}$. Therefore, the use of special optimization methods such as the Levenberg–Marquardt method is preferable. The vector $F(X)$, however, is computed numerically, so that the Jacobian $J(X) := \left[F_i/x_j\right]$, where $x = \operatorname{vec}(X)$, cannot be found in closed form. A possible workaround for this problem is proposed in [GP96], where an approximation called quasi-Jacobian is used instead. The quasi-Jacobian can be evaluated in a similar way to the one for the gradient, which allows us to use the Levenberg–Marquardt method for the solution of the WTLS problem.

Note 3.26 (Kernel vs. input/output representation) In Note 3.16 we comment that an optimal approximation $\hat{\mathscr{B}}$ might have no input/output representation (I/O repr). In practice, even when such a representation exists, the parameter $\hat{X}$ might be large, which might cause numerical problems because of ill conditioning. In this respect the use of a kernel or image representation is preferable over an input/output representation.

The input/output representation (I/O repr), however, has the advantage that the parameter X is unique, while the parameters R and P in the kernel and image representations are not. The misfit M_{wtls} depends only on $\ker(R)$ and $\operatorname{col}\operatorname{span}(P)$ and not on all elements of R and P. This makes the optimization problems $\min_R M_{\text{wtls}}(R)$ and $\min_P M_{\text{wtls}}(P)$ more difficult than (WTLS$_X$). Additional constraints such as $RR^\top = I$ and $P^\top P = I$ have to be imposed and the misfit and its derivative have to be derived as a function of a unique parameterization of $\ker(R)$ and $\operatorname{col}\operatorname{span}(P)$. The mathematical structure appropriate to treat this type of problem is the Grassman manifold. Classical optimization methods for optimization over a Grassman manifold are proposed in [MMH03].

3.6 Simulation Examples

We outlined the following algorithms for the WTLS problem:

MLPCA—the alternating least squares algorithm,

PR—the algorithm of Premoli and Rastello,

QN—the algorithm based on a quasi-Newton optimization method, and

LM—the algorithm based on the Levenberg–Marquardt optimization method.

In this section, we compare the performance of the algorithms on a simulation example. The considered model class is $\mathscr{L}^4_{2,0}$ and the experiment has $N = 25$ outcomes. The data $D \in \mathbb{R}^{4\times 25}$, used in the example, is simulated according to the WTLS EIV model with a true model $\mathscr{B}_{\text{i/o}}(\bar{X})$, where $\bar{X} \in \mathbb{R}^{2\times 2}$ is a random matrix, and with random positive definite weight matrices $\{W_i\}_{i=1}^{25}$.

The algorithms are compared on the basis of the

- achieved misfit $M_{\text{wtls}}(D, \hat{\mathscr{B}})$, where $\hat{\mathscr{B}}$ is the computed approximate model;
- number of iterations needed for convergence to the specified convergence tolerance;

Table 3.2. *Simulation results comparing four algorithms for the WTLS problem.*

Method	MLPCA	PR	QN	LM	GTLS	TLS	WLS
misfit	0.4687	0.4687	0.4687	0.4687	0.7350	0.7320	1.0500
error	0.2595	0.2582	0.2581	0.2582	0.4673	0.4698	0.5912
# iter.	51	6	5	4	—	—	—
# fun. eval.	—	—	18	29	—	—	—
megaflops	54	0.06	0.11	0.15	0.010	0.005	0.010
time, sec	2.29	0.40	0.59	0.54	0.004	0.002	0.003

- execution time, measured in seconds;
- relative error of approximation $\|\bar{X} - \hat{X}\|_{\mathrm{F}} / \|\hat{X}\|_{\mathrm{F}}$, where $\hat{X}$ is such that $\hat{\mathscr{B}} = \mathscr{B}_{\mathrm{i/o}}(\hat{X})$;
- computational cost, measured in floating point operations (flops); and
- number of cost function evaluations.

The criteria that are not relevant for a method are marked with "—". The experiment is repeated 10 times with different noise realizations and the averaged results are shown in Table 3.2.

In addition to the four WTLS algorithms, the table shows the results for the GTLS, TLS, and weighted least squares (WLS) methods. The GTLS method is applied by taking W to be the average $1/N \sum_{i=1}^{N} W_i$ of the weight matrices and the WLS method uses only the information $\{W_{b,i}\}_{i=1}^{N}$, where $W_i =: \left[\begin{smallmatrix} W_{a,i} & W_{ab,i} \\ * & W_{b,i} \end{smallmatrix}\right]$ and $W_{b,i} \in \mathbb{R}^{\mathtt{p} \times \mathtt{p}}$.

The results indicate that although (in this example) the four WTLS algorithms converge to the same local minimum of the cost function, their convergence properties and numerical efficiency are different. In order to achieve the same accuracy, the MLPCA algorithm needs more iterations than the other algorithms and as a result its computational efficiency is lower. The large number of iterations needed for convergence of the MLPCA algorithm is due to its linear convergence rate. In contrast, the convergence rate of the other algorithms is superlinear. They need approximately the same number of iterations and the execution times and numerical efficiency are similar. The PR algorithm is about two times more efficient than the standard local optimization algorithms, but it has no global convergence property, which is a serious drawback for most applications.

3.7 Conclusions

In this chapter we considered approximate modeling problems for linear static problems. The most general problem formulation that we treated is the WTLS problem, where the distance from each of the outcomes to the model is measured by a weighted norm with possibly different weight matrix. The WTLS problem is motivated by the relative error TLS and EIV estimation problems, where the weighted misfit naturally occurs and the weight matrices have interpretation, e.g., in the EIV estimation problem, the weight matrices are related to the covariance matrices of the measurement errors.

We showed the relations among the kernel, image, and input/output representations. Except for the nongeneric cases that occur in the input/output representation, one representation can be transformed into another one. Thus in misfit approximation problems the choice of the representation is a matter of convenience. Once computed, the approximate model $\hat{\mathscr{B}}$ can be transformed to any desired representation. We noted that the numerical algorithms proposed in the literature for the WTLS problem differ mainly because they use different model representations.

We showed how the TLS and GTLS problems are solved by an SVD. In linear algebra terms, these problems boil down to finding a closest rankdeficient matrix to a given matrix. The main tool for obtaining the solution is the matrix approximation lemma. The solution always exists but in certain nongeneric cases it could be nonunique.

The numerical solution of the general WTLS problem was approached in two steps:

1. compute analytically the misfit and
2. solve numerically the resulting optimization problem.

We showed two different expressions for the WTLS misfit: one for a model given by a kernel representation and the other for a model given by an image representation. In the first case, the misfit computation problem is a weighted least norm problem, and in the second case, it is a WLS problem.

Apart from the model representation used, another way of obtaining different computational algorithms is the use of a different optimization method in the second step. We outlined three algorithms: alternating least squares, the algorithm of Premoli and Rastello, and an algorithm based on classical local optimization methods. The alternating least squares algorithm has a linear convergence rate, which makes it rather slow compared to the other two algorithms, whose convergence rate is superlinear. The algorithm of Premoli and Rastello is not globally convergent, while the other two algorithms have this property. For these reasons, the algorithm based on standard local optimization methods is recommended for solving WTLS problems.

The proposed Algorithm 3.4, based on local optimization methods, however, uses the input/output representation $\mathscr{B}_{\text{i/o}}(X)$ of the model. As already discussed, this representation is not general; there are cases when the data D is such that the optimal approximation $\hat{\mathscr{B}}$ has no input/output representation $\mathscr{B}_{\text{i/o}}(\hat{X})$. Such cases are nongeneric, but even when $\hat{X}$ exists, it might be large, which might cause ill conditioning. A solution for this problem is to use a kernel or an image representation of the model. The choice of a kernel representation leads to the optimization methods presented in [MMH03].

Chapter 4

Structured Total Least Squares

The weighted total least squares problem generalizes the TLS problem by introducing weights in the misfit function. The structured total least squares problem generalizes the TLS problem by introducing a structure in the data matrix. The motivation for the special type of block-Hankel structure comes from system identification. The global total least squares problem is closely related to the structured total least squares problem with block-Hankel structured data matrix.

Section 4.1 gives an overview of the existing literature. Section 4.2 defines the type of structure we restrict ourselves to and derives an equivalent unconstrained optimization problem. The data matrix is partitioned into blocks and each of the blocks is block-Toeplitz/Hankel structured, unstructured, or exact. In Section 4.3, the properties of the equivalent problem are established. The special structure of the equivalent problem enables us to improve the computational efficiency of the numerical solution methods. By exploiting the structure, the computational complexity of the algorithms (local optimization methods) per iteration is linear in the sample size.

4.1 Overview of the Literature

History of the Problem

The origin of the STLS problem dates back to the work of Aoki and Yue [AY70a], although the name "structured total least squares" did not appear until 23 years later in the literature [DM93]. Aoki and Yue consider a single-input single-output system identification problem, where both the input and the output are noisy (EIV setting) and derive a maximum likelihood solution. Under the normality assumption for the measurement errors, a maximum likelihood estimate turns out to be a solution of the STLS problem. Furthermore, Aoki and Yue approach the optimization problem in a similar way to the one we adopt: they use classical nonlinear least squares minimization methods for solving an equivalent unconstrained problem.

The STLS problem occurs frequently in signal processing applications. Cadzow [Cad88] and Bresler and Macovski [BM86] propose heuristic solution methods that turn

out to be *suboptimal* [DM94, Section V] with respect to the STLS criterion. These methods, however, became popular because of their simplicity. For example, the method of Cadzow is an iterative method that alternates between unstructured low-rank approximation and structure enforcement, thereby requiring only SVD computations and manipulation of the matrix entries.

Abatzoglou, Mendel, and Harda [AMH91] are considered to be the first who formulated an STLS problem. They called their approach *constrained total least squares* and motivate the problem as an extension of the TLS method to matrices with structure. The solution approach adopted in [AMH91] is closely related to the one of Aoki and Yue. Again, an equivalent optimization problem is derived, but it is solved numerically via a Newton-type optimization method.

Shortly after the publication of the work on the constrained total least squares problem, De Moor lists many applications of the STLS problem and outlines a new framework for deriving analytical properties and numerical methods [DM93]. His approach is based on the Lagrange multipliers and the basic result is an equivalent problem, called *Riemannian singular value decomposition*, that can be considered as a "nonlinear" extension of the classical SVD. As an outcome of the new problem formulation, an iterative solution method based on the inverse power iteration is proposed.

Another algorithm for solving the STLS problem (even with ℓ_1 and ℓ_∞ norms in the cost function), called *structured total least norm*, is proposed by Rosen, Park, and Glick [RPG96]. In contrast to the approaches of Aoki and Yue and Abatzoglou et al., Rosen et al. solve the problem in its original formulation. The constraint is linearized around the current iteration point, which results in a linearly constrained least squares problem. In the algorithm of [RPG96], the constraint is incorporated in the cost function by adding a multiple of its residual norm.

The weighted low-rank approximation framework of Manton, Mahony, and Hua [MMH03] has been extended in Schuermans, Lemmerling, and Van Huffel [SLV04, SLV05] to *Hankel structured low-rank approximation* problems. All problem formulations and solution methods cited above, except for the ones in the WLRA framework, aim at rank reduction of the data matrix C by one. A generalization of the algorithm of Rosen et al. to problems with rank reduction by more than one is proposed by Van Huffel, Park, and Rosen [VPR96]. It involves, however, Kronecker products that unnecessarily inflate the dimension of the involved matrices. The solution methods in the WLRA framework [SLV04, SLV05] are also computationally demanding.

When dealing with a general affine structure, the constrained total least squares, Riemannian singular value decomposition, and structured total least norm methods have cubic computational complexity in the number of measurements. Fast algorithms with linear computational complexity are proposed by Lemmerling, Mastronardi, and Van Huffel [LMV00] and Mastronardi, Lemmerling, and Van Huffel [MLV00] for special STLS problems with data matrix $\mathscr{S}(p) =: \begin{bmatrix} A & b \end{bmatrix}$ that is Hankel or composed of a Hankel block A and an unstructured column b. They use the structured total least norm approach but recognize that a matrix appearing in the kernel subproblem of the algorithm has low displacement rank. This is exploited via the Schur algorithm.

Motivation for Our Work

The STLS solution methods outlined above point out the following issues:

- *structure:* the structure specification for the data matrix $\mathscr{S}(p)$ varies from general affine [AMH91, DM93, RPG96] to specific affine, such as Hankel/Toeplitz [LMV00], or Hankel/Toeplitz block augmented with an unstructured column [MLV00],
- *rank reduction:* all methods, except for the ones of [VPR96, SLV04, SLV05], reduce the rank of the data matrix by $d = 1$;
- *computational efficiency:* the efficiency varies from cubic for the methods that use a general affine structure to linear for the efficient methods of [LMV00, MLV00] that use a Hankel/Toeplitz-type structure.

No efficient algorithms exist for problems with block-Hankel/Toeplitz structure and rank reduction $d > 1$. In addition, the proposed methods lack a numerically reliable and robust software implementation that would make possible their use in real-life applications. Due to the above reasons, the STLS methods, although attractive for theoretical studies and relevant for applications, did not become popular for solving real-life problems.

The motivation for our work is to make the STLS method practically useful by deriving algorithms that are general enough for various applications and computationally efficient for real-life examples. We complement the theoretical study by a software implementation.

4.2 The Structured Total Least Squares Problem

The STLS problem

$$\min_{\hat{p}} \|p - \hat{p}\| \quad \text{subject to} \quad \operatorname{rank}\big(\mathscr{S}(\hat{p})\big) \leq n \tag{STLS}$$

defined in Section 2.7 is a structured low-rank approximation problem. The function $\mathscr{S} : \mathbb{R}^{n_p} \to \mathbb{R}^{m\times(n+d)}$, $m > n$, defines the structure of the data as follows: a matrix $C \in \mathbb{R}^{m\times(n+d)}$ is said to have structure defined by $\mathscr{S}$ if there exists a $p \in \mathbb{R}^{n_p}$, such that $C = \mathscr{S}(p)$. The vector p is called a parameter vector of the structured matrix C.

> The aim of the STLS problem is to perturb as little as possible a given parameter vector p by a vector Δp, so that the perturbed structured matrix $\mathscr{S}(p+\Delta p)$ becomes rank deficient with rank at most n.

A kernel representation of the rank deficiency constraint $\operatorname{rank}\big(\mathscr{S}(p)\big) = n$ yields the equivalent problem

$$\min_{RR^\top = I_d} \min_{\hat{p}} \|p - \hat{p}\| \quad \text{subject to} \quad R\mathscr{S}^\top(\hat{p}) = 0. \tag{STLS$_R$}$$

In this chapter, we use an input/output representation, so that the considered STLS problem is defined as follows.

Problem 4.1 (STLS). *Given a data vector $p \in \mathbb{R}^{n_p}$, a structure specification $\mathscr{S} : \mathbb{R}^{n_p} \to \mathbb{R}^{m\times(n+d)}$, and a rank specification n, solve the optimization problem*

$$\hat{X}_{\text{stls}} = \arg\min_{X,\Delta p} \|\Delta p\| \quad \textit{subject to} \quad \mathscr{S}(p-\Delta p)\begin{bmatrix} X \\ -I_d \end{bmatrix} = 0. \tag{STLS$_X$}$$

Define the matrices

$$X_{\text{ext}} := \begin{bmatrix} X \\ -I \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} A & B \end{bmatrix} := C := \mathscr{S}(p), \quad \text{where } A \in \mathbb{R}^{m\times n} \text{ and } B \in \mathbb{R}^{m\times d},$$

and note that $CX_{\text{ext}} = 0$ is equivalent to the structured system of equations $AX = B$.

The STLS problem is said to be affine structured if the function $\mathscr{S}$ is affine, i.e.,

$$\mathscr{S}(p) = S_0 + \sum_{i=1}^{n_p} S_i p_i, \quad \text{for all } p \in \mathbb{R}^{n_p} \text{ and for some } S_i,\ i = 1,\ldots,n_p. \tag{4.1}$$

In an affine STLS problem, the constraint $\mathscr{S}(p-\Delta p)X_{\text{ext}} = 0$ is bilinear in the decision variables X and Δp.

Lemma 4.2. *Let $\mathscr{S} : \mathbb{R}^{n_p} \to \mathbb{R}^{m\times(n+d)}$ be an affine function. Then*

$$\mathscr{S}(p-\Delta p)X_{\text{ext}} = 0 \quad \Longleftrightarrow \quad G(X)\Delta p = r(X),$$

where

$$G(X) := \begin{bmatrix} \operatorname{vec}\left((S_1 X_{\text{ext}})^\top\right) & \cdots & \operatorname{vec}\left((S_{n_p} X_{\text{ext}})^\top\right) \end{bmatrix} \in \mathbb{R}^{md\times n_p}, \tag{4.2}$$

and

$$r(X) := \operatorname{vec}\Big(\big(\mathscr{S}(p)X_{\text{ext}}\big)^\top\Big) \in \mathbb{R}^{md}.$$

Proof.

$$\begin{aligned} \mathscr{S}(p-\Delta p)X_{\text{ext}} = 0 &\iff \sum_{i=1}^{n_p} S_i \Delta p_i X_{\text{ext}} = \mathscr{S}(p)X_{\text{ext}} \\ &\iff \sum_{i=1}^{n_p} \operatorname{vec}\left((S_i X_{\text{ext}})^\top\right)\Delta p_i = \operatorname{vec}\Big(\big(\mathscr{S}(p)X_{\text{ext}}\big)^\top\Big) \\ &\iff G(X)\Delta p = r(X). \quad \square \end{aligned}$$

Using Lemma 4.2, we rewrite the affine STLS problem as follows:

$$\min_X \Big(\min_{\Delta p} \|\Delta p\| \quad \text{subject to} \quad G(X)\Delta p = r(X) \Big). \tag{4.3}$$

The inner minimization problem has an analytic solution, which allows us to derive an equivalent optimization problem.

Theorem 4.3 (Equivalent optimization problem for affine STLS). *Assuming that $n_p \geq md$, the affine STLS problem (4.3) is equivalent to*

$$\min_X f_0(X), \quad \textit{where} \quad f_0(X) := r^\top(X)\Gamma^\dagger(X)r(X) \quad \textit{and} \quad \Gamma(X) := G(X)G^\top(X). \tag{4.4}$$

Proof. Under the assumption $n_p \geq md$, the inner minimization problem of (4.3) is equivalent to a least norm problem. Its minimum point (as a function of X) is

$$\Delta p^*(X) = G^\top(X)\big(G(X)G^\top(X)\big)^\dagger r(X),$$

so that

$$f_0(X) = \Delta\big(p^*(X)\big)^\top \Delta p^*(X) = r^\top(X)\big(G(X)G^\top(X)\big)^\dagger r(X) = r^\top(X)\Gamma^\dagger(X)r(X). \qquad \square$$

The significance of Theorem 4.3 is that the constraint and the decision variable Δp in problem (4.3) are eliminated. Typically, the number of elements nd in X is much smaller than the number of elements n_p in the correction Δp. Thus the reduction in the complexity is significant.

The equivalent optimization problem (4.4) is a nonlinear least squares problem, so that classical optimization methods can be used for its solution. The optimization methods require a cost function and first derivative evaluation. In order to evaluate the cost function f_0 for a given value of the argument X, we need to form the weight matrix $\Gamma(X)$ and to solve the system of equations $\Gamma(X)y(X) = r(X)$. This straightforward implementation requires $O(m^3)$ floating point operation (flops). For large m (the applications that we aim at) this computational complexity becomes prohibitive.

It turns out, however, that for special affine structures $\mathscr{S}$, the weight matrix $\Gamma(X)$ has a block-Toeplitz and block-banded structure, which can be exploited for efficient cost function and first derivative evaluations. The set of structures of $\mathscr{S}$ for which we establish the special properties of $\Gamma(X)$ is specified next.

Assumption 4.4 (Flexible structure specification). *The structure specification $\mathscr{S} : \mathbb{R}^{n_p} \to \mathbb{R}^{m\times(n+d)}$ is such that for all $p \in \mathbb{R}^{n_p}$, the data matrix $\mathscr{S}(p) =: C =: \begin{bmatrix} A & B \end{bmatrix}$ is of the type $\mathscr{S}(p) = \begin{bmatrix} C^1 & \cdots & C^q \end{bmatrix}$, where C^l, for $l = 1, \ldots, q$, is block-Toeplitz, block-Hankel, unstructured, or exact and all block-Toeplitz/Hankel structured blocks C^l have equal row dimension K of the blocks.*

Assumption 4.4 says that $\mathscr{S}(p)$ is composed of blocks, each one of which is block-Toeplitz, block-Hankel, unstructured, or exact. A block C^l that is exact is not modified in the solution $\hat{C} := \mathscr{S}(p - \Delta p)$, i.e., $\hat{C}^l = C^l$. Assumption 4.4 is the essential structural assumption that we impose on the problem (STLS$_X$). As shown in Section 4.6, it is fairly general and covers many applications.

Example 4.5 Consider the following block-Toeplitz matrix:

$$C = \left[\begin{array}{c|c|c} \begin{matrix}5\\6\end{matrix} & \begin{matrix}3\\4\end{matrix} & \begin{matrix}1\\2\end{matrix} \\ \hline \begin{matrix}7\\8\end{matrix} & \begin{matrix}5\\6\end{matrix} & \begin{matrix}3\\4\end{matrix} \\ \hline \begin{matrix}9\\10\end{matrix} & \begin{matrix}7\\8\end{matrix} & \begin{matrix}5\\6\end{matrix} \end{array}\right]$$

with row dimension of the block $K = 2$. Next, we specify the matrices S_i that define via (4.1) an affine function $\mathscr{S}$, such that $C = \mathscr{S}(p)$ for certain parameter vector p. Let $==$ be the element-wise comparison operator

$$(A==B) := C, \qquad \text{for all } A, B \in \mathbb{R}^{m\times n}, \text{ where } C_{ij} := \begin{cases} 1 & \text{if } A_{ij} = B_{ij}, \\ 0 & \text{otherwise.} \end{cases}$$

Let E be the 6×3 matrix with all elements equal to 1 and define $S_0 := 0_{6\times 3}$ and $S_i := (C==iE)$, for $i = 1, \ldots, 10$. We have

$$C = \sum_{i=1}^{10} S_i i = S_0 + \sum_{i=1}^{10} S_i p_i =: \mathscr{S}(p), \qquad \text{with } p = \begin{bmatrix} 1 & 2 & \cdots & 10 \end{bmatrix}^\top.$$

The matrix C considered in the example is special; it allowed us to easily write down a corresponding affine function $\mathscr{S}$. Clearly with the constructed $\mathscr{S}$, *any* 6×3 block-Toeplitz matrix C with row dimension of the block $K = 2$ can be written as $C = \mathscr{S}(p)$ for certain $p \in \mathbb{R}^{10}$. ∎

We use the notation $\mathbf{n}_l$ for the number of *block* columns of the block C^l. For unstructured and exact blocks, $\mathbf{n}_l := 1$.

4.3 Properties of the Weight Matrix*

For the evaluation of the cost function f_0 of the equivalent optimization problem (4.4), we have to solve the system of equations $\Gamma(X)y(X) = r(X)$, where $\Gamma(X) \in \mathbb{R}^{md\times n_p}$ with both m and n_p large. In this section, we investigate the structure of the matrix $\Gamma(X)$. In the notation, occasionally we drop the explicit dependence of r and Γ on X.

Theorem 4.6 (Structure of the weight matrix Γ). *Consider the equivalent optimization problem (4.4) from Theorem 4.3. If, in addition to the assumptions of Theorem 4.3, the structure $\mathscr{S}$ is such that Assumption 4.4 holds, then the weight matrix $\Gamma(X)$ has the block-*

banded and block-Toeplitz structure

$$\Gamma(X) = \begin{bmatrix} \mathbf{\Gamma}_0 & \mathbf{\Gamma}_1^\top & \cdots & \mathbf{\Gamma}_s^\top & & \mathbf{0} \\ \mathbf{\Gamma}_1 & \ddots & \ddots & \ddots & \ddots & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \mathbf{\Gamma}_s^\top \\ \mathbf{\Gamma}_s & \ddots & \ddots & \ddots & \ddots & \vdots \\ & \ddots & \ddots & \ddots & \ddots & \mathbf{\Gamma}_1^\top \\ \mathbf{0} & & \mathbf{\Gamma}_s & \cdots & \mathbf{\Gamma}_1 & \mathbf{\Gamma}_0 \end{bmatrix} \in \mathbb{R}^{md \times md}, \tag{4.5}$$

where $\mathbf{\Gamma}_k \in \mathbb{R}^{dK \times dK}$*, for* $k = 0, 1, \ldots, s$*, and* $s = \max_{l=1,\ldots,q}(\mathbf{n}_l - 1)$*, where* $\mathbf{n}_l$ *is the number of block columns in the block* C^l *of the data matrix* $\mathscr{S}(p)$*.*

The proof is developed in a series of lemmas. First, we reduce the original problem with multiple blocks C^l (see Assumption 4.4) to three independent problems—one for the unstructured case, one for the block-Hankel case, and one for the block-Toeplitz case.

Lemma 4.7. *Consider a structure specification of the form*

$$\mathscr{S}(p) = \begin{bmatrix} \mathscr{S}^1(p^1) & \cdots & \mathscr{S}^q(p^q) \end{bmatrix}, \quad p^l \in \mathbb{R}^{n_{p,l}}, \quad \textstyle\sum_{l=1}^q n_{p,l} =: n_p,$$

where $p =: \operatorname{col}(p^1, \ldots, p^q)$ *and* $\mathscr{S}(p^l) := S_0^l + \sum_{i=1}^{n_{p,l}} S_i^l p_i^l$*, for all* $p^l \in \mathbb{R}^{n_{p,l}}$*,* $l = 1, \ldots, q$*. Then*

$$\Gamma(X) = \sum_{l=1}^{q} \Gamma^l(X), \tag{4.6}$$

where $\Gamma^l := G^l (G^l)^\top$*,* $G^l := \begin{bmatrix} \operatorname{vec}\left((S_1^l X_{\text{ext}}^l)^\top\right) & \cdots & \operatorname{vec}\left((S_{n_{p,l}}^l X_{\text{ext}}^l)^\top\right) \end{bmatrix}$*, and*

$$X_{\text{ext}} =: \operatorname{col}(X_{\text{ext}}^1, \ldots, X_{\text{ext}}^q), \quad \textit{with} \quad X_{\text{ext}}^l \in \mathbb{R}^{n_l \times d}, \quad \textstyle\sum_{l=1}^q n_l = n + d.$$

Proof. The result is a refinement of Lemma 4.2. Let $\Delta p =: \operatorname{col}(\Delta p^1, \ldots, \Delta p^q)$, where $\Delta p^l \in \mathbb{R}^{n_{p,l}}$, for $l = 1, \ldots, q$. We have

$$\begin{aligned} \mathscr{S}(p - \Delta p) X_{\text{ext}} = 0 &\iff \textstyle\sum_{l=1}^q \mathscr{S}^l(p^l - \Delta p^l) X_{\text{ext}}^l = 0 \\ &\iff \textstyle\sum_{l=1}^q \sum_{i=1}^{n_p} S_i^l \Delta p_i^l X_{\text{ext}}^l = \mathscr{S}(p) X_{\text{ext}} \\ &\iff \textstyle\sum_{l=1}^q G^l \Delta p^l = r(X) \\ &\iff \underbrace{\begin{bmatrix} G^1 & \cdots & G^q \end{bmatrix}}_{G(X)} \Delta p = r(X), \end{aligned}$$

so that $\Gamma = GG^\top = \sum_{l=1}^q G^l (G^l)^\top = \sum_{l=1}^q \Gamma^l$. ∎

Next, we establish the structure of Γ for an STLS problem with unstructured data matrix.

Lemma 4.8. *Let*

$$\mathscr{S}(p) := \begin{bmatrix} p_1 & p_2 & \cdots & p_{n+d} \\ p_{n+d+1} & p_{n+d+2} & \cdots & p_{2(n+d)} \\ \vdots & \vdots & & \vdots \\ p_{(m-1)(n+d)+1} & p_{(m-1)(n+d)+2} & \cdots & p_{m(n+d)} \end{bmatrix} \in \mathbb{R}^{m\times(n+d)}.$$

Then

$$\Gamma = I_m \otimes (X_{\text{ext}}^\top X_{\text{ext}}); \tag{4.7}$$

i.e., the matrix Γ has the structure (4.5) with $s = 0$ and $\mathbf{\Gamma}_0 = I_K \otimes (X_{\text{ext}}^\top X_{\text{ext}})$.

Proof. We have

$$\begin{aligned} \mathscr{S}(p-\Delta p)X_{\text{ext}} = 0 &\iff \operatorname{vec}\left(X_{\text{ext}}^\top \mathscr{S}^\top(\Delta p)\right) = \operatorname{vec}\left(\left(\mathscr{S}(p)X_{\text{ext}}\right)^\top\right) \\ &\iff \underbrace{(I_m \otimes X_{\text{ext}}^\top)}_{G(X)} \underbrace{\operatorname{vec}\left(\mathscr{S}^\top(\Delta p)\right)}_{\Delta p} = r(X). \end{aligned}$$

Therefore, $\Gamma = GG^\top = (I_m \otimes X_{\text{ext}}^\top)(I_m \otimes X_{\text{ext}}^\top)^\top = I_m \otimes (X_{\text{ext}}^\top X_{\text{ext}})$. □

Next, we establish the structure of Γ for an STLS problem with block-Hankel data matrix.

Lemma 4.9. *Let*

$$\mathscr{S}(p) := \begin{bmatrix} C_1 & C_2 & \cdots & C_{\mathbf{n}} \\ C_2 & C_3 & \cdots & C_{\mathbf{n}+1} \\ \vdots & \vdots & & \vdots \\ C_{\mathbf{m}} & C_{\mathbf{m}+1} & \cdots & C_{\mathbf{m}+\mathbf{n}-1} \end{bmatrix} \in \mathbb{R}^{m\times(n+d)}, \qquad \begin{aligned} \mathbf{n} &:= \frac{n+d}{L}, \\ \mathbf{m} &:= \frac{m}{K}, \end{aligned}$$

where C_i are $K \times L$ unstructured blocks, parameterized by $p^{(i)} \in \mathbb{R}^{KL}$ as follows:

$$C_i := \begin{bmatrix} p_1^{(i)} & p_2^{(i)} & \cdots & p_L^{(i)} \\ p_{L+1}^{(i)} & p_{L+2}^{(i)} & \cdots & p_{2L}^{(i)} \\ \vdots & \vdots & & \vdots \\ p_{(K-1)L+1}^{(i)} & p_{(K-1)L+2}^{(i)} & \cdots & p_{KL}^{(i)} \end{bmatrix} \in \mathbb{R}^{K\times L}.$$

Define a partitioning of X_{ext} as $X_{\text{ext}}^\top =: \begin{bmatrix} X_1 & \cdots & X_{\mathbf{n}} \end{bmatrix}$, where $X_j \in \mathbb{R}^{d\times L}$. Then Γ has the block-banded and block-Toeplitz structure (4.5) with $s = \mathbf{n} - 1$ and with

$$\mathbf{\Gamma}_k = \sum_{j=1}^{\mathbf{n}-k} \mathbf{X}_j \mathbf{X}_{j+k}^\top, \quad \textit{where} \quad \mathbf{X}_k := I_K \otimes X_k. \tag{4.8}$$

Proof. Define the residual $R := \mathscr{S}(\Delta p)X_{\text{ext}}$ and the partitioning $R^\top =: \begin{bmatrix} R_1 & \cdots & R_{\mathbf{m}} \end{bmatrix}$,

where $R_i \in \mathbb{R}^{d\times K}$. Let $\Delta C := \mathscr{S}(\Delta p)$, with block entries ΔC_i. We have

$$\begin{aligned}
&\mathscr{S}(p-\Delta p)X_{\text{ext}} = 0 \iff \mathscr{S}(\Delta p)X_{\text{ext}} = \mathscr{S}(p)X_{\text{ext}} \\
&\iff \begin{bmatrix} X_1 & X_2 & \cdots & X_{\mathbf{n}} & & \\ & X_1 & X_2 & \cdots & X_{\mathbf{n}} & \\ & & \ddots & \ddots & & \ddots \\ & & & X_1 & X_2 & \cdots & X_{\mathbf{n}} \end{bmatrix} \begin{bmatrix} \Delta C_1^\top \\ \Delta C_2^\top \\ \vdots \\ \Delta C_{\mathbf{m}+\mathbf{n}-1}^\top \end{bmatrix} = \begin{bmatrix} R_1^\top \\ R_2^\top \\ \vdots \\ R_{\mathbf{m}}^\top \end{bmatrix} \\
&\iff \underbrace{\begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_{\mathbf{n}} & & \\ & \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_{\mathbf{n}} & \\ & & \ddots & \ddots & & \ddots \\ & & & \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_{\mathbf{n}} \end{bmatrix}}_{G(X)} \underbrace{\begin{bmatrix} \operatorname{vec}(\Delta C_1^\top) \\ \operatorname{vec}(\Delta C_2^\top) \\ \vdots \\ \operatorname{vec}(\Delta C_{\mathbf{m}+\mathbf{n}-1}^\top) \end{bmatrix}}_{\Delta p} = \underbrace{\begin{bmatrix} \operatorname{vec}(R_1^\top) \\ \operatorname{vec}(R_2^\top) \\ \vdots \\ \operatorname{vec}(R_{\mathbf{m}}^\top) \end{bmatrix}}_{r(X)}.
\end{aligned}$$

Therefore, $\Gamma = GG^\top$ has the structure (4.5), with $\mathbf{\Gamma}_k$'s given by (4.8). □

The derivation of the Γ matrix for an STLS problem with block-Toeplitz data matrix is analogous to the one for an STLS problem with block-Hankel data matrix. We state the result in the next lemma.

Lemma 4.10. *Let*

$$\mathscr{S}(p) := \begin{bmatrix} C_{\mathbf{n}} & C_{\mathbf{n}-1} & \cdots & C_1 \\ C_{\mathbf{n}+1} & C_{\mathbf{n}} & \cdots & C_2 \\ \vdots & \vdots & & \vdots \\ C_{\mathbf{m}+\mathbf{n}-1} & C_{\mathbf{m}+\mathbf{n}-2} & \cdots & C_{\mathbf{m}} \end{bmatrix} \in \mathbb{R}^{m\times(n+d)},$$

with the blocks C_i defined as in Lemma 4.9. Then Γ has the block-banded and block-Toeplitz structure (4.5) with $s = \mathbf{n}-1$ and

$$\mathbf{\Gamma}_k = \sum_{j=k+1}^{\mathbf{n}} \mathbf{X}_j \mathbf{X}_{j-k}^\top. \tag{4.9}$$

Proof. Following the same derivation as in the proof of Lemma 4.9, we find that

$$G = \begin{bmatrix} \mathbf{X}_{\mathbf{n}} & \mathbf{X}_{\mathbf{n}-1} & \cdots & \mathbf{X}_1 & & & \\ & \mathbf{X}_{\mathbf{n}} & \mathbf{X}_{\mathbf{n}-1} & \cdots & \mathbf{X}_1 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & \mathbf{X}_{\mathbf{n}} & \mathbf{X}_{\mathbf{n}-1} & \cdots & \mathbf{X}_1 \end{bmatrix}.$$

Therefore, $\Gamma = GG^\top$ has the structure (4.5), with $\mathbf{\Gamma}_k$'s given by (4.9). □

Proof of Theorem 4.6. Lemmas 4.7–4.10 show that the weight matrix Γ for the original problem has the block-banded and block-Toeplitz structure (4.5) with $s = \max_{l=1,\ldots,q}(\mathbf{n}_l - 1)$, where $\mathbf{n}_l$ is the number of block columns in the lth block of the data matrix.

Apart from revealing the structure of Γ, the proof of Theorem 4.6 gives an algorithm for the construction of the blocks $\mathbf{\Gamma}_0$, ..., $\mathbf{\Gamma}_s$ that define Γ:

$$\mathbf{\Gamma}_k = \sum_{l=1}^{q} \mathbf{\Gamma}_k^l, \text{ where } \mathbf{\Gamma}_k^l = \begin{cases} \sum_{j=k+1}^{\mathbf{n}_l} \mathbf{X}_j^l (\mathbf{X}_{j-k}^l)^\top & \text{if } C^l \text{ is block-Toeplitz,} \\ \sum_{j=1}^{\mathbf{n}_l - k} \mathbf{X}_j^l (\mathbf{X}_{j+k}^l)^\top & \text{if } C^l \text{ is block-Hankel,} \\ \delta_k I_K \otimes \left((X_{\text{ext}}^l)^\top X_{\text{ext}}^l \right) & \text{if } C^l \text{ is unstructured,} \\ 0_{dK} & \text{if } C^l \text{ is exact,} \end{cases} \tag{4.10}$$

where δ is the Kronecker delta function: $\delta_0 = 1$ and $\delta_k = 0$ for $k \neq 0$.

Corollary 4.11 (Positive definiteness of the weight matrix Γ). *Assume that the structure of $\mathscr{S}$ is given by Assumption 4.4 with the block C^q being block-Toeplitz, block-Hankel, or unstructured and having at least d columns. Then the matrix $\Gamma(X)$ is positive definite for all $X \in \mathbb{R}^{n \times d}$.*

Proof. We will show that $\Gamma^q(X) > 0$ for all $X \in \mathbb{R}^{n \times d}$. From (4.6), it follows that Γ has the same property. By the assumption $\operatorname{col\,dim}(C^q) \geq d$, it follows that $X_{\text{ext}}^q = \left[\begin{smallmatrix} * \\ -I_d \end{smallmatrix} \right]$, where the $*$ denotes a block (possibly empty) depending on X. In the unstructured case, $\Gamma^q = I_m \otimes \left((X_{\text{ext}}^q)^\top X_{\text{ext}}^q \right)$; see (4.10). But $\operatorname{rank} \left((X_{\text{ext}}^q)^\top X_{\text{ext}}^q \right) = d$, so that Γ^q is nonsingular. In the block-Hankel/Toeplitz case, G^q is block-Toeplitz and block-banded; see Lemmas 4.9 and 4.10. One can verify by inspection that, independent of X, $G^q(X)$ has full row rank due to its row echelon form. Then $\Gamma^q = G^q (G^q)^\top > 0$. □

The positive definiteness of Γ is studied in a statistical setting in [KMV05, Section 4], where more general conditions are given. The restriction of Assumption 4.4 that ensures $\Gamma > 0$ is fairly minor, so that in what follows we will consider STLS problems of this type and replace the pseudoinverse in (4.4) with the inverse.

In the next section, we give an interpretation of the result from a statistical point of view, and in Section 4.5, we consider in more detail the algorithmic side of the problem.

4.4 Stochastic Interpretation*

Our work on the STLS problem has its origin in the field of estimation theory. Consider the EIV model

$$AX \approx B, \quad \text{where} \quad A = \bar{A} + \tilde{A}, \quad B = \bar{B} + \tilde{B}, \quad \text{and} \quad \bar{A}\bar{X} = \bar{B}. \tag{EIV$_X$}$$

The data A and B is obtained from true values $\bar{A}$ and $\bar{B}$ with measurement errors $\tilde{A}$ and $\tilde{B}$ that are zero mean random matrices. Define the extended matrix $\tilde{C} := \begin{bmatrix} \tilde{A} & \tilde{B} \end{bmatrix}$ and the vector $\tilde{c} := \operatorname{vec}(\tilde{C}^\top)$ of the measurement errors. It is well known (see [VV91, Chapter 8]) that the TLS problem (TLS$_X$) provides a consistent estimator for the true value of the parameter $\bar{X}$ in the model (EIV$_X$) if $\operatorname{cov}(\tilde{c}) = \sigma^2 I$ (and additional technical conditions are satisfied). If in addition to $\operatorname{cov}(\tilde{c}) = \sigma^2 I$, $\tilde{c}$ is normally distributed, i.e., $\tilde{c} \sim \mathrm{N}(0, \sigma^2 I)$,then the solution $\hat{X}_{\text{tls}}$ of the TLS problem is the maximum likelihood estimate of $\bar{X}$.

The model (EIV$_X$) is called structured EIV model if the observed data C and the true value $\bar{C} := \begin{bmatrix} \bar{A} & \bar{B} \end{bmatrix}$ have a structure defined by a function $\mathscr{S}$. Therefore,

$$C = \mathscr{S}(p) \quad \text{and} \quad \bar{C} = \mathscr{S}(\bar{p}),$$

where $\bar{p} \in \mathbb{R}^{n_p}$ is a true value of the parameter p. As a consequence, the matrix of measurement errors is also structured. Let $\mathscr{S}$ be affine and defined by (4.1). Then

$$\tilde{C} = \sum_{i=1}^{n_p} S_i \tilde{p}_i \quad \text{and} \quad p = \bar{p} + \tilde{p},$$

where the random vector $\tilde{p}$ represents the measurement error on the structure parameter $\bar{p}$. In [KMV05], it is proven that the STLS problem (STLS_X) provides a consistent estimator for the true value of the parameter $\bar{X}$ if $\operatorname{cov}(\tilde{p}) = \sigma^2 I$ (and additional technical conditions are satisfied). If $\tilde{p} \sim \mathrm{N}(0, \sigma^2 I)$ then a solution $\hat{X}$ of the STLS problem is a maximum likelihood estimate of $\bar{X}$.

Let $\tilde{r}(X) := \operatorname{vec}\left(\mathscr{S}(\tilde{p}) X_{\text{ext}}\right)$ be the random part of the residual r.

> In the stochastic setting, the weight matrix Γ is, up to the scale factor σ^2, equal to the covariance matrix $V_{\tilde{r}} := \operatorname{cov}(\tilde{r})$.

Indeed, $\tilde{r} = G\tilde{p}$, so that

$$V_{\tilde{r}} := \mathbf{E}(\tilde{r}\tilde{r}^\top) = G\ \mathbf{E}(\tilde{p}\tilde{p}^\top) G^\top = \sigma^2 G G^\top = \sigma^2 \Gamma.$$

Next, we show that the structure of Γ is in a one-to-one correspondence with the structure of $V_{\tilde{c}} := \operatorname{cov}(\tilde{c})$. Let $\mathbf{\Gamma}_{ij} \in \mathbb{R}^{dK \times dK}$ be the (i,j)th block of Γ and let $V_{\tilde{\mathbf{c}},ij} \in \mathbb{R}^{(n+d)K \times (n+d)K}$ be the (i,j)th block of $V_{\tilde{c}}$. Define also the following partitionings of the vectors $\tilde{r}$ and $\tilde{c}$:

$$\tilde{r} =: \operatorname{col}(\tilde{\mathbf{r}}_1, \ldots, \tilde{\mathbf{r}}_{\mathbf{m}}), \quad \tilde{\mathbf{r}}_i \in \mathbb{R}^{dK} \qquad \text{and} \qquad \tilde{c} =: \operatorname{col}(\tilde{\mathbf{c}}_1, \ldots, \tilde{\mathbf{c}}_{\mathbf{m}}), \quad \tilde{\mathbf{c}}_i \in \mathbb{R}^{(n+d)K},$$

where $\mathbf{m} := m/K$. Using $\mathbf{r}_i = \mathbf{X}_{\text{ext}} \mathbf{c}_i$, where $\mathbf{X}_{\text{ext}} := (I_K \otimes X_{\text{ext}}^\top)$, we have

$$\sigma^2 \mathbf{\Gamma}_{ij} = \mathbf{E}(\tilde{\mathbf{r}}_i \tilde{\mathbf{r}}_j^\top) = \mathbf{X}_{\text{ext}}\, \mathbf{E}(\tilde{\mathbf{c}}_i \tilde{\mathbf{c}}_j^\top) \mathbf{X}_{\text{ext}}^\top = \mathbf{X}_{\text{ext}} V_{\tilde{\mathbf{c}},ij} \mathbf{X}_{\text{ext}}^\top. \tag{4.11}$$

The one-to-one relation between the structures of Γ and $V_{\tilde{c}}$ allows us to relate the structural properties of Γ, established in Theorem 4.6, with statistical properties of the measurement errors. Define stationarity and s-dependence of a centered sequence of random vectors $\tilde{\mathbf{c}} := \{\tilde{\mathbf{c}}_1, \tilde{\mathbf{c}}_2, \ldots\}$, $\tilde{\mathbf{c}}_i \in \mathbb{R}^{(n+d)K}$ as follows:

- $\tilde{\mathbf{c}}$ is *stationary* if the covariance matrix $V_{\tilde{c}}$ is block-Toeplitz with block size $(n+d)K \times (n+d)K$, and
- $\tilde{\mathbf{c}}$ is *s-dependent* if the covariance matrix $V_{\tilde{c}}$ is block-banded with block size $(n+d)K \times (n+d)K$ and block bandwidth $2s+1$.

> The sequence of measurement errors $\tilde{\mathbf{c}}_1, \ldots, \tilde{\mathbf{c}}_{\mathbf{m}}$ being stationary and s-dependent corresponds to Γ being block-Toeplitz and block-banded.

The statistical setting gives an insight into the relation between the structure of the weight matrix Γ and the structure of the data matrix C. It can be verified that the structure specification of Assumption 4.4 implies stationarity and s-dependen for $\tilde{\mathbf{c}}$. This indicates an alternative (statistical) proof of Theorem 4.6.

The blocks of Γ are quadratic functions of X, $\mathbf{\Gamma}_{ij}(X) = \mathbf{X}_{\text{ext}} W_{\tilde{\mathbf{c}},ij} \mathbf{X}_{\text{ext}}^\top$, where $W_{\tilde{\mathbf{c}},ij} := V_{\tilde{\mathbf{c}},ij}/\sigma^2$; see (4.11). Moreover, by Theorem 4.6, we have that under Assumption 4.4, $W_{\tilde{\mathbf{c}},ij} = W_{\tilde{\mathbf{c}},|i-j|}$, for certain matrices $W_{\tilde{\mathbf{c}},k}$, $k = 1, \ldots, \mathbf{m}$, and $W_{\tilde{\mathbf{c}},ij} = 0$, for $|i-j| > s$, where s is defined in Theorem 4.6. Therefore,

$$\mathbf{\Gamma}_k(X) = \mathbf{X}_{\text{ext}} W_{\tilde{\mathbf{c}},k} \mathbf{X}_{\text{ext}}^\top, \quad \text{for } k = 1, \ldots, s, \quad \text{where} \quad W_{\tilde{\mathbf{c}},k} := \frac{1}{\sigma^2} V_{\tilde{\mathbf{c}},k}.$$

In (4.10) we show how the matrices $\{\mathbf{\Gamma}_k\}_{k=0}^s$ are determined from the structure specification of Assumption 4.4. Now we give the corresponding expressions for the matrices $\{W_{\tilde{\mathbf{c}},k}\}_{k=0}^s$:

$$W_{\tilde{\mathbf{c}},k} := \operatorname{diag}(W_k^1, \ldots, W_k^q), \quad \text{and} \quad W_k^l = \begin{cases} (J_{n_l K}^\top)^{t_l k} & \text{if } C^l \text{ is block-Toeplitz,} \\ (J_{n_l K})^{t_l k} & \text{if } C^l \text{ is block-Hankel,} \\ \delta_k I_{n_l K} & \text{if } C^l \text{ is unstructured,} \\ 0_{n_l K} & \text{if } C^l \text{ is exact,} \end{cases} \tag{4.12}$$

where J_{n_l} is the $n_l \times n_l$ shift matrix

$$J_{n_l} := \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \in \mathbb{R}^{n_l \times n_l}.$$

In the computational algorithm described in Section 4.5, we use the partitioning of the matrix Γ into blocks of size $d \times d$. Let $\Gamma_{ij} \in \mathbb{R}^{d \times d}$ be the (i,j)th block of Γ and let $V_{\tilde{c},ij} \in \mathbb{R}^{(n+d)\times(n+d)}$ be the (i,j)th block of $V_{\tilde{c}}$. Define the following partitionings of the vectors $\tilde{r}$ and $\tilde{c}$:

$$\tilde{r} =: \operatorname{col}(\tilde{r}_1, \ldots, \tilde{r}_m), \quad \tilde{r}_i \in \mathbb{R}^d \qquad \text{and} \qquad \tilde{c} =: \operatorname{col}(\tilde{c}_1, \ldots, \tilde{c}_m), \quad \tilde{c}_i \in \mathbb{R}^{n+d}.$$

Using $\tilde{r}_i = X_{\text{ext}}^\top \tilde{c}_i$, we have

$$\Gamma_{ij} = \frac{1}{\sigma^2} \mathbf{E}(\tilde{r}_i \tilde{r}_j^\top) = \frac{1}{\sigma^2} X_{\text{ext}}^\top \mathbf{E}(\tilde{c}_i \tilde{c}_j^\top) X_{\text{ext}} = \frac{1}{\sigma^2} X_{\text{ext}}^\top V_{\tilde{c},ij} X_{\text{ext}} =: X_{\text{ext}}^\top W_{\tilde{c},ij} X_{\text{ext}}.$$

4.5 Efficient Cost Function and First Derivative Evaluation*

We consider an efficient numerical method for solving the STLS problem (STLS$_X$) by applying standard local optimization algorithms to the equivalent problem (4.4). With this approach, the main computational effort is in the cost function and its first derivative evaluation.

First, we describe the evaluation of the cost function: given X, compute $f_0(X)$. For given X, and with $\{\mathbf{\Gamma}_k\}_{k=0}^{s}$ constructed according to (4.12), the weight matrix $\Gamma(X)$ is specified. Then, from the solution of the system $\Gamma(X)y_r(X) = r(X)$, the cost function is found as $f_0(X) = r^\top(X)y_r(X)$.

The properties of $\Gamma(X)$ can be exploited in the solution of the system $\Gamma y_r = r$. The subroutine `MB02GD` from the SLICOT library [VSV+04] exploits both the block-Toeplitz and the block-banded structure to compute a Cholesky factor of Γ in $O((dK)^2sm)$ flops. In combination with the LAPACK subroutine `DPBTRS` that solves block-banded triangular systems of equations, the cost function is evaluated in $O(m)$ flops. Thus an algorithm for local optimization that uses only cost function evaluations has computational complexity $O(m)$ flops per iteration, because the computations needed internally for the optimization algorithm do not depend on m.

Next, we describe the evaluation of the derivative. The derivative of the cost function f_0 is (see Appendix A.2)

$$f_0'(X) = 2\sum_{i,j=1}^{m} a_j r_i^\top(X) M_{ij}(X) - 2\sum_{i,j=1}^{m} \begin{bmatrix} I & 0 \end{bmatrix} W_{\tilde{c},ij} \begin{bmatrix} X \\ -I \end{bmatrix} N_{ji}(X), \tag{4.13}$$

where $A^\top =: \begin{bmatrix} a_1 & \cdots & a_m \end{bmatrix}$, with $a_i \in \mathbb{R}^n$,

$$M(X) := \Gamma^{-1}(X), \qquad N(X) := \Gamma^{-1}(X) r(X) r^\top(X) \Gamma^{-1}(X),$$

and $M_{ij} \in \mathbb{R}^{d\times d}$, $N_{ij} \in \mathbb{R}^{d\times d}$ are the (i,j)th blocks of M and N, respectively.

Consider the following two partitionings of $y_r \in \mathbb{R}^{md}$:

$$y_r =: \operatorname{col}(y_{r,1}, \ldots, y_{r,m}), \quad y_{r,i} \in \mathbb{R}^d \quad \text{and} \quad y_r =: \operatorname{col}(\mathbf{y}_{r,1}, \ldots, \mathbf{y}_{r,\mathbf{m}}), \quad \mathbf{y}_{r,i} \in \mathbb{R}^{dK}, \tag{4.14}$$

where $\mathbf{m} := m/K$. The first sum in (4.13) becomes

$$\sum_{i,j=1}^{m} a_j r_i^\top M_{ij} = A^\top Y_r, \quad \text{where} \quad Y_r^\top := \begin{bmatrix} y_{r,1} & \cdots & y_{r,m} \end{bmatrix}. \tag{4.15}$$

Define the sequence of matrices

$$\mathbf{N}_k := \sum_{i=1}^{\mathbf{m}-k} \mathbf{y}_{r,i+k} \mathbf{y}_{r,i}^\top, \quad \mathbf{N}_k = \mathbf{N}_{-k}^\top, \quad k = 0, \ldots, s.$$

The second sum in (4.13) can be written as

$$\sum_{i,j=1}^{m} \begin{bmatrix} I & 0 \end{bmatrix} W_{\tilde{c},ij} \begin{bmatrix} X \\ -I \end{bmatrix} N_{ji} = \sum_{k=-s}^{s} \sum_{i,j=1}^{K} (W_{\tilde{\mathbf{a}},k,ij} X - W_{\tilde{\mathbf{a}}\tilde{\mathbf{b}},k,ij}) \mathbf{N}_{k,ij}^\top,$$

where $W_{\tilde{\mathbf{c}},k,ij} \in \mathbb{R}^{(n+d)\times(n+d)}$ is the (i,j)th block of $W_{\tilde{\mathbf{c}},k} \in \mathbb{R}^{K(n+d)\times K(n+d)}$, $W_{\tilde{\mathbf{a}},k,ij} \in \mathbb{R}^{n\times n}$ and $W_{\tilde{\mathbf{a}}\tilde{\mathbf{b}},k,ij} \in \mathbb{R}^{n\times d}$ are defined as blocks of $W_{\tilde{\mathbf{c}},k,ij}$ as

$$W_{\tilde{\mathbf{c}},k,ij} =: \begin{bmatrix} W_{\tilde{\mathbf{a}},k,ij} & W_{\tilde{\mathbf{a}}\tilde{\mathbf{b}},k,ij} \\ W_{\tilde{\mathbf{b}}\tilde{\mathbf{a}},k,ij} & W_{\tilde{\mathbf{b}},k,ij} \end{bmatrix},$$

and $\mathbf{N}_{k,ij} \in \mathbb{R}^{d\times d}$ is the (i,j)th block of $\mathbf{N}_k \in \mathbb{R}^{dK\times dK}$.

Thus the evaluation of the derivative $f_0'(X)$ uses the solution of $\Gamma y_r = r$, already computed for the cost function evaluation and additional operations of $O(m)$ flops. The steps described above are summarized in Algorithms 4.1 and 4.2.

> The structure of $\mathscr{S}(\cdot)$ is specified by the integer K, the number of rows in a block of a block-Toeplitz/Hankel structured block $C^{(i)}$, and the array $\mathtt{S} \in (\{\mathtt{T},\mathtt{H},\mathtt{U},\mathtt{E}\}\times\mathbb{N}\times\mathbb{N})^q$ that describes the structure of the blocks $\{C^{(i)}\}_{i=1}^q$.

The ith element $\mathtt{S}_i$ of the array $\mathtt{S}$ specifies the block $C^{(i)}$ by giving its type $\mathtt{S}_i(1)$, the number of columns $n_i = \mathtt{S}_i(2)$, and (if $C^{(i)}$ is block-Hankel or block-Toeplitz) the column dimension $t_i = \mathtt{S}_i(3)$ of a block in $C^{(i)}$. Therefore, the input data for the STLS problem is the data matrix $\mathscr{S}(p)$ (alternatively the parameter vector p) and the structure specification K and $\mathtt{S}$.

Algorithm 4.1 outlines the steps for the construction of the $W_{\tilde{\mathbf{c}},k}$ matrices. It requires arithmetic operation only for indexing matrix-vector elements. The $s+1$ matrices $\{W_{\tilde{\mathbf{c}},k}\}_{k=0}^s$ are sparse. For the typical applications that we address, however, their dimension $(n+d)K\times(n+d)K$ is relatively small (compared to the row dimension m of the data matrix), so that we do not take into account their structure.

Algorithm 4.1 From structure specification $K, \mathtt{S}$ to $\{W_{\tilde{\mathbf{c}},k}\}$ `decode_struct`

Input: structure specification $K, \mathtt{S}$.

Define $s := \max_{l=1,\ldots,q}(\mathbf{n}_l - 1)$, where $\mathbf{n}_l := n_l/t_l$, for block-Toeplitz/Hankel structured block C^l, and $\mathbf{n}_l := 1$, otherwise.
for $k = 1,\ldots,s$ **do**
  for $l = 1,\ldots,q$ **do**
    if $\mathtt{S}_l(1)$==$\mathtt{T}$ **then**
      $W_k^l = (J_{n_l}^\top)^{t_l k}$
    else if $\mathtt{S}_l(1)$==$\mathtt{H}$ **then**
      $W_k^l = (J_{n_l})^{t_l k}$
    else if $\mathtt{S}_l(1)$==$\mathtt{U}$ **then**
      $W_k^l = \delta_k I_{n_l}$
    else
      $W_k^l = 0_{n_l}$
    end if
  end for
  $W_{\tilde{\mathbf{c}},k} := \mathrm{diag}(W_k^1,\ldots,W_k^q)$
end for

Output: $\{W_{\tilde{\mathbf{c}},k}\}_{k=0}^s$.

Algorithm 4.2 specifies the steps needed for the cost function and its first derivative evaluation. The flops per step for Algorithm 4.2 are as follows:

2. $(n+d)(n+2d)dK^3$
3. $m(n+1)d$
4. msd^2K^2
5. md
8. $msd^2K - s(s+1)d^2K^2/2$
9. $mnd + (2s+1)(nd+n+1)dK^2$

Thus in total $O(md(sdK^2+n)+n^2dK^3+3nd^2K^3+2d^3K^3+2snd^2K^2)$ flops are required for cost function and first derivative evaluation. Note that the flop counts depend on the structure through s.

Algorithm 4.2 STLS cost function and first derivative evaluation `cost`

Input: A, B, X, $\{W_{\tilde{\mathbf{c}},k}\}_{k=0}^{s}$.

1: $\mathbf{\Gamma}_k = (I_K \otimes \begin{bmatrix} X^\top & -I \end{bmatrix}) W_{\tilde{\mathbf{c}},k} (I_K \otimes \begin{bmatrix} X^\top & -I \end{bmatrix})^\top$, for $k = 0, 1, \ldots, s$.

2: $r = \operatorname{vec}\left((AX-B)^\top\right)$.

3: Solve $\Gamma y_r = r$ exploiting the block-banded and block-Toeplitz structure of Γ, e.g., by using the routines `MB02GD` from the SLICOT library and `DPBTRS` from the LAPACK library.

4: $f_0 = r^\top y_r$.

5: If only the cost function evaluation is required, output f_0 and stop.

6: Define $\operatorname{col}(y_{r,1}, \ldots, y_{r,m}) := y_r$, where $y_{r,i} \in \mathbb{R}^d$; $\operatorname{col}(\mathbf{y}_{r,1}, \ldots, \mathbf{y}_{r,\mathbf{m}}) =: y_r$, where $\mathbf{y}_{r,i} \in \mathbb{R}^{dK}$, $\mathbf{m} := m/K$; and $Y_r^\top := \begin{bmatrix} y_{r,1} & \cdots & y_{r,m} \end{bmatrix}$.

7: $\mathbf{N}_k = \sum_{i=1}^{\mathbf{m}-k} \mathbf{y}_{r,i+k} \mathbf{y}_{r,i}^\top$, for $k = 0, 1, \ldots, s$.

8: $f_0' = 2A^\top Y_r - 2\sum_{k=-s}^{s} \sum_{i,j=1}^{K} (W_{\tilde{\mathbf{a}},k,ij} X - W_{\tilde{\mathbf{a}}\tilde{\mathbf{b}},k,ij}) \mathbf{N}_{k,ij}^\top$, where $W_{\tilde{\mathbf{c}},k,ij} \in \mathbb{R}^{(n+d)\times(n+d)}$ is the (i,j)th block of $W_{\tilde{\mathbf{c}},k} \in \mathbb{R}^{K(n+d)\times K(n+d)}$, $W_{\tilde{\mathbf{a}},k,ij} \in \mathbb{R}^{n\times n}$; $W_{\tilde{\mathbf{a}}\tilde{\mathbf{b}},k,ij} \in \mathbb{R}^{n\times d}$ are defined as blocks of $W_{\tilde{\mathbf{c}},k,ij}$ as

$$W_{\tilde{\mathbf{c}},k,ij} =: \begin{bmatrix} W_{\tilde{\mathbf{a}},k,ij} & W_{\tilde{\mathbf{a}}\tilde{\mathbf{b}},k,ij} \\ W_{\tilde{\mathbf{b}}\tilde{\mathbf{a}},k,ij} & W_{\tilde{\mathbf{b}},k,ij} \end{bmatrix};$$

and $\mathbf{N}_{k,ij} \in \mathbb{R}^{d\times d}$ is the (i,j)th block of $\mathbf{N}_k \in \mathbb{R}^{dK\times dK}$.

Output: f_0, f_0'.

Using the computation of the cost function and its first derivative, we can apply the Broyden, Fletcher, Goldfarb, and Shanno (BFGS) quasi-Newton method. Thus the overall algorithm for the computation of the STLS solution is Algorithm 4.3.

A more efficient alternative, however, is to apply a nonlinear least squares optimization algorithm, such as the Levenberg–Marquardt algorithm. Let $\Gamma = U^\top U$ be the Cholesky factorization of Γ. Then $f_0 = F^\top F$, with $F := U^{-1}r$. (Note that the evaluation of $F(X)$ is cheaper than that of $f_0(X)$.) We do not know an analytic expression for the Jacobian matrix $J(X) = [\partial F_i/\partial x_j]$, but instead we use the so-called pseudo-Jacobian J_+ proposed in [GP96]. The evaluation of J_+ can be done efficiently, using the approach described above for $f'(X)$.

Moreover, by using the nonlinear least squares approach and the pseudo-Jacobian J_+, we have as a byproduct of the optimization algorithm an estimate of the covariance matrix

Algorithm 4.3 Algorithm for solving the STLS problem `stls`

Input: the structure specification $K, \mathtt{S}$ and the matrices A and B.

1: Compute the matrices $\{W_{\bar{\mathtt{c}},k}\}$ via Algorithm 4.1.
2: Compute the TLS solution $X^{(0)}$ of $AX \approx B$ by, e.g., the function `MB02MD` from the SLICOT library.
3: Execute a standard optimization algorithm, e.g., the BFGS quasi-Newton method, for the minimization of f_0 over X with initial approximation $X^{(0)}$ and with cost function and first derivative evaluation performed via Algorithm 4.2.

Output: $\hat{X}$ the approximation found by the optimization algorithm upon convergence.

Table 4.1. *Standard approximation problems that are special cases of the STLS problem for particular structure specification $K, \mathtt{S}$.*

Problem	Structure $\mathtt{S}$	K
Least squares (LS)	$\left[\left[\mathtt{E} \quad n\right], \left[\mathtt{U} \quad d\right]\right]$	1
Total least squares (TLS)	$\left[\mathtt{U} \quad n+d\right]$	1
Mixed least squares–total least squares (LS-TLS)	$\left[\left[\mathtt{E} \quad n_1\right], \left[\mathtt{U} \quad n_2\right], \left[\mathtt{U} \quad d\right]\right]$	1
Hankel low-rank approximation (HLRA)	$\left[\mathtt{H} \quad \mathtt{n}+\mathtt{p} \quad \mathtt{m}\right]$	p
SISO deconvolution	$\left[\left[\mathtt{T} \quad \mathtt{n}\right], \left[\mathtt{U} \quad 1\right]\right]$	1
SISO EIV system identification	$\left[\left[\mathtt{H} \quad \mathtt{n}+1\right], \left[\mathtt{H} \quad \mathtt{n}+1\right]\right]$	1

$V_{\hat{x}} = \operatorname{cov}\left(\operatorname{vec}(\hat{X})\right)$. As shown in [PS01, Chapter 17.4.7, equations (17)–(35)],

$$V_{\hat{x}} \approx \left(J_+^\top(\hat{X}) J_+(\hat{X})\right)^{-1}.$$

Using $V_{\hat{x}}$, we can compute statistical confidence bounds for the estimate $\hat{X}$.

The solution method outlined in Algorithm 4.3, using the Levenberg–Marquardt algorithm, is implemented in C language. A description of the software package is given in Appendix B.2.

4.6 Simulation Examples

The approximation problems listed in Table 4.1 are special cases of the block-Toeplitz/Hankel STLS problem for particular choices of the structure specification $K, \mathtt{S}$. If not given, the third element of $\mathtt{S}_l$ is by default equal to one.

Our goal is to show the flexibility of the STLS problem formulation (STLS_X) with a structure of Assumption 4.4. More realistic applications of the STLS package are described in Chapter 11, where real-life data sets for multi-input multi-output (MIMO) system identification are used. Special problems such as LS, TLS, and mixed LS-TLS should be solved by the corresponding special methods. Still, they serve as benchmarks for the STLS package.

We show simulation examples for the problems of Table 4.1. The data is a perturbed version of a "true" data generated by a "true" model. True model and true data refer to the

particular problem (see the description below) and are selected randomly. The perturbation is a Gaussian noise with a covariance matrix $\sigma^2 I$.

Table 4.2 shows the scaled computation time t, cost function value $f_0(\hat{X})$, i.e., error of approximation, and relative error of estimation

$$e := \|\bar{X} - \hat{X}\|_{\mathrm{F}} / \|\bar{X}\|_{\mathrm{F}}, \qquad \text{where } \bar{X} \text{ is the parameter of the "true" model}$$

for the STLS package and for an alternative computational method, if there is one. The scaling is done by the smaller of the two values: the one achieved by the STLS package and the one achieved by the alternative method.

Next, we describe the simulation setup for the examples.

Least Squares

The LS problem $AX \approx B$, where $A \in \mathbb{R}^{m \times n}$ is exact and unstructured and $B \in \mathbb{R}^{m \times d}$ is perturbed and unstructured, is solved as an STLS problem with structure specification $\mathtt{S} = \left[\left[\mathtt{E} \quad n\right], \left[\mathtt{U} \quad d\right]\right]$. In the simulation example, the solution of the STLS package is checked by the MATLAB least squares solver `\`. In the example, $m = 100$, $n = 5$, $d = 2$, and $\sigma = 0.1$.

In the LS case, the STLS optimization algorithm converges in two iteration steps. The reason for this is that the second order approximation used in the algorithm is actually the exact LS cost function. Therefore, independent of the initial approximation, in one iteration step the algorithm finds the global optimum point. An additional iteration is needed in order to conclude that the computed approximation in the first step is optimal.

Total Least Squares

The TLS problem $AX \approx B$, where the data matrix $C := \begin{bmatrix} A & B \end{bmatrix} \in \mathbb{R}^{m \times (n+d)}$ is perturbed and unstructured, is solved as an STLS problem with structure specification $\mathtt{S} = \begin{bmatrix} \mathtt{U} & n+d \end{bmatrix}$. In the simulation example, the solution of the STLS package is checked by the function `tls.m` that implements the SVD method for the computation of the TLS solution; see Theorem 3.14. In the example, $m = 100$, $n = 5$, $d = 2$, and $\sigma = 0.1$.

Table 4.2. *Comparison of the STLS package and alternative methods on simulation examples. t—scaled execution time, f_0—scaled cost function value, e—scaled error of estimation. The scaling is the smaller of the values achieved by the methods.*

Problem	STLS package			Alternative method			
	t	f_0	e	t	f_0	e	function
LS	40	1	1.000000	1	1.000000000000	1	`\`
TLS	1	1	1.000000	2	1.000000000000	1	`tls`
LS-TLS	1	1	1.000000	5	1.000000000000	1	`lstls`
HLRA	1	1	1.000087	147	1.000000056132	1	`faststln2`
Deconvolution	1	1	1.000009	631	1.000000000002	1	`faststln1`
System ident.	1	1	1.000000	—	—	—	—

In the TLS case, the STLS algorithm converges in one iteration step, because the default initial approximation used in the STLS package is the TLS solution.

Mixed Least Squares–Total Least Squares

The mixed LS-TLS problem [VV91, Section 3.5] is defined as follows: $AX \approx B$, where $A = \begin{bmatrix} A_{\mathrm{e}} & A_{\mathrm{p}} \end{bmatrix}$, $A_{\mathrm{p}} \in \mathbb{R}^{m \times n_1}$ and $B \in \mathbb{R}^{m \times d}$ are perturbed and unstructured, and $A_{\mathrm{e}} \in \mathbb{R}^{m \times n_2}$ is exact and unstructured. This problem is solved as an STLS problem with structure specification $\mathtt{S} = \begin{bmatrix} \begin{bmatrix} \mathtt{E} & n_1 \end{bmatrix}, \begin{bmatrix} \mathtt{U} & n_2 \end{bmatrix}, \begin{bmatrix} \mathtt{U} & d \end{bmatrix} \end{bmatrix}$. In [VV91] an SVD based method for the computation of the mixed LS-TLS solution is proposed. In the simulation example the solution of the STLS package is checked by a MATLAB implementation `lstls.m` of the exact mixed LS-TLS solution method. In the example, $m = 100$, $n = 5$, $d = 2$, $n_1 = 1$, and $\sigma = 0.1$.

Hankel Low-Rank Approximation

The Hankel low-rank approximation problem [DM93, Section 4.5], [SLV04] is defined as follows:

$$\min_{\Delta p} \|\Delta p\|_2^2 \quad \text{subject to} \quad \mathscr{H}(p - \Delta p) \text{ has given rank } \mathtt{n}. \tag{4.16}$$

Here $\mathscr{H}$ is a mapping from the parameter space $\mathbb{R}^{n_p}$ to the set of the $m \times (\mathtt{n} + \mathtt{p})$ block-Hankel matrices, with block size $\mathtt{p} \times \mathtt{m}$. If the rank constraint is expressed as $\mathscr{H}(\hat{p}) \left[\begin{smallmatrix} X \\ -I \end{smallmatrix} \right] = 0$, where $X \in \mathbb{R}^{\mathtt{n} \times \mathtt{p}}$ is an additional variable, then (4.16) becomes an STLS problem with $K = \mathtt{p}$ and $\mathtt{S} = \begin{bmatrix} \mathtt{H} & \mathtt{n} + \mathtt{p} & \mathtt{m} \end{bmatrix}$.

The Hankel low-rank approximation problem has a system theoretic meaning of approximate realization or (finite-time) model reduction; see Section 11.4. In the single-input single-output (SISO) case, i.e., when $\mathtt{p} = \mathtt{m} = 1$, the STLS package is checked by a MATLAB implementation `faststln2` of the method of [LMV00]. In the example, the true parameter vector is $\bar{p} = \mathrm{col}(1, \ldots, 12)$ (to which corresponds $\bar{x} = \mathrm{col}(-1, 2)$) and the given vector is $p = \bar{p} + \mathrm{col}(5, 0, \ldots, 0)$.

The computed solutions by the STLS package and `faststln2` approximate the same locally optimal solution. In the example, however, the STLS package achieves better approximation of the minimum point for 147 times less computation time. The huge difference in the execution times is due to the MATLAB implementation of `faststln1`: m-files that extensively use `for` loops are executed slowly in MATLAB (versions ≤ 7.0).

Single-Input Single-Output Deconvolution

The convolution of the sequences $(\ldots, a_{-1}, a_0, a_1, \ldots)$ and $(\ldots, x_{-1}, x_0, x_1, \ldots)$ is the sequence $(\ldots, b_{-1}, b_0, b_1, \ldots)$ defined as follows:

$$b_i = \sum_{j=-\infty}^{\infty} x_j a_{i-j}. \tag{4.17}$$

Assume that $x_j = 0$ for all $j < 1$ and for all $j > \mathtt{n}$. Then (4.17) for $i = 1, \ldots, m$ can be written as the following structured system of equations:

$$\underbrace{\begin{bmatrix} a_0 & a_{-1} & \cdots & a_{1-\mathtt{n}} \\ a_1 & a_0 & \cdots & a_{2-\mathtt{n}} \\ \vdots & \vdots & & \vdots \\ a_{m-1} & a_{m+\mathtt{n}-2} & \cdots & a_{m-\mathtt{n}} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{\mathtt{n}} \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}}_{b}. \tag{4.18}$$

Note that the matrix A is Toeplitz structured and is parameterized by the vector

$$a = \operatorname{col}(a_{1-\mathtt{n}}, \ldots, a_{m-1}) \in \mathbb{R}^{m+\mathtt{n}-1}.$$

The aim of the deconvolution problem is to find x, given a and b. With exact data the problem boils down to solving the system of equations (4.18). By construction it has an exact solution. Moreover, the solution is unique whenever A is of full column rank, which can be translated to a condition on a (persistency of excitation).

The deconvolution problem is more realistic and more challenging when the data a, b is perturbed. We assume that $m > \mathtt{n}$, so that the system of equations (4.18) is overdetermined. Because both a and b are perturbed and the A matrix is structured, the deconvolution problem is an STLS problem with the structure specification $\mathtt{S} = \big[\big[\mathtt{T} \quad \mathtt{n}\big], \big[\mathtt{U} \quad 1\big]\big]$. Moreover, under the assumption that the observations are obtained from true values with additive noise that is zero mean and normal, with covariance matrix a multiple of the identity, the STLS method provides a maximum likelihood estimate of the true values.

We compare the solution obtained by the STLS package with the solution obtained by the MATLAB implementation `faststln1` of the method of [MLV00]. In the particular simulation example, $m = 200$, $\mathtt{n} = 2$, and $\sigma = 0.05$. The STLS package computes slightly more accurate approximation of a minimum point using 631 times less computation time. The difference in the execution time is again due to the MATLAB implementation of `faststln1`.

Single-Input Single-Output Errors-in-Variables System Identification

Consider the SISO linear time-invariant system described by the difference equation

$$y_t + \sum_{\tau=1}^{\mathtt{n}} a_\tau y_{t+\tau} = \sum_{\tau=0}^{\mathtt{n}} b_\tau u_{t+\tau} \tag{4.19}$$

and define the parameter vector

$$x := \operatorname{col}(b_0, \ldots, b_{\mathtt{n}}, -a_0, \ldots, -a_{\mathtt{n}-1}) \in \mathbb{R}^{2\mathtt{n}+1}.$$

Given a set of input/output data $(u_1, y_1), \ldots, (u_T, y_T)$ and an order specification $\mathtt{n}$, we want to find the parameter x of a system that fits the data.

For the time horizon $t = 1, \ldots, T$, (4.19) can be written as the structured system of equations

$$\left[\begin{array}{cccc|cccc} u_1 & u_2 & \cdots & u_{\mathtt{n}+1} & y_1 & y_2 & \cdots & y_{\mathtt{n}} \\ u_2 & u_3 & \cdots & u_{\mathtt{n}+2} & y_2 & y_3 & \cdots & y_{\mathtt{n}+1} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ u_m & u_{m+1} & \cdots & u_T & y_m & y_{m+1} & \cdots & y_{T-1} \end{array}\right] x = \begin{bmatrix} y_{n+1} \\ y_{n+2} \\ \vdots \\ y_T \end{bmatrix}, \qquad (4.20)$$

where $m := T - \mathtt{n}$. We assume that the time horizon is large enough to ensure $m > 2\mathtt{n} + 1$. The system (4.20) is satisfied for exact data and a solution is the true value of the parameter x. Moreover, under additional assumption on the input (persistency of excitation) the solution is unique.

For perturbed data an approximate solution is sought, and the fact that the system of equations (4.20) is structured suggests the use of the STLS method. Again, under appropriate conditions for the data generating mechanism, an STLS solution provides a maximum likelihood estimator.

The structure arising in the SISO identification problem is

$$\mathtt{S} = \left[\begin{bmatrix} \mathtt{H} & \mathtt{n}+1 \end{bmatrix}, \begin{bmatrix} \mathtt{H} & \mathtt{n}+1 \end{bmatrix}\right],$$

where $\mathtt{n}$ is the order of the system. Unfortunately, in this case we do not have an alternative method by which the result of the STLS package can be verified. In the simulation example we choose $\mathtt{n} = 3$,

$$\bar{a} = 0.151 \begin{bmatrix} 1 & 0.9 & 0.49 & 0.145 \end{bmatrix}, \qquad \bar{b} = \begin{bmatrix} 1 & -1.2 & 0.81 & -0.27 \end{bmatrix},$$

$T = 1000$, $\bar{u}$ white noise with unit variance, and $\sigma^2 = 0.1$. From the compared LS, TLS, and STLS solutions, the relative error of estimation e is largest for the LS method and is smallest for the STLS method. (The numerical values are not shown in Table 4.2.) This relation of the estimation errors can be expected with high probability for large sample size ($T \to \infty$) due to the statistical consistency of the TLS and STLS methods and the inconsistency of the LS method. In addition, the STLS method being a maximum likelihood method is statistically more efficient than the TLS method.

4.7 Conclusions

We considered an STLS problem with block-wise specified structure of the data matrix. Each of the blocks can be block-Toeplitz/Hankel structured, unstructured, or exact. It is shown that such a formulation is flexible and covers as special cases many previously studied structured and unstructured matrix approximation problems.

The proposed numerical solution method is based on an equivalent unconstrained optimization problem (4.4). We proved that our Assumption 4.4 about the structure of the data matrix implies that the weight matrix Γ in the equivalent problem is block-Toeplitz and block-banded. These properties are used for cost function and first derivative evaluation with linear in the sample size computational cost.

Our results show that a large variety of STLS problems can be solved efficiently with a single kernel computational tool—efficient Cholesky factorization of a block-Toeplitz and block-banded matrix.

The block-Toeplitz/Hankel structure is motivated by approximate modeling problems for MIMO linear time-invariant dynamical systems. For example, EIV system identification, approximate realization, and model reduction problems can be solved via the proposed STLS algorithm.

Useful extensions of the results are

1. weighted STLS problems with cost function $\Delta p^\top W \Delta p$, where $W > 0$ is diagonal, and
2. regularized STLS problems, where the cost function is augmented with the regularization term $\operatorname{vec}^\top(X) Q \operatorname{vec}(X)$.

These extensions are still computable in $O(m)$ flops per iteration by our approach with small modifications [MV06]. For example, the weighted STLS problem leads to weight matrix Γ that is no longer block-Toeplitz but still block-banded with bandwidth independent of m. This property is sufficient for cost function and first derivative evaluation with computational complexity $O(m)$.

Chapter 5

Bilinear Errors-in-Variables Model

A bilinear EIV model is considered. It corresponds to an overdetermined set of linear equations $AXB = C$, in which the data A, B, C is perturbed by errors. An ALS estimator is constructed that converges to the true value of the parameter X as the number of rows in A and the number of columns in B tend to infinity.

The estimator is modified for an application in computer vision. A pair of corresponding points in two images are related via a bilinear equation, in which a parameter is the fundamental matrix. The fundamental matrix contains information about the relative orientation of the two images, and its estimation is a central problem in two-view motion analysis.

5.1 Introduction

In this section, we generalize the linear model

$$AX \approx B \tag{5.1}$$

to the bilinear in the measurements model

$$AXB \approx C. \tag{5.2}$$

An example where the bilinear model (5.2) occurs is the total production cost model.

Example 5.1 (Total production cost model) Assume that p production inputs (materials, parts, labor, etc.) are combined to make n products. Let b_k, $k = 1, \ldots, p$, be the price per unit of the kth production input and x_{jk}, $j = 1, \ldots, n$, $k = 1, \ldots, p$, be the number of units of the kth production input required to produce one unit of the jth product. The production cost per unit of the jth product is the jth element of the vector

$$y = Xb, \qquad y \in \mathbb{R}^n.$$

Let a_j, $j = 1, \ldots, n$, be a required quantity to be produced of the jth product. The total quantity needed of the kth production input is the kth element of the vector

$$z^\top = a^\top X, \qquad z \in \mathbb{R}^p.$$

The total production cost c is $z^\top b = a^\top y$, which gives a "single measurement" $AXB = C$ model

$$a^\top X b = c.$$

Multiple measurements occur when a set of quantities $a^{(1)}, \ldots, a^{(N_1)}$, to be produced of the n products, a set of prices per unit $b^{(1)}, \ldots, b^{(N_2)}$ of the production inputs, and a set of total costs c_{il}, corresponding to all combinations of the required quantities and prices, are given. Then the model is

$$\underbrace{\begin{bmatrix} a^{(1)\top} \\ \vdots \\ a^{(N_1)\top} \end{bmatrix}}_{A} X \underbrace{\begin{bmatrix} b^{(1)} & \cdots & b^{(N_2)} \end{bmatrix}}_{B} = \underbrace{\begin{bmatrix} c_{11} & \cdots & c_{1N_2} \\ \vdots & & \vdots \\ c_{N_1 1} & \cdots & c_{N_1 N_2} \end{bmatrix}}_{C}. \quad \blacksquare$$

Another example that involves the bilinear model (5.2) is the estimation of the fundamental matrix in two-view motion analysis. The fundamental matrix estimation problem is treated in detail in the latter sections of this chapter.

The TLS method applied for (5.2) results in the following optimization problem:

$$\min_{X, \Delta A, \Delta B, \Delta C} \left\| \begin{bmatrix} \Delta A & \Delta B & \Delta C \end{bmatrix} \right\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad (A - \Delta A) X (B - \Delta B) = C - \Delta C. \tag{5.3}$$

As mentioned in [Ful87], the TLS estimate $\hat{X}_{\text{tls}}$ (a global minimum point of (5.3)) is biased. Moreover (5.3) is a nonconvex optimization problem, whose solution requires computationally demanding optimization methods that are not guaranteed to find a global minimum point.

We use instead an ALS estimator that is consistent and computationally cheap. A strong assumption needed for the ALS estimator, however, is that the covariance structure of the measurement noises is known exactly. In contrast, the TLS method needs the covariances up to a scaling factor. In the fundamental matrix estimation problem, we derive a noise variance estimation procedure that overcomes this deficiency.

5.2 Adjusted Least Squares Estimation of a Bilinear Model

In the model (5.2), $A \in \mathbb{R}^{N_1 \times n}$, $B \in \mathbb{R}^{p \times q}$, and $C \in \mathbb{R}^{N_1 \times q}$ are observations and $X \in \mathbb{R}^{n \times p}$ is a parameter of interest. We assume that the observations are noisy measurements of true values $\bar{A}$, $\bar{B}$, and $\bar{C}$, i.e.,

$$A = \bar{A} + \tilde{A}, \qquad B = \bar{B} + \tilde{B}, \qquad C = \bar{C} + \tilde{C}. \tag{5.4}$$

The true values $\bar{A}$, $\bar{B}$, and $\bar{C}$ of the observations are assumed to satisfy the bilinear model $\bar{A}\bar{X}\bar{B} = \bar{C}$ for some true value $\bar{X}$ of the parameter. From the point of view of EIV modeling, $\tilde{C}$ represents the equation error, while $\tilde{A}$ and $\tilde{B}$ represent the measurement errors.

Looking for asymptotic results in the estimation of X, we fix the dimensions of X—n and p—and let the number of measurements, m and q, increase. The measurements

are represented by the rows of A, the columns of B, and the elements of C. Define the covariance matrices

$$V_{\tilde{A}} := \mathbf{E}\,\tilde{A}^\top \tilde{A}, \qquad V_{\tilde{B}} := \mathbf{E}\,\tilde{B}\tilde{B}^\top.$$

The assumptions are enumerated with Roman numerals.

(i) The elements of $\tilde{A}$, $\tilde{B}$, and $\tilde{C}$ are centered random variables with finite second order moments. The elements of any one of the matrices $\tilde{A}$, $\tilde{B}$, $\tilde{C}$ are independent of the elements of the other two matrices. The covariance matrices $V_{\tilde{A}}$ and $V_{\tilde{B}}$ are known.

Consider first the LS cost function

$$Q_{\text{ls}}(X; A, B, C) := \|AXB - C\|_{\text{F}}^2. \tag{5.5}$$

In the space of matrices $\mathbb{R}^{n\times p}$, we introduce a scalar product $\langle T, S\rangle := \operatorname{trace}(TS^\top)$. The derivative $\partial Q_{\text{ls}}/\partial X$ is a linear functional on $\mathbb{R}^{n\times p}$:

$$\begin{aligned}\frac{1}{2}\frac{\partial Q_{\text{ls}}}{\partial X}(H) &= \operatorname{trace}\big((AXB - C)(AHB)^\top\big)\\ &= \operatorname{trace}\big(A^\top(AXB - C)B^\top H^\top\big)\\ &= \langle A^\top(AXB - C)B^\top,\ H\rangle.\end{aligned} \tag{5.6}$$

We identify the derivative with the matrix that represents it in (5.6), so that we redefine

$$\frac{1}{2}\frac{\partial Q_{\text{ls}}}{\partial X} := A^\top(AXB - C)B^\top.$$

The LS estimator $\hat{X}_{\text{ls}}$ is the solution of the optimization problem

$$\min_X Q_{\text{ls}}(X; A, B, C)$$

or, equivalently, the solution of the score equation

$$\Psi_{\text{ls}}(X; A, B, C) := \partial Q_{\text{ls}}/\partial X = (A^\top A)X(BB^\top) - A^\top CB^\top = 0. \tag{5.7}$$

For the estimator, we can take

$$\hat{X}_{\text{ls}} := (A^\top A)^\dagger A^\top CB^\top (BB^\top)^\dagger,$$

which satisfies (5.7) if $A^\top A$ and $BB^\top$ are nonsingular. In the absence of measurement errors, i.e., when $\tilde{A} = 0$ and $\tilde{B} = 0$, the LS estimator is consistent.

If $\tilde{A} = 0$, the "partial least squares" estimator

$$\hat{X}_{\text{pa}} := \text{ TLS solution of } XB = (A^\top A)^\dagger A^\top C \tag{5.8}$$

is consistent. Similarly, if $\tilde{B} = 0$, the estimator

$$\hat{X}_{\text{pb}} := \text{ TLS solution of } AX = CB^\top (BB^\top)^\dagger \tag{5.9}$$

is consistent. The partial least squares estimators $\hat{X}_{\text{pa}}$ and $\hat{X}_{\text{pb}}$ are inconsistent when both A and B are noisy.

Next, we are looking for a corrected score function Ψ_{als}, such that

$$\mathbf{E}[\,\Psi_{\text{als}}(X;\bar{A}+\tilde{A},\bar{B}+\tilde{B},C)\mid C\,]=\Psi_{\text{ls}}(X;\bar{A},\bar{B},C),\quad \text{for all } X,\ \bar{A},\ \bar{B},\ \text{and } C.$$

The ALS estimator $\hat{X}_{\text{als}}$ is defined from the equation

$$\Psi_{\text{als}}(X;A,B,C)=0. \tag{5.10}$$

In order to solve (5.2), we look for a correction $\Delta\Psi$ applied on the LS score function Ψ_{ls}, such that $\Psi_{\text{als}}=\Psi_{\text{ls}}-\Delta\Psi$. By assumption (i),

$$\begin{aligned}&\mathbf{E}[\,\Psi_{\text{ls}}(X;\bar{A}+\tilde{A},\bar{B}+\tilde{B},C)\mid C\,]\\&=\Psi_{\text{ls}}(X;\bar{A},\bar{B},C)+\mathbf{E}\,\tilde{A}^\top\tilde{A}X\bar{B}\bar{B}^\top+\mathbf{E}\,\bar{A}^\top\bar{A}X\tilde{B}\tilde{B}^\top+V_{\tilde{A}}XV_{\tilde{B}}\\&=\Psi_{\text{ls}}+\Delta\Psi_1(\bar{B})+\Delta\Psi_2(\bar{A})+V_{\tilde{A}}XV_{\tilde{B}},\end{aligned}$$

where

$$\Delta\Psi_1(\bar{B}):=V_{\tilde{A}}X\bar{B}\bar{B}^\top\quad\text{and}\quad\Delta\Psi_2(\bar{A}):=\bar{A}^\top\bar{A}XV_{\tilde{B}}.$$

To find a proper correction term $\Delta\Psi$, consider

$$\mathbf{E}\,\Delta\Psi_1(\bar{B}+\tilde{B})=V_{\tilde{A}}X\bar{B}\bar{B}^\top+V_{\tilde{A}}XV_{\tilde{B}} \tag{5.11}$$

and

$$\mathbf{E}\,\Delta\Psi_2(\bar{A}+\tilde{A})=\bar{A}^\top\bar{A}XV_{\tilde{B}}+V_{\tilde{A}}XV_{\tilde{B}}. \tag{5.12}$$

Then

$$\Delta\Psi(A,B)=\Delta\Psi_1(B)+\Delta\Psi_2(A)-V_{\tilde{A}}XV_{\tilde{B}}$$

and

$$\begin{aligned}&\Psi_{\text{als}}(X;A,B,C)\\&=(A^\top A)X(BB^\top)-A^\top CB^\top-V_{\tilde{A}}X(BB^\top)-(A^\top A)XV_{\tilde{B}}+V_{\tilde{A}}XV_{\tilde{B}}\\&=(A^\top A-V_{\tilde{A}})X(BB^\top-V_{\tilde{B}})-A^\top CB^\top.\end{aligned}$$

As an estimator we can take

$$\hat{X}_{\text{als}}:=(A^\top A-V_{\tilde{A}})^\dagger(A^\top CB^\top)(BB^\top-V_{\tilde{B}})^\dagger. \tag{5.13}$$

If $A^\top A-V_{\tilde{A}}$ and $BB^\top-V_{\tilde{B}}$ are nonsingular, then (5.13) satisfies (5.10). These matrices are nonsingular with probability tending to one as the number of measurements tend to infinity.

5.3 Properties of the Adjusted Least Squares Estimator

We introduce further assumptions.

(ii) The rows of $\tilde{A}$ are independent, the columns of $\tilde{B}$ are independent, and all elements of $\tilde{C}$ are independent.

(iii) $\mathbf{E}\,\tilde{a}_{ij}^4 \leq \text{const}$, $\mathbf{E}\,\tilde{b}_{kl}^4 \leq \text{const}$, and $\mathbf{E}\,\tilde{c}_{il}^2 \leq \text{const}$.

(iv) With $V_{\bar{A}} := \bar{A}^\top \bar{A}$ and $V_{\bar{B}} := \bar{B}\bar{B}^\top$,

$$\frac{\lambda_{\max}(V_{\bar{A}}) + m}{\lambda_{\min}^2(V_{\bar{A}})} \to 0 \text{ as } N_1 \to \infty \quad \text{and} \quad \frac{\lambda_{\max}(V_{\bar{B}}) + q}{\lambda_{\min}^2(V_{\bar{B}})} \to 0 \text{ as } N_2 \to \infty.$$

Assumption (iv) corresponds to the condition of weak consistency, given in [Gal82] for the maximum likelihood estimator in the linear model (5.1).

Theorem 5.2 (Weak consistency). *Under assumptions (i) to (iv), the ALS estimator $\hat{X}_{\text{als}}$ converges to $\bar{X}$ in probability as $N_1 \to \infty$ and $N_2 \to \infty$.*

Proof. See [KMV03, Theorem 1]. □

Under more restrictive assumptions than (iii) and (iv), the ALS estimator is strongly consistent.

(iii') $\mathbf{E}\,|\tilde{a}_{ij}|^{2r} \leq \text{const}$, $\mathbf{E}\,|\tilde{b}_{kl}|^{2r} \leq \text{const}$, and $\mathbf{E}\,|\tilde{c}_{il}|^{2r} \leq \text{const}$ for a fixed real number $r \geq 2$.

(iv') For a fixed $N_1' \geq 1$,

$$\sum_{N_1=N_1'}^{\infty} \left(\frac{N_1^{r/2}}{\lambda_{\min}^r(V_{\bar{A}})} + \frac{\lambda_{\max}^r(V_{\bar{A}})}{\lambda_{\min}^{2r}(V_{\bar{A}})} \right) < \infty,$$

and for a fixed $N_2' \geq 1$,

$$\sum_{N_2=N_2'}^{\infty} \left(\frac{N_2^{r/2}}{\lambda_{\min}^r(V_{\bar{B}})} + \frac{\lambda_{\max}^r(V_{\bar{B}})}{\lambda_{\min}^{2r}(V_{\bar{B}})} \right) < \infty,$$

where r is defined in assumption (iii').

Theorem 5.3 (Strong consistency). *Under assumptions (i), (ii), (iii'), and (iv'), the ALS estimator $\hat{X}_{\text{als}}$ converges to $\bar{X}$ almost surely as $N_1 \to \infty$ and $N_2 \to \infty$.*

Proof. See [KMV03, Theorem 2]. □

In [KMV03, Section 5], we prove the following rate of convergence:

$$\|\hat{X}_{\text{als}} - \bar{X}\|_{\text{F}} = \left(\frac{\sqrt{N_1} + \sqrt{\lambda_{\max}(V_{\bar{A}})}}{\lambda_{\min}(V_{\bar{A}})} + \frac{\sqrt{N_2} + \sqrt{\lambda_{\max}(V_{\bar{B}})}}{\lambda_{\min}(V_{\bar{B}})} \right) \mathrm{O_p}(1)\,.$$

Under additional assumptions, the ALS estimator is asymptotically normal; see [KMV03, Section 6]. It turns out that the asymptotic covariance matrix of the estimator does not depend upon the covariance structure of $\tilde{C}$.

In [KMV03, Section 7] a heuristic small sample modification of the ALS estimator is proposed. The approach is similar to the one of [CST00]. The modified ALS estimator has the same asymptotic properties as the ALS estimator but improves the results for small sample size.

5.4 Simulation Examples

In this section, we apply the ALS estimator to a hypothetical example. Consider the bilinear model (5.2) with $N_1 = N_2 = N$ and $n = p = 2$, i.e.,

$$\underbrace{A}_{N\times 2}\underbrace{X}_{2\times 2}\underbrace{B}_{2\times N} = \underbrace{C}_{N\times N}.$$

The true data is

$$\bar{A} = \begin{bmatrix} I_2 \\ \vdots \\ I_2 \end{bmatrix}, \quad \bar{B} = \begin{bmatrix} I_2 & \cdots & I_2 \end{bmatrix}, \quad \text{and} \quad \bar{C} = \begin{bmatrix} I_2 & \cdots & I_2 \\ \vdots & & \vdots \\ I_2 & \cdots & I_2 \end{bmatrix},$$

so that the true value of the parameter is $\bar{X} = I_2$. The perturbations $\tilde{A}$, $\tilde{B}$, and $\tilde{C}$ are selected in three different ways.

- *Equally sized errors.* All errors $\tilde{a}_{ij}$, $\tilde{b}_{kl}$, $\tilde{c}_{il}$ are independent, centered, and normally distributed with common variance 0.01.
- *Differently sized errors.* All errors $\tilde{a}_{ij}$, $\tilde{b}_{kl}$, $\tilde{c}_{il}$ are independent, centered, and normally distributed. The elements in the first column of $\tilde{A}$ have variance 0.05 and the elements in the second column of $\tilde{A}$ have variance 0.01. The elements in the first row of $\tilde{B}$ have variance 0.01 and the elements in the second row of $\tilde{B}$ have variance 0.05. All elements of $\tilde{C}$ have variance 0.01.
- *Correlated errors.* All errors $\tilde{a}_{ij}$, $\tilde{b}_{kl}$, $\tilde{c}_{il}$ are independent and normally distributed. All rows of $\tilde{A}$ have covariance $0.01\left[\begin{smallmatrix} 5 & 1 \\ 1 & 1 \end{smallmatrix}\right]$ and the elements are independent from row to row. All columns of $\tilde{B}$ have covariance $0.01\left[\begin{smallmatrix} 1 & 1 \\ 1 & 5 \end{smallmatrix}\right]$ and the elements are independent from column to column.

The estimation is performed for increasing number of measurements N. As a measure of the estimation quality, we use the empirical relative mean square error

$$e(N) = \frac{1}{K}\sum_{s=1}^{K} \frac{\|\bar{X} - \hat{X}^{(s)}\|_{\mathrm{F}}^2}{\|\bar{X}\|_{\mathrm{F}}^2},$$

where $\hat{X}^{(s)}$ is the estimate computed for the sth noise realization.

The following estimators are compared:

$\hat{X}_{\text{als}}$—the ALS estimator,

$\hat{X}_{\text{m}}$—the small sample modified ALS estimator [KMV03, Section 7],

$\hat{X}_{\text{ls}}$—the LS estimator, and

$\hat{X}_{\text{pa}}$ and $\hat{X}_{\text{pb}}$—the partial least squares estimators (5.8) and (5.9).

Figure 5.1 shows the relative mean square error e for N ranging from 20 to 100. The consistency properties of the ALS estimators and the bias of the other estimators is most clearly seen in the experiment with correlated errors. Note that the small sample modification indeed improves the relative mean square error and for large sample size converges to the original ALS estimator.

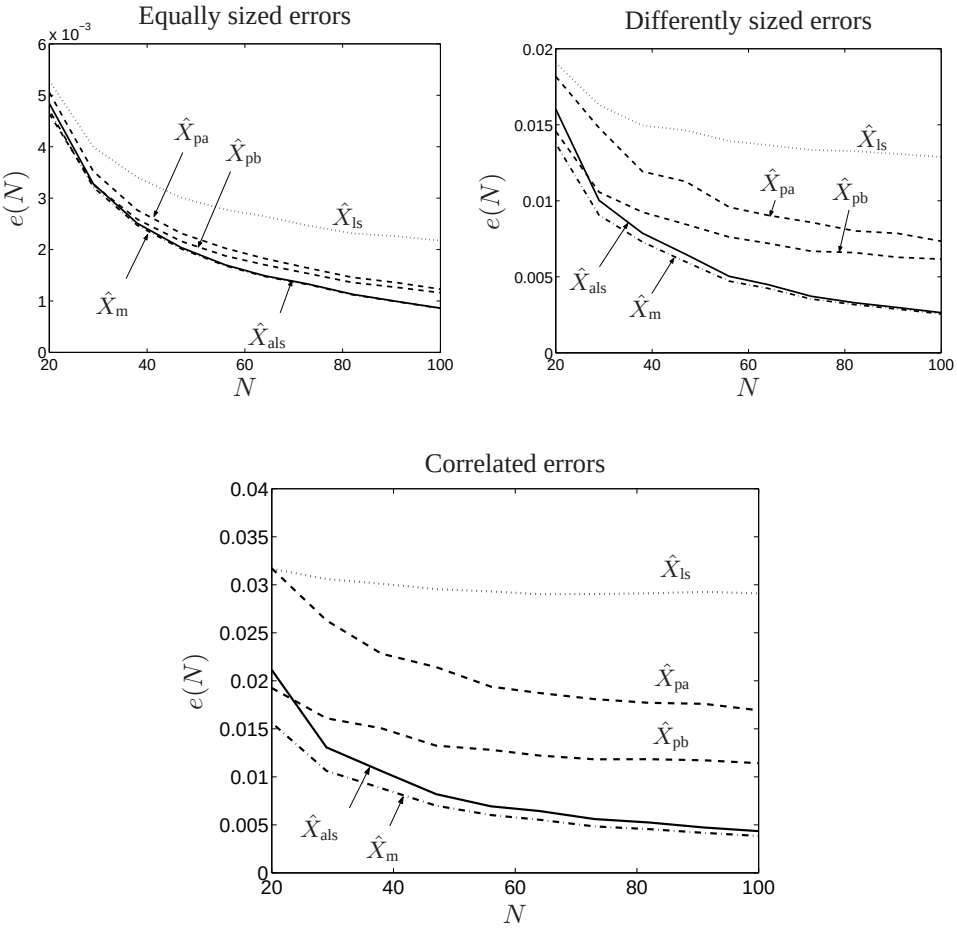


Figure 5.1. *Relative mean square error e as a function of N.*

5.5 Fundamental Matrix Estimation

Suppose that two images are captured by a mobile camera and N matching pairs of pixels are located. Let

$$u^{(i)} = \mathrm{col}(u_1^{(i)}, u_2^{(i)}, 1) \quad \text{and} \quad v^{(i)} = \mathrm{col}(v_1^{(i)}, v_2^{(i)}, 1), \quad \text{for } i = 1, \ldots, N,$$

be the homogeneous pixel coordinates in the first and second images, respectively. The so-called epipolar constraint

$$v^{(i)\top} F u^{(i)} = 0, \qquad i = 1, \ldots, N, \tag{5.14}$$

relates the corresponding matching pixels, where $F \in \mathbb{R}^{3\times 3}$, $\mathrm{rank}(F) = 2$ is the fundamental matrix. Estimation of F from the given data $v^{(i)}$, $u^{(i)}$, $i = 1, \ldots, N$, is called structure from motion problem and is a central problem in computer vision. We adapt the ALS estimator (5.13) to the fundamental matrix estimation problem.

The fundamental matrix estimation problem is not a special case of the bilinear model considered in Sections 5.1–5.4. Note that with

$$A := \begin{bmatrix} v^{(1)} & \cdots & v^{(N)} \end{bmatrix}^\top, \quad B := \begin{bmatrix} u^{(1)} & \cdots & u^{(N)} \end{bmatrix}, \quad \text{and} \quad C := 0,$$

(5.14) represents only the diagonal elements of the equation $AFB = C$. Moreover, the C matrix is noise-free, and the parameter F is of rank two and normalized. Thus the ALS estimator derived for the bilinear model (5.2) cannot be used directly for the estimation of the fundamental matrix F.

As in Section 5.1, we assume that the given points are noisy observations

$$u^{(i)} = \bar{u}^{(i)} + \tilde{u}^{(i)}, \quad \text{and} \quad v^{(i)} = \bar{v}^{(i)} + \tilde{v}^{(i)}, \quad \text{for} \quad i = 1, \ldots, N \tag{5.15}$$

of true values $\bar{u}^{(i)}$ and $\bar{v}^{(i)}$ that satisfy the model

$$\bar{v}^{(i)\top} \bar{F} \bar{u}^{(i)} = 0, \quad \text{for} \quad i = 1, \ldots, N \tag{5.16}$$

for some true value $\bar{F} \in \mathbb{R}^{3\times 3}$, $\mathrm{rank}(\bar{F}) = 2$ of the parameter F. In addition, we assume that $\bar{F}$ is normalized by $\|\bar{F}\|_{\mathrm{F}} = 1$.

In the absence of noise, equations (5.14) have an exact solution $\bar{F}$. The so-called eight-point algorithm [Har97] computes it from $N = 8$ given pairs of points. A lot of techniques have been proposed to improve the accuracy of the eight-point algorithm in the presence of noise [TM97, MM98, LM00]. From a statistical point of view, however, the corresponding estimators are inconsistent. We review in more detail the estimator proposed in [MM98].

In [MM98], the model (5.14) is transformed into the form

$$\big(\underbrace{u^{(i)} \otimes v^{(i)}}_{a^{(i)}} \big)^\top \underbrace{\mathrm{vec}(F)}_{f} = 0, \quad \text{for} \quad i = 1, \ldots, N. \tag{5.17}$$

Defining the matrix $A := \begin{bmatrix} a^{(1)} & \cdots & a^{(N)} \end{bmatrix}^\top$, (5.17) becomes the system of equations $Af = 0$. The normalization condition $\|F\|_{\mathrm{F}} = 1$ for F implies the normalization condition $\|f\| = 1$ for f. With noisy data, an estimate of f can be computed by solving the optimization problem

$$\min_f \|Af\|_2^2 \quad \text{subject to} \quad \|f\| = 1, \tag{5.18}$$

which is a quadratically constrained least squares problem, so that its global solution $\hat{f}_{\text{ls},1}$ can be computed from the SVD of A. The corresponding estimate $\hat{F}_{\text{ls},1}$ is constructed from $\hat{f}_{\text{ls},1}$ in an obvious way. We denote this construction by $\hat{F}_{\text{ls},1} := \operatorname{vec}^{-1}(\hat{f}_{\text{ls},1})$. Finally, the rank constraint is enforced by approximating $\hat{F}_{\text{ls},1}$ with the nearest (in Frobenius norm) rank-deficient matrix $\hat{F}_{\text{ls}}$. This requires another SVD.

For statistical analysis of $\hat{F}_{\text{ls}}$, the vectors $a^{(i)}$ are interpreted as observations

$$a^{(i)} = \bar{u}^{(i)} \otimes \bar{v}^{(i)} + \tilde{a}^{(i)}. \tag{5.19}$$

The estimator $\hat{F}_{\text{ls}}$ is consistent under the assumption that the errors $\tilde{a}^{(1)}, \dots, \tilde{a}^{(N)}$ are zero mean independent and identically distributed (i.i.d.) random vectors. Such an assumption, however, is not satisfied for the EIV model (5.15). Suppose that the measurements $u^{(i)}$ and $v^{(i)}$ are obtained with additive errors $\tilde{v}^{(i)}$ and $\tilde{v}^{(i)}$ that are independent zero mean i.i.d. normal random vectors. Then the vectors $\tilde{a}^{(i)}$ are not normally distributed because their elements involve the product of two coordinates. It can be shown that

$$\begin{aligned}\mathbf{E}\,\tilde{a}^{(i)}\tilde{a}^{(i)\top} = \big(\mathbf{E}\,\tilde{u}^{(i)}\tilde{u}^{(i)\top}\big) \otimes \big(\bar{v}^{(i)}\bar{v}^{(i)\top}\big) + \big(\bar{u}^{(i)}\bar{u}^{(i)\top}\big) \otimes \big(\mathbf{E}\,\tilde{v}^{(i)}\tilde{v}^{(i)\top}\big) \\ + \big(\mathbf{E}\,\tilde{v}^{(i)}\tilde{v}^{(i)\top}\big) \otimes \big(\mathbf{E}\,\tilde{u}^{(i)}\tilde{u}^{(i)\top}\big).\end{aligned}$$

In [MM98], the estimator $\hat{F}_{\text{ls}}$ is called the TLS estimator. In fact, $\hat{F}_{\text{ls}}$ is the TLS estimator for the transformed model $Af = 0$, $\|f\| = 1$. The TLS estimator for the original problem (5.14) is (compare with (5.3))

$$\min_{\substack{F \\ \Delta u^{(1)},\dots,\Delta u^{(N)} \\ \Delta v^{(1)},\dots,\Delta v^{(N)}}} \sum_{i=1}^{N} \Big(\|\Delta u^{(i)}\|^2 + \|\Delta v^{(i)}\|^2 \Big) \quad \text{subject to} \quad (u^{(i)} + \Delta u^{(i)})^\top F (v^{(i)} + \Delta v^{(i)}) = 0 \quad \text{for } i = 1, \dots, N, \tag{5.20}$$

which is a different problem. It is a nonconvex optimization problem. Moreover, as noted in Section 5.1, the TLS estimator $\hat{X}_{\text{tls}}$ (a global minimum point of (5.20)) is also biased. We refer to $\hat{F}_{\text{ls}}$ as the LS estimator because in terms of the bilinear model (5.14) it minimizes the equation error Af.

We make the following assumptions on the errors $\tilde{u}^{(i)}$ and $\tilde{v}^{(i)}$ in (5.15).

(i) $\tilde{u}^{(i)}$ and $\tilde{v}^{(i)}$, for $i = 1, \dots, N$, are zero mean independent random variables.

(ii) $\operatorname{cov}(\tilde{u}^{(i)}) = \operatorname{cov}(\tilde{v}^{(i)}) = \bar{\sigma}^2 \cdot \operatorname{diag}(1, 1, 0)$, for $i = 1, \dots, N$ and certain $\bar{\sigma} > 0$.

Let $\tilde{u}^{(i)} := \operatorname{col}(\tilde{u}_1^{(i)}, \tilde{u}_2^{(i)}, \tilde{u}_3^{(i)})$. Assumption (ii) means that the components of $\tilde{u}^{(i)}$ are uncorrelated, $\tilde{u}_3^{(i)}$ is noise-free, and $\operatorname{var}\big(\tilde{u}_1^{(i)}\big) = \operatorname{var}\big(\tilde{u}_2^{(i)}\big) = \bar{\sigma}^2$. The same holds for $\tilde{v}^{(i)}$. These are more natural assumptions for the application at hand than the assumptions on $\tilde{a}^{(i)}$ needed for consistency of the LS estimator.

5.6 Adjusted Least Squares Estimation of the Fundamental Matrix

The LS cost function is

$$Q_{\text{ls}}(F) := \sum_{i=1}^{N} q_{\text{ls}}(F; u^{(i)}, v^{(i)}), \quad \text{where} \quad q_{\text{ls}}(F; u, v) := \big(v^\top F u\big)^2.$$

Next, we construct an adjusted cost function $Q_{\text{als}}(F)$ that leads to a consistent estimator. It is defined by the identity

$$\mathbf{E}\, Q_{\text{als}}(F) = Q_{\text{ls}}(F) \quad \text{for all } F \in \mathbb{R}^{3\times 3} \text{ and } \bar{u}^{(i)}, \bar{v}^{(i)} \in \mathbb{R}^3.$$

By assumption (i),

$$Q_{\text{als}}(F) = \sum_{i=1}^{N} q_{\text{als}}(F; u^{(i)}, v^{(i)}),$$

where q_{als} satisfies the identity

$$\mathbf{E}\, q_{\text{als}}(F; \bar{u} + \tilde{u}, \bar{v} + \tilde{v}) = q_{\text{ls}}(F; \bar{u}, \bar{v}), \quad \text{for all } F \in \mathbb{R}^{3\times 3} \text{ and } \bar{u}, \bar{v} \in \mathbb{R}^3,$$

and $\tilde{u} \sim \mathrm{N}(0, V)$, $\tilde{v} \sim \mathrm{N}(0, V)$ independent, with $V := \sigma^2 \operatorname{diag}(1, 1, 0)$.

The solution of (5.6) is

$$q_{\text{als}}(F, u, v) := \operatorname{trace}\Big(\big(vv^\top - V\big)F\big(uu^\top - V\big)F^\top\Big). \tag{5.21}$$

Indeed,

$$\begin{aligned}
&\mathbf{E}\, q_{\text{als}}(F; \bar{u} + \tilde{u}, \bar{v} + \tilde{v}) \\
&= \mathbf{E} \operatorname{trace}\Big(\big((\bar{v} + \tilde{v})(\bar{v} + \tilde{v})^\top - V\big)F\big((\bar{u} + \tilde{u})^\top(\bar{u} + \tilde{u})^\top - V\big)F^\top\Big) \\
&= \mathbf{E} \operatorname{trace}\Big(\big(\bar{v}\bar{v}^\top + 2\bar{v}\tilde{v}^\top + (\tilde{v}\tilde{v}^\top - V)\big)F\big(\bar{u}\bar{u}^\top + 2\bar{u}\tilde{u}^\top + (\tilde{u}\tilde{u}^\top - V)\big)F^\top\Big).
\end{aligned}$$

After expanding the right-hand side and applying the expectation operator to the summands, assumptions (i) and (ii) imply that all summands except for the first one are equal to zero. Thus

$$\mathbf{E}\, q_{\text{als}}(F, \bar{u} + \tilde{u}, \bar{v} + \tilde{v}) = \operatorname{trace}\Big(\big(\bar{v}\bar{v}^\top\big)F\big(\bar{u}\bar{u}^\top\big)F^\top\Big).$$

But

$$\operatorname{trace}\Big(\big(\bar{v}\bar{v}^\top\big)F\big(\bar{u}\bar{u}^\top\big)F^\top\Big) = \big(\bar{u}^\top F^\top \bar{v}\big)\big(\bar{v}^\top F \bar{u}\big) = \big(\bar{v}^\top F \bar{u}\big)^2 = q_{\text{ls}}(F, \bar{u}, \bar{v}).$$

Then the solution of (5.6) is given by

$$Q_{\text{als}}(F) = \operatorname{trace}\Big(\sum_{i=1}^{N} \big(v^{(i)} v^{(i)\top} - V\big)F\big(u^{(i)} u^{(i)\top} - V\big)F^\top\Big).$$

With $f := \operatorname{vec}(F)$,

$$Q_{\text{als}}(F) = f^\top \left(\sum_{i=1}^{N} \left(u^{(i)} u^{(i)\top} - V\right) \otimes \left(v^{(i)} v^{(i)\top} - V\right) \right) f.$$

Denote

$$S_N := \sum_{i=1}^{N} \left(u^{(i)} u^{(i)\top} - V\right) \otimes \left(v^{(i)} v^{(i)\top} - V\right) \tag{5.22}$$

and let

$$\hat{f}_{\text{als},1} \in \arg\min f^\top S_N f \quad \text{subject to} \quad \|f\| = 1. \tag{5.23}$$

The matrix S_N is symmetric, so that the ALS estimator $\hat{f}_{\text{als},1}$ is a normalized eigenvector of S_N associated with the smallest eigenvalue of S_N.

Let $\hat{F}_{\text{als},1} := \operatorname{vec}^{-1}(\hat{f}_{\text{als},1})$. If $\operatorname{rank}(\hat{F}_{\text{als},1}) = 3$, it is approximated by a rank-deficient matrix. Let $\hat{F}_{\text{als},1} = U \Sigma V^\top$, where $\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \sigma_3)$ and $U, V \in \mathbb{R}^{3\times 3}$ are orthogonal matrices, be an SVD of $\hat{F}_{\text{als},1}$. The ALS estimator on the second stage is defined as

$$\hat{F}_{\text{als}} := U \operatorname{diag}(\sigma_1, \sigma_2, 0) V^\top,$$

i.e., the best low-rank approximation of $\hat{F}_{\text{als},1}$, according to the Eckart–Young–Mirsky theorem [EY36].

5.7 Properties of the Fundamental Matrix Estimator*

Consistency of the estimator $\hat{F}_{\text{als},1}$ implies consistency of the estimator $\hat{F}_{\text{als}}$. Indeed, suppose that $\|\hat{F}_{\text{als},1} - \bar{F}\|_{\text{F}} \le \varepsilon$. Because $\operatorname{rank}(\bar{F}) = 2$, for the estimator $\hat{F}_{\text{als}}$ on the second stage, we have

$$\|\hat{F}_{\text{als},1} - \hat{F}_{\text{als}}\|_{\text{F}} \le \|\hat{F}_{\text{als},1} - \bar{F}\|_{\text{F}} \le \varepsilon. \tag{5.24}$$

Then

$$\|\hat{F}_{\text{als}} - \bar{F}\|_{\text{F}} \le \|\hat{F}_{\text{als}} - \hat{F}_{\text{als},1}\|_{\text{F}} + \|\hat{F}_{\text{als},1} - \bar{F}\|_{\text{F}} \le 2\varepsilon.$$

Note that the matrix $-\bar{F}$ also satisfies (5.16), and $\| - \bar{F}\|_{\text{F}} = \|\bar{F}\|_{\text{F}} = 1$. Therefore we estimate $\bar{F}$ up to a sign.

Introduce the matrix

$$\mathscr{F}_N := \frac{1}{N} \sum_{i=1}^{N} \left(\bar{u}^{(i)} \bar{u}^{(i)\top}\right) \otimes \left(\bar{v}^{(i)} \bar{v}^{(i)\top}\right). \tag{5.25}$$

For the vector $\bar{f} := \operatorname{vec}(\bar{F})$, we have (see (5.16))

$$\bar{f}^\top \mathscr{F}_N \bar{f} = \frac{1}{N} \sum_{i=1}^{N} \operatorname{trace}(\bar{v}^{(i)} \bar{v}^{(i)\top} \bar{F} \bar{u}^{(i)} \bar{u}^{(i)\top} \bar{F}^\top) = 0,$$

and $\mathscr{F}_N \ge 0$. Thus $\lambda_{\min}(\mathscr{F}_N) = 0$. We require that there exists N' such that $\operatorname{rank}(\mathscr{F}_N) = 8$ for $N \ge N'$. Moreover, we need a stronger assumption.

Let $\lambda_1(\mathscr{F}_N) \ge \lambda_2(\mathscr{F}_N) \ge \cdots \ge \lambda_9(\mathscr{F}_N) = 0$ be the eigenvalues of $\mathscr{F}_N$.

(iii) There exist $N' \geq 1$ and $c_0 > 0$, such that for all $N \geq N'$, $\lambda_8(\mathscr{F}_N) \geq c_0$.

The minimization problem (5.23) could have a nonunique solution, but due to assumption (iii), for $N > N'(\omega)$ the smallest eigenvalue of S_N will be unique and then the estimator $\hat{f}_{\text{als},1}$ will be uniquely defined, up to a sign.

The next assumptions are needed for the convergence

$$\frac{1}{N} S_N - \mathscr{F}_N \to 0 \quad \text{almost surely} \quad \text{as} \quad N \to \infty. \tag{5.26}$$

(iv) $\frac{1}{N} \sum_{i=1}^{N} \|\bar{u}^{(i)}\|^4 \leq \text{const}$ and $\frac{1}{N} \sum_{i=1}^{N} \|\bar{v}^{(i)}\|^4 \leq \text{const}$.

(v) For fixed $\delta > 0$, $\mathbf{E}\,\|\tilde{u}^{(i)}\|^{4+\delta} \leq \text{const}$ and $\mathbf{E}\,\|\tilde{v}^{(i)}\|^{4+\delta} \leq \text{const}$.

For two matrices A and B of the same size, define the distance between A and B as the Frobenius norm of their difference, i.e.,

$$\text{dist}(A, B) := \|A - B\|_{\text{F}}.$$

Theorem 5.4 (Strong consistency). *Under assumptions (i) to (v),*

$$\text{dist}(\hat{F}_{\text{als}}, \{-\bar{F}, +\bar{F}\}) \to 0 \quad \textit{almost surely} \quad \textit{as} \quad N \to \infty. \tag{5.27}$$

Proof. See [KMV02, Theorem 1]. □

The computation of the ALS estimator needs knowledge of the noise variance $\bar{\sigma}^2$. When $\bar{\sigma}^2$ is unknown, it can be estimated as follows:

$$\hat{\sigma}^2 = \arg\min_{\sigma} \left| \lambda_{\min}\big(S_N(\sigma^2)\big) \right|.$$

In [KMV02, Section 3], the ALS estimator using the noise variance estimate $\hat{\sigma}^2$ instead of the true noise variance $\bar{\sigma}^2$ is proven to be consistent.

5.8 Simulation Examples

In this section, we present numerical results with the estimators $\hat{F}$ and $\hat{\sigma}^2$. The data is simulated. The fundamental matrix $\bar{F}$ is a randomly chosen rank two matrix with unit Frobenius norm. The true coordinates $\bar{u}^{(i)}$ and $\bar{v}^{(i)}$ have third components equal to one, and the first two components are vectors with unit norm and random direction. The perturbations $\tilde{u}^{(i)}$ and $\tilde{v}^{(i)}$ are selected according to the assumptions stated in this book; i.e., the third components $\tilde{u}_3^{(i)} = \tilde{v}_3^{(i)} = 0$, and $\tilde{u}_j^{(i)}, \tilde{v}_j^{(i)} \sim \text{N}(0, \bar{\sigma}^2)$, are independent for $i = 1, \ldots, N$ and $j = 1, 2$. In each experiment, the estimation is repeated 1000 times with the same true data and different noise realizations.

The true value of the parameter $\bar{F}$ is known, which allows evaluation of the results. We compare three estimators:

$\hat{F}_{\text{als}}(\bar{\sigma}^2)$—the ALS estimator using the true noise variance $\bar{\sigma}^2$,

$\hat{F}_{\text{als}}(\hat{\sigma}^2)$—the ALS estimator using the estimated noise variance $\hat{\sigma}^2$, and

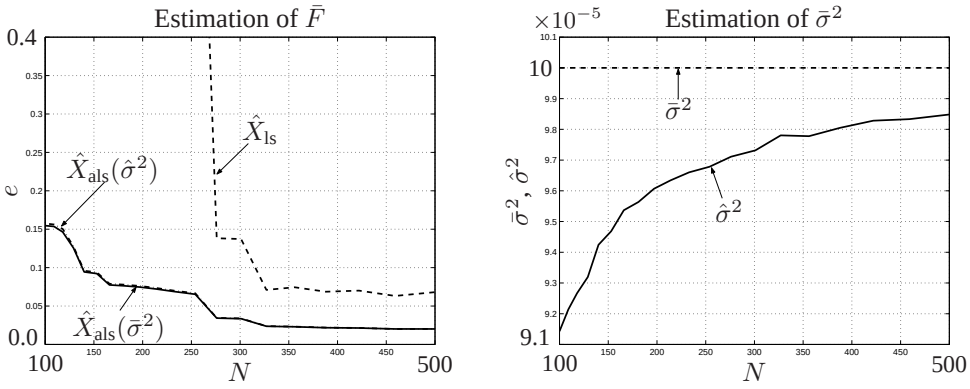


Figure 5.2. *Left: relative error of estimation* $e := \|\bar{F} - \hat{F}\|_{\text{F}}/\|\bar{F}\|_{\text{F}}$ *as a function of the sample size* N*. Right: convergence of the noise variance estimate* $\hat{\sigma}^2$ *to the true value* $\bar{\sigma}^2$*.*

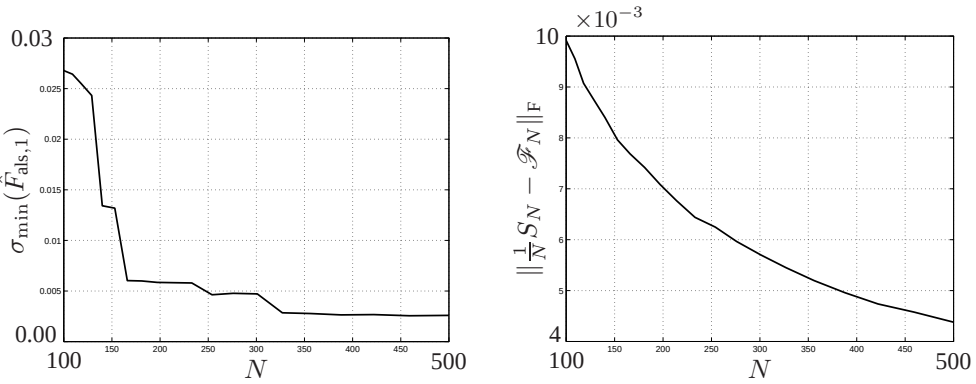


Figure 5.3. *Left: distance from* $\hat{F}_{\text{als},1}$ *to the set of rank-deficient matrices. Right: convergence of* $\frac{1}{N}S_N$ *to* $\mathscr{F}_N$*.*

$\hat{F}_{\text{ls}}$—the LS estimator.

Figure 5.2 shows the relative error of estimation $e := \|\bar{F} - \hat{F}\|_{\text{F}}/\|\bar{F}\|_{\text{F}}$ as a function of the sample size N in the left plot and the convergence of the estimate $\hat{\sigma}^2$ in the right plot. Figure 5.3, left plot, shows the convergence of the estimator $\hat{F}_{\text{als},1}$ to the set of rank-deficient matrices. This empirically confirms inequality (5.24). The right plot in Figure 5.3 confirms the convergence of $\frac{1}{N}S_N \to \mathscr{F}_N$ as $N \to \infty$; see (5.26).

Figure 5.4 shows the function $S_N(\sigma^2)$, used in the estimation of $\bar{\sigma}^2$, for $N = 500$ in the left plot and for $N = 30$ in the right plot. In general, $S_N(\sigma^2)$ is a nonconvex, nondifferentiable function with many local minima. However, we observed empirically that the number of local minima decreases as N increases. For larger sample sizes and smaller noise variance, the function $S_N(\sigma^2)$ becomes unimodal.

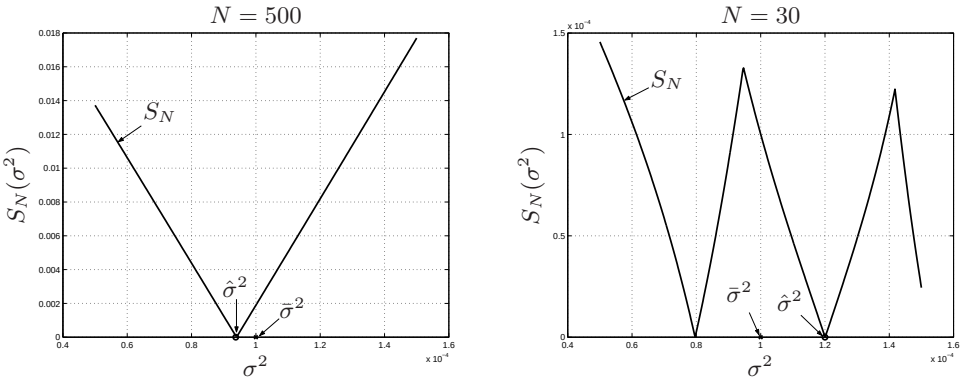


Figure 5.4. *The function $S_N(\sigma^2)$ used for the estimation of $\bar{\sigma}^2$. Left: large sample size. Right: small sample size.*

5.9 Conclusions

We considered the bilinear model $AXB = C$. The TLS estimator is inconsistent in this case. We constructed the ALS estimator, which is consistent and computationally cheap. Conditions for weak and strong consistency were listed.

An open question is, What are the optimality properties of the ALS estimator? For the linear model $AX = B$, in [KM00] it was shown that the ALS estimator is asymptotically efficient when $V_{\tilde{A}}$ is known exactly and $\mathbf{E}\,\tilde{b}_{kl}^2$ are known up to a constant factor. It would be interesting to check the following conjecture:

> In the model $AXB = C$, the ALS estimator is asymptotically efficient when $V_{\tilde{A}}$ and $V_{\tilde{B}}$ are known exactly and $\mathbf{E}\,\tilde{c}_{il}^2$ are known up to a constant factor.

Estimation of the rank-deficient fundamental matrix, yielding information about the relative orientation of two images in two-view motion analysis, was considered. A consistent estimator was derived by minimizing a corrected contrast function in a bilinear EIV model. The proposed ALS estimator was computed in three steps: first, estimate the measurement error variance; second, construct a preliminary matrix estimate; and third, project that estimate onto the space of singular matrices.

Chapter 6

Ellipsoid Fitting

A parameter estimation problem for ellipsoid fitting in the presence of measurement errors is considered. The LS estimator is inconsistent and, due to the nonlinearity of the model, the orthogonal regression estimator is inconsistent as well; i.e., these estimators do not converge to the true value of the parameters as the sample size tends to infinity. A consistent estimator is proposed, based on a proper correction of the LS estimator. The correction is explicitly given in terms of the true value of the noise variance.

In Section 6.2, we define the quadratic EIV model. The LS and ALS estimators are defined in Sections 6.3 and 6.4. Ellipsoid estimates are derived from the general quadratic model estimates in Section 6.5. An algorithm for ALS estimation is outlined in Section 6.6. We show simulation examples in Section 6.7.

6.1 Introduction

In this chapter, we consider the ellipsoid fitting problem: given a set of data points

$$x^{(1)}, \ldots, x^{(N)}, \qquad \text{where} \quad x^{(i)} \in \mathbb{R}^n,$$

find an ellipsoid

$$\mathscr{E}(A_{\mathrm{e}}, c) := \{\, x \in \mathbb{R}^n : (x-c)^\top A_{\mathrm{e}}(x-c) = 1 \,\}, \quad A_{\mathrm{e}} = \hat{A}_{\mathrm{e}}^\top > 0, \tag{6.1}$$

that "best matches" them. The freedom in the choice of the matching criterion gives rise to different estimation methods.

One approach, called algebraic fitting, is to solve the optimization problem

$$\min_{A_{\mathrm{e}}, c} \sum_{i=1}^{N} \left((x^{(i)} - c)^\top A_{\mathrm{e}} (x^{(i)} - c) - 1 \right)^2 \tag{6.2}$$

and to define the estimate as any global optimal point. We will refer to (6.2) as the LS method for the ellipsoid model.

Another approach, called geometric fitting, is to solve the optimization problem

$$\min_{A_{\mathrm{e}},c} \sum_{i=1}^{N} \Big(\operatorname{dist}\big(x^{(i)}, \mathscr{E}(A_{\mathrm{e}},c)\big)\Big)^2, \tag{6.3}$$

where $\operatorname{dist}(x,\mathscr{E})$ is the Euclidean distance from the point x to the set $\mathscr{E}$. In the statistical literature, (6.3) is called the orthogonal regression method.

Note 6.1 (Orthogonal regression $\equiv$ TLS) The TLS method applied to the ellipsoid model (6.1) results in the following optimization problem:

$$\min_{\substack{A_{\mathrm{e}},c \\ \Delta x^{(1)},\ldots,\Delta x^{(N)}}} \sum_{i=1}^{N} \big\|\Delta x^{(i)}\big\|^2 \text{ subject to } \big(x^{(i)}+\Delta x^{(i)}-c\big)^{\top} A_{\mathrm{e}} \big(x^{(i)}+\Delta x^{(i)}-c\big) = 1 \quad \text{for } i=1,\ldots,N. \tag{6.4}$$

Clearly,

$$\operatorname{dist}(x,\mathscr{E}) = \arg\Big(\min_{\Delta x} \|\Delta x\|^2 \quad \text{subject to} \quad (x+\Delta x-c)^{\top} A_{\mathrm{e}} (x+\Delta x-c) = 1\Big)$$

and (6.4) is separable in $\Delta x^{(1)},\ldots,\Delta x^{(N)}$, so that the TLS problem (6.4) is equivalent to the orthogonal regression problem (6.3). In (6.3), the auxiliary variables $\Delta x^{(1)},\ldots,\Delta x^{(N)}$ are "hidden" in the dist function.

We assume that all data points are noisy measurements $x^{(i)} := \bar{x}^{(i)} + \tilde{x}^{(i)}$ of some true points $\bar{x}^{(1)},\ldots,\bar{x}^{(N)}$ that lie on a true ellipsoid $\mathscr{E}(\bar{A}_{\mathrm{e}},\bar{c})$; i.e., the model is a quadratic EIV model. The measurement errors $\tilde{x}^{(1)},\ldots,\tilde{x}^{(N)}$ are centered, independent, and identically distributed (i.i.d.), and the distribution is normal with variance-covariance matrix $\bar{\sigma}^2 I$, where $\bar{\sigma}^2$ is the noise variance.

Due to the quadratic nature of the ellipsoid model with respect to the measurement x, both the algebraic and the geometric fitting methods are inconsistent in a statistical sense, see [NS48] and the discussion in [Ful87, page 250]. We propose an ALS estimator that is consistent.

The LS estimator, defined by (6.2), is a nonlinear least squares problem. We use a computationally cheap, but suboptimal method to solve the optimization problem (6.2). The equation defining the ellipsoid model is "embedded" in the quadratic equation

$$x^{\top} A x + b^{\top} x + d = 0, \quad A = A^{\top} > 0, \tag{6.5}$$

which is linear in the parameters A, b, and d, so that a linear least squares estimation is possible. For given estimates $\hat{A}$, $\hat{b}$, and $\hat{d}$ of the parameters in (6.5), assuming that $\hat{A} = \hat{A}^{\top} > 0$, the estimates of the original parameters in (6.2) are given by

$$\hat{c} := -\frac{1}{2}\hat{A}^{-1}\hat{b} \qquad \text{and} \qquad \hat{A}_{\mathrm{e}} := \frac{1}{\hat{c}^{\top}\hat{A}\hat{c} - \hat{d}}\,\hat{A}. \tag{6.6}$$

The necessary computation for the (suboptimal) LS estimator involves finding an eigenvector associated with the smallest eigenvalue of a symmetric matrix. We use the same indirect approach to compute the ALS estimator.

The correction needed for the ALS estimator is given explicitly in terms of the noise variance $\bar{\sigma}^2$. We give an algorithm for ellipsoid fitting that implements the theoretical results. Its computational cost increases linearly with the sample size N. In [KMV04], we present the statistical properties of the estimator and treat the case when $\bar{\sigma}^2$ is unknown.

The orthogonal regression estimator, on the other hand, is computed via a local optimization method and scales worse with N and with the dimension n of the vector space. In addition, due to the nonconvexity of the cost function in (6.3), the computed solution depends on the supplied initial approximation. In degenerate cases (see [Nie02, pages 260–261]) the global minimum of (6.3) is not unique, so that there are several "best" fitting ellipses.

We point out several papers in which the ellipsoid fitting problem is considered. Gander, Golub, and Strebel [GGS94] consider algebraic and geometric fitting methods for circles and ellipses and note the inadequacy of the algebraic fit on some specific examples. Later on, the given examples are used as benchmarks for the algebraic fitting methods. Fitting methods, specific for ellipsoids, as opposed to the more general conic sections, are first proposed in [FPF99]. The methods incorporate the ellipticity constraint into the normalizing condition and thus give better results when an elliptic fit is desired. In [Nie01] a new algebraic fitting method is proposed that does not have as singularity the special case of a hyperplane fitting; if the best fitting manifold is affine the method coincides with the TLS method.

A statistical point of view on the ellipsoid fitting problem is taken in [Kan94]. Kanatani proposed an unbiased estimation method, called a renormalization procedure. He uses an adjustment similar to the one in the present chapter, but his approach of estimating the unknown noise variance is different from the one presented in [KMV04]. Moreover, the noise variance estimate proposed in [Kan94] is still inconsistent; the bias is removed up to the first order approximation.

6.2 Quadratic Errors-in-Variables Model

A second order surface in $\mathbb{R}^n$ is the set

$$\mathscr{B}(A, b, d) := \{\, x \in \mathbb{R}^n \mid x^\top A x + b^\top x + d = 0 \,\}, \tag{6.7}$$

where the symmetric matrix $A \in \mathbb{S}$, the vector $b \in \mathbb{R}^n$, and the scalar $d \in \mathbb{R}$ are parameters of the surface. If $A = 0$ and $b \neq 0$, then the surface (6.7) is a hyperplane, and if A is positive definite and $4d < b^\top A^{-1} b$, then (6.7) is an ellipsoid. Until Section 6.5, we will only assume that $\mathscr{B}(A, b, d)$ is a nonempty set, but in Section 6.5, we will come back to the ellipsoid fitting problem, so that the parameters will be restricted.

Let $\bar{A} \in \mathbb{S}$, $\bar{b} \in \mathbb{R}^n$, and $\bar{d} \in \mathbb{R}$ be such that the set $\mathscr{B}(\bar{A}, \bar{b}, \bar{d})$ is nonempty and let the points $\bar{x}^{(1)}, \ldots, \bar{x}^{(N)}$ lie on the surface $\mathscr{B}(\bar{A}, \bar{b}, \bar{d})$, i.e.,

$$\bar{x}^{(i)\top} \bar{A} \bar{x}^{(i)} + \bar{b}^\top \bar{x}^{(i)} + \bar{d} = 0, \quad \text{for } i = 1, \ldots, N. \tag{6.8}$$

The points $x^{(1)}, \ldots, x^{(N)}$ are measurements of the points $\bar{x}^{(1)}, \ldots, \bar{x}^{(N)}$, respectively, i.e.,

$$x^{(i)} = \bar{x}^{(i)} + \tilde{x}^{(i)}, \quad \text{for } i = 1, \ldots, N, \tag{6.9}$$

where $\tilde{x}^{(1)}, \ldots, \tilde{x}^{(N)}$ are the corresponding measurement errors. We assume that the measurement errors form an i.i.d. sequence and the distribution of $\tilde{x}^{(i)}$, for all $i = 1, \ldots, N$, is normal and zero mean, with variance-covariance matrix $\bar{\sigma}^2 I_n$, i.e.,

$$\mathbf{E}\ \tilde{x}^{(i_1)} \tilde{x}^{(i_2)\top} = 0, \quad \text{for all } i_1, i_2 = 1, \ldots, N, \quad i_1 \neq i_2,$$

and

$$\tilde{x}^{(i)} \sim \mathrm{N}(0, \bar{\sigma}^2 I_n), \quad \text{for } i = 1, \ldots, N,$$

where $\bar{\sigma}^2 > 0$ is called the noise variance.

The matrix $\bar{A}$ is the true value of the parameter A, while $\bar{b}$ and $\bar{d}$ are the true values of the parameters b and d, respectively. Without additional constraints imposed on the parameters, for a given second order surface $\mathscr{B}(A, b, d)$, the model parameters A, b, and d are not unique: $\mathscr{B}(\tau A, \tau b, \tau d)$ is the same surface for any real nonzero τ. This makes the quadratic model, parameterized by A, b, and d, nonidentifiable. To resolve the problem, we impose a normalizing condition; e.g., the true values of the parameters are assumed to satisfy the constraint

$$\|\bar{A}\|_{\mathrm{F}}^2 + \|\bar{b}\|^2 + \bar{d}^2 = 1. \tag{6.10}$$

Then the estimates are unique up to a sign.

Note 6.2 (Invariance of the LS and ALS estimators) As shown in [Boo79, page 59], [GGS94, page 564, equation (3.5)], and [Pra87, page 147], the constraint (6.10) is not invariant under Euclidean transformations. As a result, the LS estimator is not invariant under Euclidean transformations. Such a dependence on the coordinate system is undesirable. Suggestions for making the LS estimator invariant can be found in [Nie01].

The following question arises. Are the ALS estimators derived with the constraint (6.10) invariant? If the noise variance is *fixed*, the answer is negative. However, if we are allowed to modify the noise variance after the transformation of the data, then the ALS estimator can be made invariant.

A modification of the noise variance that ensures invariance under Euclidean transformations is the noise variance estimation procedure derived in [KMV04]. We demonstrate the invariance properties of the ALS estimator with estimated noise variance by a simulation example in Section 6.7. Rigorous analysis is presented in [SKMH05].

6.3 Ordinary Least Squares Estimation

The LS estimator for the second order surface model (6.7), subject to the normalizing condition (6.10), is defined as a global minimum point of the following optimization problem:

$$\min_{A,b,d} \sum_{i=1}^{N} \left(x^{(i)\top} A x^{(i)} + b^\top x^{(i)} + d \right)^2 \quad \text{subject to} \quad \begin{cases} A = A^\top, \\ \|A\|_{\mathrm{F}}^2 + \|b\|^2 + d^2 = 1. \end{cases} \tag{6.11}$$

The LS cost function is

$$Q_{\mathrm{ls}}(A, b, d) = \sum_{i=1}^{N} q_{\mathrm{ls}}(A, b, d; x^{(i)}),$$

where the elementary LS cost function

$$q_{\text{ls}}(A, b, d; x) = (x^\top A x + b^\top x + d)^2$$

measures the discrepancy of a single measurement point x from the surface $\mathscr{B}(A, b, d)$.

In order to derive the solution of (6.11), we introduce a parameter vector β containing all decision variables. Let $\text{vec}_\text{s} : \mathbb{S} \to \mathbb{R}^{(n+1)n/2}$ be an operator, a symmetric matrix vectorizing operator, that stacks the upper triangular part of A in a vector. The vector of decision variables is

$$\beta := \text{col}\left(\text{vec}_\text{s}(A), b, d\right), \tag{6.12}$$

an element of the parameter space $\mathbb{R}^{n_\beta}$, $n_\beta := (n+1)n/2 + n + 1$.

Define the symmetric Kronecker product $\otimes_\text{s}$ by

$$x^\top A x = (x \otimes_\text{s} x)^\top \text{vec}_\text{s}(A) \quad \text{for all } x \in \mathbb{R}^n \text{ and } A \in \mathbb{S}. \tag{6.13}$$

We have for the elementary LS cost function

$$\begin{aligned} q_{\text{ls}}(\beta; x) &= (x^\top A x + b^\top x + d)^2 \\ &= \left(\underbrace{\begin{bmatrix}(x \otimes_\text{s} x)^\top & x^\top & 1\end{bmatrix}}_{y^\top} \begin{bmatrix}\text{vec}_\text{s}(A) \\ b \\ d\end{bmatrix} \right)^2 \\ &= (y^\top \beta)^2 = \beta^\top y y^\top \beta \end{aligned} \tag{6.14}$$

and for the LS cost function

$$Q_{\text{ls}}(\beta) = \sum_{i=1}^{N} q_{\text{ls}}(\beta; x^{(i)}) = \sum_{i=1}^{N} \left((y^{(i)\top})\beta\right)^2 = \|Y\beta\|^2 = \beta^\top Y^\top Y \beta,$$

where

$$y^{(i)} := \begin{bmatrix} x^{(i)} \otimes_\text{s} x^{(i)} \\ x^{(i)} \\ 1 \end{bmatrix}, \quad \text{for } i = 1, \ldots, N, \qquad \text{and} \qquad Y := \begin{bmatrix} y^{(1)\top} \\ \vdots \\ y^{(N)\top} \end{bmatrix}.$$

Let $H \in \mathbb{R}^{n_\beta \times n_\beta}$ be a matrix, such that

$$\|H\beta\|^2 = \|A\|_\text{F}^2 + \|b\|^2 + d^2, \quad \text{for all } A \in \mathbb{S},\ b \in \mathbb{R}^n,\ d \in \mathbb{R}, \tag{6.15}$$

where β is defined in (6.12).

The LS estimation problem (6.11) is equivalent to the following classical quadratically constrained least squares problem:

$$\min_\beta \|Y\beta\|^2 \quad \text{subject to} \quad \|H\beta\|^2 = 1. \tag{6.16}$$

The LS estimator $\hat{\beta}_\text{ls}$ is $H^{-1} v_{\min}$, where $v_{\min}$ is a normalized eigenvector of $H^{-T} Y^\top Y H^{-1}$, corresponding to the smallest eigenvalue.

In order to avoid the computation of the Gram matrix $Y^\top Y$, one can obtain the solution from the SVD of YH^{-1}. Let

$$YH^{-1} = U\Sigma V^\top, \quad \text{with} \quad U^\top U = I, \quad V^\top V = I, \quad \text{and} \\ \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n), \quad \sigma_1 \geq \cdots \geq \sigma_n \geq 0. \quad (6.17)$$

Then $\hat{\beta}_{\text{ls}}$ is $H^{-1}v_{\min}$, where $v_{\min}$ is the last column of the matrix V.

Note 6.3 The matrix H that ensures (6.15) is a diagonal matrix with diagonal elements equal to 1 or $\sqrt{2}$, where the latter correspond to the off-diagonal elements of A; see Note 6.5. Since the normalizing condition (6.10) is arbitrary, however, we can choose any nonsingular matrix H in (6.16). Particularly simple is $H = I$. The LS and ALS estimators depend on the normalizing condition, but the ALS estimator is consistent for any nondegenerate normalizing condition, i.e., for any full-rank matrix H.

Note that $\operatorname{vec}_{\text{s}}(xx^\top) \neq x \otimes_{\text{s}} x$. One can verify that $x \otimes_{\text{s}} x = D \ \operatorname{vec}_{\text{s}}(xx^\top)$, where D is a diagonal matrix with diagonal elements equal to 1 or 2; the latter corresponds to the off-diagonal elements of $xx^\top$ appearing in the product $D \ \operatorname{vec}_{\text{s}}(xx^\top)$; see Note 6.5.

6.4 Adjusted Least Squares Estimation

The LS estimator is readily computable but it is inconsistent. We propose an adjustment procedure that defines a consistent estimator. The proposed approach is due to [KZ02] and is related to the method of corrected score functions; see [CRS95, Section 6.5].

The ALS estimator $\hat{\beta}_{\text{als}}$ is defined as a global minimum point of the following optimization problem:

$$\min_{\beta} Q_{\text{als}}(\beta) \quad \text{subject to} \quad \|H\beta\|^2 = 1,$$

where the ALS cost function Q_{als} is

$$Q_{\text{als}}(\beta) = \sum_{i=1}^{N} q_{\text{als}}(\beta; x^{(i)}) \quad \text{for all } \beta \in \mathbb{R}^{n_\beta}.$$

Let $x = \bar{x} + \tilde{x}$, where $\tilde{x}$ is normally distributed with zero mean and variance-covariance matrix $\bar{\sigma}^2 I$. The elementary ALS cost function q_{als} is defined by the identity

$$\mathbf{E}\, q_{\text{als}}(\beta, \bar{x} + \tilde{x}) = q_{\text{ls}}(\beta, \bar{x}), \quad \text{for all } \beta \in \mathbb{R}^{n_\beta} \text{ and } \bar{x} \in \mathbb{R}^n. \quad (6.18)$$

We motivate the definition of the ALS cost function as follows:

$$\overline{Q}_{\text{ls}}(\beta) := \sum_{i=1}^{N} q_{\text{ls}}(\beta; \bar{x}^{(i)}), \quad \text{for all } \beta \in \mathbb{R}^{n_\beta},$$

has as a global minimum point the true value of the parameter vector

$$\bar{\beta} := \operatorname{col}\big(\operatorname{vec}_{\text{s}}(\bar{A}), \bar{b}, \bar{d}\big).$$

Indeed, $\overline{Q}_{\text{ls}} \geq 0$ and by definition $\overline{Q}_{\text{ls}}(\bar{\beta}) = 0$. From

$$\mathbf{E}\, Q_{\text{als}} = \overline{Q}_{\text{ls}},$$

we see that, as the sample size grows, Q_{als} approximates $\overline{Q}_{\text{ls}}$. Provided that $\overline{Q}_{\text{ls}}$ has $\bar{\beta}$ as a unique global minimum (the contrast condition of [KMV04]), the ALS estimator is strongly consistent.

Next, we derive an explicit expression for the ALS cost function Q_{als}. From (6.18) and (6.14), we have

$$\mathbf{E}\, q_{\text{als}}(\beta, x) = q_{\text{ls}}(\beta, \bar{x}) = \beta^\top \bar{y}\bar{y}^\top \beta =: \beta^\top \psi_{\text{ls}}(\bar{x})\beta,$$

where

$$\bar{y} := \operatorname{col}\big((\bar{x} \otimes_{\text{s}} \bar{x}), \bar{x}, 1\big) \qquad \text{and} \qquad \psi_{\text{ls}}(\bar{x}) := \bar{y}\bar{y}^\top.$$

Thus the ALS elementary cost function q_{als} is quadratic in β,

$$q_{\text{als}}(\beta; x) = \beta^\top \psi_{\text{als}}(x)\beta,$$

where

$$\mathbf{E}\, \psi_{\text{als}}(x) = \psi_{\text{ls}}(\bar{x}). \tag{6.19}$$

Under the normality assumption for the noise term $\tilde{x}$, (6.19) yields the following convolution equation:

$$\Big(\frac{1}{2\pi\bar{\sigma}^2}\Big)^{n/2} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \psi_{\text{als}}(\bar{x} + \tilde{x}) \prod_{i=1}^{n} \exp\Big(-\frac{\tilde{x}_i^2}{2\bar{\sigma}^2}\Big)\, \mathrm{d}\tilde{x}_1 \cdots \mathrm{d}\tilde{x}_n = \psi_{\text{ls}}(\bar{x}).$$

Solving for the unknown ψ_{als} is a deconvolution problem.

The deconvolution problem can be solved independently for the entries of ψ_{als}. The elements of the matrix $\psi_{\text{ls}}(\bar{x})$ are monomials of at most fourth order in $\bar{x}$. Consider the generic term

$$\eta_{\text{ls}}(\bar{x}) = \bar{x}_i \bar{x}_j \bar{x}_p \bar{x}_q, \quad \text{where } i, j, p, q \in \{\, 0, 1, \ldots, n \,\}.$$

We formally set $\bar{x}_0 = 1$ and allow any of the indices to be zero, in order to allow η_{ls} to be of order less than four.

Let $r(s)$, $s = 1, \ldots, n$, denote the number of repetitions of the index s in the monomial $\bar{x}_i \bar{x}_j \bar{x}_p \bar{x}_q$. For example, let $n = 2$. In the monomial $\bar{x}_1 \bar{x}_2^3$, $r(1) = 1$ and $r(2) = 3$, and in the monomial $\bar{x}_1^4$, $r(1) = 4$ and $r(2) = 0$.

The functions

$$\begin{gathered} t_0(\xi) := 1, \qquad t_1(\xi) := \xi, \qquad t_2(\xi) := \xi^2 - \bar{\sigma}^2, \\ t_3(\xi) := \xi^3 - 3\xi\bar{\sigma}^2, \qquad \text{and} \qquad t_4(\xi) := \xi^4 - 6\xi^2\bar{\sigma}^2 + 3\bar{\sigma}^4 \end{gathered} \tag{6.20}$$

have the property

$$\mathbf{E}\, t_k(\bar{x}_s + \tilde{x}_s) = \bar{x}_s^k, \quad \text{for all } \bar{x}_s \in \mathbb{R} \text{ and } k = 0, 1, 2, 3, 4,$$

where $\tilde{x}_s \sim \mathrm{N}(0, \bar{\sigma}^2)$. Thus the polynomial

$$\eta_{\text{als}}(x) := \prod_{s=1}^{n} t_{r(s)}(x_s) \tag{6.21}$$

has the property

$$\mathbf{E}\,\eta_{\text{als}}(x) = \bar{x}_i\bar{x}_j\bar{x}_p\bar{x}_q = \eta_{\text{ls}}(\bar{x}) \quad \text{for all } \bar{x} \in \mathbb{R}^n.$$

This shows that η_{als} is the desired solution. The matrix ψ_{als} is constructed element-wise in the described way.

The ALS cost function Q_{als} is quadratic in β,

$$Q_{\text{als}}(\beta) = \beta^\top \Psi_{\text{als}}\beta, \quad \text{for all } \beta \in \mathbb{R}^{n_\beta},$$

where

$$\Psi_{\text{als}} = \sum_{i=1}^{N} \psi_{\text{als}}(x^{(i)}).$$

Thus the function Q_{als} is described thoroughly.

Example 6.4 (The matrix ψ_{als} for $n = 2$) The model parameters are $A = \left[\begin{smallmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{smallmatrix}\right]$, $b = \left[\begin{smallmatrix} b_1 \\ b_2 \end{smallmatrix}\right]$, and the scalar d. The parameter space is 6-dimensional with

$$\beta := \operatorname{col}\big(\operatorname{vec_s}(A), b, d\big) = \begin{bmatrix} a_{11} & a_{12} & a_{22} & b_1 & b_2 & d \end{bmatrix}^\top.$$

From (6.13), we have

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top \otimes_{\text{s}} \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top = \begin{bmatrix} x_1x_1 & 2x_1x_2 & x_2x_2 \end{bmatrix}^\top,$$

so that

$$y := \operatorname{col}\big((x \otimes_{\text{s}} x), x, 1\big) = \begin{bmatrix} x_1x_1 & 2x_1x_2 & x_2x_2 & x_1 & x_2 & 1 \end{bmatrix}^\top,$$

$$\psi_{\text{ls}}(x) = yy^\top = \begin{bmatrix} x_1^4 & 2x_1^3x_2 & x_1^2x_2^2 & x_1^3 & x_1^2x_2 & x_1^2 \\ * & 4x_1^2x_2^2 & 2x_1x_2^3 & 2x_1^2x_2 & 2x_1x_2^2 & 2x_1x_2 \\ * & * & x_2^4 & x_1x_2^2 & x_2^3 & x_2^2 \\ * & * & * & x_1^2 & x_1x_2 & x_1 \\ * & * & * & * & x_2^2 & x_2 \\ * & * & * & * & * & 1 \end{bmatrix},$$

with $*$'s indicating the symmetric elements.

The adjusted matrix ψ_{als} is $\psi_{\text{als}} = \psi_{\text{ls}} + \Delta\psi_{\text{als}}$, where the correction $\Delta\psi_{\text{als}}$ is

$$\begin{bmatrix} 3\bar{\sigma}^4 - 6\bar{\sigma}^2x_1^2 & -6\bar{\sigma}^2x_1x_2 & \Delta\psi_{\text{als},13} & -3\bar{\sigma}^2x_1 & -\bar{\sigma}^2x_2 & -\bar{\sigma}^2 \\ * & \Delta\psi_{\text{als},22} & -6\bar{\sigma}^2x_1x_2 & -2\bar{\sigma}^2x_2 & -2\bar{\sigma}^2x_1 & 0 \\ * & * & 3\bar{\sigma}^4 - 6\bar{\sigma}^2x_2^2 & -\bar{\sigma}^2x_1 & -3\bar{\sigma}^2x_2 & -\bar{\sigma}^2 \\ * & * & * & -\bar{\sigma}^2 & 0 & 0 \\ * & * & * & * & -\bar{\sigma}^2 & 0 \\ * & * & * & * & * & 0 \end{bmatrix},$$

$$\Delta\psi_{\text{als},13} = \bar{\sigma}^4 - \bar{\sigma}^2(x_1^2 + x_2^2), \quad \text{and} \quad \Delta\psi_{\text{als},22} = 4\bar{\sigma}^4 - 4\bar{\sigma}^2(x_1^2 + x_2^2).$$

The correction matrix $\Delta\psi_{\text{als}}$, without the fourth order terms in $\bar{\sigma}$, is derived in [Zha97, Section 7]. The derivation in [Zha97], however, applies only for the two-dimensional case. ∎

The recommended way of computing the LS estimator is via the SVD of YH^{-1}. For the ALS estimator we use the less accurate eigenvalue decomposition because the correction is derived for $\Psi_{\text{ls}} = Y^\top Y$ and cannot be determined for the factor Y.

6.5 Ellipsoid Estimation

The ALS estimator β_{als} is derived for the general quadratic EIV model (6.8)–(6.9). Now we specialize it for the ellipsoid fitting problem; i.e., we assume that the true surface belongs to the class of surfaces

$$\mathscr{B}(A_{\text{e}}, c) = \{\, x \in \mathbb{R}^n : (x - c)^\top A_{\text{e}}(x - c) = 1 \,\} \tag{6.22}$$

for some true values $\bar{A}_{\text{e}} \in \mathbb{S}$, $\bar{A}_{\text{e}} = \bar{A}_{\text{e}}^\top > 0$, and $\bar{c}$ of the parameters A_{e} and c. The equation defining $\mathscr{B}(\bar{A}_{\text{e}}, \bar{c})$ can be written as

$$x^\top \bar{A}_{\text{e}} x - 2(\bar{A}_{\text{e}}\bar{c})^\top x + \bar{c}^\top \bar{A}_{\text{e}}\bar{c} - 1 = 0,$$

or, with $\theta := \left(\|\bar{A}_{\text{e}}\|_{\text{F}}^2 + \|2\bar{A}_{\text{e}}\bar{c}\|^2 + (\bar{c}^\top \bar{A}_{\text{e}}\bar{c} - 1)^2\right)^{1/2}$,

$$x^\top (\bar{A}_{\text{e}}/\theta) x - 2(\bar{A}_{\text{e}}\bar{c}/\theta)^\top x + (\bar{c}^\top \bar{A}_{\text{e}}\bar{c} - 1)/\theta = 0.$$

Introduce the new parameters

$$\bar{A} := \frac{\bar{A}_{\text{e}}}{\theta}, \qquad \bar{b} := -2\frac{\bar{A}_{\text{e}} c}{\theta}, \quad \text{and} \quad \bar{d} := \frac{\bar{c}^\top \bar{A}_{\text{e}}\bar{c} - 1}{\theta}.$$

As defined, $\bar{A}$, $\bar{b}$, and $\bar{d}$ satisfy the normalizing condition (6.10).

We can go back to the original parameters $\bar{A}_{\text{e}}$ and $\bar{c}$ from $\bar{A}$, $\bar{b}$, and $\bar{d}$ that satisfy (6.10) by

$$\bar{c} = -\frac{1}{2}\bar{A}^{-1}\bar{b} \qquad \text{and} \qquad \bar{A}_{\text{e}} = \frac{1}{\bar{c}^\top \bar{A}\bar{c} - \bar{d}}\,\bar{A}. \tag{6.23}$$

Note that $\theta = \bar{c}^\top \bar{A}\bar{c} - \bar{d}$ is nonzero. Let $\hat{A}$, $\hat{b}$, $\hat{d}$ be the ALS estimator of the parameters $\bar{A}$, $\bar{b}$, $\bar{d}$. The estimator of the parameters $\bar{A}_{\text{e}}$ and $\bar{c}$ is given by the transformation (6.6).

If the obtained estimate $\hat{A}_{\text{e}}$ is indefinite, we impose a posteriori positive definiteness by the projection

$$\hat{A}_{\text{e},2} := \textstyle\sum_{l:\hat{\lambda}_l > 0} \hat{\lambda}_l \hat{v}_l \hat{v}_l^\top, \tag{6.24}$$

where $\hat{A}_{\text{e}} = \sum_{l=1}^n \hat{\lambda}_l \hat{v}_l \hat{v}_l^\top$ is the eigenvalue decomposition of $\hat{A}_{\text{e}}$. Indefinite estimate $\hat{A}_{\text{e}}$ can be obtained because the estimator does not enforce the prior knowledge $\hat{A}_{\text{e}} = \hat{A}_{\text{e}}^\top > 0$. Clearly, the two-stage procedure—$\hat{A}_{\text{e}}$ obtained on the first stage and $\hat{A}_{\text{e},2}$ on the second stage—is suboptimal. Empirical results, however, suggest that the event of having the constraint $\hat{A}_{\text{e}} > 0$ active is rare. Typically, it occurs for a small sample size with nonuniform data point distribution and for data with outliers. Due to $\bar{A}_{\text{e}} = \hat{A}_{\text{e}}^\top > 0$ and the consistency of the estimator $\hat{A}_{\text{e}}$, we expect that for large sample size, $\hat{A}_{\text{e}} > 0$.

6.6 Algorithm for Adjusted Least Squares Estimation*

In this section, we summarize the estimation procedure described above by giving Algorithm 6.1 for its computation. Notation similar to the MATLAB syntax for indexing the elements of a matrix is used. For example, $A(i_1{:}i_2, j_1{:}j_2)$ stands for the submatrix of A obtained by selecting the elements with first index in the set $\{i_1, i_1+1, \ldots, i_2\}$ and with second index in the set $\{j_1, j_1+1, \ldots, j_2\}$.

Note 6.5 If a general quadratic model is estimated, the normalizing condition is given as prior knowledge, see Note 6.3. If an ellipsoid is estimated, however, the normalizing condition is arbitrary. In Algorithm 6.1, we set $H = I$, which corresponds to a normalizing condition

$$\|\operatorname{vec}_{\mathrm{s}}(A)\|^2 + \|b\|^2 + d^2 = 1.$$

The matrix H corresponding to the normalizing condition (6.10) is

$$H = \begin{bmatrix} \sqrt{D} & & \\ & I_n & \\ & & 1 \end{bmatrix},$$

where D is a diagonal matrix with diagonal elements

$$D_{ii} = \begin{cases} 2 & \text{if } i \in \mathscr{I}, \\ 1 & \text{otherwise.} \end{cases}$$

Note 6.6 (Known blocks of the matrix Ψ_{als}) Algorithm 6.1 can be improved by setting certain elements of Ψ_{als} in advance and not by following the general adjustment procedure. Consider a block partitioning of the matrices ψ_{ls}, ψ_{als}, and Ψ_{als} according to the partitioning of the vector

$$\left[(x \otimes_{\mathrm{s}} x)^\top \quad | \quad x^\top \quad | \quad 1\right]^\top;$$

e.g., for ψ_{ls}, denote

$$\psi_{\mathrm{ls}} =: \begin{bmatrix} \psi_{\mathrm{ls},11} & \psi_{\mathrm{ls},12} & \psi_{\mathrm{ls},13} \\ * & \psi_{\mathrm{ls},22} & \psi_{\mathrm{ls},23} \\ * & * & \psi_{\mathrm{ls},33} \end{bmatrix}.$$

All elements of ψ_{ls} are monomials in x; moreover all elements of:

- $\psi_{\mathrm{ls},11}(x)$ are of fourth order,
- $\psi_{\mathrm{ls},12}(x)$ are of third order,
- $\psi_{\mathrm{ls},13}(x)$ and $\psi_{\mathrm{ls},22}(x)$ are of second order,
- $\psi_{\mathrm{ls},23}(x)$ are of first order, and
- the scalar $\psi_{\mathrm{ls},33}(x) = 1$ is independent of x.

Algorithm 6.1 ALS ellipsoid fitting `als_fit`

Input: a matrix $X := \begin{bmatrix} x^{(1)} & \cdots & x^{(N)} \end{bmatrix} \in \mathbb{R}^{n \times N}$ and the noise variance $\bar{\sigma}^2$.

1: Form the tensor $T \in \mathbb{R}^{5 \times n \times N}$, $T(k,l,i) := t_k\big(X(l,i)\big)$, for $k = 0, \ldots, 4$, $l = 1, \ldots, n$, and $i = 1, \ldots, N$, where the functions t_k, $k = 0, 1, 2, 3, 4$, are given in (6.20).

2: Define the vectors $\mathbf{1}, \mathbf{l} \in \mathbb{R}^{n+1}$ by $\mathbf{1} := \mathrm{col}(1, \ldots, 1, 1)$, $\mathbf{l} := \mathrm{col}(1, \ldots, n, 0)$, and form the matrix $M \in \mathbb{R}^{n_\beta \times 2}$, $n_\beta := (n+1)n/2 + n + 1$, $M := \begin{bmatrix} \mathrm{vec_s}(\mathbf{l}\mathbf{1}^\top) & \mathrm{vec_s}(\mathbf{1}\mathbf{l}^\top) \end{bmatrix}$. We use M to find the indices of $\bar{x}$ in the entries of $\psi_{\mathrm{ls}}(\bar{x})$. Note that $\big(M(p,1), M(p,2)\big)$ are the indices of $\bar{x}$ in the pth entry of $\bar{y} := \bar{x} \otimes_{\mathrm{s}} \bar{x}$. Recall that $\psi_{\mathrm{ls}}(\bar{x}) := \bar{y}\bar{y}^\top$. Thus the indices of $\bar{x}$ in the (p,q)th entry of $\psi_{\mathrm{ls}}(\bar{x})$ are $\big(M(p,1), M(p,2), M(q,1), M(q,2)\big)$.

3: Define a binary operator $==$ by $(l_1 == l_2) := 1$ if $l_1 = l_2$ and 0, otherwise, for all $l_1, l_2 \in \mathbb{R}$. Form the tensor $R \in \mathbb{R}^{n_\beta \times n_\beta \times n}$,

$$R(p,q,l) = \big(M(p,1) == l\big) + \big(M(p,2) == l\big) + \big(M(q,1) == l\big) + \big(M(q,2) == l\big),$$

for all $q \geq p$ and $l = 1, \ldots, n$, which contains the number of repetitions of an index l in an entry (p,q)th of $\psi_{\mathrm{ls}}(\bar{x})$. In terms of the function r, defined in Section 6.4, R stores $\big(r(1), \ldots, r(n)\big)$ for the entries of $\psi_{\mathrm{ls}}(\bar{x})$.

4: Compute

$$\eta_{\mathrm{als}}(p,q) = \sum_{i=1}^{N} \prod_{l=1}^{n} T\big(R(p,q,l), l, i\big) \quad \text{for all } q \geq p.$$

This step corresponds to the correction (6.21) from Section 6.4.

5: Form the set $\mathscr{I}$ of the indices of the vector $\mathrm{vec_s}(A)$, corresponding to the off-diagonal elements of A, $\mathscr{I} = \{\, 1, \ldots, (n+1)n/2 \,\} - \{\, l(l+1)/2 : l = 1, \ldots, n \,\}$. ($\mathscr{I}_1 - \mathscr{I}_2$ denotes the set difference of the sets $\mathscr{I}_1$ and $\mathscr{I}_2$.) Note that $\{\, l(l+1)/2 \mid l = 1, \ldots, n \,\}$ are the indices of $\mathrm{vec_s}(A)$, corresponding to the diagonal elements of A.

6: Form the symmetric matrix Ψ_{als} by

$$\Psi_{\mathrm{als}}(p,q) := \begin{cases} 4\eta_{\mathrm{als}}(p,q) & \text{if } p \in \mathscr{I} \text{ and } q \in \mathscr{I}, \\ 1\eta_{\mathrm{als}}(p,q) & \text{if } p \notin \mathscr{I} \text{ and } q \notin \mathscr{I}, \\ 2\eta_{\mathrm{als}}(p,q) & \text{otherwise,} \end{cases}$$

for all $q \geq p$, and $\Psi_{\mathrm{als}}(p,q) := \Psi_{\mathrm{als}}(q,p)$, for all $q < p$.

7: Find an eigenvector $\hat{\beta}_{\mathrm{als}}$ associated with the smallest eigenvalue of Ψ_{als}.

8: Normalize $\hat{\beta}_{\mathrm{als}}$, $\hat{\beta}_{\mathrm{als}} := \hat{\beta}_{\mathrm{als}} / \|\hat{\beta}_{\mathrm{als}}\|$.

9: Reconstruct the estimates $\hat{A}$, $\hat{b}$, and $\hat{d}$ from the vector $\hat{\beta}_{\mathrm{als}}$,

$$\hat{A} := \mathrm{vec_s}^{-1}\big(\hat{\beta}_{\mathrm{als}}(1 : n(n+1)/2)\big),\ \hat{b} := \hat{\beta}_{\mathrm{als}}\big(n(n+1)/2 + 1 : n_\beta - 1\big),\ \hat{d} := \hat{\beta}_{\mathrm{als}}(n_\beta),$$

where $\mathrm{vec_s}^{-1} : \mathbb{R}^{n(n+1)/2} \to \mathbb{S}$, forms a symmetric matrix out of the vector of the elements in its upper triangular part.

10: Obtain the estimates of the ellipsoid parameters $\bar{A}_{\mathrm{e}}$ and $\bar{c}$ by (6.6).

11: If $\hat{A}_{\mathrm{e}} \leq 0$, project $\hat{A}$ on the positive definite cone by (6.24).

Output: the estimates $\hat{A}_{\mathrm{e}}$, $\hat{c}$ of the ellipsoid parameters.

For the blocks of order zero and one, there is no correction applied in the formation of the matrix ψ_{als}. The correction for the elements of the blocks of order two is $-\bar{\sigma}^2 I_n$. Thus for the corresponding blocks of ψ_{als}, we have

$$\psi_{\text{als},22}(x) = xx^\top - \bar{\sigma}^2 I_n, \qquad \psi_{\text{als},23}(x) = x,$$
$$\psi_{\text{als},13}(x) = x \otimes_{\text{s}} x - \text{vec}_{\text{s}}(\bar{\sigma}^2 I_n), \qquad \psi_{\text{als},33}(x) = 1.$$

Finally, the corresponding blocks of Ψ_{als} are

$$\Psi_{\text{als},22} = \sum_{i=1}^{N} x^{(i)} x^{(i)\top} - N\bar{\sigma}^2 I_n, \qquad \Psi_{\text{als},23} = \sum_{i=1}^{N} x^{(i)},$$
$$\Psi_{\text{als},13} = \sum_{i=1}^{N} x^{(i)} \otimes_{\text{s}} x^{(i)} - \text{vec}_{\text{s}}(N\bar{\sigma}^2 I_n), \qquad \Psi_{\text{als},33} = N,$$

and only the upper triangular part of the block $\Psi_{\text{als},11}$ and the block $\Psi_{\text{als},12}$ need to be computed in steps 4 and 5 of Algorithm 6.1.

6.7 Simulation Examples

We show the ALS, LS, and orthogonal regression (OR) estimates for a test example from [GGS94], called "special data". It is designed to illustrate the inadequacy of the algebraic fitting method and to show the advantage of the OR method.

Only data points are given; even if they are generated with a true model, we do not know it. For this reason the comparison is visual. Since the noise variance needed for the ALS estimator is unknown, we estimate it via the procedure proposed in [KMV04].

Figure 6.1 shows the data points with the estimated ellipses superimposed on them. The OR estimator is computed by a general purpose optimization algorithm (MATLAB function `fmincon`). The cost function is evaluated as explained in [Zha97, Section 5.2].

For the first test example (see Figure 6.1, left) the OR estimator is influenced by the initial approximation. Using the LS estimate as initial approximation, the optimization algorithm converges to a local minimum. The resulting estimate is the dashed-dotted ellipse closer to the LS estimate. Using the ALS estimate as initial approximation, the obtained estimate is the dashed-dotted ellipse closer to the ALS estimate. Next, we will consider the better of the two OR estimates.

Although the sample size is only $N = 8$ data points, the ALS estimator gives good estimates that are comparable with the OR estimate. The value of the OR cost function (see (6.3)) is 3.2531 for the LS estimator, 1.6284 for the ALS estimator, and 1.3733 for the OR estimate. The ALS estimator is less than 19% suboptimal. Moreover, the volume of the OR estimate is 62.09 square units, while the volume of the ALS estimate is 34.37 square units, which is nearly twice as small. Visually (as well as in other senses), "smaller" estimates are preferable.

In a second example, taken from [Spä97], the ALS estimate is close to the OR estimate; see Figure 6.1, right. In terms of the OR cost function, the ALS estimate is less than 25% suboptimal. The volume of the ALS estimate is comparable to that of the OR estimate.

Figure 6.2 illustrates the invariance properties of the ALS estimator with estimated noise variance. The data used is again the "special data" from [GGS94]. The figure shows translated, rotated, scaled, and translated and rotated data points with the corresponding ALS estimates.

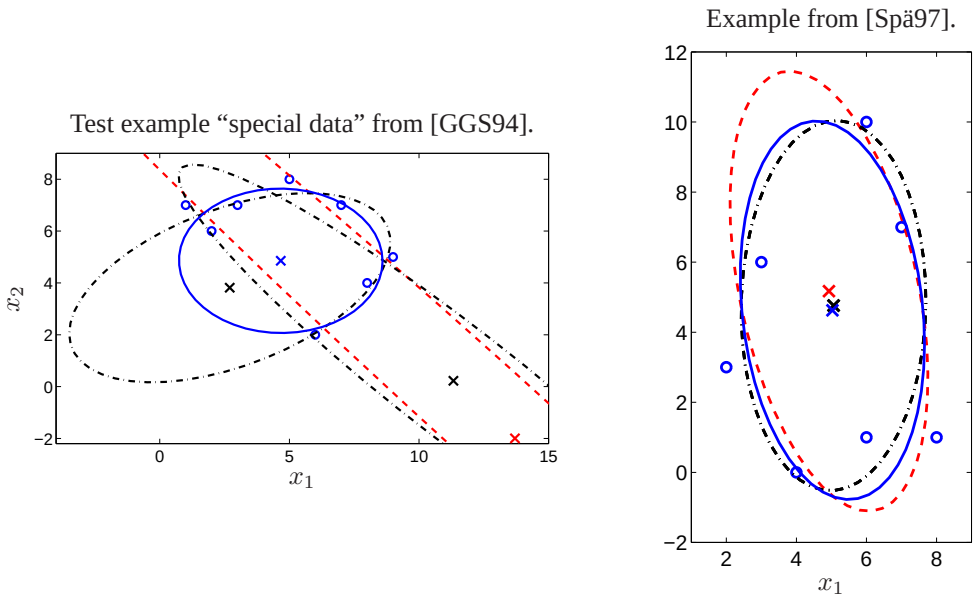


Figure 6.1. *Test examples. dashed—LS, dashed-dotted—OR, solid—ALS, ○—data points, ×—centers of the estimated ellipses.*

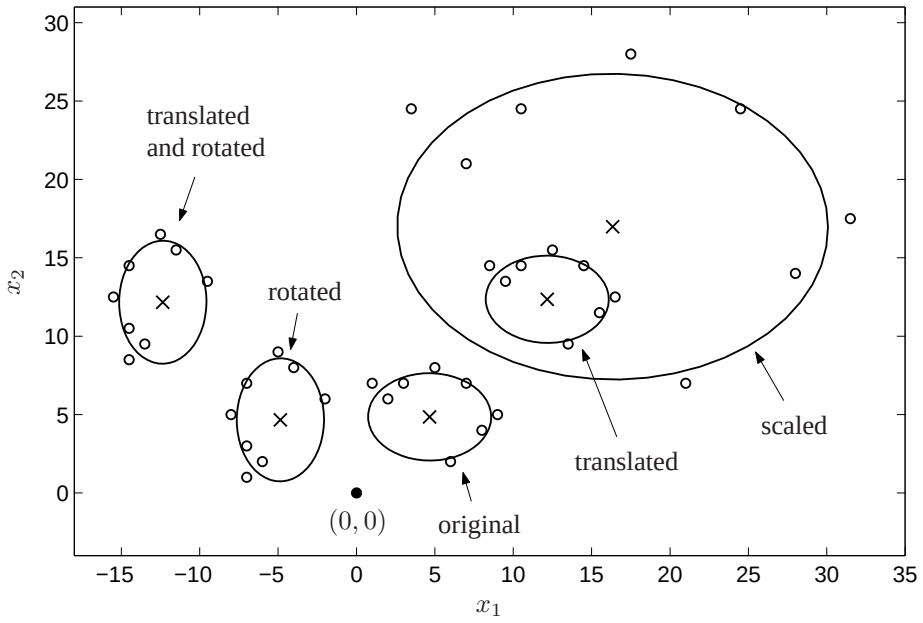


Figure 6.2. *ALS estimates of the original, translated, rotated, scaled, and translated and rotated data points. ○—data points, ×—centers of the estimated ellipses, •—point* $(0,0)$.

6.8 Conclusions

The LS estimation of the ellipsoid parameters from noisy measurements of points on its boundary is a nonlinear least squares problem. An indirect, suboptimal approach was used that transforms the ellipsoid model to a general quadratic model and applies linear least squares estimation. Due to the measurement errors, however, the LS estimator is inconsistent.

Assuming that the measurement errors are normally distributed, a correction is derived that uses the true measurement error variance and adjusts the LS cost function, so that the resulting ALS estimator is consistent. An algorithm for the necessary computation is outlined.

The ALS estimator is illustrated via simulation examples. Compared to the orthogonal regression estimator, it has the advantage of being cheaper to compute and independent of initial approximation. The computational efficiency is crucial for higher dimensional ellipsoid fitting and for problems with large sample size.

Part II

Dynamic Problems

Chapter 7

Introduction to Dynamical Models

With this chapter, we start to consider modeling problems for linear time-invariant (LTI) systems. First, we give an introduction to the LTI model class using the behavioral language. As in the static case, a key question is the representation of the model, i.e., how it is described by equations. Again, the kernel, image, and input/output representations play an important role, but other representations that bring additional structure into evidence are used as well.

Dynamical systems are much richer in properties than static systems. In the dynamic case, the memory of the system is central, i.e., the fact that the past can affect the future. The intuitive notion of memory is formalized in the definition of state. In addition, a key role is played by the controllability property of the system. Every linear static system has an image representation. In the dynamic case this is no longer true. A necessary and sufficient condition for existence of an image representation is controllability.

7.1 Linear Time-Invariant Systems

Dynamical systems describe variables that are functions of one independent variable, referred to as "time". In Chapter 2, a system was defined as a subset $\mathscr{B}$ of a universum set $\mathscr{U}$. In the context of dynamical systems, $\mathscr{U}$ is a set of functions $w : \mathbb{T} \to \mathbb{W}$, denoted by $\mathbb{W}^{\mathbb{T}}$. The sets $\mathbb{W}$ and $\mathbb{T} \subseteq \mathbb{R}$ are called, respectively, signal space and time axis. The signal space is the set where the system variables take on their values and the time axis is the set where the time variable takes on its values. We use the following definition of a dynamical system [Wil86a].

> A dynamical system Σ is a 3-tuple $\Sigma = (\mathbb{T}, \mathbb{W}, \mathscr{B})$, with $\mathbb{T} \subseteq \mathbb{R}$ the time axis, $\mathbb{W}$ the signal space, and $\mathscr{B} \subseteq \mathbb{W}^{\mathbb{T}}$ the behavior.

The behavior $\mathscr{B} \subseteq \mathbb{W}^{\mathbb{T}}$ is the set of all legitimate functions, according to the system Σ, from the universum set $\mathscr{U} = \mathbb{W}^{\mathbb{T}}$. When the time axis and the signal space are understood from the context, as is often the case, we may identify the system $\Sigma = (\mathbb{T}, \mathbb{W}, \mathscr{B})$ with its behavior $\mathscr{B}$. As with any set, the behavior can be described in a number of ways. In the context of dynamical systems, most often used are representations by equations

$f : \mathbb{W}^{\mathbb{T}} \to \mathbb{R}^g$, i.e., $\mathscr{B} = \{ w \in \mathbb{W}^{\mathbb{T}} \mid f(w) = 0 \}$. The equations $f(w) = 0$ are called annihilating behavioral equations.

Of interest are systems with special properties. In the behavioral setting,

> a property of the system Σ is always defined in terms of the behavior and then translated to equivalent statements in terms of particular representations.

Similarly, the statement that w is a trajectory of Σ, i.e., $w \in \mathscr{B}$, is translated to more convenient characterizations for numerical verification in terms of representations of Σ.

Note 7.1 (Classical vs. behavioral theory) In the classical theory, system properties are often defined on the representation level; i.e., a property of the system is defined as a property of a particular representation. (Think, for example, of controllability, which is defined as a property of a state space representation.) This has the drawback that such a definition might be representation dependent and therefore not a genuine property of the system itself. (For example, a controllable system (see Section 7.5) for definition, may have uncontrollable state representation.)

It is more natural to work instead the other way around.

1 Define the property in terms of the behavior $\mathscr{B}$.

2 Find the implications of that property on the parameters of the system in particular representations. On this level, algorithms for verification of the property are derived.

The way of developing system theory as a sequence of steps 1 and 2 is characteristic for the behavioral approach.

A static system $(\mathscr{U}, \mathscr{B})$ is *linear* when the universum set $\mathscr{U}$ is a vector space and the behavior $\mathscr{B}$ is a linear subspace. Analogously, a dynamical system $\Sigma = (\mathbb{T}, \mathbb{W}, \mathscr{B})$ is linear when the signal space $\mathbb{W}$ is a vector space and $\mathscr{B}$ is a linear subspace of $\mathbb{W}^{\mathbb{T}}$ (viewed as a vector space in the natural way).

The universum set $\mathbb{W}^{\mathbb{T}}$ of a dynamical system has special structure that is not present in the static case. For this reason dynamical systems are richer in properties than static systems. Next, we restrict ourselves to the case when the time axis is either $\mathbb{T} = \mathbb{N}$ or $\mathbb{T} = \mathbb{Z}$ and define two properties—time-invariance and completeness. In keeping with tradition, we call a function $w \in \mathbb{W}^{\mathbb{T}}$ a time series.

A system $\Sigma = (\mathbb{N}, \mathbb{W}, \mathscr{B})$ is *time-invariant* if $\mathscr{B} \subseteq \sigma\mathscr{B}$, where σ is the backward shift operator $(\sigma w)(t) := w(t+1)$ and $\sigma\mathscr{B} := \{ \sigma w \mid w \in \mathscr{B} \}$. In the case $\mathbb{T} = \mathbb{Z}$, a system $\Sigma = (\mathbb{Z}, \mathbb{W}, \mathscr{B})$ is time-invariant if $\mathscr{B} = \sigma\mathscr{B}$. Time-invariance requires that if a time series w is a trajectory of a time-invariant system, then all its backward shifts $\sigma^t w$, $t > 0$, are also trajectories of that system.

The restriction of the behavior $\mathscr{B} \subseteq (\mathbb{R}^{\mathtt{w}})^{\mathbb{T}}$ to the time interval $[t_1, t_2]$, where $t_1, t_2 \in \mathbb{T}$ and $t_1 < t_2$, is denoted by

$$\mathscr{B}|_{[t_1,t_2]} := \{ w \in (\mathbb{R}^{\mathtt{w}})^{t_2 - t_1 + 1} \mid \text{ there are } w_- \text{ and } w_+ \text{ such that } \operatorname{col}(w_-, w, w_+) \in \mathscr{B} \}.$$

A system $\Sigma = (\mathbb{T}, \mathbb{W}, \mathscr{B})$ is *complete* if

$$w|_{[t_0,t_1]} \in \mathscr{B}|_{[t_0,t_1]} \quad \text{for all } t_0, t_1 \in \mathbb{T},\ t_0 \leq t_1 \quad \Longrightarrow \quad w \in \mathscr{B};$$

i.e., by looking at the time series w through a window of finite width $t_1 - t_0$, one can decide if it is in the behavior or not. Moreover, if the window can be taken to have a fixed width $t_1 - t_0 = l$, then the system is called l-complete. It turns out that a system is complete if and only if its behavior is closed in the topology of pointwise convergence, i.e., if $w_i \in \mathscr{B}$ for $i \in \mathbb{N}$ and $w_i(t) \to w(t)$, for all $t \in \mathbb{T}$, implies $w \in \mathscr{B}$. Also, a system is l-complete if and only if there is a difference equation representation of that system with l time shifts. For LTI systems, the completeness property is also called *finite dimensionality*.

We consider the class of discrete-time complete LTI systems. Our generic notation for the signal space is $\mathbb{W} = \mathbb{R}^{\mathtt{w}}$.

The class of all complete LTI systems with $\mathtt{w}$ variables is denoted by $\mathscr{L}^{\mathtt{w}}$.

Next, we discuss representations of the class $\mathscr{L}^{\mathtt{w}}$.

7.2 Kernel Representation

Consider the difference equation

$$R_0 w(t) + R_1 w(t+1) + \cdots + R_l w(t+l) = 0, \qquad \text{where} \quad R_\tau \in \mathbb{R}^{g \times \mathtt{w}}. \tag{DE}$$

It shows the dependence among consecutive samples of the time series w. Assuming that $R_l \neq 0$, the maximum number of shifts is l. The integer l is called the *lag* of the equation. Since in general (DE) is a vector equation, l is the largest lag among the lags $l_1, \ldots, l_g$ of all scalar equations.

Obviously, (DE) induces a dynamical system via the representation

$$\mathscr{B} = \{\, w \in (\mathbb{R}^{\mathtt{w}})^{\mathbb{Z}} \mid \text{(DE) holds} \,\}.$$

One can analyze $\mathscr{B}$ using the difference equation. It turns out, however, that it is more convenient to use polynomial matrix algebra for this purpose. (DE) is compactly written in terms of the polynomial matrix

$$R(z) := R_0 + R_1 z^1 + R_2 z^2 + \cdots + R_l z^l \in \mathbb{R}^{g \times \mathtt{w}}[z]$$

as $R(\sigma)w = 0$. Consequently, operations on the system of difference equations are represented by operations on the polynomial matrix R. The system induced by (DE) is

$$\ker\big(R(\sigma)\big) := \{\, w \in (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}} \mid R(\sigma)w = 0 \,\}. \tag{KER repr}$$

We call (KER repr) a kernel representation of the system $\mathscr{B} := \ker\big(R(\sigma)\big)$.

The following theorem summarizes the representation-free characterization of the class of complete LTI systems, explained in the previous section, and states that

without loss of generality one can assume the existence of a kernel representation $\mathscr{B} = \ker\big(R(\sigma)\big)$ of a system $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$.

Theorem 7.2 (Willems [Wil86a]). *The following are equivalent:*

(i) $\Sigma = (\mathbb{Z}, \mathbb{R}^{\mathtt{w}}, \mathscr{B})$ *is linear, time-invariant, and complete.*

(ii) $\mathscr{B}$ *is linear, shift-invariant, and closed in the topology of pointwise convergence.*

(iii) There is a polynomial matrix $R \in \mathbb{R}^{\bullet \times \mathtt{w}}[z]$*, such that* $\mathscr{B} = \ker\big(R(\sigma)\big)$.

The linearity of the system induced by (DE) follows from the linearity of (DE) with respect to w. The shift-invariance follows from the time-invariance of the coefficients $R_0, \ldots, R_l$, and the completeness follows from the fact that (DE) involves a finite number l shifts of the time series. Thus (iii) $\Longrightarrow$ (i) is immediate. The reverse implication (i) $\Longrightarrow$ (iii), on the other hand, requires proof; see [Wil86a, Theorem 5].

A kernel representation associated with a given $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$ is not unique. The non-uniqueness is due to:

1. linearly dependent equations (which refers to R not being full row rank) and
2. equivalence of the representations $\ker\big(R(\sigma)\big) = 0$ and $\ker\big(U(\sigma)R(\sigma)\big) = 0$, where $U \in \mathbb{R}^{g \times g}[z]$ is a unimodular matrix.

A square polynomial matrix U is *unimodular* if it has a polynomial inverse. A necessary and sufficient condition for U to be unimodular is its determinant to be a nonzero constant. Two kernel representations of the same behavior are called equivalent.

> Premultiplication of R with a unimodular matrix is a convenient way to represent a sequence of equivalence transformations on the system of difference equations (DE).

For a given system $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$, there always exists a kernel representation in which the polynomial matrix R has full row rank [Wil91, Proposition III.3]. Such a kernel representation is called a *minimal kernel representation*. In a minimal kernel representation, the number of equations $\mathtt{p} := \operatorname{row\,dim}(R)$ is minimal among all possible kernel representations of $\mathscr{B}$. All minimal kernel representations of a given system are in fact unimodularly equivalent; i.e., if $R'(\sigma) = 0$ and $R''(\sigma) = 0$ are both minimal, then there is a unimodular matrix U, such that $R' = UR''$.

There exists a minimal kernel representation $\mathscr{B} = \ker\big(R(\sigma)\big)$, in which the number of equations $\mathtt{p} = \operatorname{row\,dim}(R)$, the maximum lag $\mathtt{l} := \max_{i=1,\ldots,\mathtt{p}} l_i$, and the total lag $\mathtt{n} := \sum_{i=1}^{\mathtt{p}} l_i$ are simultaneously all minimal over all possible kernel representations [Wil86a, Theorem 6]. Such a kernel representation is called *shortest lag representation*. A kernel representation $\mathscr{B} = \ker\big(R(\sigma)\big)$ is a shortest lag representation if and only if $R(z)$ is row proper. The polynomial matrix $R = \begin{bmatrix} r_1 & \cdots & r_{\mathtt{p}} \end{bmatrix}^\top$, $\deg(r_i) =: \mathtt{l}_i$ is *row proper* if the leading row coefficient matrix (i.e., the matrix of which the (i,j)th entry is the coefficient of the term with power $\mathtt{l}_i$ of $R_{ij}(z)$) is full row rank. It can be shown that the $\mathtt{l}_i$'s are the observability indices of the system.

> The minimal and shortest lag kernel representations correspond to special properties of the R matrix: in a minimal representation, R is full row rank, and in a shortest lag representation, R is row proper.

A shortest lag representation is a special minimal representation, because a row proper matrix is necessarily full row rank. A shortest lag representation, however, is still not unique.

The minimal number of equations $\mathtt{p}$, the lag $\mathtt{l}$, and the total lag $\mathtt{n}$ are invariants of $\mathscr{B}$. It turns out that $\mathtt{p}$ is equal to the number of outputs, called output cardinality, in an input/output representation. Correspondingly, the integer $\mathtt{m} := \mathtt{w} - \mathtt{p}$ is also an invariant of $\mathscr{B}$ and is called the input cardinality. It is equal to the number of inputs in an input/output representation. The total lag $\mathtt{n}$ is equal to the state dimension in a minimal state space representation of $\mathscr{B}$. We use the following notation:

$\mathbf{m}(\mathscr{B})$ for the input cardinality of $\mathscr{B}$,

$\mathbf{p}(\mathscr{B})$ for the output cardinality of $\mathscr{B}$,

$\mathbf{n}(\mathscr{B})$ for the minimal state dimension of $\mathscr{B}$, and

$\mathbf{l}(\mathscr{B})$ for the lag of $\mathscr{B}$.

7.3 Inputs, Outputs, And Input/Output Representation

Consider a projection operator $\Pi \in \mathbb{R}^{\mathtt{w}\times\mathtt{w}}$ and a partitioning of the time series $w \in (\mathbb{R}^{\mathtt{w}})^{\mathbb{Z}}$ into time series u and y as follows:

$$\begin{bmatrix} u \\ y \end{bmatrix} := \Pi^{\top} w, \qquad \text{where} \quad \dim\big(u(t)\big) =: \mathtt{m}, \quad \dim\big(y(t)\big) =: \mathtt{p}, \quad \text{with} \quad \mathtt{m}+\mathtt{p} = \mathtt{w}.$$

Respectively a behavior $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$ is partitioned into two subbehaviors $\mathscr{B}_u$ and $\mathscr{B}_y$. The variables in u are called free if $\mathscr{B}_u = (\mathbb{R}^{\mathtt{m}})^{\mathbb{Z}}$. If, in addition, any other partitioning results in no more free variables, then $\mathscr{B}_u$ is called maximally free in $\mathscr{B}$. A partitioning in which $\mathscr{B}_u$ is maximally free is called an input/output partitioning with u an input and y an output.

There always exists an input/output partitioning of the variables of $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$, in fact a componentwise one; see [Wil86a, Theorem 2]. It is not unique, but the number of free variables $\mathtt{m}$ and the number of dependent variables $\mathtt{p}$ are equal to, respectively, the input cardinality and the output cardinality of $\mathscr{B}$ and are invariant. In a minimal kernel representation $\ker\big(R(\sigma)\big) = \mathscr{B}$, the choice of such a partitioning amounts to the selection of a full-rank square submatrix of R. The variables corresponding to the columns of R that form the full-rank submatrix are dependent variables and the other variables are free.

The inputs together with the initial conditions determine the outputs. This property is called *processing* [Wil91, Definition VIII.2]. Also the inputs can be chosen so that they are not anticipated by the outputs. *Nonanticipation* is also called causality [Wil91, Definition VIII.4].

Let $\ker\big(R(\sigma)\big)$ be a minimal kernel representation of $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$. One can always find a permutation matrix $\Pi \in \mathbb{R}^{\mathtt{w}\times\mathtt{w}}$, such that $P \in \mathbb{R}^{\mathtt{p}\times\mathtt{p}}[z]$, defined by $R\Pi =: \begin{bmatrix} Q & -P \end{bmatrix}$, has a nonzero determinant and the rational polynomial matrix

$$G(z) := P^{-1}(z)Q(z) \in \mathbb{R}^{\mathtt{p}\times\mathtt{m}}(z) \tag{TF}$$

is proper. This requires selecting a submatrix P among all full-rank square submatrices of R that has determinant of maximal degree. Then the corresponding partitioning of w, $\mathrm{col}(u, y) := \Pi^\top w$, is an input/output partitioning. G being proper implies that u is not anticipated by y; see [Wil91, Theorem VIII.7].

The difference equation

$$P(\sigma)y = Q(\sigma)u \tag{I/O eqn}$$

with an input/output partitioning Π is called an input/output equation, and the matrix G, defined in (TF), is called the transfer function of the system $\mathscr{B} := \ker\big(R(\sigma)\big)$.

> The class of LTI complete systems with $\mathtt{w}$ variables and at most $\mathtt{m}$ inputs is denoted by $\mathscr{L}^{\mathtt{w}}_{\mathtt{m}}$.

The system $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$ induced by an input/output equation with parameters (P, Q) (and input/output partitioning defined by Π) is

$$\mathscr{B}_{\text{i/o}}(P, Q, \Pi) := \{\, w := \Pi\, \mathrm{col}(u, y) \in (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}} \mid P(\sigma)y = Q(\sigma)u \,\}. \tag{I/O repr}$$

(I/O repr) is called an input/output representation of the system $\mathscr{B} := \mathscr{B}_{\text{i/o}}(P, Q, \Pi)$. If Π is the identity matrix $I_{\mathtt{w}}$, it is skipped in the notation of the input/output representation.

7.4 Latent Variables, State Variables, and State Space Representations

Modeling from first principles invariably requires the addition to the model of other variables apart from the ones that the model aims to describe. Such variables are called latent, and we denote them by l (not to be confused with the lag of a difference equation). The variables w that the model aims to describe are called manifest variables in order to distinguish them from the latent variables.

An important result, called the elimination theorem [Wil86a, Theorem 1], states that the behavior

$$\mathscr{B}(R, M) := \Big\{\, w \in (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}} \;\Big|\; \exists\, l \in (\mathbb{R}^{\mathtt{l}})^{\mathbb{N}}, \text{ such that } R(\sigma)w = M(\sigma)l \,\Big\} \tag{LV repr}$$

induced by the latent variable equation

$$R(\sigma)w = M(\sigma)l \tag{LV eqn}$$

is LTI. The behavior $\mathscr{B}(R, M)$ is called manifest behavior of the latent variable system. The behavior of the manifest and latent variables together is called the full behavior of the system. The elimination theorem states that if the full behavior is LTI, then the manifest behavior is LTI; i.e., by eliminating the latent variables, the resulting system is still LTI.

A latent variable system is *observable* if there is a map $w \mapsto l$, i.e., if the latent variables can be inferred from the knowledge of the system and the manifest variables. The kernel representation is a special case of the latent variable representation for $R = I$.

State variables are special latent variables that specify the memory of the system. More precisely, latent variables x are called state variables if they satisfy the following axiom of state [Wil91, Definition VII.1]:

$$(w_1, x_1), (w_2, x_2) \in \mathscr{B},\ t \in \mathbb{N},\ \text{and } x_1(t) = x_2(t) \quad \Longrightarrow \quad (w, x) \in \mathscr{B},$$

where

$$\big(w(\tau), x(\tau)\big) := \begin{cases} \big(w_1(\tau), x_1(\tau)\big) & \text{for } \tau < t \\ \big(w_2(\tau), x_2(\tau)\big) & \text{for } \tau \geq t. \end{cases}$$

A latent variable representation of the system is a state variable representation if there exists an equivalent representation whose behavioral equations are first order in the latent variables and zeroth order in the manifest variables. For example, the equation

$$\sigma x = A'x + B'v, \qquad w = C'x + D'v$$

defines a state representation. It is called state representation with a driving input because v acts like the input: v is free and, together with the initial conditions, determines a trajectory $w \in \mathscr{B}$. The system induced by the parameters (A', B', C', D') is

$$\mathscr{B}_{\text{ss}}(A', B', C', D') := \big\{\, w \in (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}} \;\big|\; \exists\, v \in (\mathbb{R}^{\mathtt{v}})^{\mathbb{N}} \text{ and } x \in (\mathbb{R}^{\mathtt{n}})^{\mathbb{N}}, \\ \text{such that } \sigma x = A'x + B'v,\ w = C'x + D'v \,\big\}.$$

Any LTI system $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$ admits a representation by an input/state/output equation

$$\sigma x = Ax + Bu, \qquad y = Cx + Du, \qquad w = \Pi\, \mathrm{col}(u, y), \tag{I/S/O eqn}$$

in which both the input/output and the state structure of the system are explicitly displayed [Wil86a, Theorem 3]. The system $\mathscr{B}$, induced by an input/state/output equation with parameters (A, B, C, D) and Π, is

$$\mathscr{B}_{\text{i/s/o}}(A, B, C, D, \Pi) := \{\, w := \Pi\, \mathrm{col}(u, y) \in (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}} \mid \exists\, x \in (\mathbb{R}^{\mathtt{n}})^{\mathbb{N}}, \\ \text{such that } \sigma x = Ax + Bu,\ y = Cx + Du \,\}. \tag{I/S/O repr}$$

(I/S/O repr) is called an input/state/output representation of the system $\mathscr{B} := \mathscr{B}_{\text{i/s/o}}(A, B, C, D, \Pi)$. Again, Π is skipped whenever it is I.

An input/state/output representation is not unique. The minimal state dimension $\mathtt{n} = \dim(x)$ among all input/state/output representations of $\mathscr{B}$, however, is invariant (denoted by $\mathbf{n}(\mathscr{B})$).

We denote the class of LTI systems with $\mathtt{w}$ variables, at most $\mathtt{m}$ inputs, and minimal state dimension at most $\mathtt{n}$ by $\mathscr{L}^{\mathtt{w},\mathtt{n}}_{\mathtt{m}}$.

7.5 Autonomous and Controllable Systems

A system $\mathscr{B}$ is *autonomous* if for any trajectory $w \in \mathscr{B}$ the past

$$w_- := \big(\dots, w(-2), w(-1)\big)$$

of w completely determines its future

$$w_+ := \big(w(0), w(1), \dots\big).$$

A system $\mathscr{B}$ is autonomous if and only if its input cardinality $\mathbf{m}(\mathscr{B})$ equals 0. Therefore, an autonomous LTI system is parameterized by the pair of matrices A and C via the state space representation

$$\sigma x = Ax, \qquad y = Cx, \qquad w = y. \tag{AUT}$$

The system induced by the state space representation with parameters (A, C) is

$$\mathscr{B}_{\text{i/s/o}}(A, C) := \{\, w \in (\mathbb{R}^{\mathtt{p}})^{\mathbb{N}} \mid \exists\, x \in (\mathbb{R}^{\mathtt{n}})^{\mathbb{N}}, \text{ such that } \sigma x = Ax,\ w = Cx \,\}.$$

The behavior of an autonomous system is finite dimensional; in fact, $\dim(\mathscr{B}) = \mathbf{n}(\mathscr{B})$. Alternatively, an autonomous LTI system is parameterized in a minimal kernel representation $\mathscr{B} = \ker\big(R(\sigma)\big)$ by a square and nonsingular matrix R, i.e., $R \in \mathbb{R}^{\mathtt{p}\times\mathtt{p}}[z]$, $\det(R) \neq 0$.

The system $\mathscr{B}$ is *controllable* if for any two trajectories $w_1, w_2 \in \mathscr{B}$, there is a third trajectory $w \in \mathscr{B}$, such that $w_1(t) = w(t)$, for all $t < 0$, and $w_2(t) = w(t)$, for all $t \geq 0$. The subset of controllable systems contained in the set $\mathscr{L}^{\mathtt{w}}$ is denoted by $\mathscr{L}^{\mathtt{w}}_{\text{ctrb}}$. A noncontrollable system $\mathscr{B}$ can be represented [Wil91, Proposition V.8] as $\mathscr{B} = \mathscr{B}_{\text{ctrb}} \oplus \mathscr{B}_{\text{aut}}$, where $\mathscr{B}_{\text{ctrb}}$ is the largest controllable subsystem in $\mathscr{B}$ and $\mathscr{B}_{\text{aut}}$ is a (nonunique) autonomous subsystem.

A test for controllability of the system $\mathscr{B}$ in terms of the parameter $R \in \mathbb{R}^{g\times\mathtt{w}}[z]$ in a kernel representation $\mathscr{B} = \ker\big(R(\sigma)\big)$ is given in [Wil91, Theorem V.2]: $\mathscr{B}$ is controllable if and only if the matrix $R(z)$ has a constant rank for all $z \in \mathbb{C}$. Equivalently, $\mathscr{B}$ is controllable if and only if a matrix R that defines a minimal kernel representation of $\mathscr{B}$ is left prime. In terms of the input/output representation $\mathscr{B} = \mathscr{B}_{\text{i/o}}(P, Q)$, $\mathscr{B}$ being controllable is equivalent to P and Q being left coprime.

The controllable subsystem $\mathscr{B}_{\text{ctrb}}$ of $\mathscr{B}$ can be found via the factorization $R = FR'$, where $F \in \mathbb{R}^{g\times g}[z]$ and R' is prime: $\mathscr{B}_{\text{ctrb}} = \ker\big(R'(\sigma)\big)$. In general, left multiplication of R with a nonsingular polynomial matrix changes the behavior: it amounts to adding an autonomous subbehavior. Only left multiplication with a unimodular matrix does not alter the behavior because it adds the trivial autonomous behavior $\{0\}$.

7.6 Representations for Controllable Systems

The transfer function G parameterizes the controllable subsystem of $\mathscr{B}_{\text{i/o}}(P, Q)$. Let $\mathscr{Z}$ be the Z-transform

$$\mathscr{Z}(w) = w(0) + w(1)z^{-1} + w(2)z^{-2} + \cdots$$

and consider the input/output equation

$$\mathscr{Z}(y) = G(z)\mathscr{Z}(u). \tag{TF eqn}$$

(TF eqn) is known as a frequency domain equation because $G(e^{j\omega})$ describes how the sinusoidal input $u(t) = \sin(\omega t)$ is "processed" by the system:

$$y(t) = |G(e^{j\omega})| \sin\big(\omega t + \angle G(e^{j\omega})\big).$$

The system induced by G (with an input/output partition defined by Π) is

$$\mathscr{B}_{\text{i/o}}(G, \Pi) := \big\{ w = \Pi \operatorname{col}(u, y) \in (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}} \;\big|\; y = \mathscr{Z}^{-1}\big(G(z)\mathscr{Z}(u)\big) \big\}. \qquad \text{(TF repr)}$$

(TF repr) is called a transfer function representation of the system $\mathscr{B} := \mathscr{B}_{\text{i/o}}(G, \Pi)$. In terms of the parameters of the input/state/output representation $\mathscr{B}_{\text{i/s/o}}(A, B, C, D) = \mathscr{B}_{\text{i/o}}(G)$, the transfer function is

$$G(z) = C(Iz - A)^{-1}B + D. \qquad \text{(TF}\leftarrow\text{I/S/O)}$$

Define the matrix valued time series $H \in (\mathbb{R}^{\mathtt{p}\times\mathtt{m}})^{\mathbb{N}}$ by $H := \mathscr{Z}^{-1}(G)$, i.e.,

$$G(z) = H(0) + H(1)z^{-1} + H(2)z^{-2} + \cdots . \qquad \text{(TF}\leftarrow\text{CONV)}$$

The time series H is a parameter in an alternative, time-domain representation of the system $\mathscr{B}_{\text{i/o}}(G, \Pi)$. Let $\star$ be the convolution operator. Then

$$y(t) := (H \star u)(t) = \sum_{\tau=0}^{t-1} H(\tau)u(t-\tau). \qquad \text{(CONV eqn)}$$

The system induced by H (with an input/output partition defined by Π) is

$$\mathscr{B}_{\text{i/o}}(H, \Pi) := \big\{ w = \Pi \operatorname{col}(u, y) \in (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}} \;\big|\; y = H \star u \big\}. \qquad \text{(CONV repr)}$$

(CONV repr) is called a convolution representation of the system $\mathscr{B} := \mathscr{B}_{\text{i/o}}(H, \Pi)$.

The matrices $H(t)$, $t \geq 0$, are called Markov parameters of the representation $\mathscr{B}_{\text{i/o}}(H)$. In terms of the parameters of the state space representation $\mathscr{B}_{\text{i/s/o}}(A, B, C, D) = \mathscr{B}_{\text{i/o}}(H)$, the Markov parameters are

$$H(0) = D, \qquad H(t) = CA^{t-1}B, \quad t \geq 1. \qquad \text{(CONV}\leftarrow\text{I/S/O)}$$

In addition to the transfer function (TF repr) and convolution (CONV repr) representations, a controllable system $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$ allows an image representation [Wil91, Theorem V.3]; i.e., there is a polynomial matrix $M \in \mathbb{R}^{\mathtt{w}\times g}[z]$, such that $\mathscr{B} = \operatorname{image}\big(M(\sigma)\big)$, where

$$\operatorname{image}\big(M(\sigma)\big) := \{\, w \in (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}} \mid \exists\, l \in (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}, \text{ such that } w = M(\sigma)l \,\}. \qquad \text{(IMG repr)}$$

The image representation is minimal if the number $\mathtt{l}$ of latent variables is minimal; i.e., there are no extra external variables in the representation than necessary. The image representation $\operatorname{image}\big(M(\sigma)\big)$ of $\mathscr{B}$ is minimal if and only if M is full column rank.

7.7 Representation Theorem

The following theorem summarizes the results presented in the previous sections of this chapter.

Theorem 7.3 (LTI system representations). *The following statements are equivalent:*

(i) $\mathscr{B}$ *is a complete LTI system with* $\mathtt{w}$ *variables,* $\mathtt{m}$ *inputs, and* $\mathtt{p} := \mathtt{w} - \mathtt{m}$ *outputs, i.e.,* $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$ *and* $\mathbf{m}(\mathscr{B}) = \mathtt{m}$*;*

(ii) there is a (full row rank) polynomial matrix $R \in \mathbb{R}^{\mathtt{p}\times\mathtt{w}}[z]$*, such that* $\mathscr{B} = \ker\big(R(\sigma)\big)$*;*

(iii) there are polynomial matrices $Q \in \mathbb{R}^{\mathtt{p}\times\mathtt{m}}[z]$ *and* $P \in \mathbb{R}^{\mathtt{p}\times\mathtt{p}}[z]$*,* $\det(P) \neq 0$*,* $P^{-1}Q$ *proper, and a permutation matrix* $\Pi \in \mathbb{R}^{\mathtt{w}\times\mathtt{w}}$*, such that* $\mathscr{B} = \mathscr{B}_{\text{i/o}}(P, Q, \Pi)$*;*

(iv) there is a natural number $\mathtt{n}$*, matrices* $A \in \mathbb{R}^{\mathtt{n}\times\mathtt{n}}$*,* $B \in \mathbb{R}^{\mathtt{n}\times\mathtt{m}}$*,* $C \in \mathbb{R}^{\mathtt{p}\times\mathtt{n}}$*, and* $D \in \mathbb{R}^{\mathtt{p}\times\mathtt{m}}$*, and a permutation matrix* $\Pi \in \mathbb{R}^{\mathtt{w}\times\mathtt{w}}$*, such that* $\mathscr{B} = \mathscr{B}_{\text{i/s/o}}(A, B, C, D, \Pi)$*;*

(v) there is a natural number $\mathtt{l} \in \mathbb{N}$ *and polynomial matrices* $R \in \mathbb{R}^{\mathtt{p}\times\mathtt{m}}[z]$ *and* $M \in \mathbb{R}^{\mathtt{p}\times\mathtt{l}}[z]$*, such that* $\mathscr{B} = \mathscr{B}(R, M)$*;*

(vi) there is a natural number $\mathtt{l} \in \mathbb{N}$ *and matrices* $A' \in \mathbb{R}^{\mathtt{n}\times\mathtt{n}}$*,* $B' \in \mathbb{R}^{\mathtt{n}\times\mathtt{m}}$*,* $C' \in \mathbb{R}^{\mathtt{p}\times\mathtt{n}}$*, and* $D' \in \mathbb{R}^{\mathtt{p}\times\mathtt{m}}$*, such that* $\mathscr{B} = \mathscr{B}_{\text{ss}}(A', B', C', D')$*.*

If in addition $\mathscr{B}$ *is controllable, then the following statement is equivalent to (i)–(vi):*

(vii) there is a full column rank matrix $M \in \mathbb{R}^{\mathtt{w}\times\mathtt{m}}[z]$*, such that* $\mathscr{B} = \operatorname{image}\big(M(\sigma)\big)$*.*

A controllable system $\mathscr{B}$ *has transfer function* $\mathscr{B}_{\text{i/o}}(G, \Pi)$ *and convolution* $\mathscr{B}_{\text{i/o}}(H, \Pi)$ *representations. These representations are unique when an input/output partitioning of the variables is fixed.*

The proofs of most of the implications of Theorem 7.3 can be found in [Wil86a] and [Wil91]. These proofs are constructive and give explicit algorithms for passing from one representation to another.

Figure 7.1 shows schematically the representations discussed up to now. To the left of the vertical line are representations that have no explicit input/output separation of the variables and to the right of the vertical line are representations with input/output separation of the variables. In the first row are state space representations. The representations below the second horizontal line exist only for controllable systems.

Transition from a latent variable representation to a representation without latent variables, for example $\mathscr{B}(R', M') \rightarrow \ker(R)$, involves elimination. Transition from a representation without an input/output separation to a representation with such a separation, for example $\ker(R) \rightarrow \mathscr{B}_{\text{i/o}}(P, Q)$, involves input/output selection. Transitions from a representation in the second or third rows to a representation in the first row is a realization problem.

In principle, all transitions from one type of representation to another are of interest (and imply algorithms that implement them). Moreover, all representations have special forms such as the controller canonical form, the observer canonical form, balanced representation, etc. Making the graph in Figure 7.1 connected suffices in order to be able to

derive any representation, starting from any other one. Having a specialized algorithm that does not derive intermediate representations, however, has advantages from a computational point of view.

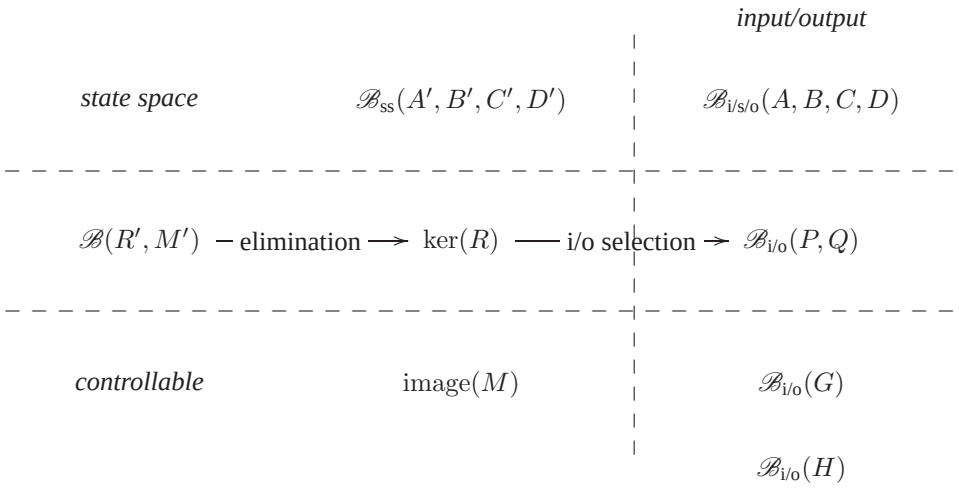


Figure 7.1. *Representations by categories: state space, input/output, and controllable.*

7.8 Parameterization of a Trajectory

A trajectory w of $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$ is parameterized by

1. a corresponding input u and
2. initial conditions x_{ini}.

If $\mathscr{B}$ is given in an input/state/output representation $\mathscr{B} = \mathscr{B}_{\text{i/s/o}}(A, B, C, D)$, then an input u is given and the initial conditions can be chosen as the initial state $x(1)$. The variation of constants formula

$$w = \text{col}(u, y), \qquad y(t) = CA^{t-1}x_{\text{ini}} + \sum_{\tau=1}^{t-1} \underbrace{CA^{t-\tau-1}B}_{H(t-\tau)} u(\tau) + Du(t), \quad t \geq 1 \qquad \text{(VC)}$$

gives a parameterization of w. Note that the second term in the expression for y is the convolution of H and u. It alone gives the zero initial conditions response. The ith column of the impulse response H is the zero initial conditions response of the system to input $u = e_i\delta$, where e_i is the ith unit vector.

For a given pair of matrices (A, B), $A \in \mathbb{R}^{\mathtt{n}\times\mathtt{n}}$, $B \in \mathbb{R}^{\mathtt{n}\times\mathtt{m}}$, and $t \in \mathbb{N}$, define the extended controllability matrix (with t block columns)

$$\mathscr{C}_t(A, B) := \begin{bmatrix} B & AB & \cdots & A^{t-1}B \end{bmatrix} \qquad (\mathscr{C})$$

and let $\mathscr{C}(A,B) := \mathscr{C}_\infty(A,B)$. The pair (A,B) is controllable if $\mathscr{C}(A,B)$ is full row rank. By the Cayley–Hamilton theorem [Bro70, page 72], $\operatorname{rank}\big(\mathscr{C}(A,B)\big) = \operatorname{rank}\big(\mathscr{C}_{\mathtt{n}}(A,B)\big)$, so that it suffices to check the rank of the finite matrix $\mathscr{C}_{\mathtt{n}}(A,B)$. The smallest natural number i, for which $\mathscr{C}_i(A,B)$ is full row rank, is denoted by $\nu(A,B)$ and is called the controllability index of the pair (A,B). The controllability index is an invariant under state transformation; i.e., $\nu(A,B) = \nu(SAS^{-1},SB)$ for any nonsingular matrix S. In fact, $\nu(A,B)$ is an invariant of any system $\mathscr{B}_{\text{i/s/o}}(A,B,\bullet,\bullet)$, so that it is legitimate to use the notation $\nu(\mathscr{B})$ for $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$. Clearly, $\nu(\mathscr{B}) \leq \mathbf{n}(\mathscr{B})$.

Similarly, for a given pair of matrices (A,C), $A \in \mathbb{R}^{\mathtt{n}\times\mathtt{n}}$, $C \in \mathbb{R}^{\mathtt{p}\times\mathtt{n}}$, and $t \in \mathbb{N}$, define the extended observability matrix (with t block rows)

$$\mathscr{O}_t(A,C) := \operatorname{col}(C, CA, \ldots, CA^{t-1}) \tag{$\mathscr{O}$}$$

and let $\mathscr{O}(A,C) := \mathscr{O}_\infty(A,C)$. The pair (A,C) is observable if $\mathscr{O}(A,C)$ is full column rank. Again, $\operatorname{rank}\big(\mathscr{O}(A,C)\big) = \operatorname{rank}\big(\mathscr{O}_{\mathtt{n}}(A,C)\big)$, so that it suffices to check the rank of $\mathscr{O}_{\mathtt{n}}(A,C)$. The smallest natural number i, for which $\mathscr{O}_i(A,C)$ is full row rank, is denoted by $\mu(A,C)$ and is called the observability index of the pair (A,C). The observability index is an invariant under state transformation; i.e., $\mu(A,C) = \mu(SAS^{-1},CS^{-1})$ for any nonsingular matrix S. In fact, $\nu(A,C)$ is equal to the lag of any system $\mathscr{B}_{\text{i/s/o}}(A,\bullet,C,\bullet)$, so that it is invariant and it is legitimate to use the notation $\nu(\mathscr{B})$ for $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$. Clearly, $\mathbf{l}(\mathscr{B}) = \nu(\mathscr{B}) \leq \mathbf{n}(\mathscr{B})$.

If the pairs (A,B) and (A,C) are understood from the context, they are skipped in the notation of the extended controllability and observability matrices.

We define also the lower triangular block-Toeplitz matrix

$$\mathscr{T}_{t+1}(H) := \begin{bmatrix} H(0) & & & & \\ H(1) & H(0) & & & \\ H(2) & H(1) & H(0) & & \\ \vdots & \vdots & \ddots & \ddots & \\ H(t) & H(t-1) & \cdots & H(1) & H(0) \end{bmatrix} \tag{$\mathscr{T}$}$$

and let $\mathscr{T}(H) = \mathscr{T}_\infty(H)$. With this notation, equation (VC) can be written compactly as

$$\begin{bmatrix} u \\ y \end{bmatrix} = \begin{bmatrix} 0 & I \\ \mathscr{O}(A,C) & \mathscr{T}(H) \end{bmatrix} \begin{bmatrix} x_{\text{ini}} \\ u \end{bmatrix}. \tag{VC'}$$

If the behavior $\mathscr{B}$ is not given by an input/state/output representation, then the parameterization of a trajectory $w \in \mathscr{B}$ is more involved. For example, in an input/output representation $\mathscr{B} = \mathscr{B}_{\text{i/o}}(P,Q)$, w can be parameterized by the input u and the $\mathtt{l} = \deg(P)$ values of the time series $w_{\text{ini}} := \big(w(-\mathtt{l}+1), \ldots, w(0)\big)$ preceding w as follows:

$$y = \mathscr{O}_{\text{i/o}} w_{\text{ini}} + \mathscr{T}(H)u. \tag{VC i/o}$$

Here $\mathscr{O}_{\text{i/o}}$ is a matrix that induces a mapping from w_{ini} to the corresponding initial conditions response. Let $\mathscr{B}_{\text{i/s/o}}(A,B,C,D) = \mathscr{B}_{\text{i/o}}(P,Q)$. Comparing (VC') and (VC), we see that the matrix $\mathscr{O}_{\text{i/o}}$ can be factored as $\mathscr{O}_{\text{i/o}} = \mathscr{O}(A,C)X$, where X is a matrix that induces the map $w_{\text{ini}} \mapsto x_{\text{ini}}$, called a state map [RW97].

The graph in Figure 7.2 illustrates the two representations introduced in this section for a trajectory w of the system $\mathscr{B} \in \mathscr{L}^{\mathtt{w},\mathtt{n}}_{\mathtt{m}}$.

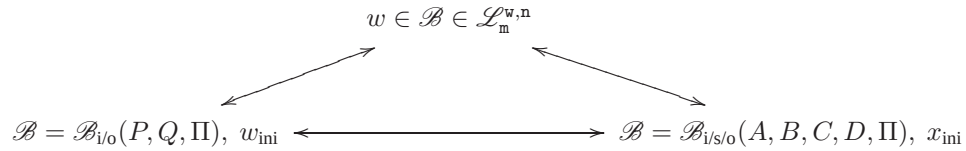


Figure 7.2. *Links among $w \in \mathscr{B} \in \mathscr{L}_{\mathtt{m}}^{\mathtt{w},\mathtt{n}}$ and its parameterizations in input/output and input/state/output form.*

7.9 Complexity of a Linear Time-Invariant System

In Chapter 2, we introduced the complexity of a linear system $\mathscr{B}$ as the dimension of $\mathscr{B}$ as a subspace of the universum set. For an LTI system $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$ and for $T \geq \mathbf{l}(\mathscr{B})$,

$$\dim(\mathscr{B}|_{[1,T]}) = \mathbf{m}(\mathscr{B})T + \mathbf{n}(\mathscr{B}), \qquad (\dim \mathscr{B})$$

which shows that the pair of natural numbers $\big(\mathbf{m}(\mathscr{B}), \mathbf{n}(\mathscr{B})\big)$ (the input cardinality and the total lag) specifies the complexity of the system. The model class $\mathscr{L}_{\mathtt{m}}^{\mathtt{w},\mathtt{n}}$ contains LTI systems of complexity bounded by the pair $(\mathtt{m}, \mathtt{n})$.

In the context of system identification problems, aiming at a kernel representation of the model, we need an alternative specification of the complexity by the input cardinality $\mathbf{m}(\mathscr{B})$ and the lag $\mathbf{l}(\mathscr{B})$. In general,

$$\big(\mathbf{l}(\mathscr{B}) - 1\big)\mathbf{p}(\mathscr{B}) < \mathbf{n}(\mathscr{B}) \leq \mathbf{l}(\mathscr{B})\mathbf{p}(\mathscr{B}),$$

so that

$$\dim(\mathscr{B}|_{[1,T]}) \leq \mathbf{m}(\mathscr{B})T + \mathbf{l}(\mathscr{B})\mathbf{p}(\mathscr{B})$$

and the pair $\big(\mathbf{m}(\mathscr{B}), \mathbf{l}(\mathscr{B})\big)$ bounds the complexity of the system $\mathscr{B}$.

> The class of LTI systems with `w` variables, at most `m` inputs, and lag at most `l` is denoted by $\mathscr{L}_{\mathtt{m},\mathtt{l}}^{\mathtt{w}}$.

This class specifies a set of LTI systems of a bounded complexity.

7.10 The Module of Annihilators of the Behavior*

Define the set of annihilators of the system $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$ as

$$\mathscr{N}_{\mathscr{B}} := \{\, r \in \mathbb{R}^{\mathtt{w}}[z] \mid r^{\top}(\sigma)\mathscr{B} = 0 \,\}$$

and the set of annihilators with length less than or equal to l as

$$\mathscr{N}_{\mathscr{B}}^{l} := \{\, r \in \mathscr{N}_{\mathscr{B}} \mid \deg(r) < l \,\}.$$

The sets $\mathscr{N}_{\mathscr{B}}$ and $\mathscr{N}_{\mathscr{B}}^l$ are defined as subsets of $\mathbb{R}^{\mathtt{w}}[z]$. With some abuse of notation, we consider also the annihilators as vectors; i.e., for $r(z) =: r_0 + r_1 z + \cdots + r_l z^l \in \mathscr{N}_{\mathscr{B}}$, we also write $\mathrm{col}(r_0, r_1, \ldots, r_l) \in \mathscr{N}_{\mathscr{B}}$.

Lemma 7.4. *Let $r(z) = r_0 + r_1 z + \cdots + r_{l-1} z^{l-1}$. Then $r \in \mathscr{N}_{\mathscr{B}}^l$ if and only if*

$$\mathrm{col}^\top(r_0, r_1, \ldots, r_{l-1})\mathscr{B}|_{[1,l]} = 0.$$

The set of annihilators $\mathscr{N}_{\mathscr{B}}$ is the dual $\mathscr{B}^\perp$ of the behavior $\mathscr{B}$.

The proof of the following facts can be found in [Wil86a]. The structure of $\mathscr{N}_{\mathscr{B}}$ is that of the module of $\mathbb{R}[z]$ generated by $\mathtt{p}$ polynomial vectors, say $r^{(1)}, \ldots, r^{(\mathtt{p})}$. The polynomial matrix $R := [r^{(1)} \ \ldots \ r^{(\mathtt{p})}]^\top$ yields a kernel representation of the behavior $\mathscr{B}$, i.e., $\mathscr{B} = \ker\big(R(\sigma)\big)$.

Without loss of generality, assume that R is row proper; i.e., $\ker\big(R(\sigma)\big)$ is a shortest lag kernel representation. By the row properness of R, the set of annihilators $\mathscr{N}_{\mathscr{B}}^l$ can be constructed from the $r^{(k)}$'s and their shifts

$$\mathscr{N}_{\mathscr{B}}^l = \mathrm{image}\big(r^{(1)}(z), zr^{(1)}(z), \ldots, z^{l-\mu_1-1}r^{(1)}(z) \ ; \ \ldots \ ; \\ r^{(\mathtt{p})}(z), zr^{(\mathtt{p})}(z), \ldots, z^{l-\mu_{\mathtt{p}}-1}r^{(\mathtt{p})}(z)\big).$$

The dimension of $\mathscr{N}_{\mathscr{B}}^l$ is $l - \mu_1 + l - \mu_2 + \cdots + l - \mu_p = \mathtt{p}l - \mathtt{n}$.

In the proof of the fundamental lemma (see Appendix A.3), we need the following simple fact.

Lemma 7.5. *Let $r^{(1)}, \ldots, r^{(\mathtt{p})}$, where $\deg(r_i) =: \mu_i$, be independent over the ring of polynomials. Then*

$$r^{(1)}(z), zr^{(1)}(z), \ldots, z^{l-\mu_1-1}r^{(1)}(z) \ ; \ \ldots \ ; \ r^{(\mathtt{p})}(z), zr^{(\mathtt{p})}(z), \ldots, z^{l-\mu_{\mathtt{p}}-1}r^{(\mathtt{p})}(z)$$

are independent over the field of reals.

Chapter 8

Exact Identification

With this chapter, we start to consider identification problems. The first problem is the simplest of this type: given a trajectory of an LTI system, find a representation of the system that produced this trajectory. The problem is defined and motivated in Sections 8.1–8.3.

Exact identification is closely related to the construction of the most powerful unfalsified model (MPUM). In Section 8.2, we define the MPUM, and in Section 8.3, we define the identifiability property. Under identifiability, the MPUM of the data, which is explicitly constructible from the data, coincides with the data generating system. This allows us to find the data generating system from data. An identifiability test in terms of the given data is presented in Section 8.4. This key result is repeatedly used in what follows and is called the fundamental lemma.

In Section 8.5, we review algorithms for exact identification. Section 8.6 presents algorithms for passing from data to a convolution representation. Section 8.7 reviews realization theory and algorithms. Section 8.8 presents algorithms for computation of sequential free responses, which are a key ingredient of direct algorithms for the construction of an input/state/output representation of the MPUM.

In Section 8.9, we explain the relation of the algorithms presented to classical algorithms for deterministic subspace identification. In particular, the orthogonal and oblique projections correspond to computation of, respectively, free responses and sequential free responses of the system. We comment on the inherent inefficiency of the orthogonal and oblique projections for the purpose of exact identification. Simulation results that compare the efficiency of various exact identification algorithms are shown in Section 8.10.

8.1 Introduction

In this chapter, we consider the following problem:

> Given a trajectory w_{d} of an LTI system $\mathscr{B}$, find a representation of $\mathscr{B}$.

We refer to this most basic identification problem as an exact identification problem. It is of interest to find algorithms that make the transition from w_{d} directly to any one of the various possible representations of $\mathscr{B}$; cf., Figure 7.1.

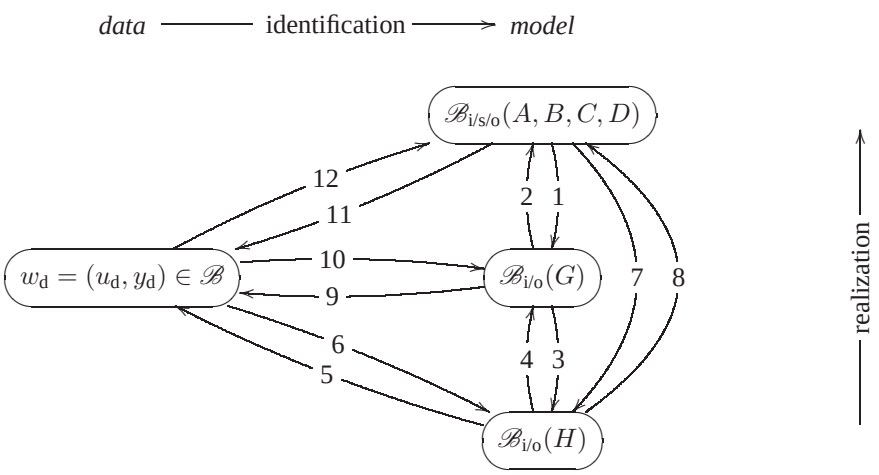


Figure 8.1. *Data, input/output model representations, and links among them.*

1. $G(z) = C(Iz - A)^{-1}B + D$
2. Realization of a transfer function
3. $H = \mathscr{Z}^{-1}(G)$
4. $G = \mathscr{Z}(H) = \sum_{t=0}^{\infty} H(t)z^{-t}$
5. Convolution $y_{\text{d}}(t) = \sum_{\tau=0}^{t} H(\tau)u_{\text{d}}(t-\tau)$
6. Exact identification; see Algorithms 8.6 and 8.7
7. $H(0) = D$, $H(t) = CA^{t-1}B$, for $t \geq 1$
8. Realization of an impulse response; see Algorithm 8.8
9. Simulation of the response under the input u_{d}
10. Exact identification; see Algorithm 8.1
11. Simulation of the response under the input u_{d} and initial conditions $x(1) = x_{\text{ini}}$
12. Exact identification; see Algorithms 8.4 and 8.5

Figure 8.1 shows the representations with an input/output partition of the variables that we considered before and the trajectory $w_{\text{d}} =: (u_{\text{d}}, y_{\text{d}})$. The transitions from w_{d} to convolution, transfer function, and input/state/output representations are exact identification problems. The transitions among the representations themselves are representation problems. Most notable of the representation problems are the realization ones: passing from an impulse response or transfer function to an input/state/output representation.

The exact identification problem is an important system theoretic problem. It includes as a special case the classical impulse response realization problem and is a prerequisite

for the study of more involved approximate, stochastic, and stochastic/approximate identification problems (e.g., the GlTLS misfit minimization problem, which is an approximate identification problem). In addition, numerical algorithms for exact identification are useful computational tools and appear as subproblems in other identification algorithms. By itself, however, exact identification is not a practical identification problem. The data is assumed to be exact and unless $\mathscr{B}$ is the trivial system $\mathscr{B} = (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$, a randomly chosen time series $w_{\mathrm{d}} \in (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$ is a trajectory of $\mathscr{B}$ with probability zero.

Modified exact identification algorithms can be applied on data that is not necessarily generated by a finite dimensional LTI system by replacing exact linear algebra operations with approximate operations. For example, rank determination is replaced by numerical rank determination (via SVD) and solution of a system of linear equations by LS or TLS approximation. A lot of research is devoted to the problem of establishing alternatives to $w_{\mathrm{d}} \in \mathscr{B}$, under which such modified algorithms have desirable properties. Often this problem is treated in the stochastic setting of the ARMAX model and the properties aimed at are consistency and asymptotic efficiency.

Note 8.1 (Multiple time series) In general, the given data for identification is a finite set of time series $w_{\mathrm{d},1}, \ldots, w_{\mathrm{d},N}$. In the presentation, however, we define and solve the identification problems for a single time series. The generalization for multiple time series of equal length is trivial and the one for nonequal length is an open problem.

Note 8.2 (Finite amount of data) An important aspect of the identification problems that we address is the finiteness of the available data. Previous studies of exact identification either assume an infinite amount of data or do not address the issue of finiteness of the data.

Note 8.3 (Given input/output partitioning) Although the exact identification problem is defined in the behavioral setting, most of the established results are in the input/output setting. In our treatment, some problems are also solved in the input/output setting.

Software implementation of the algorithms presented in this and the following chapter is described in Appendix B.3.

8.2 The Most Powerful Unfalsified Model

The notion of the most powerful unfalsified model (MPUM) is introduced in [Wil86b, Definition 4]. It plays a fundamental role in the exact identification problem.

Definition 8.4 (MPUM in the model class $\mathscr{L}^{\mathtt{w}}$ [Wil86b]). *The system $\mathscr{B} \subseteq (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$ is an MPUM of the time series $w_{\mathrm{d}} \in (\mathbb{R}^{\mathtt{w}})^{T}$ in the model class $\mathscr{L}^{\mathtt{w}}$ if it is*

1. *finite dimensional LTI, i.e., $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$,*
2. *unfalsified, i.e., $w_{\mathrm{d}} \in \mathscr{B}|_{[1,T]}$, and*
3. *most powerful among all finite dimensional LTI unfalsified systems, i.e.,*

$$\mathscr{B}' \in \mathscr{L}^{\mathtt{w}} \text{ and } w_{\mathrm{d}} \in \mathscr{B}'|_{[1,T]} \quad \Longrightarrow \quad \mathscr{B}|_{[1,T]} \subseteq \mathscr{B}'|_{[1,T]}.$$

The MPUM of w_{d} is denoted by $\mathscr{B}_{\mathrm{mpum}}(w_{\mathrm{d}})$. We skip the explicit dependence on w_{d} when w_{d} is understood from the context.

The existence and uniqueness of the MPUM are proven in the following theorem.

Theorem 8.5 (Existence and uniqueness of the MPUM [Wil86b]). *The MPUM of $w_{\mathrm{d}} \in (\mathbb{R}^{\mathtt{w}})^T$ exists and is unique. Moreover,*

$$\mathscr{B}_{\mathrm{mpum}}(w_{\mathrm{d}}) = \cap_{\substack{w_{\mathrm{d}} \in \mathscr{B}|_{[1,T]} \\ \mathscr{B} \in \mathscr{L}^{\mathtt{w}}}} \mathscr{B};$$

i.e., $\mathscr{B}_{\mathrm{mpum}}(w_{\mathrm{d}})$ is the smallest shift-invariant closed in the topology of pointwise convergence subspace of $(\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$ that contains w_{d}.

Proof. Define $\mathscr{B}' := \cap_{\substack{w_{\mathrm{d}} \in \mathscr{B}|_{[1,T]} \\ \mathscr{B} \in \mathscr{L}^{\mathtt{w}}}} \mathscr{B}$. We will show that $\mathscr{B}'$ is an MPUM.

Lemma 8.6 (Intersection property of $\mathscr{L}^{\mathtt{w}}$). $\mathscr{B}_1, \mathscr{B}_2 \in \mathscr{L}^{\mathtt{w}} \implies \mathscr{B}_1 \cap \mathscr{B}_2 \in \mathscr{L}^{\mathtt{w}}$.

Proof. See [Wil86b, Proposition 11]. □

Lemma 8.6 implies that $\mathscr{B}' \in \mathscr{L}^{\mathtt{w}}$. Obviously, $w_{\mathrm{d}} \in \mathscr{B}'$, so that $\mathscr{B}'$ is unfalsified. Moreover, $\mathscr{B}'$ is in the intersection of all finite dimensional LTI unfalsified models, so that it is most powerful. Therefore, $\mathscr{B}'$ is an MPUM.

We proved the existence of an MPUM. In order to prove uniqueness, assume that there is $\mathscr{B}'' \neq \mathscr{B}'$ that is also an MPUM of w_{d}. By Lemma 8.6, $\mathscr{B} := \mathscr{B}'' \cap \mathscr{B}' \in \mathscr{L}^{\mathtt{w}}$ and $\mathscr{B}$ is obviously unfalsified. But $\mathscr{B} \subset \mathscr{B}'$, so that $\mathscr{B}'$ is not an MPUM, which is a contradiction. □

The next proposition shows another characterization of the MPUM for infinite w_{d}.

Proposition 8.7. *Let $w_{\mathrm{d}} \in (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$. Then*

$$\mathscr{B}_{\mathrm{mpum}}(w_{\mathrm{d}}) = \mathrm{closure}\big(\,\mathrm{image}(w_{\mathrm{d}}, \sigma w_{\mathrm{d}}, \sigma^2 w_{\mathrm{d}}, \ldots)\big);$$

i.e., $\mathscr{B}_{\mathrm{mpum}}(w_{\mathrm{d}})$ is the closure of the span of w_{d} and all its shifts.

Proof. Let $\mathscr{B}' := \mathrm{closure}\big(\,\mathrm{image}(w_{\mathrm{d}}, \sigma w_{\mathrm{d}}, \sigma^2 w_{\mathrm{d}}, \ldots)\big)$. By definition, $\mathscr{B}'$ is a closed, linear, and shift-invariant subspace. Then [Wil86a, Theorem 5] implies that $\mathscr{B}' \in \mathscr{L}^{\mathtt{w}}$. By definition, $w_{\mathrm{d}} \in \mathscr{B}'$, so that $\mathscr{B}'$ is unfalsified. From conditions 1 and 2 of Definition 8.4, it is easy to see that any unfalsified model contains $\mathscr{B}'$. Therefore, $\mathscr{B}'$ is the MPUM of w_{d}. □

Note 8.8 (Algorithms for construction of the MPUM) Proposition 8.7 shows that the MPUM $\mathscr{B}_{\mathrm{mpum}}(w_{\mathrm{d}})$ is explicitly constructible from the given data w_{d}. However, algorithms that pass from w_{d} to concrete representations of $\mathscr{B}_{\mathrm{mpum}}(w_{\mathrm{d}})$ are needed. Such algorithms are described in Section 8.5.

Note 8.9 (Generically $\mathscr{B}_{\text{mpum}}(w_{\text{d}}) = (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$ for infinite data $w_{\text{d}} \in (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$) The existence of the MPUM is guaranteed in the model class $\mathscr{L}^{\mathtt{w}}$ of unbounded complexity. For "rough" data $w_{\text{d}} \in (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$ (the generic case in $(\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$), the MPUM is the trivial system $\mathscr{B}_{\text{mpum}}(w_{\text{d}}) = (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$, i.e., a system with $\mathtt{w}$ inputs. Therefore, generically the MPUM of an infinite time series does not exist in a model class $\mathscr{L}^{\mathtt{w}}_{\mathtt{m}}$ with $\mathtt{m} < \mathtt{w}$. Therefore, an approximation is needed in order to find a nontrivial model. Approximate identification is treated in Chapter 11.

Note 8.10 (Generically $\mathscr{B}_{\text{mpum}}(w_{\text{d}})|_{[1,T]} = (\mathbb{R}^{\mathtt{w}})^T$ for finite data $w_{\text{d}} \in (\mathbb{R}^{\mathtt{w}})^T$)
For finite data $w_{\text{d}} \in (\mathbb{R}^{\mathtt{w}})^T$, the MPUM always exists in a model class $\mathscr{L}^{\mathtt{w}}_{\mathtt{m}}$ with any number $0 \leq \mathtt{m} \leq \mathtt{w}$ of inputs. For rough data the solution is still a trivial system $\mathscr{B}_{\text{mpum}}(w_{\text{d}})|_{[1,T]} = (\mathbb{R}^{\mathtt{w}})^T$. Now, however, the possibility of fitting an arbitrary T samples long time series is achieved by the initial conditions as well as the inputs. Indeed, any observable system $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$ of order $\mathbf{n}(\mathscr{B}) \geq \mathbf{p}(\mathscr{B})T$ is unfalsified by any T samples long time series $w_{\text{d}} \in (\mathbb{R}^{\mathtt{w}})^T$.

8.3 Identifiability

Not every trajectory w_{d} of a system $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$ allows the reconstruction of $\mathscr{B}$ from w_{d}. For example, the trajectory $w_{\text{d}} = 0 \in \mathscr{B}$ does not carry any information about $\mathscr{B}$ because any LTI system is compatible with the zero trajectory. The possibility of identifying $\mathscr{B}$ from w_{d} is a property of both w_{d} and $\mathscr{B}$. In order to formalize the notion of the "possibility of identifying a system from exact data", we define the identifiability property as follows.

Definition 8.11 (Identifiability). *The system $\mathscr{B} \subseteq (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$ is identifiable from the data $w_{\text{d}} \in (\mathbb{R}^{\mathtt{w}})^T$ in the model class $\mathscr{L}^{\mathtt{w},\mathtt{n}}_{\mathtt{m},\mathtt{l}}$ if*

1. *$\mathscr{B} \in \mathscr{L}^{\mathtt{w},\mathtt{n}}_{\mathtt{m},\mathtt{l}}$,*
2. *$w_{\text{d}} \in \mathscr{B}|_{[1,T]}$, and*
3. *there is no other system $\mathscr{B}' \in \mathscr{L}^{\mathtt{w},\mathtt{n}}_{\mathtt{m},\mathtt{l}}$, $\mathscr{B}' \neq \mathscr{B}$, that fits the data, i.e.,*

$$\mathscr{B}' \in \mathscr{L}^{\mathtt{w},\mathtt{n}}_{\mathtt{m},\mathtt{l}} \quad \text{and} \quad w_{\text{d}} \in \mathscr{B}'|_{[1,T]} \quad \Longrightarrow \quad \mathscr{B}' = \mathscr{B}.$$

Identifiability in $\mathscr{L}^{\mathtt{w},\mathtt{n}}_{\mathtt{m},\mathtt{l}}$ implies that the MPUM of the data w_{d} is in $\mathscr{L}^{\mathtt{w},\mathtt{n}}_{\mathtt{m},\mathtt{l}}$ and coincides with the data generating system $\mathscr{B}$.

Theorem 8.12. *If $\mathscr{B} \subseteq (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$ is identifiable in the model class $\mathscr{L}^{\mathtt{w},\mathtt{n}}_{\mathtt{m},\mathtt{l}}$ from the data $w_{\text{d}} \in (\mathbb{R}^{\mathtt{w}})^T$ in the model class $\mathscr{L}^{\mathtt{w},\mathtt{n}}_{\mathtt{m},\mathtt{l}}$, then $\mathscr{B} = \mathscr{B}_{\text{mpum}}(w_{\text{d}})$.*

Proof. The first condition for $\mathscr{B}$ being identifiable from w_{d} implies the first condition for $\mathscr{B}$ being the MPUM of w_{d}, and the second conditions are equivalent. Condition 3 for $\mathscr{B}$ being identifiable from w_{d} implies that there is a unique unfalsified system in the model class $\mathscr{L}^{\mathtt{w},\mathtt{n}}_{\mathtt{m},\mathtt{l}}$. Therefore, $\mathscr{B}$ is the MPUM of w_{d}. □

Since the MPUM is explicitly computable from the given data (see Note 8.8) identifiability indeed implies the "possibility of identifying the system from exact data". In Section 8.5, we list algorithms for passing from w_{d} to kernel, convolution, and input/state/output

representations of the MPUM. For example, consider Algorithm 8.1, which constructs a kernel representation of the MPUM $\mathscr{B}_{\text{mpum}}(w_{\text{d}})$.

Next, we define the considered exact identification problem.

Problem 8.13 (Exact identification). *Given $w_{\text{d}} \in \mathscr{B} \in \mathscr{L}^{\mathtt{w}}$ and complexity specification $(\mathtt{m}, \mathtt{l}_{\max}, \mathtt{n}_{\max})$, determine whether $\mathscr{B}$ is identifiable from w_{d} in the model class $\mathscr{L}^{\mathtt{w},\mathtt{n}_{\max}}_{\mathtt{m},\mathtt{l}_{\max}}$, and if so, find an algorithm that computes a representation of $\mathscr{B}$.*

8.4 Conditions for Identifiability

The block-Hankel matrix with t_1 block rows and t_2 block columns, constructed from (in general matrix valued) time series $w = \big(w(1), w(2), \dots\big)$, is denoted by

$$\mathscr{H}_{t_1,t_2}(w) := \begin{bmatrix} w(1) & w(2) & w(3) & \cdots & w(t_2) \\ w(2) & w(3) & w(4) & \cdots & w(t_2+1) \\ w(3) & w(4) & w(5) & \cdots & w(t_2+2) \\ \vdots & \vdots & \vdots & & \vdots \\ w(t_1) & w(t_1+1) & w(t_1+2) & \cdots & w(t_1+t_2-1) \end{bmatrix}. \qquad (\mathscr{H})$$

If both block dimensions t_1 and t_2 are infinite, we skip them in the notation; i.e., we define $\mathscr{H}(w) := \mathscr{H}_{\infty,\infty}(w)$. If the time series is finite $w = \big(w(1), \dots, w(T)\big)$, i.e., then $\mathscr{H}_{t_1}(w)$ denotes the Hankel matrix with t_1 block rows and as many block columns as the finite time horizon T allows; i.e., $\mathscr{H}_{t_1}(w) := \mathscr{H}_{t_1,t_2}(w)$, where $t_2 = T - t_1 + 1$.

With some abuse of notation (w is viewed as both the matrix $\begin{bmatrix} w(1) & w(2) & \cdots \end{bmatrix}$ and the vector $\operatorname{col}\big(w(1), w(2), \dots\big)$, the infinite Hankel matrix $\mathscr{H}(w)$ can be block partitioned in the following two ways:

$$\mathscr{H}(w) = \begin{bmatrix} w \\ \sigma w \\ \sigma^2 w \\ \vdots \end{bmatrix} = \begin{bmatrix} w & \sigma w & \sigma^2 w & \cdots \end{bmatrix},$$

which shows that it is composed of w and its shifts $\sigma^t w$, $t \geq 1$, stacked next to each other. Therefore, $w \in \mathscr{B}$ implies that $\operatorname{col}\operatorname{span}\big(\mathscr{H}(w)\big) \subseteq \mathscr{B}$. We establish conditions on w and $\mathscr{B}$ under which equality holds, i.e., conditions under which w specifies $\mathscr{B}$ exactly.

Definition 8.14 (Persistency of excitation). *The time series $u_{\text{d}} = \big(u_{\text{d}}(1), \dots, u_{\text{d}}(T)\big)$ is persistently exciting of order L if the Hankel matrix $\mathscr{H}_L(u_{\text{d}})$ is of full row rank.*

Lemma 8.15 (Fundamental lemma [WRMM05]). *Let*

1. *$w_{\text{d}} = (u_{\text{d}}, y_{\text{d}})$ be a T samples long trajectory of the LTI system $\mathscr{B}$, i.e.,*

$$w_{\text{d}} = \begin{bmatrix} u_{\text{d}} \\ y_{\text{d}} \end{bmatrix} = \left(\begin{bmatrix} u_{\text{d}}(1) \\ y_{\text{d}}(1) \end{bmatrix}, \dots, \begin{bmatrix} u_{\text{d}}(T) \\ y_{\text{d}}(T) \end{bmatrix} \right) \in \mathscr{B}|_{[1,T]};$$

2. *the system $\mathscr{B}$ be controllable; and*

3. *the input sequence u_{d} be persistently exciting of order $L + \mathbf{n}(\mathscr{B})$.*

Then any L samples long trajectory $w = (u, y)$ of $\mathscr{B}$ can be written as a linear combination of the columns of $\mathscr{H}_L(w_{\mathrm{d}})$ and any linear combination $\mathscr{H}_L(w_{\mathrm{d}})g$, $g \in \mathbb{R}^{T-L+1}$, is a trajectory of $\mathscr{B}$, i.e.,

$$\operatorname{col\,span}\left(\mathscr{H}_L(w_{\mathrm{d}})\right) = \mathscr{B}|_{[1,L]}.$$

Proof. See Appendix A.3. □

The fundamental lemma gives conditions under which the Hankel matrix $\mathscr{H}_L(w_{\mathrm{d}})$ has the "correct" image (and as a consequence the "correct" left kernel). For sufficiently large L, namely $L \geq \mathbf{l}(\mathscr{B}) + 1$, it answers the identifiability question.

Theorem 8.16 (Identifiability conditions). *The system $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$ is identifiable from the exact data $w_{\mathrm{d}} = (u_{\mathrm{d}}, y_{\mathrm{d}}) \in \mathscr{B}$ if $\mathscr{B}$ is controllable and u_{d} is persistently exciting of order $\mathbf{l}(\mathscr{B}) + 1 + \mathbf{n}(\mathscr{B})$.*

Note that for applying Theorem 8.16, we need to know a priori the order and the lag of $\mathscr{B}$ and that $\mathscr{B}$ is controllable. These assumptions can be relaxed as follows. Knowledge of upper bounds $\mathtt{n}_{\max}$ and $\mathtt{l}_{\max}$ of, respectively, $\mathbf{n}(\mathscr{B})$ and $\mathbf{l}(\mathscr{B})$ suffice to verify identifiability. Moreover, the condition "$\mathscr{B}$ controllable and u_{d} persistently exciting of order $\mathtt{l}_{\max} + 1 + \mathtt{n}_{\max}$" is the sharpest necessary condition for identifiability that is verifiable from the data, $\mathtt{n}_{\max}$, and $\mathtt{l}_{\max}$ only. In other words, if u_{d} is not persistently exciting of order $\mathtt{l}_{\max} + 1 + \mathtt{n}_{\max}$, then there is a controllable system $\mathscr{B} \in \mathscr{L}^{\mathtt{w},\mathtt{n}_{\max}}_{\mathtt{m},\mathtt{l}_{\max}}$, such that $w_{\mathrm{d}} \in \mathscr{B}$ and $\mathscr{B}$ is not identifiable from w_{d}.

We will need the following corollary of the fundamental lemma.

Corollary 8.17 (Willems et al. [WRMM05]). *Consider the minimal input/state/output representation of the controllable system $\mathscr{B}$, $\mathscr{B}_{\mathrm{i/s/o}}(A, B, C, D)$, and let x_{d} be the state sequence of $\mathscr{B}_{\mathrm{i/s/o}}(A, B, C, D)$, corresponding to the trajectory $w_{\mathrm{d}} = (u_{\mathrm{d}}, y_{\mathrm{d}})$ of $\mathscr{B}$.*

(i) If u_{d} is persistently exciting of order $\mathbf{n}(\mathscr{B}) + 1$, then

$$\operatorname{rank}\left(\begin{bmatrix} x_{\mathrm{d}}(1) & x_{\mathrm{d}}(2) & \cdots & x_{\mathrm{d}}(T) \end{bmatrix}\right) = \mathbf{n}(\mathscr{B})$$

and

$$\operatorname{rank}\begin{bmatrix} u_{\mathrm{d}}(1) & \cdots & u_{\mathrm{d}}(T) \\ x_{\mathrm{d}}(1) & \cdots & x_{\mathrm{d}}(T) \end{bmatrix} = \mathbf{n}(\mathscr{B}) + \mathtt{m}.$$

(ii) If u_{d} is persistently exciting of order $\mathbf{n}(\mathscr{B}) + L$, then

$$\operatorname{rank}\begin{bmatrix} X_{\mathrm{d}} \\ \mathscr{H}_L(u_{\mathrm{d}}) \end{bmatrix} = \mathbf{n}(\mathscr{B}) + L\mathtt{m}, \quad \textit{where} \quad X_{\mathrm{d}} := \begin{bmatrix} x_{\mathrm{d}}(1) & \cdots & x_{\mathrm{d}}(T - L + 1) \end{bmatrix}.$$

The rest of the chapter is devoted to the second part of the exact identification problem: algorithms that compute a representation of the MPUM.

8.5 Algorithms for Exact Identification

If the conditions of Theorem 8.16 are satisfied, then there are algorithms that compute a representation of the data generating system $\mathscr{B}$ from the data w_{d}. In fact, such algorithms compute the MPUM of the data w_{d}. In this section, we outline four classes of algorithms for exact identification. The first one derives a kernel representation and the second one derives a convolution representation. Composed with realization algorithms, they give (indirect) algorithms for the computation of state space representations. The last two classes of algorithms construct (directly) an input/state/output representation.

Algorithms for Computation of a Kernel Representation

Under the assumption of the fundamental lemma,

$$\ker\left(\mathscr{H}_{\mathtt{l}_{\max}+1}(w_{\mathrm{d}})\right) = \mathscr{B}|_{[0,\mathtt{l}_{\max}]}.$$

Therefore, a basis for the left kernel of $\mathscr{H}_{\mathtt{l}_{\max}+1}(w_{\mathrm{d}})$ defines a kernel representation of $\mathscr{B} \in \mathscr{L}^{\mathtt{w},\mathtt{n}_{\max}}_{\mathtt{m},\mathtt{l}_{\max}}$. Let

$$\begin{bmatrix}\tilde{R}_0 & \tilde{R}_1 & \cdots & \tilde{R}_{\mathtt{l}_{\max}}\end{bmatrix} \mathscr{H}_{\mathtt{l}_{\max}+1}(w_{\mathrm{d}}) = 0,$$

where $\tilde{R}_i \in \mathbb{R}^{g\times \mathtt{w}}$ with $g = \mathtt{p}(\mathtt{l}_{\max}+1) - \mathbf{n}(\mathscr{B})$. Then

$$\mathscr{B} = \ker\left(\tilde{R}(\sigma)\right), \quad \text{where} \quad \tilde{R}(z) = \sum_{i=0}^{\mathtt{l}_{\max}} \tilde{R}_i z^i.$$

This (in general nonminimal) kernel representation can be made minimal by standard polynomial linear algebra algorithms: find a unimodular matrix $\tilde{U} \in \mathbb{R}^{g\times g}[z]$, such that $\tilde{U}\tilde{R} = \left[\begin{smallmatrix} R \\ 0 \end{smallmatrix}\right]$, where R is full row rank. Then $\ker\left(R(\sigma)\right) = 0$ is a minimal kernel representation of $\mathscr{B}$.

The above procedure is summarized in Algorithm 8.1.

Note 8.18 (Approximate identification) The SVD in step 2 of Algorithm 8.1 is used for the computation of the left kernel of the block-Hankel matrix $\mathscr{H}_{\mathtt{l}_{\max}+1}(w_{\mathrm{d}})$. Other algorithms can be used for the same purpose as well. The SVD, however, has an important advantage when an approximate model is desired.

Suppose that $\operatorname{rank}\left(\mathscr{H}_{\mathtt{l}_{\max}+1}(w_{\mathrm{d}})\right) = \mathtt{w}(\mathtt{l}_{\max}+1)$, so that $\mathscr{B}_{\mathtt{mpum}}$ is the trivial model $(\mathbb{R}^{\mathtt{w}})^T$. Nevertheless, one can proceed heuristically with steps 5 and 6 in order to compute a nontrivial approximate model. The parameter g can either be chosen from the decay of the singular values (e.g., the number of singular values smaller than a user-given tolerance) or be fixed. The selection of g determines the number of inputs of the identified model and thus its complexity. The motivation for this heuristic for approximate modeling is that U_2 spans a space that in a certain sense is an "approximate left kernel" of $\mathscr{H}_{\mathtt{l}_{\max}+1}(w_{\mathrm{d}})$.

In [Wil86b, Section 15], Algorithm 8.1 is refined. An efficient recursive algorithm for the computation of a kernel representation of the MPUM is proposed. Moreover, the algorithm of [Wil86b] computes a shortest lag kernel representation and as a byproduct finds an input/output partition of the variables.

Algorithm 8.1 Kernel representation of the MPUM `w2r`

Input: $w_{\mathrm{d}} \in (\mathbb{R}^{\mathtt{w}})^T$ and $\mathtt{l}_{\max}$.

1: Compute the SVD of $\mathscr{H}_{\mathtt{l}_{\max}+1}(w_{\mathrm{d}}) = U\Sigma V^\top$ and let r be the rank of $\mathscr{H}_{\mathtt{l}_{\max}+1}(w_{\mathrm{d}})$.

2: **if** $r = \mathtt{w}(\mathtt{l}_{\max}+1)$ **then**

3: $R(z) = 0_{1\times \mathtt{w}}$ {the MPUM is the trivial model $(\mathbb{R}^{\mathtt{w}})^T$}.

4: **else**

5: Let $U := \begin{bmatrix} \overset{r}{U_1} & \overset{g}{U_2} \end{bmatrix}$ and define $U_2^\top =: \begin{bmatrix} \tilde{R}_0 & \tilde{R}_1 & \cdots & \tilde{R}_{\mathtt{l}_{\max}} \end{bmatrix}$, where $\tilde{R}_i \in \mathbb{R}^{g\times \mathtt{w}}$.

6: Compute a unimodular matrix $\tilde{U} \in \mathbb{R}^{g\times g}[z]$, such that

$$\tilde{U}(z)\left(\sum_{i=0}^{\mathtt{l}_{\max}} \tilde{R}_i z^i\right) = \begin{bmatrix} R(z) \\ 0 \end{bmatrix}, \qquad \text{where } R \text{ is full row rank.}$$

7: **end if**

Output: $R(z)$—a minimal kernel representation of the MPUM.

Algorithm 8.2 is an indirect algorithm for computation of an input/state/output representation of the MPUM that uses Algorithm 8.1 for computing a kernel representation first. The transition from a kernel representation to an input/state/output representation is a standard one. First, a maximal-degree, full-rank submatrix $P \in \mathbb{R}^{\mathtt{p}\times \mathtt{p}}$ of R is selected and Q is defined as the complementary to P submatrix of R. Then the left matrix fraction description (P, Q) is realized by standard realization algorithms.

Algorithm 8.2 I/S/O representation of the MPUM via a kernel representation `w2r2ss`

Input: $w_{\mathrm{d}} \in (\mathbb{R}^{\mathtt{w}})^T$ and $\mathtt{l}_{\max}$.

1: Compute a minimal kernel representation of the MPUM via Algorithm 8.1.

2: Select a maximal-degree, full-rank submatrix $P \in \mathbb{R}^{\mathtt{p}\times \mathtt{p}}$ of R and let Q be the complementary to P submatrix of R {select an input/output partition of the variables}.

3: Realize (P, Q) via a state space system $\mathscr{B}_{\mathrm{i/s/o}}(A, B, C, D)$.

Output: (A, B, C, D)—a minimal input/state/output representation of the MPUM.

If an input/output partition of the time series w_{d} is a priori given, then step 2 is skipped. For the computation of the transfer function $P^{-1}(z)Q(z)$ of $\mathscr{B}$, matrix polynomial linear operations are needed that are not an integral part of most popular numerical linear algebra packages and libraries such as MATLAB.

Algorithms for Computation of a Convolution Representation

The convolution representation is parameterized by the impulse response. Algorithm 8.7 from Section 8.6 computes the impulse response directly from data. This algorithm is a consequence of the fundamental lemma with the refinement that iteratively sequential pieces of the impulse response are computed.

The impulse response is used in the algorithms for balanced model identification, presented in Chapter 9. Previously proposed algorithms for balanced model identification

compute a Hankel matrix of the Markov parameters and thus recompute most samples of the impulse response many times. The algorithm presented in Section 8.6 avoids this and as a result is more efficient.

Algorithm 8.3 is an indirect algorithm for computation of an input/state/output representation of the MPUM that uses Algorithm 8.7 for computing a convolution representation first. The transition from a convolution representation to an input/state/output representation is a standard problem of realization theory; see Section 8.7.

Algorithm 8.3 I/S/O representation of the MPUM via an impulse response `uy2h2ss`

Input: u_{d}, y_{d}, $\mathtt{n}_{\max}$, and $\mathtt{l}_{\max}$.

1: Compute the first $\mathtt{l}_{\max}+1+\mathtt{n}_{\max}$ samples of the impulse response H of the MPUM via Algorithm 8.7.

2: Compute a realization $\mathscr{B}_{\mathrm{i/s/o}}(A,B,C,D)$ of H via Algorithm 8.8.

Output: (A,B,C,D)—a minimal input/state/output representation of the MPUM.

Algorithms Based on Computation of an Observability Matrix

Let $\mathscr{B}=\mathscr{B}_{\mathrm{i/s/o}}(A,B,C,D)$. If, in addition to $w_{\mathrm{d}}=(u_{\mathrm{d}},y_{\mathrm{d}})$, the extended observability matrix $\mathscr{O}_{\mathtt{l}_{\max}+1}(A,C)$ were known, we could find (A,B,C,D) by solving two linear systems of equations. The first block row of $\mathscr{O}_{\mathtt{l}_{\max}+1}(A,C)$ immediately gives C, and A is computed from the so-called shift equation

$$\big(\sigma^*\mathscr{O}_{\mathtt{l}_{\max}+1}(A,C)\big)A=\big(\sigma\mathscr{O}_{\mathtt{l}_{\max}+1}(A,C)\big).$$

(σ and σ^*, acting on a block matrix, remove, respectively, the first and the last block rows.) Once A and C are known, computing D, B, and the initial condition x_{ini}, under which w_{d} is obtained, is also a linear problem. The system of equations (see (VC))

$$y_{\mathrm{d}}(t)=CA^tx_{\mathrm{ini}}+\sum_{\tau=1}^{t-1}CA^{t-1-\tau}Bu_{\mathrm{d}}(\tau)+D\delta(t+1),\quad \text{for } t=1,\dots,\mathtt{l}_{\max}+1, \tag{8.1}$$

is linear in the unknowns D, B, and x_{ini} and can be solved explicitly by using Kronecker products.

Thus the identification problem boils down to the computation of $\mathscr{O}_{\mathtt{l}_{\max}+1}(A,C)$. Observe that the columns of $\mathscr{O}_{\mathtt{l}_{\max}+1}(A,C)$ are $\mathbf{n}(\mathscr{B})$ linearly independent free responses of $\mathscr{B}$. Moreover, *any* $\mathbf{n}(\mathscr{B})$ linearly independent free responses $y_1,\dots,y_{\mathbf{n}(\mathscr{B})}$ of $\mathscr{B}$, stacked next to each other, determine the extended observability matrix up to a similarity transformation. Let $x_1,\dots,x_{\mathbf{n}(\mathscr{B})}$ be the initial conditions for $y_1,\dots,y_{\mathbf{n}(\mathscr{B})}$. The matrix

$$X_{\mathrm{ini}}:=\begin{bmatrix}x_1 & \cdots & x_{\mathbf{n}(\mathscr{B})}\end{bmatrix}\in\mathbb{R}^{\mathbf{n}(\mathscr{B})\times\mathbf{n}(\mathscr{B})}$$

is full rank because, by assumption, the corresponding responses are linearly independent. Then

$$Y_0:=\begin{bmatrix}y_1 & \cdots & y_{\mathbf{n}(\mathscr{B})}\end{bmatrix}=\mathscr{O}_{\mathtt{l}_{\max}+1}(A,C)X_{\mathrm{ini}},$$

which shows that Y_0 is equivalent to $\mathscr{O}_{\mathtt{l}_{\max}+1}(A,C)$.

We have further reduced the identification problem to the problem of computing $\mathbf{n}(\mathscr{B})$ linearly independent free responses of the MPUM. Under the assumptions of the fundamental lemma, such responses can be computed in the same way as the one used for the computation of the impulse response directly from data. The details are described in Section 8.8.

Since $\mathbf{n}(\mathscr{B})$ is unknown, however, $\mathtt{n}_{\max}$ free responses $y_1, \ldots, y_{\mathtt{n}_{\max}}$ are computed such that the corresponding matrix $Y_0 := \begin{bmatrix} y_1 & \cdots & y_{\mathtt{n}_{\max}} \end{bmatrix}$ has its maximal possible rank $\mathbf{n}(\mathscr{B})$. The matrix Y_0 in this case can be viewed as an extended observability matrix $\mathscr{O}_{\mathtt{l}_{\max}+1}(\tilde{A}, \tilde{C})$ for a nonminimal input/state/output representation of $\mathscr{B}$ with $\tilde{A} \in \mathbb{R}^{\mathtt{n}_{\max} \times \mathtt{n}_{\max}}$ and $\tilde{C} \in \mathbb{R}^{\mathtt{p} \times \mathtt{n}_{\max}}$. In order to find a minimal representation, a rank revealing factorization $Y_0 = \Gamma X_{\text{ini}}$ of Y_0 is computed. The matrix Γ is equal to $\mathscr{O}_{\mathtt{l}_{\max}+1}(A, C)$ up to a similarity transformation. The nonuniqueness of the state space basis in which Γ and X_{ini} are obtained corresponds precisely to the nonuniqueness of the rank revealing factorization.

The procedure outlined above is summarized in Algorithm 8.4. An alternative approach for computing a state sequence directly from data, based on the shift-and-cut map [WR02], is presented in [MWD05].

Algorithm 8.4 I/S/O representation of the MPUM via an observability matrix `uy2o2ss`

Input: $u_{\mathtt{d}}$, $y_{\mathtt{d}}$, $\mathtt{l}_{\max}$, and $\mathtt{n}_{\max}$.

1: Compute $\mathtt{n}_{\max}$, $\mathtt{l}_{\max}+1$ samples long free responses Y_0 of the MPUM via Algorithm 8.9.
2: Compute a rank revealing factorization $Y_0 = \Gamma X_{\text{ini}}$.
3: Solve the linear system of equations $(\sigma^* \Gamma) A = (\sigma \Gamma)$ for A and define C to be the first block entry of Γ.
4: Solve the linear system of equations (8.1) for D, B, and x_{ini}.

Output: (A, B, C, D)—a minimal input/state/output representation of the MPUM.

Algorithms Based on Computation of a State Sequence

If a state sequence $x_{\mathtt{d}}(1), \ldots, x_{\mathtt{d}}(\mathbf{n}(\mathscr{B}) + \mathtt{m} + 1)$ of an input/state/output representation of the MPUM were known, then the parameters (A, B, C, D) could be computed by solving the linear system of equations

$$\begin{bmatrix} x_{\mathtt{d}}(2) & \cdots & x_{\mathtt{d}}(\mathbf{n}(\mathscr{B}) + \mathtt{m} + 1) \\ y_{\mathtt{d}}(1) & \cdots & y_{\mathtt{d}}(\mathbf{n}(\mathscr{B}) + \mathtt{m}) \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x_{\mathtt{d}}(1) & \cdots & x_{\mathtt{d}}(\mathbf{n}(\mathscr{B}) + \mathtt{m}) \\ u_{\mathtt{d}}(1) & \cdots & u_{\mathtt{d}}(\mathbf{n}(\mathscr{B}) + \mathtt{m}) \end{bmatrix}. \quad (8.2)$$

Therefore, the identification problem is reduced to the problem of computing a state sequence of the MPUM. This can be done by computing $\mathbf{n}(\mathscr{B}) + \mathtt{m} + 1$ *sequential* free responses. By "sequential" we mean that the corresponding sequence of initial conditions for the responses is a valid state sequence. Under the conditions of the fundamental lemma, such responses can be computed from data by an algorithm similar to the ones used for the computation of the impulse response and free responses. Since $\mathbf{n}(\mathscr{B})$ is unknown, however, $\mathtt{n}_{\max} + \mathtt{m} + 1$ sequential free responses should be computed. The details are described in Section 8.8.

The procedure outlined above is summarized in Algorithm 8.5.

Algorithm 8.5 I/S/O representation of the MPUM via a state sequence `uy2x2ss`

Input: u_d, y_d, $\mathtt{l}_{\max}$, and $\mathtt{n}_{\max}$.

1: Compute $\mathtt{n}_{\max}$, $\mathtt{l}_{\max}+1$ samples long sequential free responses Y_0 of the MPUM via Algorithm 8.9.
2: Compute a rank revealing factorization $Y_0 = \Gamma X_\mathrm{d}$.
3: Solve the system of equations (8.2) for A, B, C, D, where

$$\begin{bmatrix} x_\mathrm{d}(1) & \cdots & x_\mathrm{d}(\mathtt{n}_{\max}+\mathtt{m}+1) \end{bmatrix} := X_\mathrm{d}.$$

Output: (A, B, C, D)—a minimal input/state/output representation of the MPUM.

8.6 Computation of the Impulse Response from Data

In this section, we consider the following problem:

> Given a trajectory $w_\mathrm{d} = (u_\mathrm{d}, y_\mathrm{d})$ of a system $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$, find the first t samples of the impulse response of $\mathscr{B}$.

Under the conditions of the fundamental lemma, we have that

$$\operatorname{col\,span}\left(\mathscr{H}_t(w_\mathrm{d})\right) = \mathscr{B}|_{[1,t]}.$$

This implies that there exists a matrix G, such that $\mathscr{H}_t(y_\mathrm{d})G = H$. Thus the problem reduces to finding a particular G.

Define U_p, U_f, Y_p, and Y_f as follows:

$$\mathscr{H}_{\mathtt{l}_{\max}+t}(u_\mathrm{d}) =: \begin{bmatrix} U_\mathrm{p} \\ U_\mathrm{f} \end{bmatrix}, \qquad \mathscr{H}_{\mathtt{l}_{\max}+t}(y_\mathrm{d}) =: \begin{bmatrix} Y_\mathrm{p} \\ Y_\mathrm{f} \end{bmatrix}, \tag{8.3}$$

where $\operatorname{row\,dim}(U_\mathrm{p}) = \operatorname{row\,dim}(Y_\mathrm{p}) = \mathtt{l}_{\max}$ and $\operatorname{row\,dim}(U_\mathrm{f}) = \operatorname{row\,dim}(Y_\mathrm{f}) = t$.

Theorem 8.19 (Impulse response from data). *Let $w_\mathrm{d} = (u_\mathrm{d}, y_\mathrm{d})$ be a trajectory of a controllable LTI system $\mathscr{B} \in \mathscr{L}^{\mathtt{w},\mathtt{n}_{\max}}_{\mathtt{m},\mathtt{l}_{\max}}$ and let u_d be persistently exciting of order $t + \mathtt{l}_{\max} + \mathtt{n}_{\max}$. Then the system of equations*

$$\begin{bmatrix} U_\mathrm{p} \\ U_\mathrm{f} \\ Y_\mathrm{p} \end{bmatrix} G = \begin{bmatrix} 0_{\mathtt{m}\mathtt{l}_{\max}\times\mathtt{m}} \\ \begin{bmatrix} I_\mathtt{m} \\ 0_{\mathtt{m}(t-1)\times\mathtt{m}} \end{bmatrix} \\ 0_{\mathtt{p}\mathtt{l}_{\max}\times\mathtt{m}} \end{bmatrix} \tag{8.4}$$

is solvable for $G \in \mathbb{R}^{\bullet\times\mathtt{m}}$. Moreover, for any particular solution $\bar{G}$, the matrix $Y_\mathrm{f}\bar{G}$ contains the first t samples of the impulse response of $\mathscr{B}$, i.e.,

$$Y_\mathrm{f}\bar{G} = H.$$

Proof. Under the assumptions of the theorem, we can apply the fundamental lemma with $L = \mathtt{l}_{\max} + t$. Thus

$$\operatorname{col\,span}\left(\mathscr{H}_{\mathtt{l}_{\max}+t}(w_\mathrm{d})\right) = \mathscr{B}|_{[1,\mathtt{l}_{\max}+t]}.$$

First, we show that (8.4) is solvable. The impulse response $(\begin{bmatrix} I_{\mathtt{m}} \\ 0_{\mathtt{m}(t-1)\times\mathtt{m}} \end{bmatrix}, H)$ is a (matrix valued) response of $\mathscr{B}$ obtained under zero initial conditions. Because of the zero initial conditions, $(\begin{bmatrix} I_{\mathtt{m}} \\ 0_{\mathtt{m}(t-1)\times\mathtt{m}} \end{bmatrix}, H)$ preceded by any number of zeros remains a response of $\mathscr{B}$. Therefore, there exists a matrix $\bar{G}$, such that

$$\begin{bmatrix} U_{\mathrm{p}} \\ U_{\mathrm{f}} \\ Y_{\mathrm{p}} \\ Y_{\mathrm{f}} \end{bmatrix} \bar{G} = \begin{bmatrix} 0_{\mathtt{ml}_{\max}\times\mathtt{m}} \\ \begin{bmatrix} I_{\mathtt{m}} \\ 0_{\mathtt{m}(t-1)\times\mathtt{m}} \end{bmatrix} \\ 0_{\mathtt{pl}_{\max}\times\mathtt{m}} \\ H \end{bmatrix}.$$

This shows that there exists a solution $\bar{G}$ of (8.4) and therefore $Y_{\mathrm{f}}\bar{G}$ is the impulse response.

Conversely, let G be a solution of (8.4). We have

$$\begin{bmatrix} U_{\mathrm{p}} \\ U_{\mathrm{f}} \\ Y_{\mathrm{p}} \\ Y_{\mathrm{f}} \end{bmatrix} G = \begin{bmatrix} 0_{\mathtt{ml}_{\max}\times\mathtt{m}} \\ \begin{bmatrix} I_{\mathtt{m}} \\ 0_{\mathtt{m}(t-1)\times\mathtt{m}} \end{bmatrix} \\ 0_{\mathtt{pl}_{\max}\times\mathtt{m}} \\ Y_{\mathrm{f}}G \end{bmatrix} \tag{8.5}$$

and the fundamental lemma guarantees that the right-hand side of (8.5) is a response of $\mathscr{B}$. The response is identically zero during the first $\mathtt{l}_{\max}$ samples, which (using the assumption $\mathtt{l}_{\max} \geq \mathbf{l}(\mathscr{B})$) guarantees that the initial conditions are set to zero. The input $\begin{bmatrix} I_{\mathtt{m}} \\ 0_{\mathtt{m}(t-1)\times\mathtt{m}} \end{bmatrix}$ is a matrix valued impulse, so that the corresponding output $Y_{\mathrm{f}}G$ is indeed the impulse response H. □

Theorem 8.19 gives the following block algorithm for the computation of H.

Algorithm 8.6 Block computation of the impulse response from data `uy2hblk`

Input: u_{d}, y_{d}, $\mathtt{l}_{\max}$, and t.

1: Solve the system of equations (8.4). Let $\bar{G}$ be the computed solution.

2: Compute $H = Y_{\mathrm{f}}\bar{G}$.

Output: the first t samples of the impulse response H of the MPUM.

Note 8.20 (Efficient implementation via QR factorization) The system of equations (8.4) of step 1 of Algorithm 8.6 can be solved efficiently by first "compressing the data" via the QR factorization

$$\left[\begin{array}{c} U_{\mathrm{p}} \\ U_{\mathrm{f}} \\ Y_{\mathrm{p}} \\ \hline Y_{\mathrm{f}} \end{array}\right]^{\top} = QR, \qquad R^{\top} =: \left[\begin{array}{ccc} R_{11} & 0 & 0 \\ \hline R_{21} & R_{22} & 0 \end{array}\right],$$

where $R_{11} \in \mathbb{R}^{j\times j}$, $j = \mathtt{m}(\mathtt{l}_{\max} + t) + \mathtt{pl}_{\max}$, and then computing the pseudoinverse of the R_{11} block. We have

$$H = Y_{\mathrm{f}} \begin{bmatrix} U_{\mathrm{p}} \\ U_{\mathrm{f}} \\ Y_{\mathrm{p}} \end{bmatrix}^{\dagger} \begin{bmatrix} 0 \\ I \\ 0 \end{bmatrix} = R_{21}R_{11}^{\dagger} \begin{bmatrix} 0 \\ I \\ 0 \end{bmatrix}.$$

We proceed to point out an inherent limitation of Algorithm 8.6 when dealing with finite amount of data. Let a T samples long trajectory be given. The persistency of excitation assumption in Theorem 8.19 requires that $\mathscr{H}_{t+\mathtt{l}_{\max}+\mathtt{n}_{\max}}(u_{\mathrm{d}})$ be full row rank, which implies that

$$\mathtt{m}(t+\mathtt{l}_{\max}+\mathtt{n}_{\max}) \leq T-(t+\mathtt{l}_{\max}+\mathtt{n}_{\max})+1 \quad \Longrightarrow \quad t \leq \frac{T+1}{\mathtt{m}+1}-\mathtt{l}_{\max}-\mathtt{n}_{\max}.$$

Thus, using Algorithm 8.6, we are limited in the number of samples of the impulse response that can be computed. Moreover, for efficiency and accuracy (in the presence of noise), we want to have Hankel matrices U_{p}, U_{f}, etc., with many more columns than rows, which implies small t.

In fact, according to Theorem 8.16, u_{d} persistently exciting of order $1+\mathtt{l}_{\max}+\mathtt{n}_{\max}$ is sufficient for computation of the whole impulse response of the system. Indeed, this can be done by weaving trajectories. (See Figure 8.2 for an illustration.)

Lemma 8.21 (Weaving trajectories). *Consider a system $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}$ and let*

1. *$w_{\mathrm{d},1}$ be a T_1 samples long trajectory of $\mathscr{B}$, i.e., $w_{\mathrm{d},1} \in \mathscr{B}|_{[1,T_1]}$;*
2. *$w_{\mathrm{d},2}$ be a T_2 samples long trajectory of $\mathscr{B}$, i.e., $w_{\mathrm{d},2} \in \mathscr{B}|_{[1,T_2]}$; and*
3. *the last $\mathtt{l}_{\max}$ samples, where $\mathtt{l}_{\max} \geq \mathtt{l}(\mathscr{B})$, of $w_{\mathrm{d},1}$ coincide with the first $\mathtt{l}_{\max}$ samples of $w_{\mathrm{d},2}$, i.e.,*

$$\big(w_{\mathrm{d},1}(T_1-\mathtt{l}_{\max}+1),\dots,w_{\mathrm{d},1}(T_1)\big) = \big(w_{\mathrm{d},2}(1),\dots,w_{\mathrm{d},2}(\mathtt{l}_{\max})\big).$$

Then the trajectory

$$w := \big(w_{\mathrm{d},1}(1),\dots,w_{\mathrm{d},1}(T_1),w_{\mathrm{d},2}(\mathtt{l}_{\max}+1),\dots,w_{\mathrm{d},2}(T_2)\big) \tag{8.6}$$

obtained by weaving together $w_{\mathrm{d},1}$ and $w_{\mathrm{d},2}$ is a trajectory of $\mathscr{B}$, i.e., $w \in \mathscr{B}|_{[1,T_1+T_2-\mathtt{l}_{\max}]}$.

Proof. Let $x_{\mathrm{d},1} := \big(x_{\mathrm{d},1}(1),\dots,x_{\mathrm{d},1}(T_1+1)\big)$ and $x_{\mathrm{d},2} := \big(x_{\mathrm{d},2}(1),\dots,x_{\mathrm{d},2}(T_2+1)\big)$ be state sequences of $\mathscr{B}$ associated with $w_{\mathrm{d},1}$ and $w_{\mathrm{d},2}$, respectively. Assumption 3 implies that $x_{\mathrm{d},1}(T_1+1) = x_{\mathrm{d},2}(\mathtt{l}_{\max}+1)$. Therefore, (8.6) is a trajectory of $\mathscr{B}$. □

Algorithm 8.7 overcomes the above-mentioned limitation of the block algorithm by iteratively computing blocks of L consecutive samples, where

$$1 \leq L \leq \frac{T+1}{\mathtt{m}+1}-\mathtt{l}_{\max}-\mathtt{n}_{\max}. \tag{8.7}$$

Moreover, monitoring the decay of H (provided the system is stable) while computing it gives a heuristic way to determine a value for t that is sufficiently large to show the transient.

In the recursive algorithm, the matrices U_{p}, U_{f}, Y_{p}, and Y_{f} defined above are redefined as follows:

$$\mathscr{H}_{\mathtt{l}_{\max}+L}(u_{\mathrm{d}}) =: \begin{bmatrix} U_{\mathrm{p}} \\ U_{\mathrm{f}} \end{bmatrix}, \qquad \mathscr{H}_{\mathtt{l}_{\max}+L}(y_{\mathrm{d}}) =: \begin{bmatrix} Y_{\mathrm{p}} \\ Y_{\mathrm{f}} \end{bmatrix},$$

where $\operatorname{row\,dim}(U_{\mathrm{p}}) = \operatorname{row\,dim}(Y_{\mathrm{p}}) = \mathtt{l}_{\max}$ and $\operatorname{row\,dim}(U_{\mathrm{f}}) = \operatorname{row\,dim}(Y_{\mathrm{f}}) = L$.

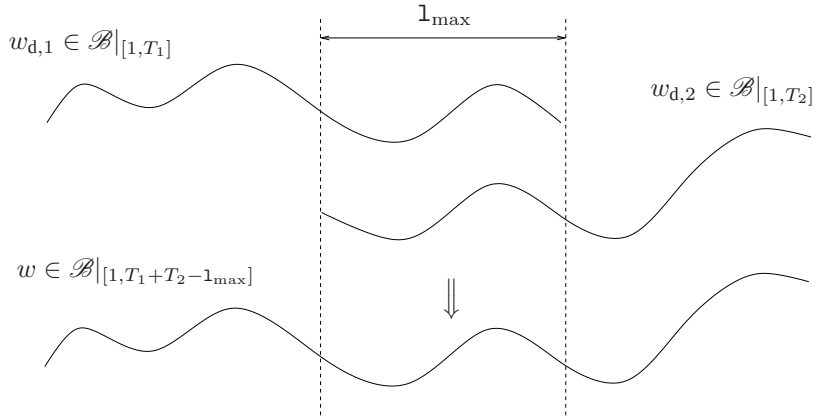


Figure 8.2. *Weaving trajectories.*

Algorithm 8.7 Iterative computation of the impulse response from data uy2h

Input: u_{d}, y_{d}, $\mathtt{n}_{\max}$, $\mathtt{l}_{\max}$, and either t or a convergence tolerance ε.

1: Choose the number of samples L computed in one iteration step according to (8.7).

2: Initialization: $k := 0$, $F_{\mathrm{u}}^{(0)} := \begin{bmatrix} 0_{\mathtt{m}\mathtt{l}_{\max} \times \mathtt{m}} \\ \begin{bmatrix} I_{\mathtt{m}} \\ 0_{\mathtt{m}(L-1)\times \mathtt{m}} \end{bmatrix} \end{bmatrix}$, and $F_{\mathrm{y,p}}^{(0)} := 0_{\mathtt{p}\mathtt{l}_{\max}}$.

3: **repeat**

4: Solve the system $\begin{bmatrix} U_{\mathrm{p}} \\ U_{\mathrm{f}} \\ Y_{\mathrm{p}} \end{bmatrix} G^{(k)} = \begin{bmatrix} F_{\mathrm{u}}^{(k)} \\ F_{\mathrm{y,p}}^{(k)} \end{bmatrix}$.

5: Compute the response $H^{(k)} := F_{\mathrm{y,f}}^{(k)} := Y_{\mathrm{f}} G^{(k)}$.

6: Define $F_{\mathrm{y}}^{(k)} := \begin{bmatrix} F_{\mathrm{y,p}}^{(k)} \\ F_{\mathrm{y,f}}^{(k)} \end{bmatrix}$.

7: Shift F_{u} and F_{y}: $F_{\mathrm{u}}^{(k+1)} := \begin{bmatrix} \sigma^L F_{\mathrm{u}}^{(k)} \\ 0_{\mathtt{m}L\times \mathtt{m}} \end{bmatrix}$, $F_{\mathrm{y,p}}^{(k+1)} := \sigma^L F_{\mathrm{y}}^{(k)}$.

8: Increment the iteration counter $k := k + 1$.

9: **until** $\begin{cases} kL < t & \text{if } t \text{ is given,} \\ \|H^{(k-1)}\|_{\mathrm{F}} \leq \varepsilon & \text{otherwise.} \end{cases}$

Output: $H = \operatorname{col}\left(H^{(0)}, \ldots, H^{(k-1)}\right)$.

Proposition 8.22. *Let $w_{\mathrm{d}} = (u_{\mathrm{d}}, y_{\mathrm{d}})$ be a trajectory of a controllable LTI system $\mathscr{B}$ of order $\mathbf{n}(\mathscr{B}) \leq \mathtt{n}_{\max}$ and lag $\mathbf{l}(\mathscr{B}) \leq \mathtt{l}_{\max}$, and let u_{d} be persistently exciting of order $L + \mathtt{l}_{\max} + \mathtt{n}_{\max}$. Then Algorithm 8.7 computes the first t samples of the impulse response of $\mathscr{B}$.*

Proof. Under the assumptions of the proposition, we can apply Theorem 8.19, with the parameter t replaced by the parameter L, selected in step 1 of the algorithm (Algorithm 8.6). Steps 4 and 5 of the recursive algorithm correspond to steps 1 and 2 of the block algorithm.

The right-hand side $\begin{bmatrix} F_{\mathrm{u}}^{(k)} \\ F_{\mathrm{y,p}}^{(k)} \end{bmatrix}$ of the system of equations, solved in step 4, is initialized so that $H^{(0)}$ is indeed the matrix of the first L samples of the impulse response.

The response computed on the $(k+1)$st iteration step, $k \geq 1$, is a response due to zero input and its first $\mathtt{l}_{\max}$ samples overlap the last $\mathtt{l}_{\max}$ samples of the response computed on the kth iteration step. By the weaving lemma (Lemma 8.21), their concatenation is a valid response. Applying this argument recursively, we have that H computed by the algorithm is the impulse response of the system. □

With $L = 1$, the persistency of excitation condition required by Proposition 8.22 is $\mathtt{l}_{\max} + 1 + \mathtt{n}_{\max}$, which is the identifiability condition of Theorem 8.16 (with the unknown lag $\mathbf{l}(\mathscr{B})$ and order $\mathbf{n}(\mathscr{B})$ replaced by their given upper bounds $\mathtt{l}_{\max}$ and $\mathtt{n}_{\max}$).

Note 8.23 (Data driven simulation) In [MWRM05], Theorem 8.19 and Algorithms 8.6 and 8.7 are modified to compute an arbitrary response directly from data. This procedure is called data driven simulation and is shown to be related to deterministic subspace identification algorithms.

Note 8.24 (Efficient implementation via QR factorization) The most expensive part of Algorithm 8.7 is solving the system of equations in step 4. It can be solved efficiently via the QR factorization, as described in Note 8.20. Moreover, since the matrix on the left-hand side of the system is fixed, the pseudo-inverse can be computed outside the iteration loop and used for all iterations.

8.7 Realization Theory and Algorithms

The problem of passing from an impulse response to another representation (typically input/state/output or transfer function) is called realization. Given a sequence $H : \mathbb{N} \to \mathbb{R}^{\mathtt{p}\times\mathtt{m}}$, we say that a system $\mathscr{B} \in \mathscr{L}_{\mathtt{m}}^{\mathtt{w}}$, $\mathtt{w} := \mathtt{m}+\mathtt{p}$, realizes H if $\mathscr{B}$ has a convolution representation with an impulse response H. In this case, we say that H is realizable (by a system in the model class $\mathscr{L}_{\mathtt{m}}^{\mathtt{w}}$). A sequence H might not be realizable by a *finite dimensional* LTI system, but if it is realizable, the realization is unique.

Theorem 8.25 (Test for realizability). *The sequence $H : \mathbb{N} \to \mathbb{R}^{\mathtt{p}\times\mathtt{m}}$ is realizable by a finite dimensional LTI system with $\mathtt{m}$ inputs if and only if the two-sided infinite Hankel matrix $\mathscr{H}(\sigma H)$ has a finite rank. Moreover, if the rank of $\mathscr{H}(\sigma H)$ is $\mathtt{n}$, then there is a unique system $\mathscr{B} \in \mathscr{L}_{\mathtt{m}}^{\mathtt{w},\mathtt{n}}$ that realizes H.*

Let H be realizable by a system $\mathscr{B} \in \mathscr{L}_{\mathtt{m}}^{\mathtt{w}}$ with an input/state/output representation $\mathscr{B} = \mathscr{B}_{\text{i/s/o}}(A, B, C, D)$. We have that

$$\mathscr{H}_{i,j}(\sigma H) = \mathscr{O}_i(A, C)\mathscr{C}_j(A, B),$$

and from the properties of the controllability and observability matrices, it follows that

$$\operatorname{rank}\big(\mathscr{H}_{i,j}(\sigma H)\big) = \begin{cases} \min(\mathtt{p}i, \mathtt{m}j) & \text{for all } i < \mu(\mathscr{B}) \text{ and } j < \nu(\mathscr{B}), \\ \mathbf{n}(\mathscr{B}) & \text{otherwise.} \end{cases}$$

Therefore, if we know that H is an impulse response of a finite dimensional LTI system $\mathscr{B}$ of order $\mathbf{n}(\mathscr{B}) \leq \mathtt{n}_{\max}$ and lag $\mathbf{l}(\mathscr{B}) \leq \mathtt{l}_{\max}$, where $\mathtt{n}_{\max}$ and $\mathtt{l}_{\max}$ are given, we can find $\mathbf{n}(\mathscr{B})$ by a rank computation as follows:

$$\mathbf{n}(\mathscr{B}) = \operatorname{rank}\big(\mathscr{H}_{\mathtt{l}_{\max}+1,\mathtt{n}_{\max}}(\sigma H)\big).$$

This fact is often used in subspace identification. Moreover, the SVD $\mathscr{H}_{t,t}(\sigma H) = U\Sigma V^{\top}$, $t > \mathtt{n}_{\max}$, allows us to find a finite time t balanced approximation of the MPUM, so that the numerical rank computation of the block-Hankel matrix of the Markov parameters is a good heuristic for approximate identification.

Note 8.26 (Realization and exact identification) Clearly, realization is a special exact identification problem. Realization of $H : \mathbb{N} \to \mathbb{R}^{\mathtt{p}\times\mathtt{m}}$ is equivalent to exact identification of the time series

$$\begin{aligned} w_{\mathtt{d},1} = (u_{\mathtt{d},1}, y_{\mathtt{d},1}) &:= \big(\operatorname{col}(0, \delta e_1), \operatorname{col}(0, h_1)\big), \\ &\cdots \\ w_{\mathtt{d},\mathtt{m}} = (u_{\mathtt{d},\mathtt{m}}, y_{\mathtt{d},\mathtt{m}}) &:= \big(\operatorname{col}(0, \delta e_{\mathtt{m}}), \operatorname{col}(0, h_{\mathtt{m}})\big), \end{aligned}$$

where $\begin{bmatrix} h_1 & \cdots & h_{\mathtt{m}} \end{bmatrix} := H$, δ is the Kronecker delta function, $\begin{bmatrix} e_1 & \cdots & e_{\mathtt{m}} \end{bmatrix} := I_{\mathtt{m}}$, and the zero prefix is $\mathtt{l}_{\max}$ samples long. (The zero prefix fixes the initial conditions to be zero, which otherwise are free in the exact identification problem.) Special purpose realization methods, however, are more efficient than a general exact identification algorithm.

Note 8.27 (Realization and exact identification of an autonomous system) An alternative point of view of realization is as an exact identification of an autonomous system: realization of $H : \mathbb{N} \to \mathbb{R}^{\mathtt{p}\times\mathtt{m}}$ is equivalent to exact identification of the time series

$$w_{\mathtt{d},1} = (u_{\mathtt{d},1}, y_{\mathtt{d},1}) := (0, \sigma h_1), \quad \ldots \quad , w_{\mathtt{d},\mathtt{m}} = (u_{\mathtt{d},\mathtt{m}}, y_{\mathtt{d},\mathtt{m}}) := (0, \sigma h_{\mathtt{m}}).$$

Consider the impulse response H of the system

$$\mathscr{B}_{\mathtt{i/s/o}}\left(A, \begin{bmatrix} b_1 & \cdots & b_{\mathtt{m}} \end{bmatrix}, C, \bullet\right)$$

and the responses $y_1, \ldots, y_{\mathtt{m}}$ of the autonomous system $\mathscr{B}_{\mathtt{i/s/o}}(A, C)$ due to the initial conditions $b_1, \ldots, b_{\mathtt{m}}$. It is easy to verify that

$$\sigma H = \begin{bmatrix} y_1 & \cdots & y_{\mathtt{m}} \end{bmatrix}.$$

Thus, with an obvious substitution,

> realization algorithms can be used for exact identification of an autonomous system and vice versa; algorithms for identification of an autonomous systems can be used for realization.

Once we know from Theorem 8.25 or from prior knowledge that a given time series $H := \big(H(0), H(1), \ldots, H(T)\big)$ is realizable in the model class $\mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}_{\max}}$, we can proceed with the problem of finding a representation of the system that realizes H. General exact identification problems can be used but in the special case at hand there are more efficient alternatives. Algorithm 8.8 is a typical realization algorithm.

Algorithm 8.8 Realization algorithm `h2ss`

Input: H and $\mathtt{l}_{\max}$ satisfying the conditions of Theorem 8.25.

1: Compute a rank revealing factorization of the Hankel matrix $\mathscr{H}_{\mathtt{l}_{\max}+1}(\sigma H) = \Gamma\Delta$.
2: Let $D = H(0)$, C be the first block row of Γ, and B be the first block column of Δ.
3: Solve the shift equation $(\sigma^*\Gamma)A = \sigma\Gamma$.

Output: parameters (A, B, C, D) of a minimal input/state/output realization of H.

The key computational step is the rank revealing factorization of the Hankel matrix $\mathscr{H}_{\mathtt{l}_{\max}+1}(H)$. Moreover, this step determines the state basis in which the parameters A, B, C, D are computed. In case of finite precision arithmetic, it is well known that rank computation is a nontrivial problem. The rank revealing factorization is crucial for the outcome of the algorithm because the rank of $\mathscr{H}_{\mathtt{l}_{\max}+1}(H)$ is the order of the realization.

When the given time series H is not realizable by an LTI system of order $\mathtt{n}_{\max} := \mathtt{pl}_{\max}$, i.e., $\mathscr{H}_{\mathtt{l}_{\max}+1}(\sigma H)$ is full rank, the SVD offers a possibility to find approximate realization in the model class $\mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}_{\max}}$; see also Note 8.18 on page 122. Replace the rank revealing factorization in step 1 of Algorithm 8.8 by the SVD $\mathscr{H}_{\mathtt{l}_{\max}+1}(H) = U\Sigma V^\top$ and the definitions $\Gamma := U\sqrt{\Sigma}$ and $\Delta := \sqrt{\Sigma}V^\top$. This can be viewed as computation of an "approximate rank revealing factorization". Note that in this case the finite time controllability and observability gramians are equal,

$$\Gamma^\top\Gamma = \Delta\Delta^\top = \Sigma,$$

so that the computed realization $\mathscr{B}_{\mathtt{i/s/o}}(A, B, C, D)$ is in a finite time $\mathtt{l}_{\max}$ balanced form. Algorithm 8.8 with the above modification is Kung's algorithm [Kun78].

8.8 Computation of Free Responses

In this section, we consider the following problem:

Given $w_{\mathrm{d}} = (u_{\mathrm{d}}, y_{\mathrm{d}}) \in \mathscr{B}$, find (sequential) free responses Y_0 of $\mathscr{B}$.

By "sequential", we mean that the initial conditions corresponding to the columns of Y_0 form a valid state sequence of $\mathscr{B}$.

First, we consider computation of general free responses. Using the fundamental lemma, a set of t samples long free responses can be computed from data as follows:

$$\begin{bmatrix} \mathscr{H}_t(u_{\mathrm{d}}) \\ \mathscr{H}_t(y_{\mathrm{d}}) \end{bmatrix} G = \begin{bmatrix} 0 \\ Y_0 \end{bmatrix}. \tag{8.8}$$

Therefore, for any G that satisfies $\mathscr{H}_t(u_{\mathrm{d}})G = 0$, the columns of $Y_0 := \mathscr{H}_t(y_{\mathrm{d}})G$ are free responses. The columns of G are vectors in the null space of $\mathscr{H}_t(u_{\mathrm{d}})$ and can be computed explicitly; however, in general, $\mathrm{rank}(Y_0) \leq \mathbf{n}(\mathscr{B})$. The condition $\mathrm{rank}(Y_0) = \mathbf{n}(\mathscr{B})$ is needed for identification of an input/state/output representation of the MPUM, as outlined in Algorithm 8.3.

In order to ensure the rank condition, we use the splitting of the data into "past" and "future" as defined in (8.3). The blocks in the past allow us to restrict the matrix G, so that

the initial conditions X_{ini} under which the responses Y_0 are generated satisfy $\operatorname{rank}(X_{\text{ini}}) = \mathbf{n}(\mathscr{B})$. This implies $\operatorname{rank}(Y_0) = \mathbf{n}(\mathscr{B})$. In turns out, however, that in choosing the initial conditions X_{ini}, we can furthermore produce sequential free responses.

Using the fundamental lemma, we know that the right-hand side of the equation

$$\begin{bmatrix} U_{\text{p}} \\ U_{\text{f}} \\ Y_{\text{p}} \\ Y_{\text{f}} \end{bmatrix} G = \begin{bmatrix} U_{\text{p}} \\ 0 \\ Y_{\text{p}} \\ Y_0 \end{bmatrix}$$

is a trajectory. Therefore, a set of free responses can be computed from data by solving the system of equations

$$\begin{bmatrix} U_{\text{p}} \\ U_{\text{f}} \\ Y_{\text{p}} \end{bmatrix} G = \begin{bmatrix} U_{\text{p}} \\ 0 \\ Y_{\text{p}} \end{bmatrix} \tag{8.9}$$

and setting $Y_0 = Y_{\text{f}} G$. Moreover, the Hankel structure of U_{p} and Y_{p} imply that Y_0 is a matrix of sequential responses. System (8.9) and $Y_0 = Y_{\text{f}} G$ give a block algorithm for the computation of sequential free responses. It is analogous to the block algorithm for the computation of the impulse response and again the computation can be performed efficiently via the QR factorization.

We proceed to present a recursive algorithm for the computation of Y_0, analogous to Algorithm 8.7 for the computation of the impulse response. An advantage of the recursive algorithm over the block one is that one is not restricted by the finite amount of data w_{d} to a finite length responses Y_0.

Proposition 8.28. *Under the assumptions of Proposition 8.22, Algorithm 8.9 computes a matrix of sequential free responses of $\mathscr{B}$ with t block rows.*

Proof. This is similar to the proof of Proposition 8.22. □

8.9 Relation to Subspace Identification Methods*

MOESP-Type Algorithms

The multivariable output error state space (MOESP)-type subspace identification algorithms correspond to the algorithm based on the computation of free responses as outlined in Section 8.5, Algorithm 8.4. However, in the MOESP algorithms, step 1—the computation of free responses—is implemented via the *orthogonal projection*

$$Y_0 := \mathscr{H}_{1_{\max}+1}(y_{\text{d}}) \underbrace{\Big(I - \mathscr{H}_{1_{\max}+1}^{\top}(u_{\text{d}}) \big(\mathscr{H}_{1_{\max}+1}(u_{\text{d}}) \mathscr{H}_{1_{\max}+1}^{\top}(u_{\text{d}}) \big)^{-1} \mathscr{H}_{1_{\max}+1}(u_{\text{d}}) \Big)}_{\Pi_{u_{\text{d}}}^{\perp}}; \tag{8.10}$$

i.e., the MOESP algorithms compute the orthogonal projection of the rows of $\mathscr{H}_{1_{\max}+1}(y_{\text{d}})$ on the orthogonal complement of the row space of $\mathscr{H}_{1_{\max}+1}(u_{\text{d}})$. In subspace identification it is customary to think in terms of geometric operations: projection of the rows of a certain

Algorithm 8.9 Iterative computation of sequential free responses `uy2y0`

Input: u_d, y_d, $\mathtt{n}_{\max}$, $\mathtt{l}_{\max}$, and either the desired number of samples t or a convergence tolerance ε.

1: Choose the number of samples L computed in one iteration step according to (8.7).

2: Initialization: $k := 0$, $F_\mathrm{u}^{(0)} := \begin{bmatrix} U_\mathrm{p} \\ 0 \end{bmatrix}$, and $F_\mathrm{y,p}^{(0)} := Y_\mathrm{p}$.

3: **repeat**

4: Solve the system $\begin{bmatrix} U_\mathrm{p} \\ U_\mathrm{f} \\ Y_\mathrm{p} \end{bmatrix} G^{(k)} = \begin{bmatrix} F_\mathrm{u}^{(k)} \\ F_\mathrm{y,p}^{(k)} \end{bmatrix}$.

5: Compute the response $Y_0^{(k)} := F_\mathrm{y,f}^{(k)} := Y_\mathrm{f} G^{(k)}$.

6: Define $F_\mathrm{y}^{(k)} := \begin{bmatrix} F_\mathrm{y,p}^{(k)} \\ F_\mathrm{y,f}^{(k)} \end{bmatrix}$.

7: Shift F_u and F_y: $F_\mathrm{u}^{(k+1)} := \begin{bmatrix} \sigma^L F_\mathrm{u}^{(k)} \\ 0_{\mathtt{m}L \times \mathtt{m}} \end{bmatrix}$ and $F_\mathrm{y,p}^{(k+1)} := \sigma^L F_\mathrm{y}^{(k)}$.

8: Increment the iteration counter $k := k + 1$.

9: **until** $\begin{cases} kL < t & \text{if } t \text{ is given,} \\ \|Y_0^{(k-1)}\|_\mathrm{F} \le \varepsilon & \text{otherwise.} \end{cases}$

Output: $Y_0 = \operatorname{col}\left(Y_0^{(0)}, \ldots, Y_0^{(k-1)}\right)$.

matrix onto the row space of another matrix. The fact that these matrices have special (block-Hankel) structure is ignored and the link with system theory is lost. Still, as we show next,

> the orthogonal projection (8.10) has the simple and useful system theoretic interpretation of computing a maximal number of free responses.

Observe that

$$\begin{bmatrix} \mathscr{H}_{\mathtt{l}_{\max}+1}(u_\mathrm{d}) \\ \mathscr{H}_{\mathtt{l}_{\max}+1}(y_\mathrm{d}) \end{bmatrix} \Pi_{u_\mathrm{d}}^{\perp} = \begin{bmatrix} 0 \\ Y_0 \end{bmatrix},$$

which corresponds to (8.8) except that now the projector $\Pi_{u_\mathrm{d}}^{\perp}$ is a square matrix, while in (8.8) G is in general a rectangular matrix. In [VD92, Section 3.3], it is shown that a sufficient condition for $\operatorname{rank}(Y_0) = \mathbf{n}(\mathscr{B})$ is

$$\operatorname{rank}\left(\begin{bmatrix} X_\mathrm{ini} \\ \mathscr{H}_{\mathtt{l}_{\max}+1}(u_\mathrm{d}) \end{bmatrix}\right) = \mathbf{n}(\mathscr{B}) + (\mathtt{l}_{\max}+1)\mathtt{m}. \tag{8.11}$$

This condition, however, is not verifiable from the data $w_\mathrm{d} = (u_\mathrm{d}, y_\mathrm{d})$. Therefore, given w_d, one cannot check in general whether the data generating system $\mathscr{B}$ is identifiable by the MOESP algorithms. Under the identifiability condition

> u_d persistently exciting of order $\mathtt{l}_{\max} + 1 + \mathtt{n}_{\max}$,

which is verifiable from the data, Corollary 8.17 implies (8.11).

Finally, note that the $j = T - \mathtt{l}_{\max}$ free responses that the orthogonal projection (8.10) computes are typically more than necessary for exact identification, i.e., $j \gg \mathbf{n}(\mathscr{B})$. Therefore, in general, the orthogonal projection is a computationally inefficient operation for exact identification. This deficiency of the MOESP algorithms is partially corrected on the level of the numerical implementation. First, the QR factorization

$$\begin{bmatrix} \mathscr{H}_{\mathtt{n}_{\max}}(u_{\mathrm{d}}) \\ \mathscr{H}_{\mathtt{n}_{\max}}(y_{\mathrm{d}}) \end{bmatrix}^{\top} = QR$$

is computed and then only the block entry R_{22} of the R factor is used, where

$$R^{\top} =: \begin{array}{c} \begin{matrix} \mathtt{n}_{\max}\mathtt{m} & \mathtt{n}_{\max}\mathtt{p} & \end{matrix} \\ \begin{bmatrix} R_{11} & 0 & 0 \\ R_{21} & R_{22} & 0 \end{bmatrix} \end{array} \begin{matrix} \\ \mathtt{n}_{\max}\mathtt{m} \\ \mathtt{n}_{\max}\mathtt{p} \end{matrix}.$$

It can be shown (see [VD92, Section 4.1]) that

$$\operatorname{col\,span}(Y_0) = \operatorname{col\,span}(R_{22}).$$

The column dimension of R_{22} is $\mathtt{n}_{\max}\mathtt{p}$, which is (typically) comparable with $\mathtt{n}_{\max}$ and is (typically) much smaller than j.

N4SID-Type Algorithms

The numerical algorithms for subspace state space system identification (N4SID) correspond to the algorithm based on the computation of a state sequence as outlined in Section 8.5, Algorithm 8.5. However, in the N4SID-type algorithms, step 1—the computation of sequential free responses—is implemented via the *oblique projection*. Consider the splitting of the data into "past" and "future",

$$\mathscr{H}_{2(\mathtt{l}_{\max}+1)}(u_{\mathrm{d}}) =: \begin{bmatrix} U_{\mathrm{p}} \\ U_{\mathrm{f}} \end{bmatrix}, \qquad \mathscr{H}_{2(\mathtt{l}_{\max}+1)}(y_{\mathrm{d}}) =: \begin{bmatrix} Y_{\mathrm{p}} \\ Y_{\mathrm{f}} \end{bmatrix}, \tag{8.12}$$

with $\operatorname{row\,dim}(U_{\mathrm{p}}) = \operatorname{row\,dim}(U_{\mathrm{f}}) = \operatorname{row\,dim}(Y_{\mathrm{p}}) = \operatorname{row\,dim}(Y_{\mathrm{f}}) = \mathtt{l}_{\max} + 1$, and let

$$W_{\mathrm{p}} := \begin{bmatrix} U_{\mathrm{p}} \\ Y_{\mathrm{p}} \end{bmatrix}.$$

As the key computational step of the MOESP algorithms is the orthogonal projection, the key computational step of the N4SID algorithms is the oblique projection of Y_{f} along the space spanned by the rows of U_{f} onto the space spanned by the rows of W_{p}. This geometric operation, denoted by $Y_{\mathrm{f}}/_{U_{\mathrm{f}}}W_{\mathrm{p}}$, is defined as follows (see [VD96, equation (1.4), page 21]):

$$Y_0 := Y_{\mathrm{f}}/_{U_{\mathrm{f}}}W_{\mathrm{p}} := Y_{\mathrm{f}} \underbrace{\begin{bmatrix} W_{\mathrm{p}}^{\top} & U_{\mathrm{f}}{}^{\top} \end{bmatrix} \begin{bmatrix} W_{\mathrm{p}}W_{\mathrm{p}}^{\top} & W_{\mathrm{p}}U_{\mathrm{f}}^{\top} \\ U_{\mathrm{f}}W_{\mathrm{p}}^{\top} & U_{\mathrm{f}}U_{\mathrm{f}}{}^{\top} \end{bmatrix}^{+} \begin{bmatrix} W_{\mathrm{p}} \\ 0 \end{bmatrix}}_{\Pi_{\mathrm{obl}}}. \tag{8.13}$$

Next, we show that

the oblique projection computes sequential free responses of the system.

Note that

$$\begin{bmatrix} W_{\mathrm{p}} \\ U_{\mathrm{f}} \\ Y_{\mathrm{f}} \end{bmatrix} \Pi_{\text{obl}} = \begin{bmatrix} W_{\mathrm{p}} \\ 0 \\ Y_0 \end{bmatrix}$$

corresponds to (8.9) except that the oblique projector Π_{obl} is a square matrix, while in (8.9), G is in general rectangular. Therefore, the columns of the oblique projection Y_0 given in (8.13) are $j := T - 2\mathtt{l}_{\max} - 1$ sequential free responses. However, as with the orthogonal projection, the oblique projection also computes in general more responses than the $\mathtt{n}_{\max} + \mathtt{m} + 2$ ones needed for applying Algorithm 8.5.

In [VD96, Section 2, Theorem 2], it is (implicitly) proven that a sufficient condition for $\operatorname{rank}(X_{\mathrm{d}}) = \mathbf{n}(\mathscr{B})$, which is needed for the exact identification Algorithm 8.5, is

1. u_{d} persistently exciting of order $2\mathtt{n}_{\max}$ and
2. $\operatorname{row\,span}(X_{\mathrm{d}}) \cap \operatorname{row\,span}(U_{\mathrm{f}}) = \{0\}$;

see assumptions 1 and 2 of [VD96, Section 2, Theorem 2]. As with assumption (8.11) in the MOESP algorithms, however, assumption 2 is again not verifiable from the given data. Persistency of excitation of u_{d} of order $2(\mathtt{l}_{\max} + 1) + \mathbf{n}(\mathscr{B})$ (i.e., the assumption of the fundamental lemma) is a sufficient condition, verifiable from the data $(u_{\mathrm{d}}, y_{\mathrm{d}})$, for assumptions 1 and 2 of [VD96, Section 2, Theorem 2].

8.10 Simulation Examples

Impulse Response Computation

First we consider the problem of computing the first t samples of the impulse response H of a system $\mathscr{B}$ from data $w_{\mathrm{d}} := (u_{\mathrm{d}}, y_{\mathrm{d}})$. We choose a random stable system $\mathscr{B}$ of order $\mathtt{n} = 4$ with $\mathtt{m} = 2$ inputs and $\mathtt{p} = 2$ outputs. The data w_{d} is obtained according to the EIV model $w_{\mathrm{d}} = \bar{w} + \tilde{w}$, where $\bar{w} := (\bar{u}, \bar{y}) \in \mathscr{B}|_{[1,T]}$ with $T = 500$, $\bar{u}$ is zero mean unit variance white Gaussian noise, and $\tilde{w}$ is a zero mean white Gaussian noise with variance σ^2. Varying σ, we study empirically the effect of random perturbation on the results.

We apply Algorithm 8.6 with $t = 27$, $\mathtt{n}_{\max} = \mathtt{n}$, and $\mathtt{l}_{\max} = \lceil \mathtt{n}_{\max}/\mathtt{p} \rceil$. The computed impulse response is denoted by $\hat{H}$ and is compared with the "true" impulse response H obtained from $\mathscr{B}$ by simulation. The comparison is in terms of the Frobenius norm $e = ||H - \hat{H}||_{\mathrm{F}}$ of the approximation error $H - \hat{H}$. We also apply Algorithm 8.7 with parameters $\mathtt{n}_{\max} = \mathtt{n}$, $\mathtt{l}_{\max} = \lceil \mathtt{n}_{\max}/\mathtt{p} \rceil$, $L = 12$, and the function `impulse` from the System Identification Toolbox of MATLAB that estimates impulse response from data.

Table 8.1 shows the approximation errors e and execution times for four different noise levels and for the three compared algorithms. (The efficiency is measured by the execution time and not by the floating point operations (flops) because the function `impulse` is available only in the latter versions of MATLAB that do not support flop counts.)

In the absence of noise, both Algorithm 8.6 and Algorithm 8.7 compute up to numerical errors exactly the impulse response H, while the same is not true for the function `impulse`. The simulation results show that the iterative algorithm is faster than the block algorithm.

Table 8.1. *Error of approximation* $e = ||H - \hat{H}||_{\mathrm{F}}$ *and execution time in seconds for Algorithm 8.6, Algorithm 8.7 with* $L = 12$*, and the function* `impulse`.

	$\sigma = 0.0$		$\sigma = 0.01$		$\sigma = 0.05$		$\sigma = 0.1$	
Method	e	time, s	e	time, s	e	time, s	e	time, s
Algorithm 8.6	10^{-14}	0.293	0.029	0.277	0.096	0.285	0.251	0.279
Algorithm 8.7	10^{-14}	0.066	0.023	0.086	0.066	0.068	0.201	0.087
`impulse`	0.059	0.584	0.067	0.546	0.109	0.573	0.249	0.558

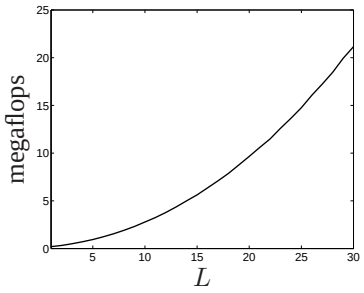


Figure 8.3. *Number of flops as a function of the parameter* L.

Also, when the given data w_{d} is noisy, the iterative algorithm outperforms the block algorithm and the function `impulse`.

Next, we show the effect of the parameter L on the number of flops and the error of approximation e. The plot in Figure 8.3 shows the number of flops, measured in megaflops, as a function of L. The function is monotonically increasing, so that most efficient is the computation for $L = 1$. The plots in Figure 8.4 show the approximation error e as a function of L for four different noise levels. The results are averaged for 100 noise realizations. The function $e(t)$ is complicated and is likely to depend on many factors. The graphs, however, indicate that in the presence of noise, there is a trade-off between computational efficiency and approximation error. For small L the computational cost is small, but the error e tends to be large.

Comparison of Exact Identification Algorithms

We compare the numerical efficiency of the following algorithms for deterministic identification:

`uy2ssmr` Algorithm of Moonen and Ramos [MR93]; see Algorithm 9.6;

`uy2ssvd` Algorithm of Van Overschee and De Moor [VD96]; see Algorithm 9.5;

"Deterministic algorithm 1" of Section 2.4.1 in [VD96] is combined with the choice of the weight matrices W_1 and W_2 given in Theorem 13, Section 5.4.1. Our implementation, however, differs from the outline of the algorithms given in [VD96]; see Note 9.4 on page 145;

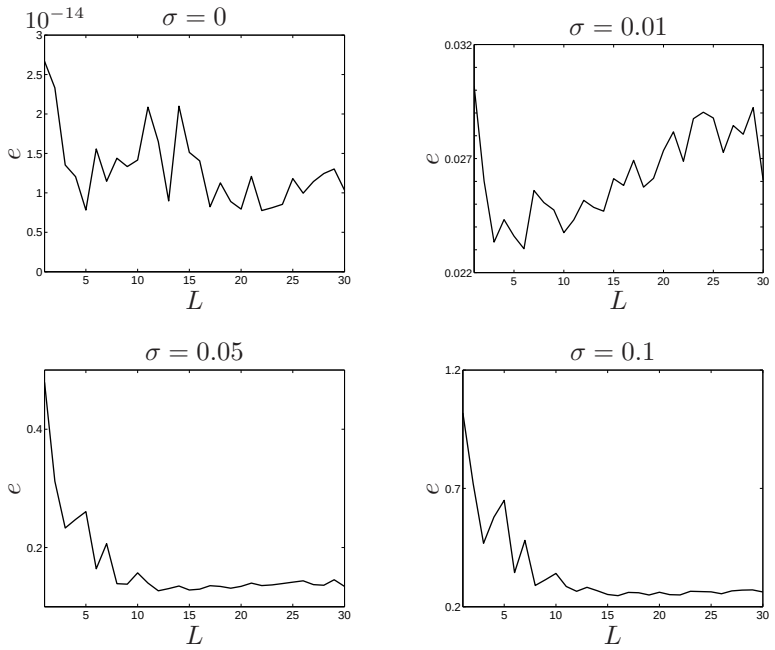


Figure 8.4. *Error of approximation $e = ||H - \hat{H}||_{\mathrm{F}}$ as a function of the parameter L for different noise levels σ.*

`det_stat` "Deterministic algorithm 1" of [VD96, Section 2.4.1]l;
(implementation `det_stat.m` supplementing the book)

`det_alt` "Deterministic algorithm 2" of [VD96, Section 2.4.2];
(implementation `det_alt.m` supplementing the book);

`projec` "Projection algorithm" of [VD96, Section 2.3.1];
(implementation `projec.m` supplementing the book);

`intersec` "Intersection algorithm" of [VD96, Section 2.3.2];
(implementation `intersec.m` supplementing the book);

`moesp` A deterministic version of the MOESP algorithm;

`uy2ssbal` The algorithm for deterministic balanced subspace identification proposed in Chapter 9 (with parameter $L = 1$); see Algorithm 9.4;

`uy2h2ss` Algorithm 8.7 (with $L = 1$) applied for the computation of the first $2\mathtt{n}_{\max} + 1$ samples of the impulse response, followed by Kung's algorithm for the realization of the impulse response; see Algorithm 8.3.

For the experiments we generate a random stable `n`th order system $\mathscr{B}$ with $\mathtt{m} = 2$ inputs and $\mathtt{p} = 2$ outputs. The input is T samples long, zero mean unit variance white Gaussian sequence and the initial condition x_{ini} is a zero mean random vector. We assume that the

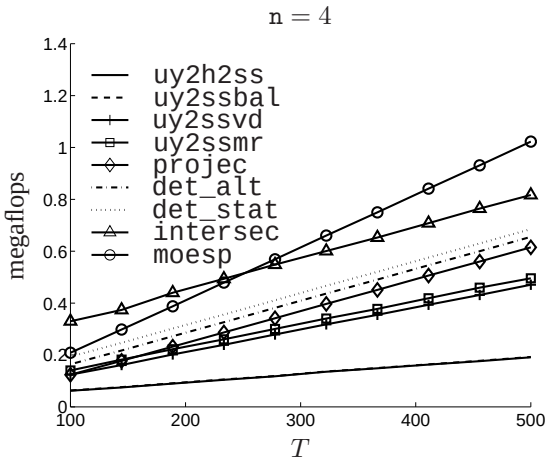


Figure 8.5. *Number of flops for the algorithms as a function of the length T of the given time series.*

true order is known; i.e., $\mathtt{n}_{\max} = \mathtt{n}$ and $\mathtt{l}_{\max}$ is selected as $\lceil \mathtt{n}_{\max}/\mathtt{p} \rceil$. The parameter i (the number of block rows of the Hankel matrix constructed from data) in the subspace identification algorithms is selected as $i = \lceil \mathtt{n}_{\max}/\mathtt{p} \rceil$.

First, we illustrate the amount of work (measured in megaflops) for the compared algorithms as a function of T; see Figure 8.5. The order is chosen as $\mathtt{n} = 4$ and T varies from 100 to 500. The computational complexity of all compared algorithms is linear in T but with different initial cost and different slope. The initial cost and the slope are smallest (almost the same) for `uy2ssbal` and `uy2h2ss`.

The second experiment shows the flops for the compared algorithms as a function of the system order `n`; see Figure 8.6. The length T of the given time series is chosen as 50 and the order `n` is varied from 1 to 18. We deliberately choose T small to show the limitations of the algorithms to identify a system from a finite amount of data. At a certain value of `n`, the graphs in Figure 8.6 stop. The value of `n` where a graph stops is the highest possible order of a system that the corresponding algorithm can identify from the given $T = 50$ data points. (At higher values of `n`, the algorithm either exits with an error message or gives a wrong result.) The flops as a function of `n` are quadratic for all compared algorithms but again the actual number of flops depends on the implementation. Again, most efficient are `uy2ssbal` and `uy2h2ss`. Also, they outperform all other methods except `moesp` in the ability to identify a (high order) system from (small) amount of data. This is a consequence of the fact that Algorithm 8.7 is more parsimonious in the persistency of excitation assumption than Algorithm 8.6.

8.11 Conclusions

We have presented theory and algorithms for exact identification of LTI systems. Although the exact identification problem is not a realistic identification problem, it is interesting and nontrivial from theoretic and algorithmic points of view. In addition, it is an ingredient and

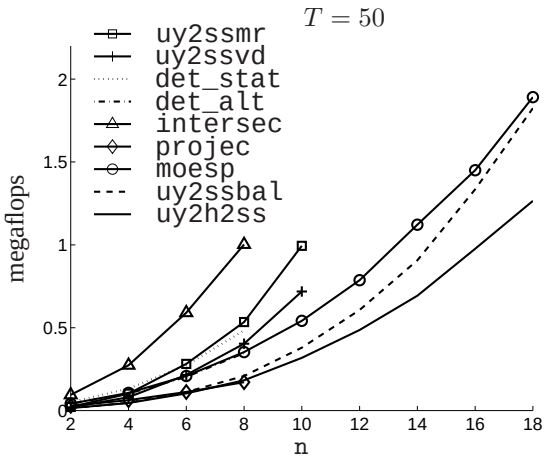


Figure 8.6. *Number of flops for the algorithms as a function of the order* n *of the system.*

prerequisite for proper understanding of other more complicated and realistic identification problems incorporating uncertainty.

The main result is the answer to the identifiability question; Under what conditions, verifiable from w_{d}, does the MPUM coincide with the data generating system? Once this question is answered positively, one can consider algorithms for passing from the data to a representation of the unknown system. In fact, the algorithms compute a representation of the MPUM.

We have presented algorithms for exact identification aiming at kernel, convolution, and input/state/output representations. The latter ones were analyzed in the most detail. We showed links and a new interpretation of the classical MOESP and N4SID deterministic subspace identification algorithms.

Chapter 9

Balanced Model Identification

In this chapter, algorithms for identification of a balanced state space representation are considered. They are based on the algorithms for computation of the impulse response and sequential zero input responses presented in Chapter 8. The proposed algorithms are more efficient than the existing alternatives that compute the whole Hankel matrix of Markov parameters. Moreover, using a finite amount of data, the existing algorithms compute a finite time balanced representation, and the identified models have a lower bound on the distance from an exact balanced representation. The proposed algorithms can approximate arbitrarily closely an exact balanced representation. The finite time balancing parameter can be selected automatically by monitoring the decay of the impulse response. We show what is optimal in terms of the minimal identifiability condition partitioning of the data into "past" and "future" for deterministic subspace identification.

9.1 Introduction

In this chapter, we consider the following deterministic identification problem:

> Given a T samples long input/output trajectory $w_{\mathrm{d}} = (u_{\mathrm{d}}, y_{\mathrm{d}})$ of an LTI system $\mathscr{B} \in \mathscr{L}^{\mathtt{w},\mathtt{n}_{\max}}_{\mathtt{m},\mathtt{l}_{\max}}$, determine a balanced input/state/output representation
>
> $$\mathscr{B} = \mathscr{B}_{\text{i/s/o}}(A_{\text{bal}}, B_{\text{bal}}, C_{\text{bal}}, D_{\text{bal}})$$
>
> of the system, i.e., a representation such that
>
> $$\mathscr{O}^{\top}(A_{\text{bal}}, C_{\text{bal}})\mathscr{O}(A_{\text{bal}}, C_{\text{bal}}) = \mathscr{C}(A_{\text{bal}}, B_{\text{bal}})\mathscr{C}^{\top}(A_{\text{bal}}, B_{\text{bal}}) = \Sigma,$$
>
> where $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_{\mathbf{n}(\mathscr{B})})$ and $\sigma_1 \leq \sigma_1 \leq \cdots \leq \sigma_{\mathbf{n}(\mathscr{B})}$.

The problem is to find conditions and algorithms to construct $(A_{\text{bal}}, B_{\text{bal}}, C_{\text{bal}}, D_{\text{bal}})$ directly from w_{d}. Equivalently, we want to find a balanced input/state/output representation of the MPUM.

Algorithm 9.1 Balanced identification via a state sequence `uy2ssbal`

Input: u_d, y_d, $\mathtt{n}_{\max}$, $\mathtt{l}_{\max}$, and $\Delta > \mathtt{n}_{\max}$.

1: Compute the first 2Δ samples H of the impulse response matrix of $\mathscr{B}$.

2: Compute $\mathtt{n}_{\max} + \mathtt{m} + 1$, Δ samples long sequential free responses Y_0 of $\mathscr{B}$.

3: Compute the SVD, $\mathfrak{H} = U\Sigma V^\top$, of the block-Hankel matrix $\mathfrak{H} = \mathscr{H}_\Delta(\sigma H)$.

4: Compute the balanced state sequence $X_\mathrm{bal} := \sqrt{\Sigma^{-1}}U^\top Y_0$,

$$X_\mathrm{bal} = \begin{bmatrix} x_\mathrm{bal}(\mathtt{n}_{\max}+1) & \cdots & x_\mathrm{bal}(2\mathtt{n}_{\max}+2+\mathtt{m}) \end{bmatrix}.$$

5: Compute the balanced realization A_bal, B_bal, C_bal, D_bal by solving the linear system of equations

$$\begin{bmatrix} x_\mathrm{bal}(\mathtt{n}_{\max}+2) & \cdots & x_\mathrm{bal}(2\mathtt{n}_{\max}+2+\mathtt{m}) \\ y_\mathrm{d}(\mathtt{n}_{\max}+1) & \cdots & y_\mathrm{d}(2\mathtt{n}_{\max}+1+\mathtt{m}) \end{bmatrix} = \begin{bmatrix} A_\mathrm{bal} & B_\mathrm{bal} \\ C_\mathrm{bal} & D_\mathrm{bal} \end{bmatrix} \begin{bmatrix} x_\mathrm{bal}(\mathtt{n}_{\max}+1) & \cdots & x_\mathrm{bal}(2\mathtt{n}_{\max}+1+\mathtt{m}) \\ u_\mathrm{d}(\mathtt{n}_{\max}+1) & \cdots & u_\mathrm{d}(2\mathtt{n}_{\max}+1+\mathtt{m}) \end{bmatrix}. \quad (9.1)$$

Output: $A_\mathrm{bal}, B_\mathrm{bal}, C_\mathrm{bal}, D_\mathrm{bal}$.

Although the assumption that w_d is exact is mainly of theoretical importance, solving the exact identification problem is a prerequisite for the study of the realistic approximate identification problem, where w_d is approximated by a trajectory $\hat{w}$ of an LTI system. In a balanced basis, one can apply truncation as a very effective heuristic for model reduction, which yields a method for approximate identification.

The balanced state space identification problem is studied in [MR93] and [VD96, Chapter 5]. The proposed algorithms fit the outline of Algorithm 9.1.

In [MR93, VD96], it is not mentioned that the Hankel matrix of Markov parameters $\mathscr{H}_\Delta(\sigma H)$ is computed. Also, in [MR93], it is not mentioned that the matrix Y_0 of sequential zero input responses is computed. In this chapter, we interpret these algorithms as implementations of Algorithm 9.1 and reveal their structure.

Note 9.1 (Finite time-Δ balancing) The basic algorithm factors a finite $\Delta \times \Delta$ block-Hankel matrix of Markov parameters $\mathfrak{H}$, so that the obtained representation (A_bal, B_bal, C_bal, D_bal) is *finite time-Δ balanced*. For large Δ, the representation obtained is close to an infinite time balanced one. Determining an appropriate value for the parameter Δ, however, is a problem in its own right and is addressed here. The important difference among the algorithms of [MR93], [VD96], and the ones proposed here is the method of computing the matrix Y_0 and the impulse response H.

Note 9.2 (Model reduction) Identification of a state space model in a balanced basis is motivated by the effective heuristic for model reduction by truncation in that basis. In principle it is possible to identify the model in any basis and then apply standard algorithms for state transformation to a balanced basis. The direct algorithms discussed in this chapter, however, have the advantage over the indirect approach that they allow us to *identify* a reduced order model directly from data without ever computing a full order model.

Algorithm 9.2 Balanced identification via the impulse response `uy2h2ss`

Input: u_d, y_d, $\mathtt{n}_{\max}$, and $\Delta > \mathtt{n}_{\max}$.

1: Find the first 2Δ samples $H(0), \ldots, H(2\Delta-1)$ of the impulse response of $\mathscr{B}$ and let $H := \operatorname{col}\big(H(0), \ldots, H(2\Delta-1)\big)$.

2: Compute the SVD, $\mathfrak{H} = U\Sigma V^\top$, of the block-Hankel matrix of Markov parameters $\mathfrak{H} = \mathscr{H}_\Delta(\sigma H) \in \mathbb{R}^{\Delta\mathtt{p}\times\Delta\mathtt{m}}$.

3: Define $\mathscr{O}_\mathrm{bal} := U\sqrt{\Sigma}$ and $\mathscr{C}_\mathrm{bal} := \sqrt{\Sigma}V^\top$.

4: Let $D_\mathrm{bal} = H(0)$, B_bal be equal to the first $\mathtt{m}$ columns of $\mathscr{C}_\mathrm{bal}$ (the first block column), C_bal be equal to the first $\mathtt{p}$ rows of $\mathscr{O}_\mathrm{bal}$ (the first block row), and A_bal be the solution of the shift equation $(\sigma^*\mathscr{O}_\mathrm{bal})A_\mathrm{bal} = \sigma\mathscr{O}_\mathrm{bal}$.

Output: $A_\mathrm{bal}, B_\mathrm{bal}, C_\mathrm{bal}, D_\mathrm{bal}$.

The model reduction can be done in step 5 of Algorithm 9.1. Let r be the desired order of the reduced model and let X_red be the truncated to the first r rows balanced state sequence X_bal. As a heuristic model reduction procedure, we derive the reduced model parameters by solving the *least squares problem*

$$\begin{bmatrix} x_\mathrm{red}(\mathtt{n}_{\max}+2) & \cdots & x_\mathrm{red}(2\mathtt{n}_{\max}+2+\mathtt{m}) \\ y_\mathrm{d}(\mathtt{n}_{\max}+1) & \cdots & y_\mathrm{d}(2\mathtt{n}_{\max}+1+\mathtt{m}) \end{bmatrix} \\ = \begin{bmatrix} A_\mathrm{red} & B_\mathrm{red} \\ C_\mathrm{red} & D_\mathrm{red} \end{bmatrix} \begin{bmatrix} x_\mathrm{red}(\mathtt{n}_{\max}+1) & \cdots & x_\mathrm{red}(2\mathtt{n}_{\max}+1+\mathtt{m}) \\ u_\mathrm{d}(\mathtt{n}_{\max}+1) & \cdots & u_\mathrm{d}(2\mathtt{n}_{\max}+1+\mathtt{m}) \end{bmatrix}$$

in place of the exact system of equations (9.1). The obtained model $(A_\mathrm{red}, B_\mathrm{red}, C_\mathrm{red}, D_\mathrm{red})$ is *not* the same as the model obtained by truncation of the (finite time-Δ) balanced model. In particular, we do not know about error bounds similar to the ones available for the (infinite time) balanced model reduction.

Step 1, computation of the impulse response, is the crucial one. Once H is computed, a balanced model can be obtained directly via Kung's algorithm. This gives the alternative deterministic balanced model identification algorithm, outlined in Algorithm 9.2.

In Algorithm 9.2, once the impulse response is computed, the parameters A_bal, B_bal, C_bal, and D_bal are obtained without returning to the original observed data. Yet another alternative for computing a balanced representation directly from data is to obtain the parameters A_bal and C_bal as in Algorithm 9.2 from $\mathscr{O}_\mathrm{bal}$ and the parameters B_bal and D_bal (as well as the initial condition $x_\mathrm{bal}(1)$, under which w_d is obtained) from the linear system of equations

$$y_\mathrm{d}(t) = C_\mathrm{bal}A_\mathrm{bal}^t x_\mathrm{bal}(1) + \sum_{\tau=1}^{t-1} C_\mathrm{bal}A_\mathrm{bal}^{t-1-\tau}B_\mathrm{bal}u_\mathrm{d}(\tau) + D_\mathrm{bal}\delta(t+1), \quad \text{for } t = 1, \ldots, T, \tag{9.2}$$

using the original data. (By using Kronecker products, (9.2) can be solved explicitly.) The resulting Algorithm 9.3 is in the spirit of the MOESP-type algorithms.

Simulation results show that in the presence of noise, "going back to the data", as done in Algorithms 9.1 and 9.3, leads to more accurate results. This gives an indication that Algorithms 9.1 and 9.3 might be superior to Algorithm 9.2.

Algorithm 9.3 Balanced identification via an observability matrix `uy2h2o2ss`

Input: u_d, y_d, $\mathtt{n}_{\max}$, and $\Delta > \mathtt{n}_{\max}$.

1: Find the first 2Δ samples $H(0), \ldots, H(2\Delta-1)$ of the impulse response of the MPUM and let $H := \operatorname{col}\big(H(0), \ldots, H(2\Delta-1)\big)$.
2: Compute the SVD, $\mathfrak{H} = U\Sigma V^\top$, of the block-Hankel matrix of Markov parameters $\mathfrak{H} = \mathscr{H}_\Delta(\sigma H) \in \mathbb{R}^{\Delta\mathtt{p}\times\Delta\mathtt{m}}$.
3: Define $\mathscr{O}_\mathrm{bal} := U\sqrt{\Sigma}$.
4: Let C_bal be equal to the first $\mathtt{p}$ rows of $\mathscr{O}_\mathrm{bal}$ (the first block row) and A_bal be the solution of the shift equation $(\sigma^*\mathscr{O}_\mathrm{bal})A_\mathrm{bal} = \sigma\mathscr{O}_\mathrm{bal}$.
5: Solve the system of equations (9.2) for B_bal, D_bal, and $x_\mathrm{bal}(1)$.

Output: $A_\mathrm{bal}, B_\mathrm{bal}, C_\mathrm{bal}, D_\mathrm{bal}$.

9.2 Algorithm for Balanced Identification

In Chapter 8, we specified steps 1 and 2 of Algorithm 9.1. Steps 3, 4, and 5 follow from standard derivations, which we now detail. Let $\mathfrak{H}$ be the Hankel matrix of the Markov parameters $\mathfrak{H} := \mathscr{H}_\Delta(\sigma H)$. By factoring $\mathfrak{H}$ into $\mathscr{O}_\mathrm{bal}$ and $\mathscr{C}_\mathrm{bal}$ via the *restricted* SVD

$$\mathfrak{H} = U\Sigma V^\top = \underbrace{U\sqrt{\Sigma}}_{\mathscr{O}_\mathrm{bal}}\underbrace{\sqrt{\Sigma}V^\top}_{\mathscr{C}_\mathrm{bal}},$$

we obtain an extended observability matrix $\mathscr{O}_\mathrm{bal} = \mathscr{O}_\Delta(A_\mathrm{bal}, C_\mathrm{bal})$ and a corresponding extended controllability matrix $\mathscr{C}_\mathrm{bal} = \mathscr{C}_\Delta(A_\mathrm{bal}, B_\mathrm{bal})$ in a finite time balanced basis. The basis is finite time-Δ balanced, because the finite time-Δ observability gramian $\mathscr{O}_\mathrm{bal}^\top\mathscr{O}_\mathrm{bal} = \Sigma$ and the finite time-Δ controllability gramian $\mathscr{C}_\mathrm{bal}\mathscr{C}_\mathrm{bal}^\top = \Sigma$ are equal and diagonal.

The matrix of sequential zero input responses Y_0 can be written as $Y_0 = \Gamma X$ for a certain extended observability matrix Γ and a state sequence X in the same basis. We find the balanced state sequence

$$X_\mathrm{bal} := \begin{bmatrix} x_\mathrm{bal}(\mathtt{l}_{\max}+1) & \cdots & x_\mathrm{bal}(\mathtt{l}_{\max}+1+\mathtt{n}_{\max}+\mathtt{m}) \end{bmatrix}$$

corresponding to $\mathscr{O}_\mathrm{bal} = U\sqrt{\Sigma}$ from

$$Y_0 = \mathscr{O}_\mathrm{bal}X_\mathrm{bal} \implies X_\mathrm{bal} = \sqrt{\Sigma^{-1}}U^\top Y_0.$$

The corresponding balanced representation $(A_\mathrm{bal}, B_\mathrm{bal}, C_\mathrm{bal}, D_\mathrm{bal})$ is computed from the system of equations

$$\begin{bmatrix} x_\mathrm{bal}(\mathtt{l}_{\max}+2) & \cdots & x_\mathrm{bal}(\mathtt{l}_{\max}+2+\mathtt{n}_{\max}+\mathtt{m}) \\ y_\mathrm{d}(\mathtt{l}_{\max}+1) & \cdots & y_\mathrm{d}(\mathtt{l}_{\max}+1+\mathtt{n}_{\max}+\mathtt{m}) \end{bmatrix} = \begin{bmatrix} A_\mathrm{bal} & B_\mathrm{bal} \\ C_\mathrm{bal} & D_\mathrm{bal} \end{bmatrix} \begin{bmatrix} x_\mathrm{bal}(\mathtt{l}_{\max}+1) & \cdots & x_\mathrm{bal}(\mathtt{l}_{\max}+1+\mathtt{n}_{\max}+\mathtt{m}) \\ u_\mathrm{d}(\mathtt{l}_{\max}+1) & \cdots & u_\mathrm{d}(\mathtt{l}_{\max}+1+\mathtt{n}_{\max}+\mathtt{m}) \end{bmatrix}. \quad (9.3)$$

This yields Algorithm 9.4.

The preceding presentation in this section and Propositions 8.22 and 8.28 prove the following main result.

Algorithm 9.4 Algorithm for balanced subspace identification `uy2ssbal`

Input: u_d, y_d, $\mathtt{n}_{\max}$, $\mathtt{l}_{\max}$, and either Δ or a convergence tolerance ε.

1: Apply Algorithm 8.9 with inputs u_d, y_d, $\mathtt{n}_{\max}$, $\mathtt{l}_{\max}$, L, and Δ, in order to compute the sequential zero input responses Y_0.
2: Apply Algorithm 8.7 with inputs u_d, y_d, $\mathtt{n}_{\max}$, $\mathtt{l}_{\max}$, and either Δ or ε, in order to compute the impulse response H and, if not given, the parameter Δ.
3: Form the Hankel matrix $\mathfrak{H} := \mathscr{H}_\Delta(\sigma H)$ and compute the SVD $\mathfrak{H} = U\Sigma V^\top$.
4: Compute a balanced state sequence $X_\mathrm{bal} = \sqrt{\Sigma^{-1}}U^\top Y_0$.
5: Compute a balanced representation by solving (9.3).

Output: A_bal, B_bal, C_bal, D_bal, and Δ.

Algorithm 9.5 Algorithm of Van Overschee and De Moor `uy2ssvd`

Input: u_d, y_d, and a parameter i.

Define: $\begin{bmatrix} U_\mathrm{p} \\ U_\mathrm{f} \end{bmatrix} := \mathscr{H}_{2i}(u_\mathrm{d})$, where $\operatorname{row\,dim}(U_\mathrm{p}) = i$, and $\begin{bmatrix} Y_\mathrm{p} \\ Y_\mathrm{f} \end{bmatrix} := \mathscr{H}_{2i}(y_\mathrm{d})$, where $\operatorname{row\,dim}(Y_\mathrm{p}) = i$.

Compute the weight matrix $W := U_\mathrm{p}^\top (U_\mathrm{p} U_\mathrm{p}^\top)^{-1} J$, where J is the left–right flipped identity matrix.

1: Compute the oblique projection $Y_0 := Y_\mathrm{f}/_{U_\mathrm{f}} \begin{bmatrix} U_\mathrm{p} \\ Y_\mathrm{p} \end{bmatrix}$; see (8.13).
2: Compute the matrix $\hat{\mathfrak{H}} := Y_0 W$.
3: Compute the SVD $\hat{\mathfrak{H}} = U\Sigma V^\top$.
4: Compute a balanced state sequence $X_\mathrm{bal} = \sqrt{\Sigma^{-1}}U^\top Y_0$.
5: Compute a balanced representation by solving (9.3).

Output: A_bal, B_bal, C_bal, D_bal.

Theorem 9.3. *Let $w_\mathrm{d} = (u_\mathrm{d}, y_\mathrm{d})$ be a trajectory of a controllable LTI system $\mathscr{B}$ of order $\mathbf{n}(\mathscr{B}) \le \mathtt{n}_{\max}$ and lag $\mathbf{l}(\mathscr{B}) \le \mathtt{l}_{\max}$, and let u_d be persistently exciting of order $L + \mathtt{l}_{\max} + \mathtt{n}_{\max}$. Then $(A_\mathrm{bal}, B_\mathrm{bal}, C_\mathrm{bal}, D_\mathrm{bal})$ computed by Algorithm 9.4 is a finite time-Δ balanced representation of $\mathscr{B}$.*

9.3 Alternative Algorithms

We outline the algorithms of Van Overschee and De Moor [VD96] and Moonen and Ramos [MR93].

Note 9.4 (Weight matrix W) The weight matrix W is different from the one in [VD96]. In terms of the final result $\hat{\mathfrak{H}}$, however, it is equivalent. Another difference between Algorithm 9.5 and the deterministic balanced subspace algorithm of [VD96] is that the shifted state sequence appearing on the left-hand side of (9.3) is recomputed in [VD96] by another oblique projection.

In the algorithms of Van Overschee and De Moor and Moonen and Ramos, the parameter i plays the role of the finite time balancing parameter Δ. Note that i is given and the "past" and the "future" are taken with equal length i.

Algorithm 9.6 Algorithm of Moonen and Ramos `uy2ssmr`

Input: u_d, y_d, and a parameter i.

Define: $\begin{bmatrix} U_\mathrm{p} \\ U_\mathrm{f} \end{bmatrix} := \mathscr{H}_{2i}(u_\mathrm{d})$, where $\operatorname{row\,dim}(U_\mathrm{p}) = i$, and $\begin{bmatrix} Y_\mathrm{p} \\ Y_\mathrm{f} \end{bmatrix} := \mathscr{H}_{2i}(y_\mathrm{d})$, where $\operatorname{row\,dim}(Y_\mathrm{p}) = i$.

Compute a matrix $[T_1 \ T_2 \ T_3 \ T_4]$, whose rows form a basis for the left kernel of $\begin{bmatrix} U_\mathrm{p} \\ Y_\mathrm{p} \\ U_\mathrm{f} \\ Y_\mathrm{f} \end{bmatrix}$.

1: Compute a matrix of zero input responses $Y_0 = T_4^\dagger [T_1 \ T_2] \begin{bmatrix} U_\mathrm{p} \\ Y_\mathrm{p} \end{bmatrix}$.
2: Compute the Hankel matrix of Markov parameters $\mathfrak{H} = T_4^\dagger (T_2 T_4^\dagger T_3 - T_1) J$.
3: Compute the SVD, $\mathfrak{H} = U \Sigma V^\top$.
4: Compute a balanced state sequence $X_\mathrm{bal} = \sqrt{\Sigma^{-1}} U^\top Y_0$.
5: Compute a balanced representation by solving (9.3).

Output: A_bal, B_bal, C_bal, D_bal.

Both Algorithm 9.5 and Algorithm 9.6 fit the outline of Algorithm 9.1, but steps 1 and 2 are implemented in rather different ways. As shown in Section 8.8, the oblique projection $Y_\mathrm{f}/_{U_\mathrm{f}} \begin{bmatrix} U_\mathrm{p} \\ Y_\mathrm{p} \end{bmatrix}$ is a matrix of sequential zero input responses. The weight matrix W in the algorithm of Van Overschee and De Moor is constructed so that $\hat{\mathfrak{H}} = Y_0 W$ is an approximation of the Hankel matrix of Markov parameters $\mathfrak{H}$; it is the sum of $\mathfrak{H}$ and a matrix of zero input responses.

The most expensive computation in the algorithm of Moonen and Ramos is the computation of the annihilators $\begin{bmatrix} T_1 & \cdots & T_4 \end{bmatrix}$. The matrix $[T_1 \ T_2] \begin{bmatrix} U_\mathrm{p} \\ Y_\mathrm{p} \end{bmatrix}$ is a nonminimal state sequence (the shift-and-cut operator [RW97]) and $T_4^\dagger$ is a corresponding extended observability matrix. Thus $T_4^\dagger [T_1 \ T_2] \begin{bmatrix} U_\mathrm{p} \\ Y_\mathrm{p} \end{bmatrix}$ is a matrix of sequential zero input responses. It turns out that $(T_2 T_4^\dagger T_3 - T_1) J$ is an extended controllability matrix (in the same basis), so that $T_4^\dagger (T_2 T_4^\dagger T_3 - T_1) J$ is the Hankel matrix of Markov parameters $\mathfrak{H}$.

A major difference between the proposed Algorithm 9.4, on one hand, and the algorithms of Van Overschee and De Moor and Moonen and Ramos, on the other hand, is that in Algorithm 9.4 the Hankel matrix $\mathfrak{H}$ is not computed but *constructed* from the impulse response that parameterizes it. This is a big computational saving because recomputing the same elements of $\mathfrak{H}$ is avoided. In addition, in approximate identification, where w_d is not a trajectory of $\mathscr{B}$, the matrices $\hat{\mathfrak{H}}$ and $\mathfrak{H}$ computed by the algorithms of Van Overschee and De Moor and Moonen and Ramos are in general no longer Hankel, while the matrix $\mathfrak{H}$ in Algorithm 9.4 is by construction Hankel.

9.4 Splitting of the Data into "Past" and "Future"*

In the algorithms of Moonen and Ramos and Van Overschee and De Moor, the block-Hankel matrices $\begin{bmatrix} U_\mathrm{p} \\ U_\mathrm{f} \end{bmatrix}$ and $\begin{bmatrix} Y_\mathrm{p} \\ Y_\mathrm{f} \end{bmatrix}$ are split into "past" and "future" of equal length. Natural questions are why is this necessary and furthermore what is "optimal" according to certain relevant criteria partitionings. These questions have been open for a long time, in particular in the

context of the stochastic identification problem; see [DM03].

In Chapter 8, we showed that the past U_p, Y_p is used to assign the initial conditions and the future U_f, Y_f is used to compute a response. By weaving consecutive segments of the response, as done in Algorithms 8.7 and 8.9, the number of block rows in the future does not need to be equal to the required length of the response. Thus from the perspective of deterministic identification, the answer to the above question is as follows:

> $\operatorname{row}\dim(U_\mathrm{p}) = \operatorname{row}\dim(Y_\mathrm{p}) = \mathtt{l}_{\max}$, i.e., the given least upper bound on the system lag $\mathbf{l}(\mathscr{B})$, and $\operatorname{row}\dim(U_\mathrm{f}) = \operatorname{row}\dim(Y_\mathrm{f}) \in \{1, \ldots, \gamma - \mathtt{l}_{\max} + \mathtt{n}_{\max}\}$, where γ is the order of persistency of excitation of the input u_d.

By using the iterative algorithms for computation of the impulse response and sequential free responses with parameter $L = 1$, Algorithms 9.2, 9.3, and 9.4 require the same assumption as the identifiability assumption of Theorem 8.16, so that the partitioning "past = $\mathtt{l}_{\max}$ and future = 1" is consistent with our previous results.

Using the fundamental lemma, we can prove the following result.

Proposition 9.5. *Let $w_\mathrm{d} = (u_\mathrm{d}, y_\mathrm{d})$ be a trajectory of a controllable LTI system $\mathscr{B} \in \mathscr{L}_{\mathtt{m},i}^{\mathtt{w},\mathtt{n}_{\max}}$, and let u_d be persistently exciting of order $2i + \mathtt{n}_{\max}$. Then the representations computed by Algorithms 9.5 and 9.6 are equivalent to $\mathscr{B}$. Moreover, the representation computed by Algorithm 9.6 is in a finite time-i balanced basis.*

Proposition 9.5 shows that Algorithms 9.5 and 9.6 are not parsimonious with respect to the available data. In particular, the system $\mathscr{B}$ can be identifiable with Algorithms 9.2, 9.3, and 9.4 but not with Algorithms 9.5 and 9.6.

Note that the persistency of excitation required by Algorithms 9.5 and 9.6 is a function of the finite time balancing parameter. This implies that with a finite amount of data, Algorithms 9.5 and 9.6 are limited in the ability to identify a balanced representation. In fact,

$$i \leq \left\lfloor \frac{T+1}{2\big(\max(\mathtt{m},\mathtt{p})+1\big)} \right\rfloor,$$

where $\lfloor a \rfloor$ denotes the highest integer smaller than a. In contrast, the persistency of excitation required by Algorithms 9.2, 9.3, and 9.4 depends only on the upper bounds on the system order and the lag and thus these algorithms can compute an infinite time balanced representation if the identifiability condition holds.

9.5 Simulation Examples

In this section, we show examples that illustrate some of the advantages of the proposed Algorithm 9.4. In all experiments the system $\mathscr{B}$ is given by a minimal input/state/output representation with transfer function

$$C(Iz - A)^{-1}B + D = \frac{0.89172(z - 0.5193)(z + 0.5595)}{(z - 0.4314)(z + 0.4987)(z + 0.6154)}.$$

The input is a unit variance white noise and the data available for identification is the corresponding trajectory w_d of $\mathscr{B}$, corrupted by white noise with standard deviation σ.

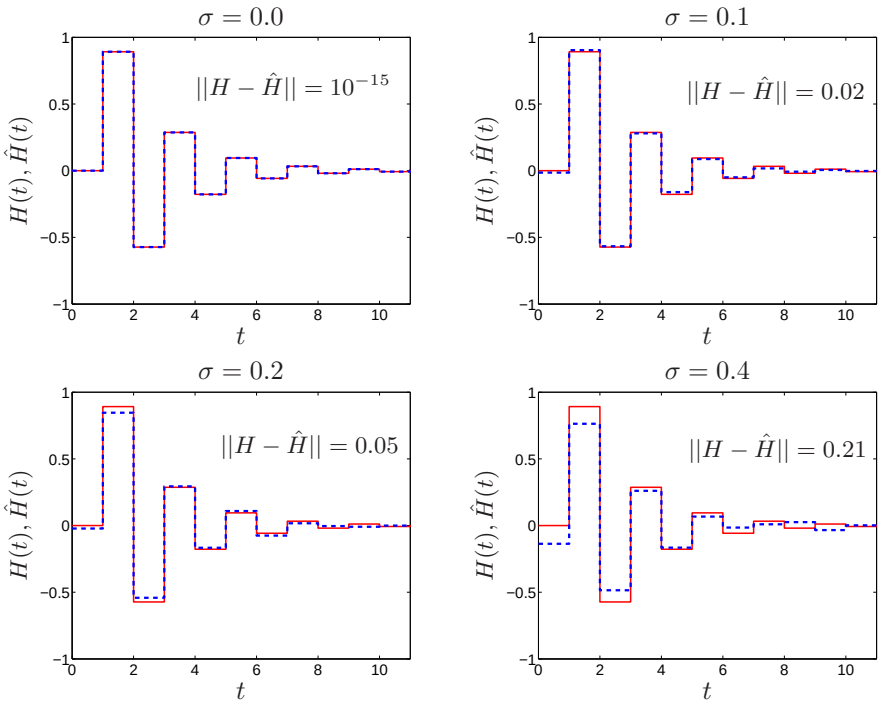


Figure 9.1. *Impulse response estimation. Solid red line—exact impulse response H, dashed blue line—impulse response $\hat{H}$ computed from data via Algorithm 8.7.*

Although our main concern is the correct work of the algorithms for exact data, i.e., with $\sigma = 0$, by varying the noise variance σ^2, we can investigate empirically the performance under noise. The simulation time is $T = 100$. In all experiments the upper bounds $\mathtt{n}_{\max}$ and $\mathtt{l}_{\max}$ are taken equal to the system order $\mathtt{n} = 3$ and the parameter L is taken equal to 3.

Consider first the estimation of the impulse response. Figure 9.1 shows the exact impulse response H of $\mathscr{B}$ and the estimate $\hat{H}$ computed by Algorithm 8.7. With exact data, $||H - \hat{H}||_{\mathrm{F}} = 10^{-15}$, so that up to the numerical precision the match is exact. The plots in Figure 9.1 show the deterioration of the estimates when the data is corrupted by noise.

Consider next the computation of the zero input response. Table 9.1 shows the error of estimation $e := ||Y_0 - \hat{Y}_0||_{\mathrm{F}}$ and the corresponding number of operations, where Y_0 is a matrix of exact sequential zero input responses with length $\Delta = 10$ and $\hat{Y}_0$ is its estimate computed from data. The estimate is computed in three ways: by Algorithm 8.9, implemented with the QR factorization; by the oblique projection, computed directly from (8.13); and by the oblique projection, computed via the QR factorization; see Section 8.8.

Algorithm 8.9 needs fewer computations and gives more accurate results than the alternatives. As already emphasized, the reason for this is that selecting the parameter $L = \mathtt{n}_{\max} = 3$ instead of $L = \Delta = 10$, as in a block computation, results in a more overdetermined system of equations in step 4 of Algorithm 8.9 compared with system (8.9) used in the block algorithm. (For the example, the difference is 95 vs. 88 columns.) As a result, the noise is averaged over more samples, which leads to a better estimate in a

Table 9.1. *Error of estimation $e = ||Y_0 - \hat{Y}_0||_{\mathrm{F}}$ and the corresponding number of operations f in megaflops, where Y_0 is an exact sequence of zero input responses and $\hat{Y}_0$ is the estimate computed from data.*

Method	$\sigma = 0.0$		$\sigma = 0.1$		$\sigma = 0.2$		$\sigma = 0.4$	
	e	f	e	f	e	f	e	f
Alg. 8.9 with QR	10^{-14}	130	1.2990	131	2.5257	132	4.7498	132
formula (8.13)	10^{-10}	182	1.6497	186	3.2063	187	6.0915	189
(8.13) with QR	10^{-14}	251	1.6497	251	3.2063	251	6.0915	252

statistical sense. Solving several more overdetermined systems of equations instead of one more rectangular system can be more efficient, as it is in the example.

All algorithms return a finite time balanced model. The next experiment illustrates the effect of the parameter Δ on the balancing. Let $W_{\mathrm{c}}/W_{\mathrm{o}}$ be the controllability/observability gramians of an infinite time balanced model and $\hat{W}_{\mathrm{c}}/\hat{W}_{\mathrm{o}}$ be the controllability/observability gramians of an identified model. Define closeness to balancing by

$$e_{\mathrm{bal}}^2 := \frac{||W_{\mathrm{c}} - \hat{W}_{\mathrm{c}}||_{\mathrm{F}}^2 + ||W_{\mathrm{o}} - \hat{W}_{\mathrm{o}}||_{\mathrm{F}}^2}{||W_{\mathrm{c}}||_{\mathrm{F}}^2 + ||W_{\mathrm{o}}||_{\mathrm{F}}^2}.$$

Figure 9.2 shows e_{bal} as a function of Δ for the three algorithms presented in the chapter. The estimates obtained by Algorithm 9.4 and the algorithm of Moonen and Ramos are identical. The estimate obtained by the algorithm of Van Overschee and De Moor is asymptotically equivalent, but for small Δ, it is worse. This is a consequence of the fact that this algorithm uses an approximation of the Hankel matrix of Markov parameters. Figure 9.2 also shows e_{bal} as a function of Δ for noisy data with $\sigma = 0.001$ and the total number of flops required by the three algorithms.

9.6 Conclusions

The impulse response and sequential free responses are the main tools for balanced model identification. First, a (nonbalanced) state sequence is obtained from the sequential free responses. Then a Hankel matrix is constructed from the impulse response, and from its SVD a balancing transformation is obtained. A balanced state sequence is computed via a change of basis and the corresponding balanced state representation is obtained by solving a system of equations. We called this procedure the basic algorithm and showed that the algorithms of Moonen and Ramos and Van Overschee and De Moor fit into it. Based on the algorithms for computation of the impulse response and sequential free responses directly from data, we proposed alternative algorithms for balanced model identification.

There are a number of advantages of the proposed algorithms over the existing ones. The algorithms of Moonen and Ramos and Van Overschee and De Moor compute the whole Hankel matrix of Markov parameters $\mathcal{H}$, while the proposed algorithms compute only the elements that uniquely specify $\mathcal{H}$ and then *construct* $\mathcal{H}$ from them. Because of the Hankel structure, the algorithms of Moonen and Ramos and Van Overschee and De Moor recompute most elements of $\mathcal{H}$ many times. This is an inefficient step in these algorithms that we avoid.

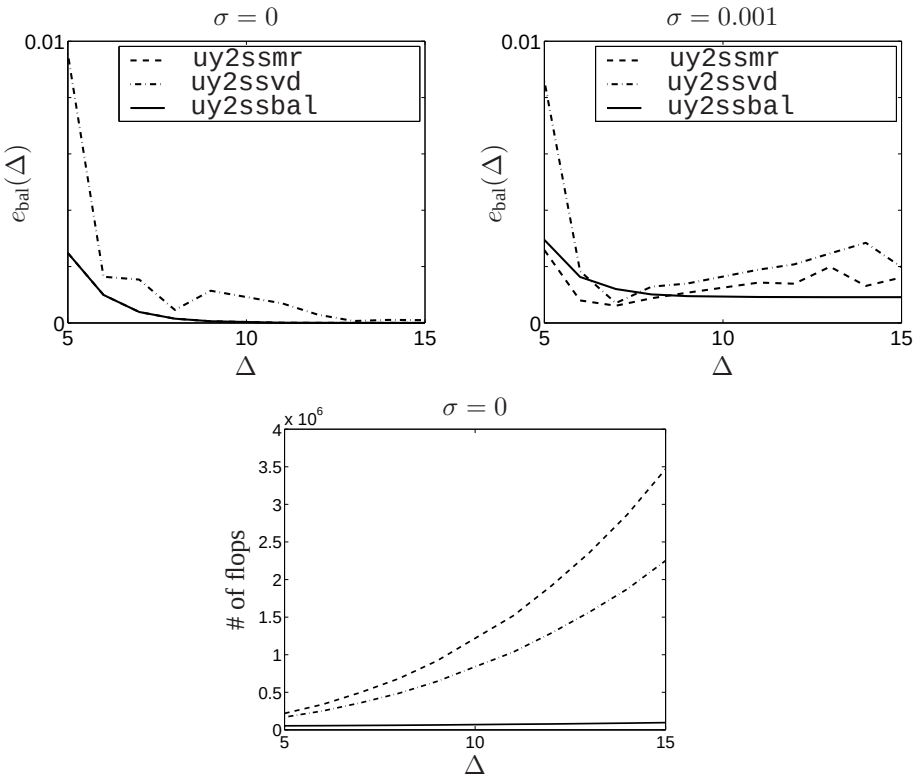


Figure 9.2. *Closeness to balancing e_{bal} and computational cost as functions of the finite time balancing parameter Δ (*`uy2ssmr`*—Algorithm 9.5,* `uy2ssvd`*—Algorithm 9.6,* `uy2ssbal`*—Algorithm 9.4).*

In the algorithms of Moonen and Ramos and Van Overschee and De Moor, the finite time balancing parameter Δ is supplied by the user. In the proposed algorithms, it can be determined automatically on the basis of a desired convergence tolerance of the impulse response, which is directly related to the closeness of the obtained representation to a balanced one.

The algorithms of Moonen and Ramos and Van Overschee and De Moor compute finite time-Δ balanced representation with $\Delta \leq \lfloor \frac{1}{2}(T+1)/\big(\max(\mathtt{m},\mathtt{p})+1\big) \rfloor$, where T is the length of the given time series w_{d}. The proposed algorithms have no such limitation and can thus compute a representation that is arbitrary close to an infinite time balanced one.

The proposed algorithms have weaker persistency of excitation condition than the one needed for the algorithms of Moonen and Ramos and Van Overschee and De Moor. As a result, in certain cases, the proposed algorithms are applicable, while the algorithms of Moonen and Ramos and Van Overschee and De Moor are not.

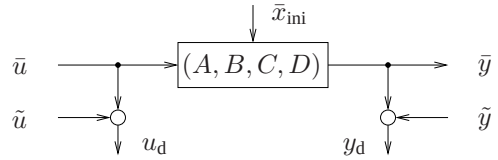


Figure 10.1. *Block scheme of the dynamic EIV model.*

Chapter 10

Errors-in-Variables Smoothing and Filtering

State estimation problems for LTI systems with noisy inputs and outputs (EIV model, see Figure 10.1) are considered.

An efficient recursive algorithm for the smoothing problem is derived. The equivalence between the optimal filter and an appropriately modified Kalman filter is established. The optimal estimate of the input signal is derived from the optimal state estimate. The result shows that the EIV filtering problem is not fundamentally different from the classical Kalman filtering problem.

10.1 Introduction

The EIV smoothing and filtering problems were first put forward by Guidorzi, Diversi, and Soverini in [GDS03], where a transfer function approach is used and recursive algorithms that solve the filtering problem are derived. The treatment, however, is limited to the SISO case and the solution obtained is not linked to the classical Kalman filter.

The MIMO case is addressed in [MD03], where the equivalence with a modified Kalman filter is established. Closely related to the approach of [MD03] is the one used in [DGS03]. The continuous-time version of the EIV state estimation problem is explicitly solved in [MWD02] by a completion of squares approach.

In this chapter, we consider the EIV model

$$w_{\text{d}} = \bar{w} + \tilde{w}, \quad \text{where} \quad \bar{w} \in \mathscr{B} \in \mathscr{L}_{\mathtt{m}}^{\mathtt{w},\mathtt{n}} \tag{EIV}$$

and $\tilde{w}$ is a white, stationary, zero mean, stochastic process with positive definite covariance matrix $V_{\tilde{w}} := \operatorname{cov}\big(\tilde{w}(t)\big)$ for all t; i.e., we assume that the observed time series w_{d} is a noise corrupted version of a true time series $\bar{w}$ that is a trajectory of an LTI system $\mathscr{B}$ with input cardinality $\mathtt{m}$ and state dimension $\mathtt{n}$. The system $\mathscr{B}$ and the noise covariance $V_{\tilde{w}}$ are assumed known.

We use the input/state/output representation of $\mathscr{B} = \mathscr{B}_{\mathrm{i/s/o}}(A, B, C, D)$, i.e.,

$$\bar{w} = \operatorname{col}(\bar{u}, \bar{y}), \qquad \text{where} \qquad \sigma\bar{x} = A\bar{x} + B\bar{u}, \quad \bar{y} = C\bar{x} + D\bar{u}, \quad \bar{x}(1) = \bar{x}_{\mathrm{ini}}. \tag{10.1}$$

Correspondingly, the observed time series w_{d} and the measurement errors $\tilde{w}$ have the input/output partitionings $w_{\mathrm{d}} = \operatorname{col}(u_{\mathrm{d}}, y_{\mathrm{d}})$ and $\tilde{w} = \operatorname{col}(\tilde{u}, \tilde{y})$. Furthermore, the covariance matrix $V_{\tilde{w}}$ is assumed to be block-diagonal, $V_{\tilde{w}} = \operatorname{diag}(V_{\tilde{u}}, V_{\tilde{y}})$, where $V_{\tilde{u}}, V_{\tilde{y}} > 0$.

The problem considered is to find the LS estimate of the state $\bar{x}$ from the observed data w_{d}. We prove that the optimal filter is the Kalman filter for the system

$$\begin{aligned} \sigma\bar{x} &= A\bar{x} + Bu_{\mathrm{d}} + v_1, \\ y_{\mathrm{d}} &= C\bar{x} + Du_{\mathrm{d}} + v_2, \end{aligned} \tag{10.2}$$

where the process noise v_1 and the measurement noise v_2 are jointly white

$$\begin{aligned} \operatorname{cov}\left(\begin{bmatrix} v_1(t_1) \\ v_2(t_1) \end{bmatrix}, \begin{bmatrix} v_1(t_2) \\ v_2(t_2) \end{bmatrix} \right) &= \begin{bmatrix} -B & 0 \\ -D & I \end{bmatrix} \begin{bmatrix} V_{\tilde{u}} & \\ & V_{\tilde{y}} \end{bmatrix} \begin{bmatrix} -B & 0 \\ -D & I \end{bmatrix}^{\top} \delta(t_1 - t_2) \\ &=: \begin{bmatrix} Q & S \\ S^{\top} & R \end{bmatrix} \delta(t_1 - t_2). \end{aligned} \tag{10.3}$$

The EIV state estimation problem is treated in [RH95] and [FW97]. The global total least squares problem of [RH95] has as a subproblem the computation of the closest trajectory in the behavior of a given system to a given time series. This is a deterministic approximation problem corresponding to the EIV smoothing problem considered in this chapter.

10.2 Problem Formulation

Consider the time horizon $[1, T]$ and define the covariance matrices of $\tilde{u}$, $\tilde{y}$, and $\tilde{w}$:

$$\mathbf{V}_{\tilde{u}} := \operatorname{cov}(\tilde{u}), \qquad \mathbf{V}_{\tilde{y}} := \operatorname{cov}(\tilde{y}), \qquad \text{and} \qquad \mathbf{V}_{\tilde{w}} := \operatorname{cov}(\tilde{w}).$$

Problem 10.1 (EIV smoothing problem). *The EIV smoothing problem is defined by*

$$\begin{aligned} &\min_{\hat{x}, \hat{w} = \operatorname{col}(\hat{u}, \hat{y})} (w_{\mathrm{d}} - \hat{w})^{\top} \mathbf{V}_{\tilde{w}}^{-1} (w_{\mathrm{d}} - \hat{w}) \quad \textit{subject to} \\ &\hat{x}(t+1) = A\hat{x}(t) + B\hat{u}(t), \quad \hat{y}(t) = C\hat{x}(t) + D\hat{u}(t), \quad \textit{for } t = 1, \ldots, T, \end{aligned} \tag{10.4}$$

and the EIV smoothed state estimate $\hat{x}(\cdot, T+1)$ *is the optimal solution of (10.4).*

Under the normality assumption for $\tilde{w}$, $\hat{x}(\cdot, T+1)$ is the maximum likelihood estimate of $\bar{x}$ [GDS03].

Problem 10.2 (EIV filtering problem). *The EIV filtering problem is to find a dynamical system*

$$\sigma z = A_{\mathrm{f}} z + B_{\mathrm{f}} w_{\mathrm{d}}, \qquad \hat{x} = C_{\mathrm{f}} z + D_{\mathrm{f}} w_{\mathrm{d}} \tag{10.5}$$

such that $\hat{x}(t) = \hat{x}(t, t+1)$*, where* $\hat{x}(\cdot)$ *is the solution of (10.5); i.e., the* EIV filtered state estimate*, and* $\hat{x}(\cdot, t+1)$ *is the EIV smoothed state estimate with a time horizon* $t+1$*.*

The EIV filtering problem is defined as a state estimation problem. When the input is measured with additive noise, an extra step is needed to find the filtered input/output signals from the state estimate. This is explained in Note 10.8.

Note 10.3 (Initial conditions) Problem 10.1 implicitly assumes no prior information about the initial condition $\bar{x}(1)$. Another possibility is that $\bar{x}(1)$ is exactly known. The standard assumption is $\bar{x}(1) \sim \mathrm{N}(x_{\mathrm{ini}}, P_{\mathrm{ini}})$, i.e., $\bar{x}(1)$ is normally distributed with mean $\bar{x}_{\mathrm{ini}}$ and covariance P_{ini}. An exactly known initial condition corresponds to $P_{\mathrm{ini}} = 0$ and an unknown initial condition corresponds to information matrix $P_{\mathrm{ini}}^{-1} = 0$.

We have chosen the initial condition assumption that results in the simplest derivation. With the classical stochastic assumption $\bar{x}_{\mathrm{ini}} \sim \mathrm{N}(x_{\mathrm{ini}}, P_{\mathrm{ini}})$, (10.4) becomes

$$\min_{\hat{u},\hat{y},\hat{x}} \left\| \sqrt{\begin{bmatrix} P_{\mathrm{ini}} & & \\ & \mathbf{V}_{\tilde{u}} & \\ & & \mathbf{V}_{\tilde{y}} \end{bmatrix}^{-1}} \begin{bmatrix} x_{\mathrm{ini}} - \hat{x}_{\mathrm{ini}} \\ u_{\mathrm{d}} - \hat{u} \\ y_{\mathrm{d}} - \hat{y} \end{bmatrix} \right\|^2 \quad \text{subject to} \quad \begin{aligned} \hat{x}(t+1) &= A\hat{x}(t) + B\hat{u}(t) \\ \hat{y}(t) &= C\hat{x}(t) + D\hat{u}(t) \\ &\text{for } t = 1, \ldots, T. \end{aligned}$$

10.3 Solution of the Smoothing Problem

Block Algorithms

We represent the input/output dynamics of the system (10.1), over the time horizon $[1, T]$, explicitly as (see (VC'))

$$\bar{y} = \mathscr{O}_T(A, C)\bar{x}_{\mathrm{ini}} + \mathscr{T}_T(H)\bar{u},$$

where H is the impulse response of $\mathscr{B}$. Using this representation, the EIV smoothing problem (10.4) becomes a classical weighted least squares problem

$$\min_{\hat{x}_{\mathrm{ini}},\hat{u}} \left\| \sqrt{\begin{bmatrix} \mathbf{V}_{\tilde{u}} & \\ & \mathbf{V}_{\tilde{y}} \end{bmatrix}^{-1}} \left(\begin{bmatrix} u_{\mathrm{d}} \\ y_{\mathrm{d}} \end{bmatrix} - \begin{bmatrix} 0 & I \\ \mathscr{O}_T(A, C) & \mathscr{T}_T(H) \end{bmatrix} \begin{bmatrix} \hat{x}_{\mathrm{ini}} \\ \hat{u} \end{bmatrix} \right) \right\|^2. \tag{10.6}$$

Alternatively, we represent the input/state/output dynamics of the system, over the time horizon $[1, T]$, as

$$\bar{\mathsf{y}} = \mathsf{A}\bar{x} + \mathsf{B}\bar{u},$$

where

$$\bar{\mathsf{y}} := \begin{bmatrix} \bar{y}(1) \\ 0 \\ \bar{y}(2) \\ 0 \\ \vdots \\ \bar{y}(T) \\ 0 \end{bmatrix}, \quad \mathsf{A} := \begin{bmatrix} C & 0 & & & & \\ A & -I & & & & \\ & C & 0 & & & \\ & A & -I & & & \\ & & \ddots & \ddots & & \\ & & & C & 0 \\ & & & A & -I \end{bmatrix}, \quad \mathsf{B} := \begin{bmatrix} D & & & & \\ B & & & & \\ & D & & & \\ & B & & & \\ & & \ddots & & \\ & & & & D \\ & & & & B \end{bmatrix}.$$

Substituting $y_\mathrm{d} - \tilde{y}$ for $\bar{y}$ and $u_\mathrm{d} - \tilde{u}$ for $\bar{u}$ (see (EIV)), we have

$$\mathsf{y}_\mathrm{d} + \mathsf{B}u_\mathrm{d} = \mathsf{A}\bar{x} + \mathsf{B}\tilde{u} + \mathsf{C}\tilde{y}, \tag{10.7}$$

where y_d is defined analogously to $\bar{\mathsf{y}}$ and

$$\mathsf{C} := \operatorname{diag}\left(\begin{bmatrix} I \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} I \\ 0 \end{bmatrix} \right).$$

Using (10.7) and defining $\Delta w := \operatorname{col}(\Delta u, \Delta y)$, (10.4) is equivalent to the problem

$$\min_{\hat{x}, \Delta w} \Delta w^\top \mathbf{V}_{\tilde{w}}^{-1} \Delta w \quad \text{subject to} \quad \mathsf{y}_\mathrm{d} + \mathsf{B}u_\mathrm{d} = \mathsf{A}\hat{x} + \begin{bmatrix} \mathsf{B} & \mathsf{C} \end{bmatrix} \Delta w, \tag{10.8}$$

which is a minimum norm-type problem, so that its solution also has closed form.

Recursive Algorithms

Next, we show a recursive solution for the case when $\bar{x}(1) = \bar{x}_\mathrm{ini}$ is given and $D = 0$. The more general case, $\bar{x}(1) \sim \mathrm{N}(x_\mathrm{ini}, P_\mathrm{ini})$ and $D \neq 0$, leads to a similar but heavier result.

Define the value function $V_\tau : \mathbb{R}^\mathtt{n} \to \mathbb{R}$, for $\tau = 1, \ldots, T$ as follows: $V_\tau(z)$ is the minimum value of (10.4) over $t = \tau, \ldots, T-1$ with $\hat{x}(\tau) = z$. Then $V_1(x_\mathrm{ini})$ is the optimum value of the EIV smoothing problem. By the dynamic programming principle, we have

$$V_\tau(z) = \min_{\hat{u}(\tau)} \Big(\Big\| \sqrt{\mathbf{V}_{\tilde{u}}^{-1}} \big(\hat{u}(\tau) - u_\mathrm{d}(\tau) \big) \Big\|^2 + \Big\| \sqrt{\mathbf{V}_{\tilde{y}}^{-1}} \big(Cz - y_\mathrm{d}(\tau) \big) \Big\|^2 + V_{\tau+1}\big(Az + B\hat{u}(\tau) \big) \Big). \tag{10.9}$$

The value function V_τ is quadratic for all τ; i.e., there are $P(\tau) \in \mathbb{R}^{n \times n}$, $s(\tau) \in \mathbb{R}^n$, and $v(\tau) \in \mathbb{R}$, such that

$$V_\tau(z) = z^\top P(\tau) z + 2 s^\top(\tau) z + v(\tau).$$

This allows us to solve (10.9).

Theorem 10.4 (Recursive smoothing). *The solution of the EIV smoothing problem with given $\bar{x}(1) = \bar{x}_\mathrm{ini}$ and $D = 0$ is*

$$\hat{u}(t) = -\big(B^\top P(t+1) B + V_{\tilde{u}}^{-1} \big)^{-1} \big(B^\top P(t+1) A \hat{x}(t) + B^\top s(t+1) - V_{\tilde{u}}^{-1} u_\mathrm{d}(t) \big), \tag{10.10}$$

$\hat{x}(t+1) = A\hat{x}(t) + B\hat{u}(t)$, *with* $\hat{x}(1) = \bar{x}_{\text{ini}}$, *and* $\hat{y}(t) = C\hat{x}(t)$ *for* $t = 0, \ldots, T-1$, *where*

$$P(t) = -A^\top P(t+1)B\big(B^\top P(t+1)B + V_{\tilde{u}}^{-1}\big)^{-1}B^\top P(t+1)A + A^\top P(t+1)A + C^\top V_{\tilde{y}}^{-1}C, \quad (10.11)$$

for $t = T-1, \ldots, 0$, *with* $P(T) = 0$, *and*

$$s(t) = -A^\top P(t+1)B\big(B^\top P(t+1)B + V_{\tilde{u}}^{-1}\big)^{-1}\big(B^\top s(t+1) - V_{\tilde{u}}^{-1}u_{\text{d}}(t)\big) + A^\top s(t+1) - C^\top V_{\tilde{y}}^{-1}y_{\text{d}}(t), \quad (10.12)$$

for $t = T-1, \ldots, 0$, *with* $s(T) = 0$.

Proof. See Appendix A.4. □

P and s are obtained from the backward-in-time recursions (10.11) and (10.12), and the estimates $\hat{u}$, $\hat{x}$, and $\hat{y}$ are obtained from the forward-in-time recursion (10.10).

Note 10.5 (Suboptimal smoothing) With (A, C) observable, (10.11) has a steady state solution $\bar{P}$ that satisfies the algebraic Riccati equation

$$\bar{P} = -A^\top \bar{P}B(B^\top \bar{P}B + V_{\tilde{u}}^{-1})^{-1}B^\top \bar{P}A + A^\top \bar{P}A + C^\top V_{\tilde{y}}^{-1}C. \quad (10.13)$$

In a suboptimal solution, the unique positive definite solution $\bar{P}_+$ of (10.13) can be substituted for $P(t)$ in (10.12) and (10.10). This is motivated by the typically fast convergence of $P(t)$ to $\bar{P}_+$. Then the smoothing procedure becomes

1. find the positive definite solution $\bar{P}_+$ of the algebraic Riccati equation (10.13),
2. simulate the LTI system (10.12) with $P(t) = \bar{P}_+$, for all t,
3. simulate the LTI system (10.10) with $P(t) = \bar{P}_+$ for all t.

10.4 Solution of the Filtering Problem

Analogously to the derivation of (10.7) in Section 10.3, now we derive an equivalent model to the EIV model representation in the form (10.2). Substitute $u - \tilde{u}$ for $\bar{u}$ and $y - \tilde{y}$ for $\bar{y}$ (see (EIV)) in (10.1):

$$\sigma\bar{x} = A\bar{x} + Bu_{\text{d}} - B\tilde{u}, \qquad y_{\text{d}} = C\bar{x} + Du_{\text{d}} - D\tilde{u} + \tilde{y}.$$

Then define a "fake" process noise v_1 and measurement noise v_2 by

$$v_1 := -B\tilde{u} \quad \text{and} \quad v_2 := -D\tilde{u} + \tilde{y}.$$

The resulting system

$$\sigma\bar{x} = A\bar{x} + Bu_{\text{d}} + v_1, \qquad y_{\text{d}} = C\bar{x}(t) + Du_{\text{d}} + v_2 \quad (10.14)$$

is in the form (10.2), where Q, S, and R are given in (10.3).

The Kalman filter corresponding to the modified system (10.14) with the covariance (10.3) is

$$\sigma z = A_{\rm kf} z + B_{\rm kf} w_{\rm d}, \qquad \hat{x} = C_{\rm kf} z + D_{\rm kf} w_{\rm d}, \tag{10.15}$$

where

$$\begin{aligned} A_{\rm kf}(t) &= \big(A - K(t)C\big), \quad B_{\rm kf}(t) = \big[B - K(t)D \;\; K(t)\big], \\ C_{\rm kf}(t) &= I - P(t)C^\top \big(CP(t)C^\top + R\big)^{-1} C, \\ D_{\rm kf}(t) &= P(t)C^\top \big(CP(t)C^\top + R\big)^{-1} \big[-D \;\; I \big], \\ K(t) &= \big(AP(t)C^\top + S\big)\big(CP(t)C^\top + R\big)^{-1}, \end{aligned} \tag{10.16}$$

and

$$P(t+1) = AP(t)A^\top - \big(AP(t)C^\top + S\big)\big(CP(t)C^\top + R\big)^{-1}\big(AP(t)C^\top + S\big)^\top + Q.$$

We call (10.15) the *modified Kalman filter*. It recursively solves (10.7) (which is equivalent to (10.14)) for the last block entry of the unknown $\bar{x}$. The solution is in the sense of the WLS problem

$$\min_{\hat{x},\hat{e}} \hat{e}^\top \Big(\big[\mathsf{B} \quad \mathsf{C}\big] \mathbf{V}_{\tilde{w}} \big[\mathsf{B} \quad \mathsf{C}\big]^\top \Big)^{-1} \hat{e} \quad \text{subject to} \quad \mathsf{y}_{\rm d} + \mathsf{B} u_{\rm d} = \mathsf{A}\hat{x} + \hat{e},$$

which is an equivalent optimization problem to the EIV smoothing problem (10.8). Therefore, the EIV filtering problem is solved by the modified Kalman filter.

Theorem 10.6. *The solution of the EIV filtering problem is $A_{\rm f} = A_{\rm kf}$, $B_{\rm f} = B_{\rm kf}$, $C_{\rm f} = C_{\rm kf}$, and $D_{\rm f} = D_{\rm kf}$, defined in (10.16).*

Note 10.7 (Suboptimal filtering) One can replace the time-varying Kalman filter with the (suboptimal) time-invariant filter, obtained by replacing $P(t)$ in (10.15) with the positive definite solution $\bar{P}_+$ of the algebraic Riccati equation

$$\bar{P} = A\bar{P}A^\top - \big(A\bar{P}C^\top + S\big)\big(C\bar{P}C^\top + R\big)^{-1}\big(A\bar{P}C^\top + S\big)^\top + Q.$$

Equivalently, one can argue that the time-invariant filter is optimal when T goes to infinity.

Note 10.8 (Optimal estimation of the input/output signals) Up to now we were interested in optimal filtering in the sense of state estimation. The optimal estimates of the input and the output, however, can be derived from the modified Kalman filter. The state estimate $\hat{x}$, the one-step-ahead prediction $z(t+1)$, and the optimal input estimate $\hat{u}$ satisfy the equation

$$z(t+1) = A\hat{x}(t) + B\hat{u}(t). \tag{10.17}$$

Then we can find $\hat{u}$ *exactly* from $\hat{x}$ and $z(t+1)$, obtained from the modified Kalman filter (10.15). In fact, (10.17) and the Kalman filter equations imply that

$$\hat{u}(t) = E(t)z(t) + F(t)w_{\rm d}(t), \tag{10.18}$$

where $E(t) := -V_{\tilde{u}} D^\top \big(CP(t)C^\top + R(t)\big)^{-1} C$ and

$$F(t) := \Big[I - V_{\tilde{u}} D^\top \big(CP(t)C^\top + R\big)^{-1} D \quad , \quad V_{\tilde{u}} D^\top \big(CP(t)C^\top + R\big)^{-1} \Big].$$

The optimal output estimate is

$$\hat{y}(t) = \big(CC_{\text{kf}}(t) + DE(t)\big) z(t) + \big(CD_{\text{kf}}(t) + DF(t)\big) w_{\text{d}}(t). \tag{10.19}$$

Appending the output equation of the EIV filter (10.5) with (10.18) and (10.19), we have an explicit solution of the EIV filtering problem of [GDS03] as a (modified) Kalman filter.

Note 10.9 (Misfit/latency) More general estimation problems occur when $\bar{w}$ is generated by the stochastic model (10.2) with a noise covariance matrix

$$V_v := \operatorname{cov} \left(\begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix} \right),$$

and the data w_{d}, available for estimation, is generated by the EIV model (EIV). The EIV smoothing and filtering problems can be defined in this case analogously to Problems 10.1 and 10.2, and the results of the chapter can be repeated mutatis mutandis for the new problems. The final result is the equivalence of the EIV filter to the modified Kalman filter (10.15)–(10.16), with the only difference that now

$$\begin{bmatrix} Q & S \\ S^\top & R \end{bmatrix} = V_v + \begin{bmatrix} -B & 0 \\ -D & I \end{bmatrix} \begin{bmatrix} V_{\tilde{u}} & \\ & V_{\tilde{y}} \end{bmatrix} \begin{bmatrix} -B & 0 \\ -D & I \end{bmatrix}^\top.$$

The more general setup is attractive because the noises v_1, v_2 have different interpretation from that of $\tilde{w}$. The former models the *latency* contribution and the latter models the *misfit* contribution; see [LD01, MWD02].

10.5 Simulation Examples

We illustrate numerically the results of the chapter. The parameters of the input/state/output representation of $\mathscr{B}$ are

$$A = \begin{bmatrix} 0.6 & -0.45 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0.48429 & -0.45739 \end{bmatrix}, \quad \text{and} \quad D = 0.5381.$$

The time horizon is $T = 100$, the initial state is $\bar{x}_{\text{ini}} = 0$, the input signal is a normal white noise sequence with unit variance, and $V_{\tilde{u}} = V_{\tilde{y}} = 0.4$.

The estimate of the EIV filter is computed directly from the definition; i.e., we solve a sequence of smoothing problems with increasing time horizon. Every smoothing problem is as a WLS problem (10.6). The last block entries of the obtained sequence of solutions form the EIV filter state estimate.

We compare the EIV filter estimate with the estimate of the modified Kalman filter (10.15). The state estimate $\hat{x}_{\text{kf}}$ obtained by the modified Kalman filter is up to numerical errors equal to the state estimate $\hat{x}_{\text{f}}$ obtained by the EIV filter, $\|\hat{x}_{\text{kf}} - \hat{x}_{\text{f}}\| < 10^{-14}$. This is the desired numerical verification of the theoretical result. The absolute errors of estimation $\|\hat{x} - \bar{x}\|^2$, $\|\hat{u} - \bar{u}\|^2$, $\|\hat{y} - \bar{y}\|^2$ for all estimation methods presented in the chapter are given in Table 10.1.

Table 10.1. *Comparison of the absolute errors of the state, input, and output estimates for all methods (MKF—modified Kalman filter).*

Method	$\|\hat{x} - \bar{x}\|^2$	$\|\hat{u} - \bar{u}\|^2$	$\|\hat{y} - \bar{y}\|^2$
optimal smoothing	75.3981	29.2195	15.5409
optimal filtering	75.7711	35.5604	16.4571
time-varying MKF	75.7711	35.5604	16.4571
time-invariant MKF	76.1835	35.7687	16.5675
noisy data	116.3374	42.4711	41.2419

10.6 Conclusions

We considered optimal EIV estimation problems for discrete-time LTI systems. A recursive solution for the smoothing problem is derived. The filtering problem is solved via a modified Kalman filter. The equivalence between the EIV filter and the modified Kalman filter is established algebraically using explicit state space representation of the system. The optimal estimate of the input is a linear function of the optimal state estimate, so that it is obtained by an extra output equation of the modified Kalman filter. The results are extended to the case when the system is driven by a measured and an unobserved input.

Chapter 11

Approximate System Identification

The following identification problem is considered:

> Minimize the 2-norm of the difference between a given time series and an approximating one under the constraint that the approximating time series is a trajectory of an LTI system of a fixed complexity.

The complexity is measured by the input cardinality and the lag. The question leads to the global total least squares problem ($\text{TLS}_{R(z)}$) and alternatively can be viewed as maximum likelihood identification in the EIV setting. Multiple time series and latent variables can be considered in the same setting. Special cases of the problem are autonomous system identification, approximate realization, and finite time optimal ℓ_2 model reduction.

The identification problem is related to the structured total least squares problem (STLS_X), so that it can be solved in practice by the methods developed in Chapter 4 and the software tool presented in Appendix B.2. The proposed system identification method and software implementation are tested on data sets from the database for the identification of systems (DAISY).

11.1 Approximate Modeling Problems

Figure 11.1 shows three approximate modeling problems. On top is the model reduction problem: given an LTI system $\bar{\mathscr{B}}$, find an LTI approximation $\hat{\mathscr{B}}$ of a desired *lower* complexity. A tractable solution that gives very good results in practice is balanced truncation [Moo81]. We consider finite time ℓ_2 optimal model reduction: the sequence of the first T Markov parameters of $\bar{\mathscr{B}}$ is approximated by the sequence of the corresponding Markov parameters of $\hat{\mathscr{B}}$ in the 2-norm sense.

The identification problem is similar to the model reduction one but starts instead from an observed response w_{d}. Various data collection models (the down arrows from $\bar{\mathscr{B}}$ to w_{d} and H in Figure 11.1) are possible. For example, the EIV model is $w_{\text{d}} = \bar{w} + \tilde{w}$, where $\bar{w}$ is a trajectory generated by $\bar{\mathscr{B}}$ and $\tilde{w}$ is measurement noise.

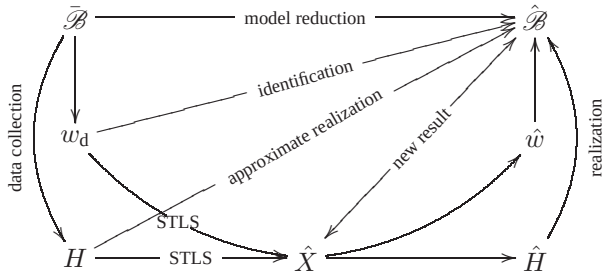


Figure 11.1. *Different problems aiming at a (low complexity) model $\hat{\mathscr{B}}$ that approximates a given (high complexity) model $\bar{\mathscr{B}}$. The time series w_{d} is an observed response and H is an observed impulse response.*

Of independent interest are the identification problems from a measurement of the impulse response $H = \bar{H} + \tilde{H}$, which is an approximate realization problem, and the autonomous system identification problem, where $\bar{w}$ and $\hat{w}$ are free responses. A classical solution to these problems is Kung's algorithm [Kun78].

The key observation that motivates the application of STLS for system identification and model reduction is that their kernel subproblem is to find a block-Hankel rank-deficient matrix $\mathscr{H}(\hat{w})$ approximating a given full-rank matrix $\mathscr{H}(w_{\mathrm{d}})$ with the same structure.

> Noniterative methods such as balanced model reduction, subspace identification, and Kung's algorithm solve the kernel approximation problem via the SVD.

For finite matrices, however, the SVD approximation of $\mathscr{H}(w_{\mathrm{d}})$ is unstructured. For this reason the algorithms based on the SVD are suboptimal with respect to an induced norm of the approximation error $\Delta w := w_{\mathrm{d}} - \hat{w}$. The STLS method, on the other hand, preserves the structure and is optimal according to this criterion.

> Our purpose is to show how system theoretic problems with misfit optimality criterion are solved as equivalent STLS problems
>
> $$\hat{X} = \arg\min_{X} \left(\min_{\hat{w}} \|w_{\mathrm{d}} - \hat{w}\| \quad \text{subject to} \quad \mathscr{S}(\hat{w}) \begin{bmatrix} X \\ -I \end{bmatrix} = 0 \right) \qquad (\mathrm{STLS}_X)$$
>
> and subsequently make use of the efficient numerical methods developed for the STLS problem.

The constraint of (STLS_X) enforces the structured matrix $\mathscr{S}(\hat{w})$ to be rank deficient, with rank at most $\mathrm{row\,dim}(X)$ and the cost function measures the distance from the given data w_{d} to its approximation $\hat{w}$. The STLS problem aims at optimal structured low-rank approximation of $\mathscr{S}(w_{\mathrm{d}})$ by $\mathscr{S}(\hat{w})$; cf. Chapter 4.

The STLS method originates from the signal processing and numerical linear algebra communities and is not widely known in the area of systems and control. The classical TLS method is known in the early system identification literature as the Koopmans–Levin

method [Lev64]. In this chapter, we show the applicability of the STLS method for system identification. We extend previous results [DR94, LD01] of the application of STLS for SISO system identification to the MIMO case and present numerical results on data sets from DAISY [DM05].

The Global Total Least Squares Problem

Let $\mathscr{M}$ be a user-specified model class and let w_{d} be an observed time series of length $T \in \mathbb{N}$. The model class restricts the maximal allowed model complexity. Within $\mathscr{M}$, we aim to find the model $\hat{\mathscr{B}}$ that best fits the data according to the misfit criterion

$$\hat{\mathscr{B}} := \arg \min_{\mathscr{B} \in \mathscr{M}} M(w_{\mathrm{d}}, \mathscr{B}), \quad \text{with} \quad M(w_{\mathrm{d}}, \mathscr{B}) := \min_{\hat{w} \in \mathscr{B}} \|w_{\mathrm{d}} - \hat{w}\|.$$

The resulting optimization problem is known as the global total least squares (GlTLS) problem [RH95].

The approach of Roorda and Heij [RH95] and Roorda [Roo95] is based on solving the inner minimization problem, the misfit computation, by isometric state representation of the system and subsequently alternating least squares or Gauss–Newton-type algorithm for the outer minimization problem. They use a state space representation with driving input. Our approach of solving the GlTLS problem is different. We use a kernel representation of the system and relate the identification problem to the STLS problem (STLS_X).

Link with the Most Powerful Unfalsified Model

In Chapter 8, we introduced the concept of the most powerful unfalsified model (MPUM). A model $\mathscr{B}$ is unfalsified by the observation w_{d} if $w_{\mathrm{d}} \in \mathscr{B}$. A model $\mathscr{B}_1$ is more powerful than $\mathscr{B}_2$ if $\mathscr{B}_1 \subset \mathscr{B}_2$. Thus the concept of the MPUM is to find the most powerful model consistent with the observations—a most reasonable and intuitive identification principle.

In practice, however, the MPUM can be unacceptably complex. For example, in the EIV setting the observation $w_{\mathrm{d}} := \big(w_{\mathrm{d}}(1), \ldots, w_{\mathrm{d}}(T)\big)$, $w_{\mathrm{d}}(t) \in \mathbb{R}^{\mathtt{w}}$, is perturbed by noise, so that with probability one the MPUM is $\mathscr{B}_{\mathrm{mpum}} = (\mathbb{R}^{\mathtt{w}})^T$; see Note 8.10. Such a model is useless because it imposes no laws.

The GlTLS problem addresses this issue by restricting the model complexity by the constraint $\hat{\mathscr{B}} \in \mathscr{M}$, where $\mathscr{M}$ is an a priori specified model class. Whenever the MPUM does not belong to $\mathscr{M}$, an approximation is needed. The idea is to

> correct the given time series as little as possible, so that the MPUM of the corrected time series belongs to $\mathscr{M}$.

This is a most reasonable adaptation of the MPUM to approximate identification. The measure of closeness is chosen as the 2-norm, which weights equally all variables over all time instants. In a stochastic setting, weighted norms can be used in order to take into account prior knowledge about nonuniform variance among the variables and/or in time.

11.2 Approximate Identification by Structured Total Least Squares

The considered approximate identification problem is defined as follows.

Problem 11.1 (GlTLS). *For given time series $w_{\mathtt{d}} \in (\mathbb{R}^{\mathtt{w}})^T$ and a complexity specification $(\mathtt{m}, \mathtt{l})$, where $\mathtt{m}$ is the maximum number of inputs and $\mathtt{l}$ is the maximum lag of the identified system, solve the optimization problem*

$$\hat{\mathscr{B}} := \arg \min_{\mathscr{B} \in \mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}}} \left(\min_{\hat{w} \in \mathscr{B}} \|w_{\mathtt{d}} - \hat{w}\| \right). \tag{GlTLS}$$

The optimal approximating time series is $\hat{w}^*$, corresponding to a global minimum point of (GlTLS), and the optimal approximating system is $\hat{\mathscr{B}}$.

The inner minimization problem of (GlTLS), i.e., the misfit $M(w_{\mathtt{d}}, \mathscr{B})$ computation, has the system theoretic meaning of finding the best approximation $\hat{w}^*$ of the given time series $w_{\mathtt{d}}$ that is a trajectory of the (fixed from the outer minimization problem) system $\mathscr{B}$. This is a *smoothing problem*.

Our goal is to express (GlTLS) as an STLS problem (STLS_X). Therefore, we need to ensure that the constraint $\mathscr{S}(\hat{w}) \left[\begin{smallmatrix} X \\ -I \end{smallmatrix} \right] = 0$ is equivalent to $\hat{w} \in \mathscr{B} \in \mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}}$. As a byproduct of doing this, we relate the parameter X in the STLS problem formulation to the system $\mathscr{B}$. The equivalence is proven under an assumption that is conjectured to hold generically in the data space $(\mathbb{R}^{\mathtt{w}})^T$.

Lemma 11.2. *Consider a time series $w \in (\mathbb{R}^{\mathtt{w}})^T$ and assume (without loss of generality) that there are natural numbers $\mathtt{l}$ and $\mathtt{p}$, $\mathtt{p} \geq 1$, and a matrix $R \in \mathbb{R}^{\mathtt{p} \times (\mathtt{l}+1)\mathtt{w}}$, such that $R\mathscr{H}_{\mathtt{l}+1}(w) = 0$. Define $R =: \begin{bmatrix} R_0 & R_1 & \cdots & R_{\mathtt{l}} \end{bmatrix}$, where $R_i \in \mathbb{R}^{\mathtt{p} \times \mathtt{w}}$, $R(z) := \sum_{i=0}^{\mathtt{l}} R_i z^i$, and $\mathscr{B} := \ker\big(R(\sigma)\big)$. Then $w \in \mathscr{B}|_{[1,T]}$ and if $R_{\mathtt{l}}$ is full rank, $\mathscr{B} \in \mathscr{L}^{\mathtt{w},\mathtt{pl}}_{\mathtt{m},\mathtt{l}}$, where $\mathtt{m} := \mathtt{w} - \mathtt{p}$.*

Proof. From the identity

$$R\mathscr{H}_{\mathtt{l}+1}(w) = 0 \quad \Longleftrightarrow \quad \textstyle\sum_{\tau=0}^{\mathtt{l}} R_\tau w(t+\tau) = 0, \quad \text{for } t = 1, \ldots, T - \mathtt{l},$$

it follows that $w \in \mathscr{B}|_{[1,T]}$.

By definition $\mathscr{B}$, is a linear system with lag $\mathbf{l}(\mathscr{B}) \leq \mathtt{l}$. The assumption that $R_{\mathtt{l}}$ is full row rank implies that $R(z)$ is row proper. Then the number of outputs of $\mathscr{B}$ is $\mathbf{p}(\mathscr{B}) = \mathtt{p}$ and therefore the number of inputs is $\mathbf{m}(\mathscr{B}) = \mathtt{w} - \mathbf{p}(\mathscr{B}) = \mathtt{m}$. Let $\mathtt{l}_i$ be the degree of the ith equation in $R(\sigma)w = 0$. The assumption that $R_{\mathtt{l}}$ is full row rank implies that $\mathtt{l}_i = \mathtt{l}$ for all i. Therefore, $\mathbf{n}(\mathscr{B}) = \sum_{i=1}^{p} \mathtt{l}_i = \mathtt{pl}$. □

The next lemma states the reverse implication.

Lemma 11.3. *Consider a time series $w \in (\mathbb{R}^{\mathtt{w}})^T$ and assume (without loss of generality) that there are natural numbers $\mathtt{l}$ and $\mathtt{m} < \mathtt{w}$ and a system $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}}$, such that $w \in \mathscr{B}|_{[1,T]}$. Let $R(\sigma)w = 0$, where $R(z) = \sum_{i=0}^{\mathtt{l}} R_i z^i$, be a shortest lag kernel representation of $\mathscr{B}$.*

Then the matrix $R := \begin{bmatrix} R_0 & R_1 & \cdots & R_{\mathtt{l}} \end{bmatrix}$ *annihilates the Hankel matrix* $\mathscr{H}_{\mathtt{l}+1}(w)$*, i.e.,* $R\mathscr{H}_{\mathtt{l}+1}(w) = 0$*. If, in addition,* $\mathbf{n}(\mathscr{B}) = \mathtt{pl}$*, then* $R_{\mathtt{l}}$ *is full row rank.*

Proof. From $w \in \mathscr{B}|_{[1,T]}$, it follows that $R\mathscr{H}_{\mathtt{l}+1}(w) = 0$.

Let $\mathtt{l}_i$ be the degree of the ith equation in $R(\sigma)w = 0$. We have $\mathtt{l}_i \leq \mathtt{l}$ and $\mathbf{n}(\mathscr{B}) = \sum_{i=1}^{p} \mathtt{l}_i$. The assumption $\mathbf{n}(\mathscr{B}) = \mathtt{pl}$ is possible only if $\mathtt{l}_i = \mathtt{l}$ for all i. Because $R(z)$ is row proper (by the shortest lag assumption of the kernel representation), the leading row coefficient matrix L has full row rank. But since $\mathtt{l}_i = \mathtt{l}$, for all i, $L = R_{\mathtt{l}}$. □

We have the following main result.

Theorem 11.4. *Let* $\mathscr{B} := \ker\big(R(\sigma)\big) \in \mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}}$*, where* $R(z) = \sum_{i=0}^{\mathtt{l}} R_i z^i$ *is row proper, and define the partitioning*

$$R_{\mathtt{l}} =: \begin{bmatrix} \overset{\mathtt{m}}{Q_{\mathtt{l}}} & \overset{\mathtt{w}-\mathtt{m}}{-P_{\mathtt{l}}} \end{bmatrix}.$$

If $P_{\mathtt{l}}$ *is nonsingular, then for any* $w \in (\mathbb{R}^{\mathtt{w}})^T$*,*

$$w \in \mathscr{B}|_{[1,T]} \iff \mathscr{H}^{\top}_{\mathtt{l}+1}(w) \begin{bmatrix} X \\ -I \end{bmatrix} = 0, \quad \textit{where} \quad X^{\top} = P_{\mathtt{l}}^{-1} \begin{bmatrix} R_0 & \cdots & R_{\mathtt{l}-1} & Q_{\mathtt{l}} \end{bmatrix}.$$

Proof. The assumption of the theorem is stronger than the assumptions of Lemmas 11.2 and 11.3 because not only is $R_{\mathtt{l}}$ required to be of full row rank but its submatrix $P_{\mathtt{l}}$ is required to have this property. In the direction of assuming $w \in \mathscr{B}|_{[1,T]}$, by Lemma 11.3, it follows that $R\mathscr{H}_{\mathtt{l}+1}(w) = 0$. Since $P_{\mathtt{l}}$ is nonsingular, $R\mathscr{H}_{\mathtt{l}+1}(w) = 0$ is equivalent to $\mathscr{H}^{\top}_{\mathtt{l}+1}(w) \left[\begin{smallmatrix} X \\ -I \end{smallmatrix}\right] = 0$, with $X^{\top} := P_{\mathtt{l}}^{-1} \begin{bmatrix} R_0 & \cdots & R_{\mathtt{l}-1} & Q_{\mathtt{l}} \end{bmatrix}$. In the opposite direction, by Lemma 11.2, $\mathscr{B} = \ker\big(\sum_{i=0}^{\mathtt{l}} R_i \sigma^i\big)$ with $\begin{bmatrix} R_0 & R_1 & \cdots & R_{\mathtt{l}} \end{bmatrix} := \begin{bmatrix} X^{\top} & -I \end{bmatrix}$. Therefore, $P_{\mathtt{l}} = I$ is nonsingular. □

Theorem 11.4 states the desired equivalence of the identification problem and the STLS problem under the assumption that the optimal approximating system $\hat{\mathscr{B}}$ admits a kernel representation

$$\hat{\mathscr{B}} = \ker\left(\sum_{i=0}^{\mathtt{l}} \hat{R}_i \sigma^i\right), \quad \hat{R}_{\mathtt{l}} := \begin{bmatrix} \hat{Q}_{\mathtt{l}} & -\hat{P}_{\mathtt{l}} \end{bmatrix} \text{ with } \hat{P}_{\mathtt{l}} \in \mathbb{R}^{\mathtt{p}\times\mathtt{p}} \text{ nonsingular.} \qquad (*)$$

We conjecture that condition $(*)$ holds true for almost all $w \in (\mathbb{R}^{\mathtt{w}})^T$. Define the subset of $(\mathbb{R}^{\mathtt{w}})^T$ consisting of all time series $w \in (\mathbb{R}^{\mathtt{w}})^T$ for which the identification problem is equivalent to the STLS problem, i.e.,

$$\Omega := \left\{ w_{\mathrm{d}} \in (\mathbb{R}^{\mathtt{w}})^T \;\middle|\; \begin{array}{l} \text{problem (GlTLS) has a unique global} \\ \text{minimizer } \hat{\mathscr{B}} \text{ that satisfies } (*) \end{array} \right\}.$$

Conjecture 11.5. *The set* Ω *is generic in* $(\mathbb{R}^{\mathtt{w}})^T$*; i.e., it contains an open subset whose complement has measure zero.*

The existence and uniqueness part of the conjecture (see the definition of Ω) is motivated in [HS99, Section 5.1]. The motivation for $(*)$ being generic is the following one.

The highest possible order of a system in the model class $\mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}}$ is $\mathtt{pl}$. Then generically in the data space $(\mathbb{R}^{\mathtt{w}})^T$, $\mathbf{n}(\hat{\mathscr{B}}) = \mathtt{pl}$. By Lemma 11.3, $\mathbf{n}(\hat{\mathscr{B}}) = \mathtt{pl}$ implies that in a kernel representation $\hat{\mathscr{B}} = \ker\left(\sum_{i=0}^{\mathtt{l}} \hat{R}_i \sigma^i\right)$, $\hat{R}_{\mathtt{l}}$ is of full row rank. But generically in $\mathbb{R}^{\mathtt{p}\times\mathtt{w}}$ the matrix $\hat{P}_{\mathtt{l}} \in \mathbb{R}^{\mathtt{p}\times\mathtt{p}}$, defined by $\hat{R}_{\mathtt{l}} =: \begin{bmatrix}\hat{Q}_{\mathtt{l}} & -\hat{P}_{\mathtt{l}}\end{bmatrix}$, is nonsingular. Although the motivation for the conjecture is quite obvious, the proof seems to be rather involved.

Properties of the Solution

The following are properties of the smoothing problem:

1. $\hat{w}$ is orthogonal to the correction $\Delta w := w_{\mathtt{d}} - \hat{w}$ and
2. Δw is generated by an LTI system $\mathscr{B}^{\perp} \in \mathscr{L}^{\mathtt{w}}_{\mathtt{p},\mathtt{l}}$.

Since the identification problem has as an inner minimization problem, the smoothing problem, the same properties hold in particular for the optimal solution of (GlTLS). These results are stated for the SISO case in [DR94] and then proven for the MIMO case in [RH95, Section VI].

Statistical properties of the identification problem (GlTLS) are studied in the literature. For the stochastic analysis, the EIV model is assumed and the basic results are consistency and asymptotic normality. Consistency in the SISO case is proven in [AY70b]. Consistency in the MIMO case is proven in [HS99] in the framework of the GlTLS problem. Complete statistical theory with practical confidence bounds is presented in [PS01] in the setting of the Markov estimator for semilinear models. Consistency of the STLS estimator for the general structure specification described in Chapter 4 is proven in [KMV05].

Numerical Implementation

A recursive solution of the smoothing problem $M(w_{\mathtt{d}}, \mathscr{B})$ is obtained by dynamic programming in Section 10.3 for the special case $D = 0$ and exactly known initial condition. An alternative derivation (for general D and unknown initial conditions) is derived by isometric state representation in [RH95]. Both solutions are derived from a system theoretic point of view. A related problem occurs in the STLS problem formulation. Because of the flexible structure specification, the inner minimization problem in the STLS formulation (STLS_X) is more general than the smoothing problem $M(w_{\mathtt{d}}, \mathscr{B})$, where a block-Hankel structure is fixed. In Chapter 4, a closed form expression is derived for the latter problem and a special structure of the involved matrices is recognized. The structure is then used on the level of the computation by employing numerical algorithms for structured matrices. The resulting computational complexity is linear in the length T of the given time series $w_{\mathtt{d}}$.

The outer minimization problem $\min_{\mathscr{B}\in\mathscr{M}} M(w_{\mathtt{d}}, \mathscr{B})$, however, is a difficult nonconvex optimization problem that requires iterative methods. Two methods are proposed in the framework of the GlTLS problem. In [RH95] an alternating least squares method is used. Its convergence is linear and can be very slow in certain cases. In [Roo95], a Gauss–Newton algorithm is proposed. For the solution of the STLS problem, a Levenberg–Marquardt algorithm is used. The convergence of all these algorithms to the desired global minimum is *not* guaranteed and depends on the provided initial approximation and the given data.

Software for solving the GlTLS problem is described in Appendix B.4.

11.3 Modifications of the Basic Problem

Input/Output Partitionings

A standard assumption in system identification is that an input/output partitioning of the variables is a priori given. Consider a $\mathtt{w} \times \mathtt{w}$ permutation matrix Π and redefine w as Πw. The first $\mathtt{m}$ variables of the redefined time series are assumed inputs and the remaining $\mathtt{p}$ variables outputs. With $\mathrm{col}(u, y) := w$ and $\begin{bmatrix}\hat{Q}(z) & -\hat{P}(z)\end{bmatrix} := \hat{R}(z)$, the kernel representation $\hat{R}(\sigma)w = 0$ becomes a left matrix fraction representation $\hat{Q}(\sigma)u = \hat{P}(\sigma)y$. The transfer function of $\hat{\mathscr{B}}$ for the fixed by Π input/output partitioning is $\hat{G}(z) := \hat{P}^{-1}(z)\hat{Q}(z)$.

Note that under the assumption $\mathbf{n}(\hat{\mathscr{B}}) = \mathbf{p}(\hat{\mathscr{B}})\mathbf{l}(\hat{\mathscr{B}})$, the state space representation

$$\hat{A} = \begin{bmatrix} 0 & \cdots & 0 & -\hat{P}_0 \\ I & & & -\hat{P}_1 \\ & \ddots & & \vdots \\ & & I & -\hat{P}_{\mathtt{l}-1} \end{bmatrix}, \hat{B} = \begin{bmatrix} \hat{Q}_0 - \hat{P}_0\hat{Q}_{\mathtt{l}} \\ \hat{Q}_1 - \hat{P}_1\hat{Q}_{\mathtt{l}} \\ \vdots \\ \hat{Q}_{\mathtt{l}-1} - \hat{P}_{\mathtt{l}-1}\hat{Q}_{\mathtt{l}} \end{bmatrix}, \hat{C} = \begin{bmatrix} 0 & \cdots & 0 & I \end{bmatrix}, \hat{D} = \hat{Q}_{\mathtt{l}}$$

is minimal. Therefore, the transition from $\hat{P}$ and $\hat{Q}$ (which is the result obtained from the optimization problem) to an input/state/output representation is trivial and requires extra computations only for the formation of the $\hat{B}$ matrix.

Conjecture 11.5 implies that generically the optimal approximation $\hat{\mathscr{B}}$ admits an input/output partitioning $\mathrm{col}(u, y) := w$, with $\Pi = I$. Moreover, we conjecture that generically $\hat{\mathscr{B}}$ admits an arbitrary input/output partitioning (i.e., $\mathrm{col}(u, y) := \Pi w$ for any permutation matrix Π).

Exact Variables

Another standard assumption is that the inputs are exact (in the EIV setting noise-free). Let $\hat{u}$ and $\hat{y}$ be the approximating input and output. The assumption that u_{d} is exact imposes the constraint $\hat{u} = u_{\mathrm{d}}$.

More generally, if some variables of w_{d} are exact, then the corresponding elements in $\hat{w}$ are fixed. In the STLS problem formulation (STLS$_X$), the exact elements of w_{d} can be separated in a block of $\mathscr{S}(w_{\mathrm{d}})$ by permuting the columns of $\mathscr{H}^{\top}_{\mathtt{l}+1}(w_{\mathrm{d}})$. The STLS package described in Appendix B.2 allows specification of exact blocks in $\mathscr{S}(w_{\mathrm{d}})$ that are not modified in the solution $\mathscr{S}(\hat{w})$. After solving the modified problem, the solution $\hat{X}$ of the original problem, with exact variables, is obtained by applying the reverse permutation.

With a given input/output partition and exact inputs, the GlTLS problem becomes the classical output error identification problem. Moreover, in the single output case the GlTLS misfit is equivalent to the cost function minimized by the prediction error methods. The following simulation example shows that the GlTLS optimal model is equivalent to the model computed by the `pem` function from the System Identification Toolbox of MATLAB, when the output error structure is specified.

Example 11.6 (Output error system identification) We use the data set "Hair dryer" from [Lju99], available via DAISY [DM05], and search for an approximate system in the model class $\mathscr{L}^2_{1,5}$. On the one hand, we use the GlTLS method, implemented by the function

Table 11.1. *Comparison of the GlTLS and prediction error methods on a SISO output error identification problem.*

Function	Time, sec	Simulation fit	GlTLS misfit
`pem`	5.5	91.31059766532992	2.27902450157299
`stlsident`	2.9	91.31059766354954	2.27902450178058

`stlsident` (see Appendix B.4) with the specification that the input is exact. On the other hand, we use the function `pem` evoked with the following calling sequence, which corresponds to output error identification:

```
sys = pem(  iddata(y,u),l,                          ...
            'nk',                   0,              ...
            'DisturbanceModel',     'None',         ...
            'SSParameterization',   'Canonical',    ...
            'InitialState',         'Estimate',     ...
            'LimitError',           0,              ...
            'Tolerance',            1e-5,           ...
            'MaxIter',              100             );
```

The identified systems by `stlsident` and `pem` are compared in Table 11.1 in terms of the simulation fit (computed by the function `compare` from the System Identification Toolbox), the GlTLS misfit, and the computation time. Note that

$$\texttt{compare}\text{'s simulation fit} = 100(1 - M_{\text{oe}}/\|y\|). \qquad \text{(FIT)}$$

Multiple Time Series

In certain cases, e.g., the approximate realization problem, multiple observed time series $w_{\text{d},1}, \ldots, w_{\text{d},N}$ are given. Assume that all time series are of the same length and define w_{d} to be the matrix valued time series $w_{\text{d}} = \begin{bmatrix} w_{\text{d},1} & \cdots & w_{\text{d},N} \end{bmatrix}$, so that $w_{\text{d}}(t) \in \mathbb{R}^{\mathtt{w} \times N}$. The only modification needed in the GlTLS solution for this case is to consider block-Hankel matrix $\mathscr{H}_{1+1}(w_{\text{d}})$ with size of blocks $\mathtt{w} \times N$ instead of $\mathtt{w} \times 1$, as for the case of a single observed time series. The software package described in Appendix B.2 treats such problems.

Known Initial Conditions

In the GlTLS problem, no prior knowledge about initial conditions is assumed. Thus the best fitting trajectory $\hat{w}$ is sought in the whole behavior $\hat{\mathscr{B}}$. If the initial conditions are a priori known, $\hat{w}$ should be searched only among the trajectories of $\hat{\mathscr{B}}$ generated by the specified initial conditions. Typical examples of identification problems with known initial conditions are approximate realization and identification from step response observations. In both cases, the initial conditions are a priori known to be zero.

Zero initial conditions can be taken into account in the identification problem by extending the given time series w_{d} with `l` zero samples. Let w_{ext} be the extended data sequences obtained in this way. In order to ensure that the approximation $\hat{w}_{\text{ext}}$ is also obtained

under zero initial conditions, the first $\mathtt{l}$ samples of w_{ext} should be preserved unmodified in $\hat{w}_{\text{ext}}$.

Note 11.7 (Known initial conditions) In the current software implementation of the GlTLS method, the specification that the $\mathtt{l}$ leading data samples are exact is *not* possible. This feature of the identification problem goes beyond the scope of the current STLS solution method and software.

Latent Inputs

The classical system identification framework [Lju99] differs from the one in this chapter in the choice of the optimization criterion and the model class. In [Lju99], an unobserved input is assumed to act on the system that generates the observations and the optimization criterion is defined as the prediction error.

An unobserved input e, of dimension $\mathtt{e}$, called *latent input*, can be accommodated in the setting of Section 11.2 by augmenting the model class $\mathscr{M} = \mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}}$ with $\mathtt{e}$ extra inputs and the cost function $\|w_{\text{d}} - \hat{w}\|^2$ with the term $\|e\|^2$. The resulting identification problem is

$$\min_{\mathscr{B} \in \mathscr{L}^{\mathtt{w}+\mathtt{e}}_{\mathtt{m}+\mathtt{e},\mathtt{l}}} \left(\min_{\hat{e},\hat{w}} \underbrace{\|w_{\text{d}} - \hat{w}\|^2}_{\text{misfit}} + \underbrace{\|\hat{e}\|^2}_{\text{latency}} \quad \text{subject to} \quad \begin{bmatrix} \hat{e} \\ \hat{w} \end{bmatrix} \in \mathscr{B} \right). \tag{M+L}$$

This problem unifies the misfit and latency descriptions of the uncertainty and is put forward by Lemmerling and De Moor [LD01]. In [LD01], it is claimed that the pure latency identification problem

$$\min_{\mathscr{B} \in \mathscr{L}^{\mathtt{w}+\mathtt{e}}_{\mathtt{m}+\mathtt{e},\mathtt{l}}} \left(\min_{\hat{e}} \|\hat{e}\|^2 \quad \text{subject to} \quad \begin{bmatrix} \hat{e} \\ w_{\text{d}} \end{bmatrix} \in \mathscr{B} \right) \tag{L}$$

is equivalent to the prediction error approach.

The misfit–latency identification problem (M+L) can easily be reformulated as an equivalent pure misfit identification problem (GlTLS). Let $w_{\text{aug}} := \operatorname{col}(e, w_{\text{d}})$, where $e := 0$ is an $\mathtt{e}$ dimensional zero time series. Then the misfit minimization problem for the time series w_{aug} and the model class $\mathscr{L}^{\mathtt{w}+\mathtt{e}}_{\mathtt{m}+\mathtt{e},\mathtt{l}}$ is equivalent to (M+L). The pure latency identification problem (L) can also be treated in our framework by considering w_{d} exact (see "Exact-variables" above) and modifying only e. Note that the latent input amounts to increasing the complexity of the model class, so that a better fit is achieved with a less powerful model.

11.4 Special Problems

In this section we consider three special identification problems in an input/output setting. In the first one the data is an observed impulse response. In the second one the data is an observed free response. In the third one the data is an exact impulse response of a high order system, i.e., a system that is not in the specified model class.

The Approximate Realization Problem

Identification from exact impulse response is the topic of (partial) realization theory; see Section 8.7. When the given data (impulse response observations) is not exact, an approximation is needed. Kung's algorithm is a well-known solution for this problem. However, Kung's algorithm is suboptimal in terms of the misfit criterion

$$M_{\text{imp}}(H_{\mathtt{d}}, \mathscr{B}) = \|H_{\mathtt{d}} - \hat{H}\|, \quad \text{where} \quad \hat{H} \text{ is an impulse response of } \mathscr{B}.$$

Note that in this special case the misfit computation does not involve optimization because the initial conditions and the input are fixed. The GlTLS problem can be used to find an optimal approximate model in terms of the misfit $M_{\text{imp}}(H_{\mathtt{d}}, \cdot)$. Next, we consider the following approximate realization problem [DM94]:

> Given a matrix valued time series $H_{\mathtt{d}} \in (\mathbb{R}^{\mathtt{p}\times\mathtt{m}})^{T+1}$ and a natural number $\mathtt{l}$, find a system $\hat{\mathscr{B}} \in \mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}}$, where $\mathtt{w} := \mathtt{m} + \mathtt{p}$, whose impulse response $\hat{H}^*$ minimizes the approximation error $\|H_{\mathtt{d}} - \hat{H}\| := \sqrt{\sum_{t=0}^{T} \|H_{\mathtt{d}}(t) - \hat{H}(t)\|_{\mathrm{F}}^2}$.

The approximate realization problem is a special GlTLS problem and can be treated as such. Now, however, the given trajectory is an observed impulse response, so that the input is a pulse and the initial conditions are zeros. For this reason the direct approach is inefficient. Moreover, known zero initial conditions cannot be specified in the current software implementation; see Note 11.7. In the rest of this section we describe an indirect solution that exploits the special features of the data and avoids specification of zero initial conditions.

The following statement is a corollary of Theorem 11.4.

Corollary 11.8. *Let $\mathscr{B} := \ker\big(R(\sigma)\big) \in \mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}}$, where $R(z) = \sum_{i=0}^{\mathtt{l}} R_i z^i$ is row proper, and define the partitioning*

$$R_{\mathtt{l}} =: \begin{bmatrix} \overset{\mathtt{m}}{Q_{\mathtt{l}}} & \overset{\mathtt{p}}{-P_{\mathtt{l}}} \end{bmatrix}.$$

If $P_{\mathtt{l}}$ is nonsingular, then for any $H \in (\mathbb{R}^{\mathtt{p}\times\mathtt{m}})^{T+1}$,

$$H \text{ is an impulse response of } \mathscr{B} \quad \Longleftrightarrow \quad \mathscr{H}^{\top}_{\mathtt{l}+1}(\sigma H) \begin{bmatrix} X \\ -I \end{bmatrix} = 0,$$

where $X^{\top} = -P_{\mathtt{l}}^{-1} \begin{bmatrix} P_0 & P_1 & \cdots & P_{\mathtt{l}} \end{bmatrix}$.

Therefore, under assumption $(*)$, the approximate realization problem can be solved as an STLS problem with structured data matrix $\mathscr{H}^{\top}_{\mathtt{l}+1}(\sigma H_{\mathtt{d}})$. Next, we show how one can obtain an input/state/output representation of the optimal approximating system $\hat{\mathscr{B}}$ from $\hat{X}$ and the $\mathtt{l}$ approximated Markov parameters $\hat{H}(1), \ldots, \hat{H}(\mathtt{l})$.

By Corollary 11.8, $\operatorname{rank}\big(\mathscr{H}^{\top}_{\mathtt{l}+1}(\sigma\hat{H})\big) =: \mathtt{n} = \mathtt{l}\mathtt{p}$. Let

$$\mathscr{H}_{\mathtt{l}+1}(\sigma\hat{H}) = \Gamma\Delta$$

be a rank revealing factorization. Since $\hat{H}$ is an impulse response of $\hat{\mathscr{B}}$, Γ and Δ must be of the form

$$\Gamma = \mathscr{O}_{\mathtt{l}+1}(\hat{A}, \hat{C}), \qquad \Delta = \mathscr{C}_{T-\mathtt{l}}(\hat{A}, \hat{B}),$$

where $\hat{\mathscr{B}} = \mathscr{B}_{\text{i/s/o}}(\hat{A}, \hat{B}, \hat{C}, \hat{D})$. (The basis of the representation is fixed by the rank revealing factorization.) We have

$$\mathscr{H}_{1+1}^{\top}(\sigma\hat{H}) \begin{bmatrix} \hat{X} \\ -I \end{bmatrix} = 0 \implies (\Gamma\Delta)^{\top} \begin{bmatrix} \hat{X} \\ -I \end{bmatrix} = 0$$
$$\implies \begin{bmatrix} \hat{X}^{\top} & -I \end{bmatrix} \Gamma = 0,$$

so that $\operatorname{col\,span}(\Gamma) \subset \ker(\begin{bmatrix} \hat{X}^{\top} & -I \end{bmatrix})$. But $\dim\big(\operatorname{col\,span}(\Gamma)\big) = \mathtt{n}$. On the other hand,

$$\dim\big(\ker(\begin{bmatrix} \hat{X}^{\top} & -I \end{bmatrix})\big) = (\mathtt{l}+1)\mathtt{p} - \mathtt{p} = \mathtt{n},$$

so that $\operatorname{col\,span}(\Gamma) = \ker(\begin{bmatrix} \hat{X}^{\top} & -I \end{bmatrix})$. Therefore, a basis for the null space of $\begin{bmatrix} \hat{X}^{\top} & -I \end{bmatrix}$ defines an observability matrix of $\hat{\mathscr{B}}$, from which $\hat{C}$ and $\hat{A}$ can be obtained up to a similarity transformation. $\hat{D} = H_{\mathrm{d}}(0)$ and $\hat{B}$ is the unique solution of the system

$$\mathscr{O}_{\mathtt{l}}(\hat{A}, \hat{C})\hat{B} = \operatorname{col}\big(\hat{H}(1), \ldots, \hat{H}(\mathtt{l})\big).$$

Example 11.9 (Approximate realization) Consider a simulation example in the EIV setup. The data $H_{\mathrm{d}} = \bar{H} + \tilde{H}$ is as a noise corrupted impulse response $\bar{H}$ of an LTI system $\bar{\mathscr{B}}$. The time horizon is $T = 50$ and the additive noise standard deviation is $\sigma = 0.25$. The true system $\bar{\mathscr{B}}$ is random stable (obtained via the MATLAB function `drss`) with $\mathtt{m} = 2$ inputs, $\mathtt{p} = 2$ outputs, and lag $\mathtt{l} = 2$. The approximate model $\hat{\mathscr{B}}$ is sought in the model class $\mathscr{L}_{\mathtt{m},\mathtt{l}}^{\mathtt{m}+\mathtt{p}}$.

We apply a direct identification from input/output data (the impulse response is extended with `l` zeros) and the indirect procedure described above. In the two cases, the optimization algorithm converges in 1.13 sec. and 0.63 sec., respectively, which shows the better efficiency of the indirect algorithm. The relative estimation errors $\|\bar{H} - \hat{H}\|/\|\bar{H}\|$ in the two cases are 0.2716 and 0.2608, respectively. (The difference is due to the wrong treatment of the initial conditions in the direct method.) For comparison, the relative error with respect to the data H_{d} is 0.9219. Figure 11.2 shows the fitting of the impulse response of the true system $\bar{\mathscr{B}}$ by the impulse response of the approximating systems.

Identification of an Autonomous System

The autonomous system identification problem is defined as follows:

> Given a time series $y_{\mathrm{d}} \in (\mathbb{R}^{\mathtt{p}})^T$ and a natural number $\mathtt{l}$, find a system $\hat{\mathscr{B}} \in \mathscr{L}_{0,\mathtt{l}}^{\mathtt{p}}$ and a vector $x_{\text{ini}}^* \in \mathbb{R}^{\mathbf{n}(\hat{\mathscr{B}})}$, such that the free response $\hat{y}^*$ of $\hat{\mathscr{B}}$ obtained under initial condition x_{ini}^* minimizes the approximation error $\|y_{\mathrm{d}} - \hat{y}\|$.

This problem is a special case of the approximate realization problem. The shifted impulse response σH of the system $\mathscr{B}_{\text{i/s/o}}(A, x_{\text{ini}}, C, \bullet)$ is equal to the free response of the system $\mathscr{B}_{\text{i/s/o}}(A, \bullet, C, \bullet)$, obtained under initial condition x_{ini}. Thus the identification problem for an autonomous system can be solved as an approximate realization problem with the obvious substitution. It is easy to generalize the autonomous system identification problem for multiple time series $y_{\mathrm{d},1}, \ldots, y_{\mathrm{d},N}$; see Note 8.27.

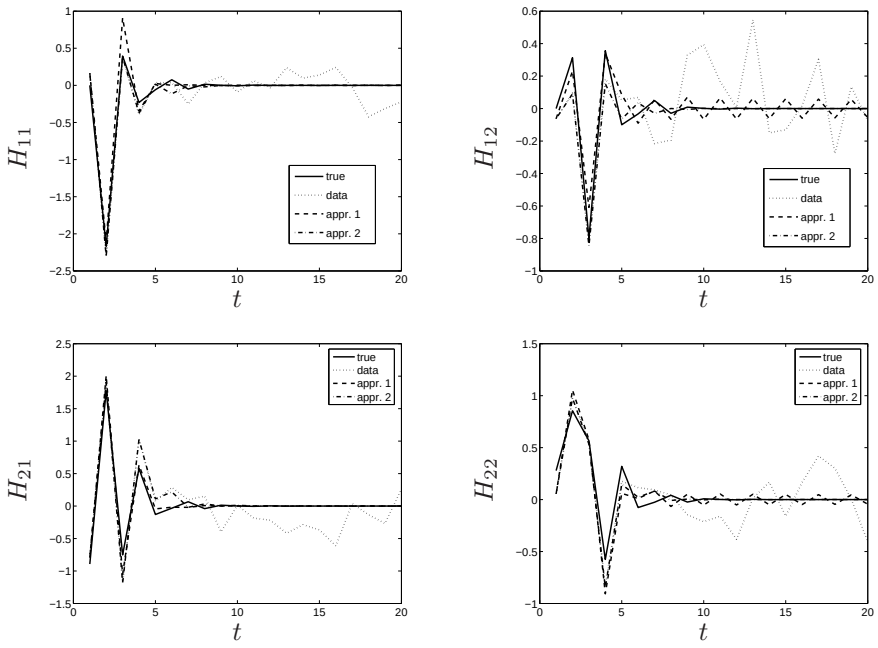


Figure 11.2. *Identification from impulse response. Solid line—exact impulse response $\bar{H}$, dotted line—data H_{d}, dashed line—approximating impulse response $\hat{H}$ from the direct approach, dashed-dotted line—approximating impulse response $\hat{H}$ from the indirect approach.*

Example 11.10 (Identification of an autonomous system) Consider the same simulation setup as in Example 11.9 with the only difference being that the true data $\bar{y}$ is a free response of length $T = 20$, obtained under random initial condition. The relative error of approximation $\|\bar{y}-\hat{y}\|/\|\bar{y}\|$ is 0.4184 versus 0.7269 for the given data y_{d}. Figure 11.3 shows the fitting of the free response of the true system $\bar{\mathscr{B}}$ by the approximating free response $\hat{y}$ of $\hat{\mathscr{B}}$.

Finite Time ℓ_2 Model Reduction

The finite time T, ℓ_2 norm of a system $\mathscr{B} \in \mathscr{L}^{\mathtt{w}}_{\mathtt{m},1}$ with an impulse response H is defined as

$$\|\mathscr{B}\|_{\ell_2,T} = \|H|_{[0,T]}\| = \sqrt{\textstyle\sum_{t=0}^{T} \|H(t)\|_{\mathrm{F}}^2}.$$

For a strictly stable system $\mathscr{B}$, $\|\mathscr{B}\|_{\ell_2,\infty}$ is well defined and is equal to its $\mathscr{H}_2$ norm.

Assume that the given time series H_{d} in the approximate realization problem is the *exact* impulse response of a higher order system $\bar{\mathscr{B}}$. Such an assumption can be made without loss of generality because any *finite* time series $H_{\mathrm{d}} \in (\mathbb{R}^{\mathtt{m}\times\mathtt{p}})^{T+1}$ can be considered as an impulse response of a system in the model class $\mathscr{L}^{\mathtt{w}}_{\mathtt{m},T}$. Then the approximate realization problem can be interpreted as the following finite time ℓ_2 model reduction problem:

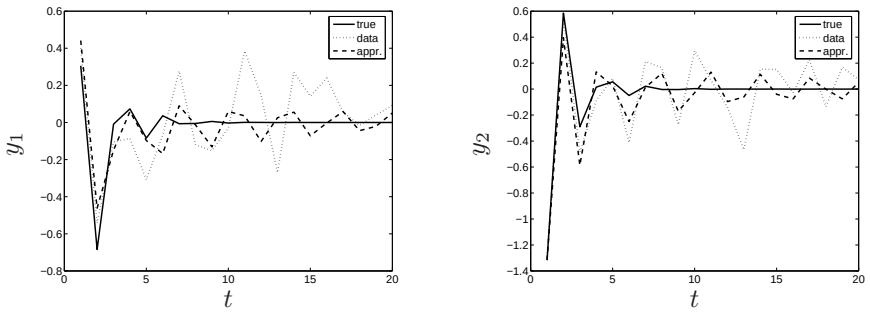


Figure 11.3. *Output-only identification. Solid line—exact trajectory $\bar{y}$, dotted line—data y_{d}, dashed line—approximating trajectory $\hat{y}$.*

> Given a system $\bar{\mathscr{B}} \in \mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}}$, a natural number $\mathtt{l}_{\mathrm{red}} < \mathtt{l}$, and a time horizon T, find a system $\hat{\mathscr{B}} \in \mathscr{L}^{\mathtt{w}}_{\mathtt{m},\mathtt{l}_{\mathrm{red}}}$ that minimizes the finite time T, ℓ_2 norm $\|\bar{\mathscr{B}} - \hat{\mathscr{B}}\|_{\ell_2,T}$ of the error system.

In the model reduction problem, the misfit is due to the low order approximation. In the approximate realization problem, assuming that the data is obtained from the EIV model, the misfit is due to the measurement errors $\tilde{H}$. The solution methods, however, are equivalent, so in this section we gave an alternative interpretation of the approximate realization problem.

Example 11.11 (Finite time ℓ_2 model reduction) The high order system $\bar{\mathscr{B}}$ is a random stable system (obtained via the MATLAB `drss` function) with $\mathtt{m} = 2$ inputs, $\mathtt{p} = 2$ outputs, and lag $\mathtt{l} = 10$. A reduced order model $\hat{\mathscr{B}}$ with lag $\mathtt{l}_{\mathrm{red}} = 1$ is sought. The time horizon T is chosen large enough for a sufficient decay of the impulse response of $\bar{\mathscr{B}}$.

Figure 11.4 shows the fitting of the impulse response of the high order system $\bar{\mathscr{B}}$ by the impulse response of the reduced order system $\hat{\mathscr{B}}$.

11.5 Performance on Real-Life Data Sets

The data base for system identification (DAISY) [DM05] is used for verification and comparison of identification algorithms. In this section, we apply the GlTLS method, described in this chapter and implemented by the software presented in Appendix B.4, on data sets from DAISY. First, we solve output error identification problems, and then, we consider the data set "Step response of a fractional distillation column", which consists of multiple vector time series.

Single Time Series Data Sets

The considered data sets are listed in Table 11.2. Since all data sets are with a given input/output partitioning, the only user-defined parameter that selects the complexity of the model class $\mathscr{M} = \mathscr{L}^{\mathtt{m}+\mathtt{p}}_{\mathtt{m},\mathtt{l}}$ is the lag $\mathtt{l}$.

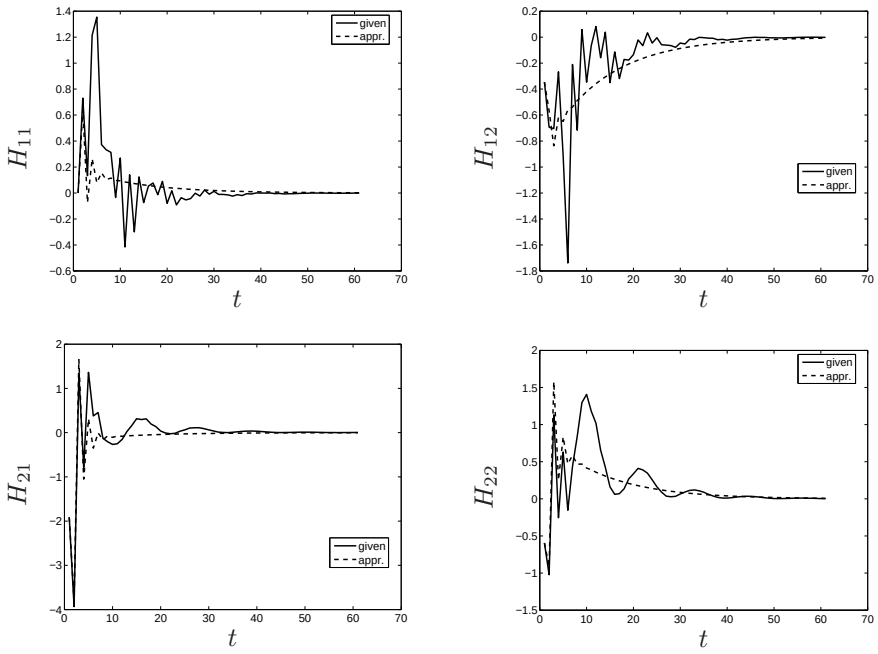


Figure 11.4. *Finite time ℓ_2 model reduction. Solid line—impulse response of the given (high-order) system, dashed line—impulse response of the reduced order system.*

The data is detrended and split into identification and validation data sets. The first 70% of the data, denoted by w_{id}, is used for identification, and the remaining 30%, denoted by w_{val}, is used for validation.

Approximate models are computed via the following methods:

`n4sid`: the N4SID method implemented in the System Identification Toolbox of MATLAB;

`stlsident`: the GlTLS method implemented by the STLS solver; and

`pem`: the prediction error method of the System Identification Toolbox of MATLAB.

Table 11.2. *Examples from DAISY. T—time horizon,* `m`*—number of inputs,* `p`*—number of outputs,* `l`*—lag.*

#	Data set name	T	`m`	`p`	`l`
1	Heating system	801	1	1	2
2	Hair dryer	1000	1	1	5
3	Flexible robot arm	1024	1	1	4
4	Heat flow density	1680	2	1	2
5	Steam heat exchanger	4000	1	1	2

The inputs are assumed exact, so that identification in the output error setting is considered. The validation is performed in terms of the misfit $M_{\text{oe}}(w_{\text{val}}, \hat{\mathscr{B}})$ obtained on the validation data set and the equivalent (see (FIT)) simulation fit computed by the function `compare`.

Note 11.12 (About the usage of the methods) The `pem` function is called with the option

```
'DisturbanceModel', 'None',
```

which specifies output error model structure. In addition, the options

```
'nk', 0, 'LimitError', 0,
```

and `'alg'` are used to disable the default for `pem` feedthrough term set to zero, robustification of the cost function, and stability constraint. (The GlTLS method does not constrain the model class by enforcing stability.)

With these options (for the single-output case), `pem` minimizes the output error misfit M_{oe}. The `stlsident` function is called with the specification that the inputs are exact, so that the GlTLS and prediction error methods solve equivalent identification problems. For both functions, we set the same convergence tolerance (`'Tolerance',1e-10`), maximum number of iterations (`'Maxiter',100`), and initial approximation (the model obtained by `n4sid`).

The identified systems by `n4sid`, `stlsident`, and `pem` are compared in Table 11.3. In all examples there is a good match between the models obtained with the `stlsident` and `pem` functions. In addition, the output error optimal model outperforms the model computed by the N4SID method. Since the criterion is checked on a part of the data that is not used for identification, there is no a priori guarantee that the output error method will outperform the N4SID method.

Identification from Step Response Measurements

Next, we consider the data set "Step response of a fractional distillation column" from DAISY. It consists of three independent time series, each one with $T = 250$ data points. The given data has a fixed input/output partitioning with $\mathtt{m} = 3$ inputs and $\mathtt{p} = 2$ outputs, so that an approximate model is sought in the model class $\mathscr{L}_2^5$. We further bound the complexity of the model class by choosing the lag $\mathtt{l} = 2$, so that the considered model class is $\mathscr{L}_{2,2}^5$.

The step response data is special because it consists of multiple time series, the inputs are exactly known, and the initial conditions are also exactly known. In order to take into account the known zero initial conditions, we precede the given time series with `l` zero samples. In order to take into account the exactly known inputs, we use the modification of the GlTLS method for time series with exact variables. Multiple time series are processed as explained in Section 11.3.

Figure 11.5 shows the data y (the measured step response) and the step response of the optimal approximating system, computed by the GlTLS method.

Table 11.3. *Comparison of the models obtained by* `n4sid`, `stlsident`, *and* `pem`.

#	Data set name	Function	Fit %	Misfit
1	Heating system	`n4sid`	51.9971	140.8018
		`stlsident`	76.0491	70.2527
		`pem`	76.0491	70.2527
2	Hair dryer	`n4sid`	88.3265	1.5219
		`stlsident`	90.8722	1.1900
		`pem`	90.8772	1.1893
3	Flexible robot arm	`n4sid`	29.5496	3.2480
		`stlsident`	96.5454	0.1593
		`pem`	96.5454	0.1593
4	Heat flow density	`n4sid`	40.7249	11.2233
		`stlsident`	83.8574	3.0565
		`pem`	83.8574	3.0565
5	Steam heat exchanger	`n4sid`	29.6890	25.5047
		`stlsident`	60.4452	14.3481
		`pem`	60.1575	14.4525

11.6 Conclusions

We generalized previous results on the application of STLS for system identification, approximate realization, and model reduction to multivariable systems. The STLS method allows us to treat identification problems without input/output partitioning of the variables and EIV identification problems. Multiple time series, latent variables, and prior knowledge about exact variables can be taken into account.

The classical identification problem, where the uncertainty is attributed solely to unobserved inputs and the observed inputs are assumed exact, is a special case of the proposed method. The relation and comparison with classical identification methods, however, have not yet been investigated.

The software tool for solving STLS problems, presented in Appendix B.2, makes the proposed identification method practically applicable. The performance of the software package was tested on data sets from DAISY. The results show that examples with a few thousands data points can be solved routinely and the optimization method is robust with respect to an initial approximation obtained from a nonoptimization based method.

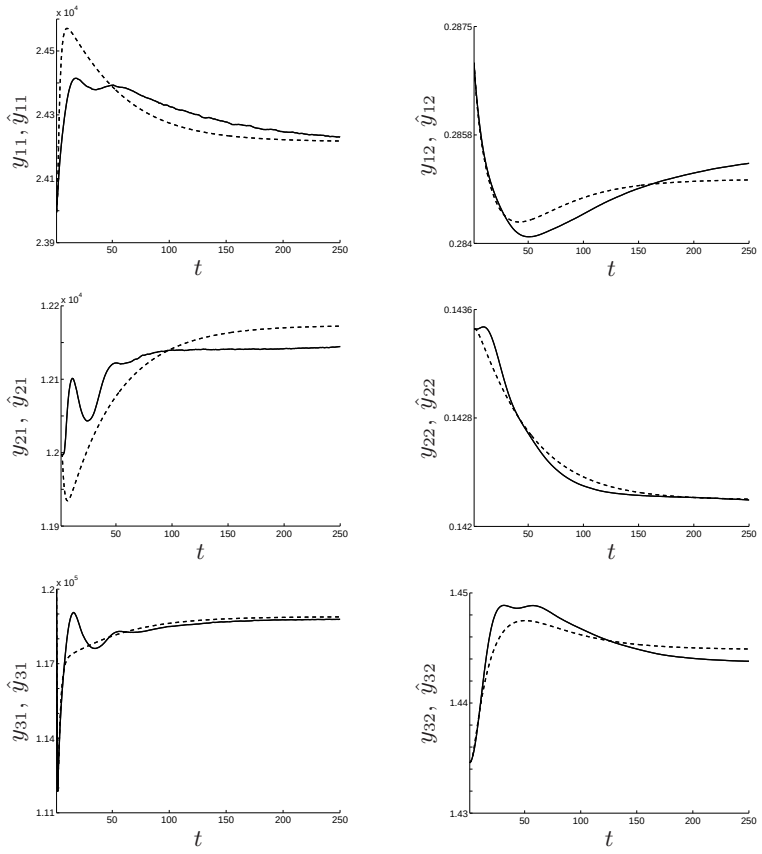


Figure 11.5. *Identification from step response measurements. Solid line—given data y, dashed line—GlTLS approximation $\hat{y}$. (y_{ij} is the step response from input i to output j.)*

Chapter 12

Conclusions

We have promoted a framework for mathematical modeling in which

1. models are disentangled from their representations and
2. data–model discrepancy is explained by correction of the data.

A basic question in our treatment is,

> When does a model in a considered model class fit the data exactly and how can we construct such a model?

This exact modeling problem leads to the notion of the most powerful unfalsified model and to identifiability conditions, i.e., under what conditions the data generating model can be recovered from the data. In the generic case when exact fit is not possible, we propose an approximate model based on misfit minimization.

The misfit approach corrects the data as little as necessary, so that the most powerful unfalsified model for the corrected data belongs to the model class. The approximate model is falsified whenever the data is not generated by a model in the model class, and the misfit is a quantitative measure of how much the model is falsified by the data.

In the errors-in-variables setting, the misfit can be chosen to be the negative log likelihood function. Such an interpretation is appealing and leads to a consistent estimator for the true model. It requires, however, strong assumptions about the data that are rarely verifiable in practice. For this reason, the approximation point of view of the modeling problem is often more appropriate than the stochastic estimation point of view.

Static approximation problems In the simplest special case of a linear static model class and unweighted misfit function, the misfit minimization problem is the classical total least squares problem. The abstract formulation is transformed to parameter optimization problems by choosing concrete representations of the model. The commonly used representations are kernel, image, and input/output.

Although concrete representations are indispensable for the actual solution of the modeling problems, their usage in problem formulations is not natural. The abstract,

representation-free formulation shows more directly what the aim of the problem is and leaves the choice of the model representation open for the solution.

The classical total least squares problem is generalized in two directions: weighted misfit function and structured data matrix. Defining the problem abstractly and then translating it to concrete parameter optimization problems, we showed links among various, seemingly unrelated, algorithms from the literature. We presented alternative algorithms that in one respect or another outperform the existing methods. However, it is a topic of future work to develop algorithms that combine all the virtues of the existing algorithms.

We presented a new flexible formulation of the structured total least squares problem that is general enough to cover various nontrivial applications and at the same time allows efficient solution methods. Algorithms with linear computational complexity in the number of data points were outlined and implemented in a software package.

Bilinear and quadratic approximation problems are solved by the adjusted least squares method, which has an analytic solution in terms of an eigenvalue decomposition. The adjusted least squares method is a stochastic latency oriented method, so in these problems we detour from the main line of the book—deterministic misfit approximation. The reason is that the adjusted least squares method leads to a significant computational simplification. In addition, although the theory of the adjusted least squares estimation is asymptotic, simulation results show that the solution is very effective even for small sample size problems.

Dynamic approximation problems In the second part of the book, we considered exact and approximate identification problems for finite time series. We gave a sharp sufficient identifiability condition: if the data generating system is controllable and an input component of the time series is persistently exciting, the most powerful unfalsified model of the data coincides with the data generating system. Exact identification algorithms find the data generating system by computing a representation of the most powerful unfalsified model.

We proposed new algorithms for exact identification of a kernel, convolution, and input/state/output representation of the most powerful unfalsified model. The latter are closely related to the deterministic subspace algorithms. However, the algorithms proposed in the book are more efficient and impose weaker assumptions on the given data. In addition, we gave system theoretic interpretation of the oblique and orthogonal projections that are used in the deterministic subspace identification.

For rough data, the exact identification problem generically identifies a trivial system that explains every time series. When a bound on the complexity of the identified system is imposed, e.g., via a bound on the model class complexity (number of inputs and lags), the exact identification problem generically has no solution. The misfit approximate modeling problem for the linear time-invariant model class is called the global total least squares problem. It is the dynamic equivalent of the classical total least squares problem. We solved the global total least squares problem by relating it to a structured total least squares problem with a block-Hankel data matrix.

Approximate identification is classically considered in a stochastic setting with the latency approach. This classical stochastic model (system with unobserved noise input), however, like the errors-in-variables model imposes unverifiable assumptions on the data. In addition, the stochastic framework addresses very indirectly the approximation issue.

Appendix A
Proofs

A.1 Weighted Total Least Squares Cost Function Gradient

Denote by Diff the differential operator. It acts on a differentiable function $M_{\text{wtls}} : U \to \mathbb{R}$, where U is an open set in $\mathbb{R}^{\mathtt{m}\times\mathtt{p}}$, and gives as a result another function, the differential of M_{wtls}, $\text{Diff}(M_{\text{wtls}}) : U \times \mathbb{R}^{\mathtt{m}\times\mathtt{p}} \to \mathbb{R}$. $\text{Diff}(M_{\text{wtls}})$ is linear in its second argument, i.e.,

$$\text{Diff}(f) := \mathrm{d}\, M_{\text{wtls}}(X, H) = \text{trace}\left(M'_{\text{wtls}}(X)H^\top\right), \tag{A.1}$$

where $M'_{\text{wtls}} : U \to \mathbb{R}^{\mathtt{m}\times\mathtt{p}}$ is the derivative of M_{wtls}, and has the property

$$M_{\text{wtls}}(X + H) = M_{\text{wtls}}(X) + \mathrm{d}\, M_{\text{wtls}}(X, H) + \mathrm{o}(\|H\|_{\mathrm{F}}), \tag{A.2}$$

for all $X \in U$ and for all $H \in \mathbb{R}^{\mathtt{m}\times\mathtt{p}}$. The notation $\mathrm{o}(\|H\|_{\mathrm{F}})$ has the usual meaning

$$g(H) = \mathrm{o}(\|H\|_{\mathrm{F}}) \quad :\Longleftrightarrow \quad g(H)/\|H\|_{\mathrm{F}} \to 0 \text{ as } \|H\|_{\mathrm{F}} \to 0.$$

We have

$$M_{\text{wtls}}(X) = \sum_{i=1}^{N} e_i^\top(X)\Gamma_i^{-1}(X)e_i(X), \qquad \text{where} \quad \Gamma_i(X) := \begin{bmatrix} X^\top & -I \end{bmatrix} W_i^{-1} \begin{bmatrix} X \\ -I \end{bmatrix}.$$

We find the derivative $M'_{\text{wtls}}(X)$ by first deriving the differential $\text{Diff}(M_{\text{wtls}})$ and then representing it in the form (A.1), from which $M'_{\text{wtls}}(X)$ is extracted. The differential of M_{wtls} is

$$\begin{aligned}
&\mathrm{d}\, M_{\text{wtls}}(X, H) \\
&= \sum_{i=1}^{N} \left(a_i^\top H\Gamma_i^{-1}(X)e_i(X) + e_i^\top(X)\Gamma_i^{-1}(X)H^\top a_i + e_i^\top(X)\,\text{Diff}\left(\Gamma_i^{-1}(X)\right)e_i(X)\right) \\
&= \sum_{i=1}^{N} \left(2\ \text{trace}\left(a_i e_i^\top(X)\Gamma_i^{-1}(X)H^\top\right) + \text{trace}\left(\text{Diff}\left(\Gamma_i^{-1}(X)\right)e_i(X)e_i^\top(X)\right)\right).
\end{aligned}$$

Using the rule for differentiation of an inverse matrix valued function, we have

$$\text{Diff}\left(\Gamma_i^{-1}(X)\right) = -\Gamma_i^{-1}(X)\,\text{Diff}\left(\Gamma_i(X)\right)\Gamma_i^{-1}(X).$$

Using the defining property (A.2), we have

$$\begin{aligned}\text{Diff}\left(\Gamma_i(X)\right) &= \text{Diff}\left(\begin{bmatrix}X^T & -I\end{bmatrix} W_i^{-1}\begin{bmatrix}X\\-I\end{bmatrix}\right)\\ &= \text{trace}\left(\begin{bmatrix}H^\top & 0\end{bmatrix} W_i^{-1}\begin{bmatrix}X\\-I\end{bmatrix} + \begin{bmatrix}X & -I\end{bmatrix} W_i\begin{bmatrix}H\\0\end{bmatrix}\right)\\ &= 2\ \text{trace}\left(\begin{bmatrix}H^\top & 0\end{bmatrix} W_i^{-1}\begin{bmatrix}X\\-I\end{bmatrix}\right).\end{aligned}$$

Let $V_i := W_i^{-1}$ and define the partitioning

$$V_i =: \begin{bmatrix}V_{a,i} & V_{ab,i}\\ V_{ba,i} & V_{b,i}\end{bmatrix}\begin{matrix}\mathtt{m}\\ \mathtt{p}\end{matrix}\quad\begin{matrix}\mathtt{m} & \mathtt{p}\end{matrix}.$$

Then

$$\text{Diff}\left(\Gamma_i(X)\right) = 2\ \text{trace}\left(H^\top(V_{a,i}X - V_{ab,i})\right).$$

Substituting backwards, we have

$$\begin{aligned}\mathrm{d}\,M_{\text{wtls}}(X,H) &= \sum_{i=1}^{N}\Big(2\ \text{trace}\left(a_i e_i^\top(X)\Gamma_i^{-1}(X)H^\top\right)\\ &\quad - 2\ \text{trace}\left(\Gamma_i^{-1}(X)H^\top(V_{a,i}X - V_{ab,i})\Gamma_i^{-1}(X)e_i(X)e_i^\top(X)\right)\Big)\\ &= \text{trace}\Bigg(\Bigg(2\sum_{i=1}^{N}\Big(a_i e_i^\top(X)\Gamma_i^{-1}(X)\\ &\quad - (V_{a,i}X - V_{ab,i})\Gamma_i^{-1}(X)e_i(X)e_i^\top(X)\Gamma_i^{-1}(X)\Big)\Bigg)H^\top\Bigg).\end{aligned}$$

Thus

$$M'_{\text{wtls}}(X) = 2\sum_{i=1}^{N}\Big(a_i e_i^\top(X)\Gamma_i^{-1}(X) - (V_{a,i}X - V_{ab,i})\Gamma_i^{-1}(X)e_i(X)e_i^\top(X)\Gamma_i^{-1}(X)\Big).$$

A.2 Structured Total Least Squares Cost Function Gradient

The differential $\text{Diff}(f_0)$ is

$$\text{Diff}(f_0) := \mathrm{d}f_0(X,H) = \text{trace}\left(f_0'(X)H^\top\right) \tag{A.3}$$

and has the property

$$f_0(X+H) = f_0(X) + \mathrm{d}f_0(X,H) + \mathrm{o}(\|H\|_{\mathrm{F}})$$

for all $X \in U$ and for all $H \in \mathbb{R}^{n\times d}$. The function $f_0' : U \to \mathbb{R}^{n\times l}$ is the derivative of f_0. As in Appendix A.1, we compute it by deriving the differential $\mathrm{Diff}(f_0)$ and representing it in the form (A.3), from which $f_0'(X)$ is extracted.

The differential of the cost function $f_0(X) = r^\top(X)\Gamma^{-1}(X)r(X)$ is (using the rule for differentiation of an inverse matrix)

$$\mathrm{d}f_0(X,H) = 2r^\top\Gamma^{-1}\begin{bmatrix} H^\top a_1 \\ \vdots \\ H^\top a_m \end{bmatrix} - r^\top\Gamma^{-1}\big(\mathrm{d}\Gamma(X,H)\big)\Gamma^{-1}r.$$

The differential of the weight matrix

$$\Gamma = V_{\tilde r} = \mathbf{E}\,\tilde r\tilde r^\top = \mathbf{E}\begin{bmatrix} X^\top\tilde a_1 - \tilde b_1 \\ \vdots \\ X^\top\tilde a_m - \tilde b_m \end{bmatrix}\begin{bmatrix}\tilde a_1^\top X - \tilde b_1^\top & \cdots & \tilde a_m^\top X - \tilde b_m^\top\end{bmatrix},$$

where $\tilde A^\top =: \begin{bmatrix}\tilde a_1 & \cdots & a_m\end{bmatrix}$, $\tilde a_i \in \mathbb{R}^n$ and $\tilde B^\top =: \begin{bmatrix}\tilde b_1 & \cdots & b_m\end{bmatrix}$, $\tilde b_i \in \mathbb{R}^d$ is

$$\mathrm{d}\Gamma(X,H) = \mathbf{E}\begin{bmatrix} H^\top\tilde a_1 \\ \vdots \\ H^\top\tilde a_m \end{bmatrix}\tilde r^\top + \mathbf{E}\,\tilde r\begin{bmatrix}\tilde a_1^\top H & \cdots & \tilde a_m^\top H\end{bmatrix}. \tag{A.4}$$

With $M_{ij} \in \mathbb{R}^{d\times d}$ denoting the (i,j)th block of Γ^{-1},

$$\begin{aligned}\mathrm{d}f_0(X,H) &= 2\left(\sum_{i,j=1}^{N} r_i^\top M_{ij}H^\top a_j - \sum_{i,j,k,l=1}^{m} r_l^\top M_{li}H^\top\,\mathbf{E}\,\tilde a_i\tilde c_j^\top X_{\text{ext}}M_{jl}r_l\right)\\ &= 2\ \mathrm{trace}\left(\Big(\sum_{i,j=1}^{m} a_j r_i^\top M_{ij} - \sum_{i,j,k,l=1}^{m}\begin{bmatrix} I & 0\end{bmatrix}V_{\tilde c,ij}X_{\text{ext}}M_{jl}r_l r_l^\top M_{li}\Big)H^\top\right),\end{aligned}$$

so that

$$f_0'(X) = 2\left(\sum_{i,j=1}^{m} a_j r_i^\top M_{ij} - \sum_{i,j=1}^{m}\begin{bmatrix} I & 0\end{bmatrix}V_{\tilde c,ij}X_{\text{ext}}N_{ji}\right),$$

where $N_{ji}(X) := \sum_{l=1}^m M_{jl}r_l \cdot \sum_{l=1}^m r_l^\top M_{li}$.

A.3 Fundamental Lemma

Of course, $\mathscr{N}_{\mathscr{B}}^l \subseteq \ker\big(\mathscr{H}_l^\top(\tilde w)\big)$. Assume by contradiction that $\ker\big(\mathscr{H}_l^\top(\tilde w)\big) \neq \mathscr{N}_{\mathscr{B}}^l$. Then there is a lowest degree polynomial $r \in \mathbb{R}^{\mathtt{w}}[z]$, $r(z) =: r_0 + r_1 z + \cdots + r_{l-1}z^{l-1}$, that annihilates $\mathscr{H}_l^\top(\tilde w)$, i.e.,

$$\mathrm{col}^\top(r_0, r_1, \ldots, r_l)\mathscr{H}_l(\tilde w) = 0,$$

but is not an element of $\mathscr{N}_{\mathscr{B}}^{l}$.

Consider $\mathscr{H}_{l+\mathtt{n}}(\tilde{w})$. Then

$$\ker\big(\mathscr{H}_{l+\mathtt{n}}^{\top}(\tilde{w})\big) = \operatorname{image}\big(r^{(1)}(z), zr^{(1)}(z), \dots, z^{l+\mathtt{n}-\mu_1} r^{(1)}(z)\ ;\ \dots\ ; \\ r^{(\mathtt{p})}(z), zr^{(\mathtt{p})}(z), \dots, z^{l+\mathtt{n}-\mu_{\mathtt{p}}-1} r^{(\mathtt{p})}(z)\ ;\ r(z), zr(z), \dots, z^{\mathtt{n}} r(z)\big).$$

Note that $r(z), zr(z), \dots, z^{\mathtt{n}}r(z)$ are additional elements due to the extra annihilator r. If all these polynomial vectors were linearly independent on $\mathbb{R}$, then the dimension of $\ker\big(\mathscr{H}_{l+\mathtt{n}}(\tilde{w})\big)$ would be (at least) $\mathtt{p}(l+\mathtt{n})+1$. But the persistency of excitation assumption implies that the number of linearly independent rows of $\mathscr{H}_{l+\mathtt{n}}(\tilde{w})$ is at least $\mathtt{m}(l+\mathtt{n})$, so that

$$\dim\Big(\ker\big(\mathscr{H}_{l+\mathtt{n}}(\tilde{w})\big)\Big) \leq \mathtt{p}(l+\mathtt{n}).$$

Therefore, not all of these elements are linearly independent. By Lemma 7.5 and the assumption that R is row proper, the generators $r^{(1)}, \dots, r^{(\mathtt{p})}$ and all their shifts are linearly independent. It follows that there is $1 \leq k \leq \mathtt{n}$, such that

$$z^k r(z) \in \operatorname{image}\big(r^{(1)}(z), zr^{(1)}(z), \dots, z^{l+\mathtt{n}-\mu_1} r^{(1)}(z)\ ;\ \dots\ ; \\ r^{(\mathtt{p})}(z), zr^{(\mathtt{p})}(z), \dots, z^{l+\mathtt{n}-\mu_{\mathtt{p}}-1} r^{(\mathtt{p})}(z)\ ;\ r(z), zr(z), \dots, z^{k-1} r(z)\big).$$

Therefore, there are $g \in \mathbb{R}[z]$ of degree $k \geq 1$ and $f \in \mathbb{R}^{1\times\mathtt{p}}[z]$, such that

$$g(z)r(z) = f(z)R(z).$$

Let λ be a root of $g(z)$. Then $f(\lambda)R(\lambda) = 0$, but by the controllability assumption, $\operatorname{rank}\big(R(\lambda)\big) = \mathtt{p}$ for all $\lambda \in \mathbb{C}$ and, consequently, $f(\lambda) = 0$. Therefore, with

$$g(z) = (z-\lambda)g'(z) \qquad \text{and} \qquad f(z) = (z-\lambda)f'(z),$$

we obtain

$$g'(z)r(z) = f'(z)R(z).$$

Proceeding with this degree lowering procedure yields $r(z) = f(z)R(z)$ and contradicts the assumption that r was an additional annihilator of $\mathscr{H}_l(w)$. Therefore, $\mathscr{H}_l(w)$ had the correct left kernel and therefore $\mathscr{N}_{\mathscr{B}}^{l} = \ker\big(\mathscr{H}_l^{\top}(\tilde{w})\big)$.

A.4 Recursive Errors-in-Variables Smoothing

By the dynamic programming principle (10.9),

$$V_t\big(\hat{x}(t)\big) = \min_{\hat{u}(t)} \Bigg(\begin{bmatrix} \hat{u}(t) \\ 1 \end{bmatrix}^{\top} \begin{bmatrix} V_{\tilde{u}}^{-1} & -V_{\tilde{u}}^{-1} u_{\mathtt{d}}(t) \\ * & u_{\mathtt{d}}(t)^{\top} V_{\tilde{u}}^{-1} u_{\mathtt{d}}(t) \end{bmatrix} \begin{bmatrix} \hat{u}(t) \\ 1 \end{bmatrix} \\ + \begin{bmatrix} \hat{x}(t) \\ 1 \end{bmatrix}^{\top} \begin{bmatrix} C^{\top} V_{\tilde{y}}^{-1} C & -C^{\top} V_{\tilde{y}}^{-1} y_{\mathtt{d}}(t) \\ * & y_{\mathtt{d}}(t)^{\top} V_{\tilde{y}}^{-1} y_{\mathtt{d}}(t) \end{bmatrix} \begin{bmatrix} \hat{x}(t) \\ 1 \end{bmatrix} + V_{t+1}\big(A\hat{x}(t) + B\hat{u}(t)\big) \Bigg), \quad \text{(A.5)}$$

where the $*$'s indicate the symmetric blocks in the matrices. Using induction, we prove that the value function V_t is quadratic for all t. At the final moment of time T, $V_T \equiv 0$ and thus it is trivially quadratic. Assume that V_{t+1} is quadratic for $t \in \{0, 1, \ldots, T\}$. Then there are $P_{t+1} \in \mathbb{R}^{\mathtt{n}\times\mathtt{n}}$, $s_{t+1} \in \mathbb{R}^{\mathtt{n}\times 1}$, and $v_{t+1} \in \mathbb{R}^{1\times 1}$, such that

$$V_{t+1}\big(\hat{x}(t)\big) = \begin{bmatrix}\hat{x}(t)\\ 1\end{bmatrix}^\top \begin{bmatrix}P_{t+1} & s_{t+1}\\ s_{t+1}^\top & v_{t+1}\end{bmatrix}\begin{bmatrix}\hat{x}(t)\\ 1\end{bmatrix}, \quad \text{for all } \hat{x}(t). \tag{A.6}$$

From (A.5) and (A.6), we have

$$\begin{aligned} V_t\big(\hat{x}(t)\big) = \min_{\hat{u}(t)} \Bigg(& \begin{bmatrix}\hat{u}(t)\\ 1\end{bmatrix}^\top \begin{bmatrix}V_{\tilde{u}}^{-1} & -V_{\tilde{u}}^{-1}u_{\mathrm{d}}(t)\\ * & u_{\mathrm{d}}(t)^\top V_{\tilde{u}}^{-1}u_{\mathrm{d}}(t)\end{bmatrix}\begin{bmatrix}\hat{u}(t)\\ 1\end{bmatrix} \\ & + \begin{bmatrix}\hat{x}(t)\\ 1\end{bmatrix}^\top \begin{bmatrix}C^\top V_{\tilde{y}}^{-1}C & -C^\top V_{\tilde{y}}^{-1}y_{\mathrm{d}}(t)\\ * & y_{\mathrm{d}}(t)^\top V_{\tilde{y}}^{-1}y_{\mathrm{d}}(t)\end{bmatrix}\begin{bmatrix}\hat{x}(t)\\ 1\end{bmatrix} \\ & + \begin{bmatrix}A\hat{x}(t)+B\hat{u}(t)\\ 1\end{bmatrix}^\top \begin{bmatrix}P_{t+1} & s_{t+1}\\ s_{t+1}^\top & v_{t+1}\end{bmatrix}\begin{bmatrix}A\hat{x}(t)+B\hat{u}(t)\\ 1\end{bmatrix}\Bigg). \end{aligned} \tag{A.7}$$

The function to be minimized in (A.7) is a convex quadratic function of $\hat{u}(t)$,

$$\begin{bmatrix}\hat{u}(t)\\ 1\end{bmatrix}^\top \begin{bmatrix}B^\top P_{t+1}B + V_{\tilde{u}}^{-1} & B^\top P_{t+1}A\hat{x}(t) + B^\top s_{t+1} - V_{\tilde{u}}^{-1}u_{\mathrm{d}}(t)\\ * & \begin{bmatrix}\hat{x}(t)\\ 1\end{bmatrix}^\top M(t)\begin{bmatrix}\hat{x}(t)\\ 1\end{bmatrix}\end{bmatrix}\begin{bmatrix}\hat{u}(t)\\ 1\end{bmatrix},$$

where

$$M(t) := \begin{bmatrix}A^\top P_{t+1}A + C^\top V_{\tilde{y}}^{-1}C & A^\top s_{t+1} - C^\top V_{\tilde{y}}^{-1}y_{\mathrm{d}}(t)\\ * & v_{t+1} + y_{\mathrm{d}}(t)^\top V_{\tilde{y}}^{-1}y_{\mathrm{d}}(t) + u_{\mathrm{d}}(t)^\top V_{\tilde{u}}^{-1}u_{\mathrm{d}}(t)\end{bmatrix}$$

so the minimizing $\hat{u}(t)$ is

$$\hat{u}(t) = -\big(B^\top P_{t+1}B + V_{\tilde{u}}^{-1}\big)^{-1}\big(B^\top P_{t+1}A\hat{x}(t) + B^\top s_{t+1} - V_{\tilde{u}}^{-1}u_{\mathrm{d}}(t)\big). \tag{A.8}$$

Substituting (A.8) back into (A.5), we have

$$\begin{aligned} V_t(\hat{x}(t)) = & \begin{bmatrix}\hat{u}(t)\\ 1\end{bmatrix}\begin{bmatrix}B^\top P_{t+1}B + V_{\tilde{u}}^{-1} & B^\top P_{t+1}A\hat{x}(t) + B^\top s_{t+1} - V_{\tilde{u}}^{-1}u_{\mathrm{d}}(t)\\ * & u_{\mathrm{d}}(t)^\top V_{\tilde{u}}^{-1}u_{\mathrm{d}}(t)\end{bmatrix}\begin{bmatrix}\hat{u}(t)\\ 1\end{bmatrix} \\ & + \begin{bmatrix}\hat{x}(t)\\ 1\end{bmatrix}^\top \begin{bmatrix}A^\top P_{t+1}A + C^\top V_{\tilde{y}}^{-1}C & A^\top s_{t+1} - C^\top V_{\tilde{y}}^{-1}y_{\mathrm{d}}(t)\\ * & v_{t+1} + y_{\mathrm{d}}(t)^\top V_{\tilde{y}}^{-1}y_{\mathrm{d}}(t)\end{bmatrix}\begin{bmatrix}\hat{x}(t)\\ 1\end{bmatrix}, \end{aligned}$$

which is a quadratic function of $\hat{x}(t)$,

$$V_t(\hat{x}(t)) = \begin{bmatrix}\hat{x}(t)\\ 1\end{bmatrix}^\top \begin{bmatrix}P_t & s_t\\ s_t^\top & v_t\end{bmatrix}\begin{bmatrix}\hat{x}(t)\\ 1\end{bmatrix}, \quad \text{for all } \hat{x}(t),$$

with P_t and s_t given in (10.11) and (10.12), respectively. By induction, V_t is quadratic for $t = 0, 1, \ldots, T$.

Appendix B
Software

This appendix describes a software implementation of the algorithms in the book. Except for the STLS solver, presented in Section B.2, all functions are written in MATLAB code. For maximum efficiency, the STLS solver is written in C with calls to BLAS, LAPACK, and SLICOT. The C function, however, is also callable from MATLAB via a mex file interface.

The software and related information are available from the following address:

`http://www.esat.kuleuven.be/~imarkovs/book.html`

B.1 Weighted Total Least Squares

Introduction

The weighted total least squares toolbox, presented in this section, contains MATLAB functions (m-files) for data approximation by linear static models. The data is a collection of N, $\mathtt{d}$-dimensional real vectors $d_1, \ldots, d_N \in \mathbb{R}^{\mathtt{d}}$, gathered in a matrix $D := \begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix} \in \mathbb{R}^{\mathtt{d} \times N}$, and a linear static model $\mathscr{B}$ for D is a subspace of $\mathbb{R}^{\mathtt{d}}$. The natural number $\mathtt{m} := \dim(\mathscr{B})$ is a measure of the model complexity and $\mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$ denotes the set of all linear static models with $\mathtt{d}$ variables of dimension *at most* $\mathtt{m}$.

A linear static model $\mathscr{B} \in \mathscr{L}^{\mathtt{d}}_{\mathtt{m},0}$ can be represented as a kernel or image of a matrix or in an input/output form; see Section 3.2. A representation of the model yields a parameterization. The model is described by equations that depend on parameter, and to a given parameter corresponds a unique model. For a given model and a chosen representation, however, the corresponding parameter might not be unique. The parameters R and P in a kernel and image representation are in general not unique, but the parameter X in the special input/output representation $\mathscr{B}_{\text{i/o}}(X)$ is unique.

We use the shorthand notation $\begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix} \in \mathscr{B} \subseteq \mathscr{U}$ for $d_i \in \mathscr{B}$, $i = 1, \ldots, N$. If $D \in \mathscr{B}$, the model $\mathscr{B}$ fits the data D exactly. If $D \notin \mathscr{B}$, the model $\mathscr{B}$ fits the data D only approximately. For optimal approximate modeling, the following misfit function is

Table B.1. *Special cases of the weighted total least squares problem (WTLS).*

Special case		Name	Acronym
$W_i = \sigma^2 I$	$\sigma \in \mathbb{R}_+$	total least squares	TLS
$W_i = \mathrm{diag}(w)$	$w \in \mathbb{R}_+^{\mathtt{d}}$	element-wise generalized TLS	EWGTLS
$W_i = W$	$W > 0$	generalized total least squares	GTLS
$M_{\text{wtls}} = M_{\text{gtls2}}$	$W_{\text{l}}, W_{\text{r}} > 0$ diag.	EWGTLS with two side weighting	EWGTLS2
$M_{\text{wtls}} = M_{\text{gtls2}}$	$W_{\text{l}}, W_{\text{r}} > 0$	GTLS with two side weighting	GTLS2
$W_i = \mathrm{diag}(w_i)$	$w_i \in \mathbb{R}_+^{\mathtt{d}}$	element-wise weighted TLS	EWTLS

adopted:

$$M_{\text{wtls}}\big(\begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix}, \mathscr{B}\big) := \min_{\hat{d}_1,\ldots,\hat{d}_N \in \mathscr{B}} \sqrt{\sum_{i=1}^{N} (d_i - \hat{d}_i)^\top W_i (d_i - \hat{d}_i)},$$

where $W_1, \ldots, W_N$ are given positive definite matrices. The weighted total least squares (WTLS) misfit $M_{\text{wtls}}(D, \mathscr{B})$ between the data D and a model $\mathscr{B} \in \mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}$ is a measure of how much the model fails to fit the data exactly. The considered approximate modeling problem is as follows:

> Given the data matrix $D = \begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix} \in \mathbb{R}^{\mathtt{d}\times N}$, a complexity bound $\mathtt{m}$, and positive definite weight matrices $W_1, \ldots, W_N$, find an approximate model
>
> $$\hat{\mathscr{B}}_{\text{wtls}} := \arg \min_{\hat{\mathscr{B}} \in \mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}} M_{\text{wtls}}(D, \hat{\mathscr{B}}). \qquad \text{(WTLS)}$$

The special cases listed in Table B.1 allow for special solution methods and are treated separately.

Note B.1 (FWTLS) The following weighted total least squares problem, called fully weighted total least square (FWTLS) problem

$$\hat{\mathscr{B}}_{\text{fwtls}} := \arg \min_{\hat{\mathscr{B}} \in \mathscr{L}_{\mathtt{m},0}^{\mathtt{d}}} \min_{\hat{D} \in \hat{\mathscr{B}}} \mathrm{vec}^\top (D - \hat{D}) W \, \mathrm{vec}(D - \hat{D}), \quad \text{where} \quad W \in \mathbb{R}^{\mathtt{d}N \times \mathtt{d}N},\ W > 0,$$

is also considered. It includes (WTLS) as a special case with $W = \mathrm{diag}(W_1, \ldots, W_N)$. The FWTLS problem, however, does not allow for efficient computational methods and its solution is prohibitive already for small sample size problems (say $\mathtt{d} = 10$ and $N = 100$). For this reason the FWTLS problem is not the central problem of interest and is included only for completeness.

Algorithms

The special cases listed in Table B.1 have increased generality from top to bottom. The more general the problem is, however, the more computationally expensive its solution is. The TLS and GTLS problems allow for analytic solutions in terms of the SVD. The more

general EWTLS, WTLS, and FWTLS problems have no similar analytic solutions and use less robust iterative solution methods.

The SVD method is computationally faster than the alternative iterative optimization methods and theoretically characterizes all globally optimal solutions. In contrast, the iterative optimization methods (used in the package) compute one locally optimal solution. (The algorithm of Premoli and Rastello [PR02, MRP$^+$05] is not globally convergent to a local solution, so that for particular initial approximations this method might not converge to a local solution. In such cases the algorithm diverges or oscillates.)

The GTLS-type problems (EWGTLS, GTLS, EWGTLS2, and GTLS2) are solved in the package via the transformation technique of Theorem 3.18. The data matrix is appropriately scaled and the corresponding TLS problem is solved for the scaled data. Then the solution of the original problem is recovered from the solution of the transformed TLS problem via the inverse transformation.

The general WTLS problem is solved via local optimization methods. The following algorithms are used/implemented in the package:

1. classical local optimization methods (from the Optimization Toolbox of MATLAB),
2. an alternating least squares algorithm,
3. the algorithm of Premoli and Rastello.

Implementation

The implementation is in MATLAB code. For problems with analytic solution, the MATLAB code is expected to compute a solution nearly as fast as an alternative code in C or FORTRAN. The general WTLS algorithms, however, are expected to benefit in terms of execution time if implemented in C or FORTRAN. The MATLAB source code could be consulted for the implementation details.

Overview of Commands

The package has three main groups of functions: transformations, misfit computations, and approximations.

The transformation functions convert a given representation of a model to an equivalent one. The considered representations are image, kernel, and input/output, so that there are in total six transformation functions among them (see Figure 3.2). In addition, a kernel or an image representation might not be minimal, so that functions that convert a given kernel or image representation to a minimal one are added. The transformation functions are summarized in Table B.2.

The misfit computation functions are used for validation: they allow the user to verify how well a given model fits given data in terms of a certain misfit function. Since the model can be specified by one of the three alternative representations—kernel, image, or input/output—all misfit functions have three versions. The following naming convention is adopted: misfit computation functions begin with `m` (for misfit), followed by the name of the approximation problem (which identifies the type of misfit to be computed), followed by a

Function	Description	
`x2r`	$X \mapsto R$	from input/output to kernel representation
`x2p`	$X \mapsto P$	from input/output to image representation
`r2p`	$R \mapsto P$	from kernel to image representation
`p2r`	$P \mapsto R$	from image to kernel representation
`r2x`	$R \mapsto X$	from kernel to input/output representation
`p2x`	$P \mapsto X$	from image to input/output representation
`minr`	$R \mapsto R_{\min}$	minimal kernel representation
`minp`	$P \mapsto P_{\min}$	minimal image representation

Table B.2. *Transformation functions.*

letter indicating the model representation: `r` for kernel, `p` for image, and `x` for input/output. Instead of a model $\mathscr{B}$, an approximating matrix $\hat{D} \in \mathbb{R}^{\mathtt{d} \times N}$ can be used for the misfit computation. In this case the last letter of the function name is `dh`.

The considered misfit functions are TLS, GTLS, GTLS2, WTLS, and FWTLS. The element-wise versions of the GTLS, GTLS2, and WTLS misfits are specified by the size of the given weight matrices: if vectors are given in `mgtls{r,p,x,dh}` and `mgtls2{r,p,x,dh}` instead of square weight matrices, then the EWGTLS and EWGTLS2 misfits are computed instead of the GTLS and GTLS2 ones. Similarly, if a $\mathtt{d} \times N$ matrix is given instead of a $\mathtt{d} \times \mathtt{d} \times N$ tensor in `mwtls{r,p,x,dh}`, then the EWTLS misfit is computed instead of the WTLS one. The general FWTLS misfit is computed by the functions `mwtls{r,p,x,dh}` if the weight matrix is of size $\mathtt{d}N \times \mathtt{d}N$. The misfit computation functions are summarized in Table B.3.

The approximation functions compute a WTLS approximation of the data. The special WTLS problems are called by special functions that are more efficient; see Table B.4. As in the misfit computation, the element-wise versions of the functions are recognized by the dimension of the weight matrices. The function `wtls` uses the quasi-Newton optimization algorithm that seems to outperform the alternatives. The alternative methods can be called by the corresponding functions; see Table B.5.

B.2 Structured Total Least Sqaures

The package uses MINPACK's Levenberg–Marquardt algorithm [Mar63] for the solution of the STLS problem (STLS_X) with the structure specification of Assumption 4.4 in its

Table B.3. *Misfit computation functions.*

Function				Description
`mtlsr`	`mtlsp`	`mtlsx`	`mtlsdh`	TLS misfit
`mgtlsr`	`mgtlsp`	`mgtlsx`	`mgtlsdh`	GTLS misfit
`mgtls2r`	`mgtls2p`	`mgtls2x`	`mgtls2dh`	GTLS2 misfit
`mwtlsr`	`mwtlsp`	`mwtlsx`	`mwtlsdh`	WTLS misfit

Table B.4. *Approximation functions.*

Function	Description
`tls`	TLS approximation
`gtls`	GTLS approximation
`gtls2`	GTLS2 approximation
`wtls`	WTLS approximation

equivalent formulation (4.4). There is no closed form expression for the Jacobian matrix $J = [\partial r_i / \partial x_j]$, where $x = \mathrm{vec}(X)$, so that the pseudo-Jacobian J_+ proposed in [GP96] is used instead of J. Its evaluation is done with computational complexity $O(m)$.

The software is written in ANSI C language. For the vector-matrix manipulations and for a C version of MINPACK's Levenberg–Marquardt algorithm, we use the *GNU Scientific Library (GSL)*. The computationally most intensive step of the algorithm—the Cholesky decomposition of the block-Toeplitz, block-banded weight matrix $\Gamma(X)$—is performed via the subroutine `MB02GD` from the SLICOT library [VSV$^+$04]. By default, the optimization algorithm is initialized with the TLS solution. Its computation is performed via the SLICOT subroutine `MB02MD`.

The package contains

- C-source code: `stls.c` and `stls.h` (the function `stls` implements Algorithm 4.3);
- MATLAB interface to the C function `stls` via C-mex file `stls.m`;
- a demo file `demo.m` with examples that illustrate the application of the STLS solver;
- user guide and papers that describe the STLS problem in more detail.

C Function

The function `stls` implements the method outlined in Section 4.5 to solve the STLS problem (STLS$_X$). Its prototype is

```
int stls(gsl_matrix* a, gsl_matrix* b, const data_struct* s,
         gsl_matrix* x, gsl_matrix* v, opt_and_info* opt)
```

Table B.5. *Auxiliary functions.*

Function	Description
`wtlsini`	initial approximation for the WTLS approximation functions
`wtlsap`	WTLS approximation by alternating projections
`wtlsopt`	WTLS approximation by classical optimization methods
`qncostderiv`	cost function and gradient for the quasi-Newton methods
`lmcostderiv`	cost function and Jacobian for the Levenberg–Marquardt method
`wtlspr`	WTLS approximation by the algorithm of [MRP$^+$05]

Description of the arguments:

- `a` and `b` are the matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times d}$, respectively, such that $\begin{bmatrix} A & B \end{bmatrix} = \mathscr{S}(p)$. We refer to the GSL reference manual for the definition of `gsl_matrix` and the functions needed to allocate and initialize variables of this type.

- `s` is the structure description $K, \mathtt{S}$ of $\mathscr{S}(p)$. The type `data_struct` is defined in `stls.h` as

```
/* structure of the data matrix C = [A B] */
#define MAXQ 10 /* maximum number of blocks in C */
typedef struct {
  int K;       /* = rowdim(block in T/H blocks) */
  int q;       /* number of blocks in C = [C1 ... Cq] */
  struct {
    char type; /* 'T'-Toeplitz, 'H'-Hankel, 'U'-unstructured,
                                                'E'-exact */
    int ncol;  /* number of columns */
    int nb;    /* = coldim(block in T/H blocks) */
  } a[MAXQ];   /* q-element array describing C1,...,Cq; */
} data_struct;
```

- `x` on input contains the initial approximation for the Levenberg–Marquardt algorithm and on exit, upon convergence of the algorithm, contains a local minimum point of the cost function f_0.

- `v` on exit contains the error covariance matrix $(J_+^\top J_+)^{-1}$ of the vectorized estimate $\hat{x} = \mathrm{vec}(\hat{X})$. It can be used for deriving confidence bounds.

- `opt` on input contains options that control the exit condition of the Levenberg–Marquardt algorithm and on exit contains information about the convergence of the algorithm. The exit condition is

$$|x_j^{(k+1)} - x_j^{(k)}| < \mathtt{epsabs} + \mathtt{epsrel}\,|x_j^{(k+1)}|, \qquad \text{for all } j = 1, \ldots, \mathtt{nd}, \quad \text{(B.1)}$$

  where $x^{(k)}$, $k = 1, 2, \ldots, \mathtt{iter} \leq \mathtt{maxiter}$, are the successive iterates, and `epsrel`, `epsabs`, `maxiter` are fields of `opt`. Convergence to the desired tolerance is indicated by a positive value of `opt.iter`. In this case, `opt.iter` is the number of iterations performed. `opt.iter = -1` indicates lack of convergence. `opt.time` and `opt.fmin` show the time in seconds used by the algorithm and the cost function f_0 value at the computed solution.

  The type `opt_and_info` is defined in `stls.h` as

```
/* optimization options and output information structure */
typedef struct {
  /* input options */
  int maxiter;
```

```
  double epsrel, epsabs;
  /* output information */
  int iter;
  double fmin;
  double time;
} opt_and_info;
```

MATLAB Mex-File

The provided C-mex file allows us to call the C solver `stls` via the MATLAB command

```
>> [xh, info, v] = stls(a, b, s, x, opt);
```

The input arguments `a`, `b`, and `s` are obligatory. `x` and `opt` are optional and can be skipped by the empty matrix `[]`. In these cases their default values are used.

Description of the arguments:

- `a` and `b` are the matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times d}$, respectively, where $\begin{bmatrix} A & B \end{bmatrix} = \mathscr{S}(p)$.
- `s` is a $q \times 3$ matrix or a structure with scalar field `k` and a $q \times 3$ matrix field `a`. In the first case, K is assumed to be 1, and in the second case it is specified by `s.k`. The array `s`, introduced in Section 4.6, is specified by `s` in the first case and by `s.a` in the second case. The first column of `s` (or `s.a`) defines the type of the blocks $C^{(1)}, \ldots, C^{(q)}$ (1 block-Toeplitz, 2 block-Hankel, 3 unstructured, 4 exact), the second column defines $n_1, \ldots, n_q$, and the third column defines $t_1, \ldots, t_q$.
- `x` is a user-supplied initial approximation. Its default value is the TLS solution.
- `opt` contains user-supplied options for the exit conditions. `opt.maxiter` defines the maximum number of iterations (default 100), `opt.epsrel` defines the relative tolerance `epsrel` (default `1e-5`), and `opt.epsabs` defines the absolute tolerance `epsabs` (default `1e-5`); see (B.1).
- `xh` is the computed solution.
- `info` is a structure with fields `iter`, `time`, and `fmin` that gives information for the termination of the optimization algorithm. These fields are the ones returned from the C function.
- `v` is the error covariance matrix $(J_+^\top J_+)^{-1}$ of the vectorized estimate $\hat{x} = \mathrm{vec}(\hat{X})$.

Compilation

The included make file, when called with argument `mex`, generates the MATLAB mex file. The GSL, BLAS, and LAPACK libraries have to be installed in advance. For their location and for the location of the `mex` command and options file, one has to edit the provided make file. Precompiled mex-files are included for Linux only.

Table B.6. *Elementary building blocks for the exact identification algorithms.*

Function	Description
`w2r`	from time series to a kernel representation
`r2pq`	from kernel representation to a left matrix fraction representation
`pq2ss`	from left matrix fraction representation to an input/state/output represent.
`uy2h`	computation of the impulse response
`uy2hblk`	block computation of the impulse response
`h2ss`	Kung's realization algorithm
`uy2y0`	computation of sequential free responses
`uy2hy0`	computation of the impulse response and sequential free responses
`y02o`	from a set of free responses to an observability matrix
`y02x`	from a set of sequential free responses to a state sequence
`uyo2ss`	from data and observability matrix to an input/state/output representation
`uyx2ss`	from data and a state sequence to an input/state/output representation
`hy02xbal`	from the impulse response and sequential free responses to a balanced state sequence

B.3 Balanced Model Identification

This section describes a MATLAB implementation of the algorithms for exact identification, presented in Chapters 8 and 9. Although the algorithms were originally designed to work with exact data, they can also be used as heuristic methods for approximate identification; see Note 8.18. By specifying the parameters $\mathtt{n}_{\max}$ and $\mathtt{l}_{\max}$ lower than the actual order and lag of the MPUM, the user obtains an approximate model in the model class $\mathscr{L}_{\mathtt{m},\mathtt{l}_{\max}}^{\mathtt{w},\mathtt{n}_{\max}}$. Another approach for deriving an approximate model via the algorithms described in this section is to do balanced model reduction of the MPUM; see Note 9.2.

The exact identification algorithms are decomposed into elementary building blocks that have independent significance. Table B.6 lists the building blocks together with short descriptions. More details can be found in the documentation of the corresponding m-files.

Table B.7 shows the implementation of the algorithms in Chapters 8 and 9 in terms of the elementary building blocks. Exceptions are Algorithms 9.5 and 9.6, which are included for completeness and are implemented as described in the original sources.

B.4 Approximate Identification

We describe MATLAB functions (m-files) for approximate LTI system identification. A discrete-time dynamical system $\mathscr{B} \subset (\mathbb{R}^{\mathtt{w}})^{\mathbb{Z}}$ is a collection of trajectories ($\mathtt{w}$-variables time series $w : \mathbb{Z} \to \mathbb{R}^{\mathtt{w}}$). No a priori distinction of the variables in inputs and outputs is made and the system is not a priori bound to a particular representation. The variables w can be partitioned into inputs u (free variables) and outputs y (dependent variables) and the system can be represented in various equivalent forms, e.g., the ubiquitous input/state/output representation

$$\sigma x = Ax + Bu, \qquad y = Cx + Du. \tag{I/S/O}$$

Table B.7. *Implementation of the algorithms in Chapters 8 and 9.*

Algorithm 8.1	`w2r`	
Algorithm 8.2	`w2r` → `r2pq` → `pq2ss`	
Algorithm 8.3	`uy2h` → `h2ss`	
Algorithm 8.4	`uy2y0` → `y02o` → `uyo2ss`	
Algorithm 8.5	`uy2y0` → `y02x` → `uyx2ss`	
Algorithm 8.6	`uy2h_blk`	
Algorithm 8.7	`uy2h`	
Algorithm 8.8	`h2ss`	
Algorithm 8.9	`uy2y0`	
Algorithm 9.1	`uy2hy0` → `hy02xbal` → `x2ss`	
Algorithm 9.2	`uy2h` → `h2ss`	(= Algorithm 8.3)
Algorithm 9.3	`uy2h` → `h2o` → `uyo2ss`	
Algorithm 9.4	`uy2hy0` → `hy02xbal` → `x2ss`	(= Algorithm 9.1)
Algorithm 9.5	`uy2ssvd`	
Algorithm 9.6	`uy2ssmr`	

The number of inputs $\mathtt{m}$, the number of outputs $\mathtt{p}$, and the minimal state dimension $\mathtt{n}$ of an input/state/output representation are invariant of the representation and in particular of the input/output partitioning.

The class of finite dimensional LTI systems with $\mathtt{w}$ variables and at most $\mathtt{m}$ inputs is denoted by $\mathscr{L}_{\mathtt{m}}^{\mathtt{w}}$. The number of inputs and the minimal state dimension specify the complexity of the system in the sense that the dimension of the restriction of $\mathscr{B}$ to the interval $[1, T]$, where $T \geq \mathtt{n}$, is a $(T\mathtt{m} + \mathtt{n})$-dimensional subspace. Equivalently, the complexity of the system can be specified by the input dimension and the *lag* of the system. The lag of $\mathscr{B}$ is the minimal natural number $\mathtt{l}$, for which there exists an $\mathtt{l}$th order difference equation

$$R_0 w(t) + R_1 w(t+1) + \cdots + R_{\mathtt{l}} w(t+\mathtt{l}) = 0 \qquad \text{(DE)}$$

representation of the system, i.e., $\mathscr{B} = \{\, w \mid \text{(DE) holds} \,\}$. The subset of $\mathscr{L}_{\mathtt{m}}^{\mathtt{w}}$ with lag at most $\mathtt{l}$ is denoted by $\mathscr{L}_{\mathtt{m},\mathtt{l}}^{\mathtt{w}}$.

The considered identification problem is the global total least squares problem [RH95, MWV+05]:

> Given a time series $w_{\mathtt{d}} \in (\mathbb{R}^{\mathtt{w}})^T$ and a complexity specification $(\mathtt{m}, \mathtt{l})$, find the system
>
> $$\hat{\mathscr{B}} := \arg \min_{\mathscr{B} \in \mathscr{L}_{\mathtt{m},\mathtt{l}}^{\mathtt{w}}} M(w_{\mathtt{d}}, \mathscr{B}), \quad \text{where} \quad M(w_{\mathtt{d}}, \mathscr{B}) := \min_{\hat{w} \in \mathscr{B}} \|w_{\mathtt{d}} - \hat{w}\|_{\ell_2}. \qquad \text{(GlTLS)}$$

The number $M(w_{\mathtt{d}}, \mathscr{B})$ is the misfit between $w_{\mathtt{d}}$ and $\mathscr{B}$. It shows how much the model $\mathscr{B}$ fails to "explain" the data $w_{\mathtt{d}}$. The optimal approximate modeling problem (GlTLS) aims to find the system $\hat{\mathscr{B}}$ in the model class $\mathscr{L}_{\mathtt{m},\mathtt{l}}^{\mathtt{w}}$ that best fits the data according to the misfit criterion.

The software presented in Section B.2 for STLS problems is the core computational tool for solving the system identification problem. In fact, the software presented in this section can be viewed as an interface to the STLS solver for the purpose of LTI system identification.

The STLS solver gives as a result a difference equation representation of the optimal approximating system $\hat{\mathscr{B}}$. The function `stlsident`, described next, converts the parameter $\hat{X}$ to the parameters $(\hat{A}, \hat{B}, \hat{C}, \hat{D})$ of an input/state/output representation of $\hat{\mathscr{B}}$. The MATLAB code of the functions in the package can be consulted for the implementation details.

Usage

The function

- `stlsident` solves the approximate identification problem (GlTLS), and

the function

- `misfit` computes the misfit $M(w_{\mathrm{d}}, \mathscr{B})$.

Both functions use the input/state/output representation (I/S/O) of the systems that are returned as an output and accepted as an input, so that they can be viewed as implementations of the following mappings:

- `stlsident`: $(w_{\mathrm{d}}, \mathtt{m}, \mathtt{l}) \mapsto (\hat{A}, \hat{B}, \hat{C}, \hat{D})$; and
- `misfit`: $\big(w_{\mathrm{d}}, (A, B, C, D)\big) \mapsto (M, \hat{w}_{\mathrm{d}})$.

The following are a special case and extensions:

- the specification $\mathtt{m} = 0$ corresponds to an output-only system identification ($\hat{\mathscr{B}}$ autonomous);
- the functions work with multiple given time series $w_k = \big(w_k(1), \ldots, w_k(T)\big)$, $k = 1, \ldots, N$; and
- some elements of w can be specified as "exact", in which case they appear unmodified in the approximation $\hat{w}$.

Using a combination of these options, one can solve approximately the realization problem, the finite time ℓ_2 model reduction problem (see [MWV+05, Section 5]), and the output error identification problem. Examples are given in Sections 11.4 and 11.5.

Calling sequences

```
[ sysh, info, wh, xini ] = stlsident( w, m, l, opt );
```

Inputs:

- `w`, the given time series w_{d}; a real MATLAB array of dimension $T \times \mathtt{w} \times N$, where T is the number of samples, `w` is the number of variables, and N is the number of time series;

- `m`, the input dimension for the identified system;
- `l`, the lag of the identified system;
- `opt`, options for the optimization algorithm:
  - `opt.exct`, (default []) a vector of indices for exact variables;
  - `opt.sys0`, (default total least squares approximation), an initial approximation: an input/state/output representation of a system, given as the MATLAB object `ss` (see `help ss`), with `m` inputs, `w-m` outputs, and order `l*(w-m)`;
  - `opt.disp`, (default `'notify'`), level of displayed information about the optimization process; the options are `'off'`—silent, `'notify'`—only if not converged, `'final'`—convergence status, `'iter'`—per iteration;
  - `opt.maxiter` (default 100), a maximum number of iterations;
  - `opt.epsrel`, `opt.epsabs`, and `opt.epsgrad` (default 10^{-5}) convergence tolerances; the convergence condition is

$$|X_{ij}^{(k+1)} - X_{ij}^{(k)}| < \texttt{opt.epsabs} + \texttt{opt.epsrel}\,|X_{ij}^{(k+1)}|, \quad \text{for all } i,j$$
$$\text{or} \qquad \|M'(X^{(k+1)})\| < \texttt{opt.epsgrad},$$

    where $X^{(k)}$, $k = 1, 2, \ldots,$ `info.iter` $\leq$ `opt.maxiter`, are the successive iterates of the parameter X and $M'(X^{(k+1)})$ is the gradient of the cost function at the current iteration step.

Outputs:

- `sysh`, an input/state/output representation of the identified system $\hat{\mathscr{B}}$;
- `info` information from the optimization solver:
  - `info.M`, the misfit $M(w_{\mathrm{d}}, \hat{\mathscr{B}})$;
  - `info.time`, the execution time for the STLS solver; not equal to the execution time of `stlsident`;
  - `info.iter`, the number of iterations. Note `info.iter = opt.maxiter` indicates lack of convergence to a desired convergence tolerance;
- `wh`, the optimal approximating time series;
- `xini`, a matrix whose columns are the initial condition, under which $\hat{w}_k$, $k = 1, \ldots, N$, are obtained.

```
[ M, wh, xini ] = misfit( w, sys, exct );
```

Inputs:

- `w`, the given time series w_{d}, a real MATLAB array of dimensions $T \times$ `w` $\times N$, where T is the number of samples, `w` is the number of variables, and N is the number of time series;

- `sys` an input/state/output representation of a system $\mathscr{B}$, given as the MATLAB object `ss` (see `help ss`), with `w` external variables (inputs and outputs) and of order which is a multiple of the number of outputs;
- `exct` (default []), a vector of indices for exact variables.

Outputs:

- `M`, the misfit $M(w_{\mathrm{d}}, \mathscr{B})$;
- `wh`, optimal approximating time series $\hat{w}$;
- `xini`, a matrix whose columns are the initial condition, under which $\hat{w}_k$, $k = 1, \ldots, N$, are obtained.

The functions `stlsidentuy` and `misfituy` are versions of `stlsident` and `misfit` that use an a priori given input/output partitioning of the variables. For details on their usage, see their MATLAB help.

Notation

Sets of numbers page

$\mathbb{R}$, $\mathbb{R}_+$	the set of real numbers, nonnegative real numbers	2, 37
$\mathbb{Z}$, $\mathbb{N}$	the set of integers, and natural numbers $\{0, 1, 2, \dots\}$	102, 35

Norms and extreme eigenvalue page

$\|x\|,\ x \in \mathbb{R}^n$	2-norm of a vector $\sqrt{\sum_{i=1}^n x_i^2}$	3
$\|A\|,\ A \in \mathbb{R}^{m\times n}$	induced 2-norm $\min_{\|x\|=1} \|Ax\|$	36
$\|A\|_{\mathrm{F}},\ A \in \mathbb{R}^{m\times n}$	Frobenius norm $\sqrt{\mathrm{trace}(AA^\top)}$	1
$\|w\|,\ w \in (\mathbb{R}^{\mathbf{w}})^T$	2-norm of a time series $\sqrt{\sum_{t=1}^T \|w(t)\|^2}$	24
$\|w\|,\ w \in (\mathbb{R}^{\mathbf{w}\times N})^T$	2-norm of a matrix valued time series $\sqrt{\sum_{t=1}^T \|w(t)\|_{\mathrm{F}}^2}$	168
$\lambda_{\min}(A)$, $\lambda_{\max}(A)$	minimum, maximum eigenvalue of a symmetric matrix	75

Matrix operations page

$A^\dagger$	pseudoinverse	53
$A^\top$	transpose of a matrix	18
$\mathrm{vec}(A)$	column-wise vectorization of a matrix	52
$\mathrm{col}(a, b)$	the column vector $\left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right]$	2
$\mathrm{col\,dim}(A)$	the number of columns of A	21
$\mathrm{row\,dim}(A)$	the number of *block* rows of A	126
$\mathrm{col\,span}(A)$	the span of the columns of A (the image or range of A)	9
$\mathrm{diag}(v)$, $v \in \mathbb{R}^n$	the diagonal matrix $\mathrm{diag}(v_1, \dots, v_n)$	30
$\mathrm{diag}(V_1, \dots, V_n)$	(block-) diagonal matrix with diagonal blocks $V_1, \dots, V_n$	41
$\otimes$	Kronecker product $A \otimes B := [a_{ij}B]$	41
$\odot$	element-wise (Hadamard) product $A \odot B := [a_{ij}b_{ij}]$	30
δ	Kronecker delta, $\delta_0 = 1$ and $\delta_t = 0$ for all $t \neq 0$	58

Expectation, covariance, and normal distribution page

$\mathbf{E}$, cov	expectation, covariance operator	59
$x \sim \mathrm{N}(m, V)$	x is normally distributed with mean m and covariance V	31

Fixed symbols page

$\mathscr{U}$	universum of outcomes from an experiment	16
$\mathscr{B}$	model behavior	16
$\mathscr{M}$	model class	16
$\mathscr{H}_l(w)$	Hankel matrix with l *block* rows; see ($\mathscr{H}$)	120
$\mathscr{S}$	structure specification for the STLS problem	50
$X_{\text{ext}} := \left[\begin{smallmatrix} X \\ -I \end{smallmatrix}\right]$	extended parameter in an input/output parameterization	52

LTI model class and invariants page

$\mathbf{m}(\mathscr{B})$	number of inputs of $\mathscr{B}$	105
$\mathbf{p}(\mathscr{B})$	number of outputs of $\mathscr{B}$	105
$\mathbf{l}(\mathscr{B})$	lag of $\mathscr{B}$	105
$\mathbf{n}(\mathscr{B})$	order of $\mathscr{B}$	105
$\mathscr{L}^{\mathtt{w},\mathtt{n}}_{\mathtt{m},\mathtt{l}}$	$:= \{\, \mathscr{B} \subset (\mathbb{R}^{\mathtt{w}})^{\mathbb{Z}} \mid \mathscr{B} \text{ is LTI},\ \mathbf{m}(\mathscr{B}) \leq \mathtt{m} \leq \mathtt{w},\ \mathbf{l}(\mathscr{B}) \leq \mathtt{l},\ \mathbf{n}(\mathscr{B}) \leq \mathtt{n} \,\}$	113

If `m`, `l`, or `n` is not specified, the corresponding invariant $\mathbf{m}(\mathscr{B})$, $\mathbf{l}(\mathscr{B})$, or $\mathbf{n}(\mathscr{B})$ is not bounded.

Miscellaneous page

$: \Longleftrightarrow$	left-hand side is defined by the right-hand side	19
$\Longleftrightarrow :$	right-hand side is defined by the left-hand side	20
σ	the backwards shift operator $\sigma f\,(t) = f(t+1)$ Acting on a vector or matrix, σ removes the first block row.	23
σ^*	the forward shift operator $\sigma^* f\,(t) = f(t-1)$ Acting on a vector or matrix, σ^* removes the last block row.	124

Abbreviations

ALS	adjusted least squares
DAISY	data base for system identification
EIV	errors-in-variables
EWTLS	element-wise weighted total least squares
GTLS	generalized total least squares
GlTLS	global total least squares
LS	least squares
LTI	linear time-invariant
MIMO	multi-input multi-output
MPUM	most powerful unfalsified model
SISO	single-input single-output
STLS	structured total least squares
SVD	singular value decomposition
TLS	total least squares
WLS	weighted least squares
WTLS	weighted total least squares

Bibliography

[AMH91] T. Abatzoglou, J. Mendel, and G. Harada. The constrained total least squares technique and its application to harmonic superresolution. *IEEE Trans. Signal Process.*, 39:1070–1087, 1991.

[AY70a] M. Aoki and P. Yue. On a priori error estimates of some identification methods. *IEEE Trans. Automat. Control*, 15(5):541–548, 1970.

[AY70b] M. Aoki and P. Yue. On certain convergence questions in system identification. *SIAM J. Control*, 8(2):239–256, 1970.

[BHN99] R. Byrd, M. Hribar, and J. Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM J. Optim.*, 9(4):877–900, 1999.

[BM86] Y. Bresler and A. Macovski. Exact maximum likelihood parameter estimation of superimposed exponential signals in noise. *IEEE Trans. Acust., Speech, Signal Process.*, 34:1081–1089, 1986.

[Boo79] F. L. Bookstein. Fitting conic sections to scattered data. *Computer Graphics and Image Processing*, 9:59–71, 1979.

[Bro70] R. Brockett. *Finite Dimensional Linear Systems*. John Wiley, New York, 1970.

[Cad88] J. Cadzow. Signal enhancement—A composite property mapping algorithm. *IEEE Trans. Signal Process.*, 36:49–62, 1988.

[CRS95] R. Carroll, D. Ruppert, and L. Stefanski. *Measurement Error in Nonlinear Models*. Chapman & Hall/CRC, London, 1995.

[CST00] C. Cheng, H. Schneeweiss, and M. Thamerus. A small sample estimator for a polynomial regression with errors in the variables. *J. R. Stat. Soc. (Ser. Stat. Methodol. B)*, 62:699–709, 2000.

[DGS03] R. Diversi, R. Guidorzi, and U. Soverini. Kalman filtering in symmetrical noise environments. In *Proceedings of the 11th IEEE Mediteranean Conference on Control and Automation*, Rhodes, Greece, 2003.

[DM93] B. De Moor. Structured total least squares and L_2 approximation problems. *Linear Algebra Appl.*, 188–189:163–207, 1993.

[DM94] B. De Moor. Total least squares for affinely structured matrices and the noisy realization problem. *IEEE Trans. Signal Process.*, 42(11):3104–3113, 1994.

[DM03] B. De Moor. On the number of rows and columns in subspace identification methods. In *Proceedings of the 13th IFAC Symposium on System Identification*, pages 1796–1801, Rotterdam, The Netherlands, 2003.

[DM05] B. De Moor. DaISy: Database for the identification of systems. Dept. EE, K.U.Leuven, `www.esat.kuleuven.be/sista/daisy/`, 2005.

[DR94] B. De Moor and B. Roorda. L_2-optimal linear system identification structured total least squares for SISO systems. In *Proceedings of the 33rd Conference on Decision and Control*, pages 2874–2879, Lake Buena, FL, 1994.

[EY36] G. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218, 1936.

[FPF99] A. Fitzgibbon, M. Pilu, and R. Fisher. Direct least-squares fitting of ellipses. *IEEE Trans. Pattern Anal. Machine Intelligence*, 21(5):476–480, 1999.

[Ful87] W. Fuller. *Measurement Error Models*. Wiley, New York, 1987.

[FW97] F. Fagnani and J. C. Willems. Deterministic Kalman filtering in a behavioral framework. *Control Lett.*, 32:301–312, 1997.

[Gal82] P. Gallo. Consistency of regression estimates when some variables are subject to error. *Comm. Statist. A—Theory Methods*, 11:973–893, 1982.

[GDS03] R. Guidorzi, R. Diversi, and U. Soverini. Optimal errors-in-variables filtering. *Automatica*, 39:281–289, 2003.

[GGS94] W. Gander, G. Golub, and R. Strebel. Fitting of circles and ellipses: Least squares solution. *BIT*, 34:558–578, 1994.

[GP96] P. Guillaume and R. Pintelon. A Gauss–Newton-like optimization algorithm for "weighted" nonlinear least-squares problems. *IEEE Trans. Signal Process.*, 44(9):2222–2228, 1996.

[GV80] G. Golub and C. Van Loan. An analysis of the total least squares problem. *SIAM J. Numer. Anal.*, 17:883–893, 1980.

[Har97] R. Hartley. In defense of the eight-point algorithm. *IEEE Trans. Pattern Anal. Machine Intelligence*, 19(6):580–593, June 1997.

[HS99] C. Heij and W. Scherrer. Consistency of system identification by global total least squares. *Automatica*, 35:993–1008, 1999.

[Kan94] K. Kanatani. Statistical bias of conic fitting and renormalization. *IEEE Trans. Pattern Anal. Machine Intelligence*, 16(3):320–326, 1994.

[KM00] A. Kukush and E.-O. Maschke. The efficiency of adjusted least squares in the linear functional relationship. DP–208, SFB 386, Univ. of Munich, 2000.

[KMV02] A. Kukush, I. Markovsky, and S. Van Huffel. Consistent fundamental matrix estimation in a quadratic measurement error model arising in motion analysis. *Comput. Statist. Data Anal.*, 41(1):3–18, 2002.

[KMV03] A. Kukush, I. Markovsky, and S. Van Huffel. Consistent estimation in the bilinear multivariate errors-in-variables model. *Metrika*, 57(3):253–285, 2003.

[KMV04] A. Kukush, I. Markovsky, and S. Van Huffel. Consistent estimation in an implicit quadratic measurement error model. *Comput. Statist. Data Anal.*, 47(1):123–147, 2004.

[KMV05] A. Kukush, I. Markovsky, and S. Van Huffel. Consistency of the structured total least squares estimator in a multivariate errors-in-variables model. *J. Statist. Plann. Inference*, 133(2):315–358, 2005.

[Kun78] S. Kung. A new identification method and model reduction algorithm via singular value decomposition. In *Proceedings of the 12th Asilomar Conference on Circuits, Systems, and Computers*, pages 705–714, Pacific Grove, CA, 1978.

[KV04] A. Kukush and S. Van Huffel. Consistency of elementwise-weighted total least squares estimator in a multivariate errors-in-variables model $AX = B$. *Metrika*, 59(1):75–97, 2004.

[KZ02] A. Kukush and S. Zwanzig. On consistent estimators in nonlinear functional EIV models. In Van Huffel and Lemmerling [VL02], pages 145–155.

[LD01] P. Lemmerling and B. De Moor. Misfit versus latency. *Automatica*, 37:2057–2067, 2001.

[Lev64] M. Levin. Estimation of a system pulse transfer function in the presence of noise. *IEEE Trans. Automat. Control*, 9:229–235, 1964.

[Lju99] L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, Upper Saddle River, NJ, 1999.

[LM00] Y. Leedan and P. Meer. Heteroscedastic regression in computer vision: Problems with bilinear constraint. *Int. J. Comput. Vision*, 37(2):127–150, 2000.

[LMV00] P. Lemmerling, N. Mastronardi, and S. Van Huffel. Fast algorithm for solving the Hankel/Toeplitz structured total least squares problem. *Numerical Algorithms*, 23:371–392, 2000.

[Mar63] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, 11:431–441, 1963.

[MD03] I. Markovsky and B. De Moor. Linear dynamic filtering with noisy input and output. In *Proceedings of the 13th IFAC Symposium on System Identification*, pages 1749–1754, Rotterdam, The Netherlands, 2003.

[MLV00] N. Mastronardi, P. Lemmerling, and S. Van Huffel. Fast structured total least squares algorithm for solving the basic deconvolution problem. *SIAM J. Matrix Anal. Appl.*, 22:533–553, 2000.

[MM98] M. Mühlich and R. Mester. The role of total least squares in motion analysis. In H. Burkhardt, editor, *Proceedings of the 5th European Conference on Computer Vision*, pages 305–321. Springer-Verlag, 1998.

[MMH03] J. Manton, R. Mahony, and Y. Hua. The geometry of weighted low-rank approximations. *IEEE Trans. Signal Process.*, 51(2):500–514, 2003.

[Moo81] B. Moore. Principal component analysis in linear systems: Controllability, observability and model reduction. *IEEE Trans. Automat. Control*, 26(1):17–31, 1981.

[MR93] M. Moonen and J. Ramos. A subspace algorithm for balanced state space system identification. *IEEE Trans. Automat. Control*, 38:1727–1729, 1993.

[MRP+05] I. Markovsky, M.-L. Rastello, A. Premoli, A. Kukush, and S. Van Huffel. The element-wise weighted total least squares problem. *Comput. Statist. Data Anal.*, 50(1):181–209, 2005.

[MV06] I. Markovsky and S. Van Huffel. On weighted structured total least squares. In I. Lirkov, S. Margenov, and J. Waśniewski, editors, *Proceedings of the 5th International Conference on "Large-Scale Scientific Computations"*, volume 3743 of *Lecture notes in computer science*, pages 695–702. Springer–Verlag, Berlin, 2006.

[MWD02] I. Markovsky, J. C. Willems, and B. De Moor. Continuous-time errors-in-variables filtering. In *Proceedings of the 41st Conference on Decision and Control*, pages 2576–2581, Las Vegas, NV, 2002.

[MWD05] I. Markovsky, J. C. Willems, and B. De Moor. State representations from finite time series. In *Proceedings of the 44th Conference on Decision and Control*, pages 832–835, Seville, Spain, 2005.

[MWRM05] I. Markovsky, J. C. Willems, P. Rapisarda, and B. De Moor. Data driven simulation with applications to system identification. In *Proceedings of the 16th IFAC World Congress*, Prague, Czech Republic, 2005.

[MWV+05] I. Markovsky, J. C. Willems, S. Van Huffel, B. De Moor, and R. Pintelon. Application of structured total least squares for system identification and model reduction. *IEEE Trans. Automat. Control*, 50(10):1490–1500, 2005.

[Nie01] Y. Nievergelt. Hyperspheres and hyperplanes fitting seamlessly by algebraic constrained total least-squares. *Linear Algebra Appl.*, 331:43–59, 2001.

[Nie02] Y. Nievergelt. A finite algorithm to fit geometrically all midrange lines, circles, planes, spheres, hyperplanes, and hyperspheres. *Numer. Math.*, 91:257–303, 2002.

[NS48] J. Neyman and E. Scott. Consistent estimates based on partially consistent observations. *Econometrica*, 16(1):1–32, 1948.

[PR02] A. Premoli and M.-L. Rastello. The parametric quadratic form method for solving TLS problems with elementwise weighting. In Van Huffel and Lemmerling [VL02], pages 67–76.

[Pra87] V. Pratt. Direct least-squares fitting of algebraic surfaces. *ACM Computer Graphics*, 21(4):145–152, 1987.

[PS01] R. Pintelon and J. Schoukens. *System Identification: A Frequency Domain Approach*. IEEE Press, Piscataway, NJ, 2001.

[PW98] J. Polderman and J. C. Willems. *Introduction to Mathematical Systems Theory*. Springer-Verlag, New York, 1998.

[RH95] B. Roorda and C. Heij. Global total least squares modeling of multivariate time series. *IEEE Trans. Automat. Control*, 40(1):50–63, 1995.

[Roo95] B. Roorda. Algorithms for global total least squares modelling of finite multivariable time series. *Automatica*, 31(3):391–404, 1995.

[RPG96] J. Rosen, H. Park, and J. Glick. Total least norm formulation and solution of structured problems. *SIAM J. Matrix Anal. Appl.*, 17:110–126, 1996.

[RW97] P. Rapisarda and J. C. Willems. State maps for linear systems. *SIAM J. Control Optim.*, 35(3):1053–1091, 1997.

[SKMH05] S. Shklyar, A. Kukush, I. Markovsky, and S. Van Huffel. On the conic section fitting problem. *Journal of Multivariate Analysis*, 2005.

[SLV04] M. Schuermans, P. Lemmerling, and S. Van Huffel. Structured weighted low rank approximation. *Numer. Linear. Algebra Appl.*, 11:609–618, 2004.

[SLV05] M. Schuermans, P. Lemmerling, and S. Van Huffel. Block-row hankel weighted low rank approximation. *Numer. Linear. Algebra Appl.*, to appear, 2005.

[SMWV05] M. Schuermans, I. Markovsky, P. Wentzell, and S. Van Huffel. On the equivalence between total least squares and maximum likelihood PCA. *Analytica Chimica Acta*, 544:254–267, 2005.

[Spä97] H. Späth. Orthogonal least squares fitting by conic sections. In S. Van Huffel, editor, *Recent Advances in Total Least Squares Techniques and Errors-in-Variables Modeling*, pages 259–264. SIAM, Philadelphia, 1997.

[TM97] P. Torr and D. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *Int. J. Computer Vision*, 24(3):271–300, 1997.

[VD92] M. Verhaegen and P. Dewilde. Subspace model identification, Part 1: The output-error state-space model identification class of algorithms. *Int. J. Control*, 56:1187–1210, 1992.

[VD96] P. Van Overschee and B. De Moor. *Subspace Identification for Linear Systems: Theory, Implementation, Applications*. Kluwer, Boston, 1996.

[VL02] S. Van Huffel and P. Lemmerling, editors. Kluwer, 2002.

[VPR96] S. Van Huffel, H. Park, and J. Rosen. Formulation and solution of structured total least norm problems for parameter estimation. *IEEE Trans. Signal Process.*, 44(10):2464–2474, 1996.

[VSV+04] S. Van Huffel, V. Sima, A. Varga, S. Hammarling, and F. Delebecque. High-performance numerical software for control. *IEEE Control Systems Magazine*, 24:60–76, 2004.

[VV91] S. Van Huffel and J. Vandewalle. *The total least squares problem: Computational aspects and analysis*. SIAM, Philadelphia, 1991.

[WAH+97] P. Wentzell, D. Andrews, D. Hamilton, K. Faber, and B. Kowalski. Maximum likelihood principle component analysis. *J. Chemometrics*, 11:339–366, 1997.

[Wil86a] J. C. Willems. From time series to linear system—Part I. Finite dimensional linear time invariant systems. *Automatica*, 22(5):561–580, 1986.

[Wil86b] J. C. Willems. From time series to linear system—Part II. Exact modelling. *Automatica*, 22(6):675–694, 1986.

[Wil87] J. C. Willems. From time series to linear system—Part I. Finite dimensional linear time invariant systems, Part II. Exact modelling, Part III. Approximate modelling. *Automatica*, 22, 23:561–580, 675–694, 87–115, 1986, 1987.

[Wil91] J. C. Willems. Paradigms and puzzles in the theory of dynamical systems. *IEEE Trans. Automat. Control*, 36(3):259–294, 1991.

[WR02] J. C. Willems and P. Rapisarda. Balanced state representations with polynomial algebra. In A. Rantzer and C. I. Byrnes, editors, *Directions in mathematical systems theory and optimization*, chapter 25, pages 345–357. Springer-Verlag, 2002.

[WRMM05] J. C. Willems, P. Rapisarda, I. Markovsky, and B. De Moor. A note on persistency of excitation. *Control Lett.*, 54(4):325–329, 2005.

[Zha97] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and Vision Computing Journal*, 15(1):59–76, 1997.

Index