

Communications and Control Engineering

For further volumes:

www.springer.com/series/61

Series Editors

A. Isidori • J.H. van Schuppen • E.D. Sontag • M. Thoma • M. Krstic

Published titles include:

Stability and Stabilization of Infinite Dimensional Systems with Applications

Zheng-Hua Luo, Bao-Zhu Guo and Omer Morgul

Nonsmooth Mechanics (Second edition)

Bernard Brogliato

Nonlinear Control Systems II

Alberto Isidori

L₂-Gain and Passivity Techniques in Nonlinear Control

Arjan van der Schaft

Control of Linear Systems with Regulation and Input Constraints

Ali Saberi, Anton A. Stoorvogel and Peddapullaiah Sannuti

Robust and H_∞ Control

Ben M. Chen

Computer Controlled Systems

Efim N. Rosenwasser and Bernhard P. Lampe

Control of Complex and Uncertain Systems

Stanislav V. Emelyanov and Sergey K. Korovin

Robust Control Design Using H_∞ Methods

Ian R. Petersen, Valery A. Ugrinovski and Andrey V. Savkin

Model Reduction for Control System Design

Goro Obinata and Brian D.O. Anderson

Control Theory for Linear Systems

Harry L. Trentelman, Anton Stoorvogel and Malo Hautus

Functional Adaptive Control

Simon G. Fabri and Visakan Kadirkamanathan

Positive 1D and 2D Systems

Tadeusz Kaczorek

Identification and Control Using Volterra Models

Francis J. Doyle III, Ronald K. Pearson and Babatunde A. Ogunnaike

Non-linear Control for Underactuated Mechanical Systems

Isabelle Fantoni and Rogelio Lozano

Robust Control (Second edition)

Jürgen Ackermann

Flow Control by Feedback

Ole Morten Aamo and Miroslav Krstic

Learning and Generalization (Second edition)

Mathukumalli Vidyasagar

Constrained Control and Estimation

Graham C. Goodwin, Maria M. Seron and José A. De Doná

Randomized Algorithms for Analysis and Control of Uncertain Systems

Roberto Tempo, Giuseppe Calafiore and Fabrizio Dabbene

Switched Linear Systems

Zhendong Sun and Shuzhi S. Ge

Subspace Methods for System Identification

Tohru Katayama

Digital Control Systems

Ioan D. Landau and Gianluca Zito

Multivariable Computer-controlled Systems

Efim N. Rosenwasser and Bernhard P. Lampe

Dissipative Systems Analysis and Control

(Second edition)

Bernard Brogliato, Rogelio Lozano, Bernhard Maschke and Olav Egeland

Algebraic Methods for Nonlinear Control Systems

Giuseppe Conte, Claude H. Moog and Anna M. Perdon

Polynomial and Rational Matrices

Tadeusz Kaczorek

Simulation-based Algorithms for Markov Decision Processes

Hyeong Soo Chang, Michael C. Fu, Jiaqiao Hu and Steven I. Marcus

Iterative Learning Control

Hyo-Sung Ahn, Kevin L. Moore and YangQuan Chen

Distributed Consensus in Multi-vehicle Cooperative Control

Wei Ren and Randal W. Beard

Control of Singular Systems with Random Abrupt Changes

El-Kébir Boukas

Nonlinear and Adaptive Control with Applications

Alessandro Astolfi, Dimitrios Karagiannis and Romeo Ortega

Stabilization, Optimal and Robust Control

Aziz Belmiloudi

Control of Nonlinear Dynamical Systems

Felix L. Chernous'ko, Igor M. Ananievski and Sergey A. Reshmin

Periodic Systems

Sergio Bittanti and Patrizio Colaneri

Discontinuous Systems

Yury V. Orlov

Constructions of Strict Lyapunov Functions

Michael Malisoff and Frédéric Mazenc

Controlling Chaos

Huaguang Zhang, Derong Liu and Zhiliang Wang

Stabilization of Navier-Stokes Flows

Viorel Barbu

Distributed Control of Multi-agent Networks

Wei Ren and Yongcan Cao

Ivan Markovsky

Low Rank Approximation

Algorithms, Implementation,
Applications

 Springer

Ivan Markovsky
School of Electronics & Computer Science
University of Southampton
Southampton, UK
im@ecs.soton.ac.uk

Additional material to this book can be downloaded from <http://extras.springer.com>

ISSN 0178-5354 Communications and Control Engineering
ISBN 978-1-4471-2226-5 e-ISBN 978-1-4471-2227-2
DOI 10.1007/978-1-4471-2227-2
Springer London Dordrecht Heidelberg New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2011942476

© Springer-Verlag London Limited 2012

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms of licenses issued by the Copyright Licensing Agency. Enquiries concerning reproduction outside those terms should be sent to the publishers.

The use of registered names, trademarks, etc., in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant laws and regulations and therefore free for general use.

The publisher makes no representation, express or implied, with regard to the accuracy of the information contained in this book and cannot accept any legal responsibility or liability for any errors or omissions that may be made.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Mathematical models are obtained from first principles (natural laws, interconnection, etc.) and experimental data. Modeling from first principles is common in natural sciences, while modeling from data is common in engineering. In engineering, often experimental data are available and a simple approximate model is preferred to a complicated detailed one. Indeed, although optimal prediction and control of a complex (high-order, nonlinear, time-varying) system is currently difficult to achieve, robust analysis and design methods, based on a simple (low-order, linear, time-invariant) approximate model, may achieve sufficiently high performance. This book addresses the problem of *data approximation by low-complexity models*.

A unifying theme of the book is low rank approximation: a prototypical data modeling problem. The rank of a matrix constructed from the data corresponds to the complexity of a linear model that fits the data exactly. The data matrix being full rank implies that there is no exact low complexity linear model for that data. In this case, the aim is to find an approximate model. One approach for approximate modeling, considered in the book, is to find small (in some specified sense) modification of the data that renders the modified data exact. The exact model for the modified data is an optimal (in the specified sense) approximate model for the original data. The corresponding computational problem is low rank approximation. It allows the user to trade off accuracy vs. complexity by varying the rank of the approximation.

The distance measure for the data modification is a user choice that specifies the desired approximation criterion or reflects prior knowledge about the accuracy of the data. In addition, the user may have prior knowledge about the system that generates the data. Such knowledge can be incorporated in the modeling problem by imposing constraints on the model. For example, if the model is known (or postulated) to be a linear time-invariant dynamical system, the data matrix has Hankel structure and the approximating matrix should have the same structure. This leads to a Hankel structured low rank approximation problem.

A tenet of the book is: the estimation accuracy of the basic low rank approximation method can be improved by exploiting prior knowledge, i.e., by adding constraints that are known to hold for the data generating system. This path of development leads to weighted, structured, and other constrained low rank approxi-

mation problems. The theory and algorithms of these new classes of problems are interesting in their own right and being application driven are practically relevant.

Stochastic estimation and deterministic approximation are two complementary aspects of data modeling. The former aims to find from *noisy data*, generated by a low-complexity system, an estimate of that data generating system. The latter aims to find from *exact data*, generated by a high complexity system, a low-complexity approximation of the data generating system. In applications both the stochastic estimation and deterministic approximation aspects are likely to be present. The data are likely to be imprecise due to measurement errors and is likely to be generated by a complicated phenomenon that is not exactly representable by a model in the considered model class. The development of data modeling methods in system identification and signal processing, however, has been dominated by the stochastic estimation point of view. If considered, the approximation error is represented in the mainstream data modeling literature as a random process. This is not natural because the approximation error is by definition deterministic and even if considered as a random process, it is not likely to satisfy standard stochastic regularity conditions such as zero mean, stationarity, ergodicity, and Gaussianity.

An exception to the stochastic paradigm in data modeling is the behavioral approach, initiated by J.C. Willems in the mid-1980s. Although the behavioral approach is motivated by the deterministic approximation aspect of data modeling, it does not exclude the stochastic estimation approach. In this book, we use the behavioral approach as a *language* for defining different modeling problems and presenting their solutions. We emphasize the importance of deterministic approximation in data modeling, however, we formulate and solve stochastic estimation problems as low rank approximation problems.

Many well known concepts and problems from systems and control, signal processing, and machine learning reduce to low rank approximation. Generic examples in system theory are model reduction and system identification. The principal component analysis method in machine learning is equivalent to low rank approximation, which suggests that related dimensionality reduction, classification, and information retrieval problems can be phrased as low rank approximation problems. Sylvester structured low rank approximation has applications in computations with polynomials and is related to methods from computer algebra.

The developed ideas lead to algorithms, which are implemented in software. The algorithms clarify the ideas and the software implementation clarifies the algorithms. Indeed, the software is the ultimate unambiguous description of how the ideas are put to work. In addition, the provided software allows the reader to reproduce the examples in the book and to modify them. The exposition reflects the sequence

theory \mapsto algorithms \mapsto implementation.

Correspondingly, the text is interwoven with code that generates the numerical examples being discussed.

Prerequisites and practice problems

A common feature of the current research activity in all areas of science and engineering is the narrow specialization. In this book, we pick applications in the broad area of data modeling, posing and solving them as low rank approximation problems. This unifies seemingly unrelated applications and solution techniques by emphasising their common aspects (e.g., complexity–accuracy trade-off) and abstracting from the application specific details, terminology, and implementation details. Despite of the fact that applications in systems and control, signal processing, machine learning, and computer vision are used as examples, the only real prerequisites for following the presentation is knowledge of linear algebra.

The book is intended to be used for self study by researchers in the area of data modeling and by advanced undergraduate/graduate level students as a complementary text for a course on system identification or machine learning. In either case, the expected knowledge is undergraduate level linear algebra. In addition, MATLAB code is used, so that familiarity with MATLAB programming language is required.

Passive reading of the book gives a broad perspective on the subject. Deeper understanding, however, requires active involvement, such as supplying missing justification of statements and specific examples of the general concepts, application and modification of presented ideas, and solution of the provided exercises and practice problems. There are two types of practice problem: analytical, asking for a proof of a statement clarifying or expanding the material in the book, and computational, asking for experiments with real or simulated data of specific applications. Most of the problems are easy to medium difficulty. A few problems (marked with stars) can be used as small research projects.

The code in the book, available from

<http://extra.springer.com/>

has been tested with MATLAB 7.9, running under Linux, and uses the Optimization Toolbox 4.3, Control System Toolbox 8.4, and Symbolic Math Toolbox 5.3. A version of the code that is compatible with Octave (a free alternative to MATLAB) is also available from the book's web page.

Acknowledgements

A number of individuals and the European Research Council contributed and supported me during the preparation of the book. Oliver Jackson—Springer's editor (engineering)—encouraged me to embark on the project. My colleagues in ESAT/SISTA, K.U. Leuven and ECS/ISIS, Southampton, UK created the right environment for developing the ideas in the book. In particular, I am in debt to Jan C. Willems (SISTA) for his personal guidance and example of critical thinking. The behavioral approach that Jan initiated in the early 1980's is present in this book.

Maarten De Vos, Diana Sima, Konstantin Usevich, and Jan Willems proofread chapters of the book and suggested improvements. I gratefully acknowledge funding

from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013)/ERC Grant agreement number 258581 “Structured low-rank approximation: Theory, algorithms, and applications”.

Southampton, UK

Ivan Markovsky

Contents

1	Introduction	1
1.1	Classical and Behavioral Paradigms for Data Modeling	1
1.2	Motivating Example for Low Rank Approximation	3
1.3	Overview of Applications	7
1.4	Overview of Algorithms	21
1.5	Literate Programming	24
1.6	Notes	27
	References	30
Part I Linear Modeling Problems		
2	From Data to Models	35
2.1	Linear Static Model Representations	35
2.2	Linear Time-Invariant Model Representations	45
2.3	Exact and Approximate Data Modeling	52
2.4	Unstructured Low Rank Approximation	60
2.5	Structured Low Rank Approximation	67
2.6	Notes	70
	References	71
3	Algorithms	73
3.1	Subspace Methods	73
3.2	Algorithms Based on Local Optimization	81
3.3	Data Modeling Using the Nuclear Norm Heuristic	96
3.4	Notes	104
	References	105
4	Applications in System, Control, and Signal Processing	107
4.1	Introduction	107
4.2	Model Reduction	108
4.3	System Identification	115
4.4	Analysis and Synthesis	122

4.5	Simulation Examples	125
4.6	Notes	128
	References	130
Part II Miscellaneous Generalizations		
5	Missing Data, Centering, and Constraints	135
5.1	Weighted Low Rank Approximation with Missing Data	135
5.2	Affine Data Modeling	147
5.3	Complex Least Squares Problem with Constrained Phase	156
5.4	Approximate Low Rank Factorization with Structured Factors	163
5.5	Notes	175
	References	177
6	Nonlinear Static Data Modeling	179
6.1	A Framework for Nonlinear Static Data Modeling	179
6.2	Nonlinear Low Rank Approximation	182
6.3	Algorithms	187
6.4	Examples	191
6.5	Notes	196
	References	197
7	Fast Measurements of Slow Processes	199
7.1	Introduction	199
7.2	Estimation with Known Measurement Process Dynamics	202
7.3	Estimation with Unknown Measurement Process Dynamics	204
7.4	Examples and Real-Life Testing	212
7.5	Auxiliary Functions	222
7.6	Notes	224
	References	225
Appendix A Approximate Solution of an Overdetermined System of Equations		
	References	231
Appendix B Proofs		
	References	237
Appendix P Problems		
	References	244
Notation		245
List of Code Chunks		247
Functions and Scripts Index		251
Index		253

Part I
Linear Modeling Problems

Part I
Linear Modeling Problems

Chapter 1

Introduction

The very art of mathematics is to say the same thing another way.

Unknown

1.1 Classical and Behavioral Paradigms for Data Modeling

Fitting linear models to data can be achieved, both conceptually and algorithmically, by solving approximately a system of linear equations

$$AX \approx B, \tag{LSE}$$

where the matrices A and B are constructed from the given data and the matrix X parametrizes the model. In this *classical paradigm*, the main tools are the ordinary linear least squares method and its variations—regularized least squares, total least squares, robust least squares, etc. The least squares method and its variations are mainly motivated by their applications for data fitting, but they invariably consider solving approximately an overdetermined system of equations.

The underlying premise in the classical paradigm is that existence of an exact linear model for the data is equivalent to existence of solution X to a system $AX = B$. Such a model is a linear map: the variables corresponding to the A matrix are inputs (or causes) and the variables corresponding to the B matrix are outputs (or consequences) in the sense that they are determined by the inputs and the model. Note that in the classical paradigm the input/output partition of the variables is postulated a priori. Unless the model is required to have the a priori specified input/output partition, imposing such structure in advance is ad hoc and leads to undesirable theoretical and numerical features of the modeling methods derived.

An alternative to the classical paradigm that does not impose an a priori fixed input/output partition is the *behavioral paradigm*. In the behavioral paradigm, fitting linear models to data is equivalent to the problem of approximating a matrix D , constructed from the data, by a matrix \hat{D} of lower rank. Indeed, existence of an exact linear model for D is equivalent to D being rank deficient. Moreover, the rank

of D is related to the complexity of the model. This fact is the tenet of the book and is revisited in the following chapters in the context of applications from systems and control, signal processing, computer algebra, and machine learning. Also its implication to the development of numerical algorithms for data fitting is explored.

To see that existence of a low-complexity exact linear model is equivalent to rank deficiency of the data matrix, let the columns d_1, \dots, d_N of D be the observations and the elements $d_{1j}, \dots, d_{\alpha j}$ of d_j be the observed variables. We assume that there are at least as many observations as observed variables, i.e., $\alpha \leq N$. A linear model for D declares that there are linear relations among the variables, i.e., there are vectors r_k , such that

$$r_k^\top d_j = 0, \quad \text{for } j = 1, \dots, N.$$

If there are p independent linear relations, then D has rank less than or equal to $m := \alpha - p$ and the observations belong to at most m -dimensional subspace \mathcal{B} of \mathbb{R}^α . We identify the model for D , defined by the linear relations $r_1, \dots, r_p \in \mathbb{R}^\alpha$, with the set $\mathcal{B} \subset \mathbb{R}^\alpha$. Once a model \mathcal{B} is obtained from the data, all possible input/output partitions can be enumerated, which is an analysis problem for the identified model. Therefore, the choice of an input/output partition in the behavioral paradigm to data modeling can be incorporated, if desired, in the modeling problem and thus need not be hypothesized as necessarily done in the classical paradigm.

The classical and behavioral paradigms for data modeling are related but not equivalent. Although existence of solution of the system $AX = B$ implies that the matrix $[A \ B]$ is low rank, it is not true that $[A \ B]$ having a sufficiently low rank implies that the system $AX = B$ is solvable. This lack of equivalence causes ill-posed (or numerically ill-conditioned) data fitting problems in the classical paradigm, which have no solution (or are numerically difficult to solve). In terms of the data fitting problem, ill-conditioning of the problem (LSE) means that the a priori fixed input/output partition of the variables is not corroborated by the data. In the behavioral setting without the a priori fixed input/output partition of the variables, ill-conditioning of the data matrix D implies that the data approximately satisfy linear relations, so that nearly rank deficiency is a good feature of the data.

The classical paradigm is included in the behavioral paradigm as a special case because approximate solution of an overdetermined system of equations (LSE) is a possible approach to achieve low rank approximation. Alternatively, low rank approximation can be achieved by approximating the data matrix with a matrix that has at least p -dimensional null space, or at most m -dimensional column space. Parametrizing the null space and the column space by sets of basis vectors, the alternative approaches are:

1. *kernel representation* there is a full row rank matrix $R \in \mathbb{R}^{p \times \alpha}$, such that

$$RD = 0,$$

2. *image representation* there are matrices $P \in \mathbb{R}^{\alpha \times m}$ and $L \in \mathbb{R}^{m \times N}$, such that

$$D = PL.$$

The approaches using kernel and image representations are equivalent to the original low rank approximation problem. Next, the use of $AX = B$, kernel, and image representations is illustrated on the most simple data fitting problem—line fitting.

1.2 Motivating Example for Low Rank Approximation

Given a (multi)set of points $\{d_1, \dots, d_N\} \subset \mathbb{R}^2$ in the plane, the aim of the line fitting problem is to find a line passing through the origin that “best” matches the given points. The classical approach for line fitting is to define

$$\text{col}(a_j, b_j) := \begin{bmatrix} a_j \\ b_j \end{bmatrix} := d_j$$

(“:=” stands for “by definition”, see page 245 for a list of notation) and solve approximately the overdetermined system

$$\text{col}(a_1, \dots, a_N)x = \text{col}(b_1, \dots, b_N) \quad (\text{lse})$$

by the least squares method. Let $x_{\text{l}s}$ be the least squares solution to (lse). Then the least squares fitting line is

$$\mathcal{B}_{\text{l}s} := \{d = \text{col}(a, b) \in \mathbb{R}^2 \mid ax_{\text{l}s} = b\}.$$

Geometrically, $\mathcal{B}_{\text{l}s}$ minimizes the sum of the squared *vertical distances* from the data points to the fitting line.

The left plot in Fig. 1.1 shows a particular example with $N = 10$ data points. The data points d_1, \dots, d_{10} are the circles in the figure, the fit $\mathcal{B}_{\text{l}s}$ is the solid line, and the fitting errors $e := ax_{\text{l}s} - b$ are the dashed lines. Visually one expects the best fit to be the vertical axis, so minimizing vertical distances does not seem appropriate in this example.

Note that by solving (lse), a (the first components of the d) is treated differently from b (the second components): b is assumed to be a *function* of a . This is an arbitrary choice; the data can be fitted also by solving approximately the system

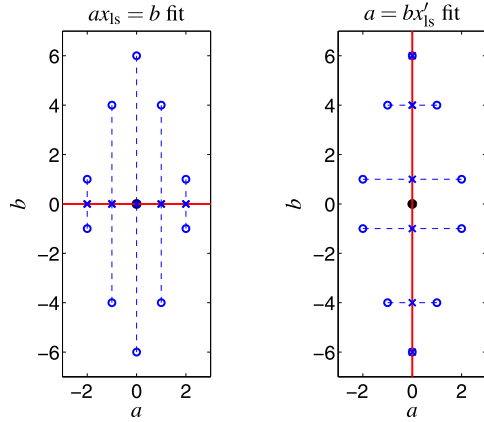
$$\text{col}(a_1, \dots, a_N) = \text{col}(b_1, \dots, b_N)x, \quad (\text{lse}')$$

in which case a is assumed to be a function of b . Let $x'_{\text{l}s}$ be the least squares solution to (lse'). It gives the fitting line

$$\mathcal{B}'_{\text{l}s} := \{d = \text{col}(a, b) \in \mathbb{R}^2 \mid a = bx'_{\text{l}s}\},$$

which minimizes the sum of the squared *horizontal distances* (see the right plot in Fig. 1.1). The line $\mathcal{B}'_{\text{l}s}$ happens to achieve the desired fit in the example.

Fig. 1.1 Least squares fits (solid lines) minimizing vertical (left plot) and horizontal (right plot) distances



In the classical approach for data fitting, i.e., solving approximately a linear system of equations in the least squares sense, the choice of the model representation affects the fitting criterion.

This feature of the classical approach is undesirable: it is more natural to specify a desired fitting criterion independently of how the model happens to be parametrized. In many data modeling methods, however, a model representation is a priori fixed and implicitly corresponds to a particular fitting criterion.

The total least squares method is an alternative to least squares method for solving approximately an overdetermined system of linear equations. In terms of data fitting, the total least squares method minimizes the sum of the squared *orthogonal distances* from the data points to the fitting line. Using the system of equations (lse), line fitting by the total least squares method leads to the problem

$$\begin{aligned} & \text{minimize} \quad \text{over } x \in \mathbb{R}, \quad \begin{bmatrix} \hat{a}_1 \\ \vdots \\ \hat{a}_N \end{bmatrix} \in \mathbb{R}^N, \text{ and } \begin{bmatrix} \hat{b}_1 \\ \vdots \\ \hat{b}_N \end{bmatrix} \in \mathbb{R}^N \quad \sum_{j=1}^N \left\| d_j - \begin{bmatrix} \hat{a}_j \\ \hat{b}_j \end{bmatrix} \right\|_2^2 \\ & \text{subject to} \quad \hat{a}_j x = \hat{b}_j, \quad \text{for } j = 1, \dots, N. \end{aligned} \quad (\text{tls})$$

However, for the data in Fig. 1.1 the total least squares problem has no solution. Informally, the approximate solution is $x_{\text{tls}} = \infty$, which corresponds to a fit by a vertical line. Formally,

the total least squares problem (tls) may have no solution and therefore fail to give a model.

The use of (lse) in the definition of the total least squares line fitting problem restricts the fitting line to be a graph of a function $ax = b$ for some $x \in \mathbb{R}$. Thus, the vertical line is a priori excluded as a possible solution. In the example, the line minimizing the sum of the squared orthogonal distances happens to be the vertical line. For this reason, x_{tls} does not exist.

Any line \mathcal{B} passing through the origin can be represented as an image and a kernel, i.e., there exist matrices $P \in \mathbb{R}^{2 \times 1}$ and $R \in \mathbb{R}^{1 \times 2}$, such that

$$\mathcal{B} = (P) := \{d = P\ell \in \mathbb{R}^2 \mid \ell \in \mathbb{R}\}$$

and

$$\mathcal{B} = \ker(R) := \{d \in \mathbb{R}^2 \mid Rd = 0\}.$$

Using the image representation of the model, the line fitting problem of minimizing the sum of the squared orthogonal distances is

$$\begin{aligned} & \text{minimize over } P \in \mathbb{R}^{2 \times 1} \text{ and } [\ell_1 \cdots \ell_N] \in \mathbb{R}^{1 \times N} && \sum_{j=1}^N \|d_j - \widehat{d}_j\|_2^2 && (\text{lra}'_P) \\ & \text{subject to } \widehat{d}_j = P\ell_j, \quad \text{for } j = 1, \dots, N. \end{aligned}$$

With

$$D := [d_1 \cdots d_N], \quad \widehat{D} := [\widehat{d}_1 \cdots \widehat{d}_N],$$

and $\|\cdot\|_F$ the *Frobenius norm*,

$$\|E\|_F := \|\text{vec}(E)\|_2 = \|[e_{11} \cdots e_{q1} \cdots e_{1N} \cdots e_{qN}]\|_2, \quad \text{for all } E \in \mathbb{R}^{q \times N}$$

(lra'_P) is more compactly written as

$$\begin{aligned} & \text{minimize over } P \in \mathbb{R}^{2 \times 1} \text{ and } L \in \mathbb{R}^{1 \times N} && \|D - \widehat{D}\|_F^2 && (\text{lra}_P) \\ & \text{subject to } \widehat{D} = PL. \end{aligned}$$

Similarly, using a kernel representation, the line fitting problem, minimizing the sum of squares of the orthogonal distances is

$$\begin{aligned} & \text{minimize over } R \in \mathbb{R}^{1 \times 2}, R \neq 0, \text{ and } \widehat{D} \in \mathbb{R}^{2 \times N} && \|D - \widehat{D}\|_F^2 && (\text{lra}_R) \\ & \text{subject to } R\widehat{D} = 0. \end{aligned}$$

Contrary to the total least squares problem (tls), problems (lra_P) and (lra_R) always have (nonunique) solutions. In the example, solutions are, e.g., $P^* = \text{col}(0, 1)$ and $R^* = [1 \ 0]$, which describe the vertical line

$$\mathcal{B}^* := (P^*) = \ker(R^*).$$

The constraints

$$\widehat{D} = PL, \quad \text{with } P \in \mathbb{R}^{2 \times 1}, L \in \mathbb{R}^{1 \times N} \quad \text{and} \quad R\widehat{D} = 0, \quad \text{with } R \in \mathbb{R}^{1 \times 2}, R \neq 0$$

are equivalent to the constraint $\text{rank}(\widehat{D}) \leq 1$, which shows that the points $\{\widehat{d}_1, \dots, \widehat{d}_N\}$ being fitted exactly by a line passing through the origin is equivalent to

$$\text{rank}([\widehat{d}_1 \ \dots \ \widehat{d}_N]) \leq 1.$$

Thus, (lra_P) and (lra_R) are instances of one and the same

abstract problem: approximate the data matrix D by a low rank matrix \widehat{D} .

In Chap. 2, the observations made in the line fitting example are generalized to modeling of q -dimensional data. The underlying goal is:

Given a set of points in \mathbb{R}^q (the data), find a subspace of \mathbb{R}^q of bounded dimension (a model) that has the least (2-norm) distance to the data points.

Such a subspace is a (2-norm) optimal fitting *model*. General *representations* of a subspace in \mathbb{R}^q are the kernel or the image of a matrix. The classical least squares and total least squares formulations of the data modeling problem exclude some subspaces. The equations $AX = B$ and $A = BX'$, used in the least squares and total least squares problem formulations to represent the subspace, might fail to represent the optimal solution, while the kernel and image representations do not have such deficiency. This suggests that the kernel and image representations are better suited for data modeling.

The equations $AX = B$ and $A = BX'$ were introduced from an algorithmic point of view—by using them, the data fitting problem is turned into the standard problem of solving approximately an overdetermined linear system of equations. An interpretation of these equations in the data modeling context is that in the model represented by the equation $AX = B$, the variable A is an input and the variable B is an output. Similarly, in the model represented by the equation $A = BX'$, A is an output and B is an input. The input/output interpretation has an intuitive appeal because it implies a causal dependence of the variables: the input is causing the output.

Representing the model by an equation $AX = B$ and $A = BX'$, as done in the classical approach, one a priori assumes that the optimal fitting model has a certain input/output structure. The consequences are:

- existence of exceptional (nongeneric) cases, which complicate the theory,
- ill-conditioning caused by “nearly” exceptional cases, which leads to lack of numerical robustness of the algorithms, and
- need of regularization, which leads to a change of the specified fitting criterion.

These aspects of the classical approach are generally considered as inherent to the data modeling problem. By choosing the alternative image and kernel model representations, the problem of solving approximately an overdetermined system of

equations becomes a low rank approximation problem, where the nongeneric cases (and the related issues of ill-conditioning and need of regularization) are avoided.

1.3 Overview of Applications

In this section, examples of low rank approximation drawn from different application areas are listed. The fact that a matrix constructed from exact data is low rank and the approximate modeling problem is low rank approximation is sometimes well known (e.g., in realization theory, model reduction, and approximate greatest common divisor). In other cases (e.g., natural language processing and conic section fitting), the link to low rank approximation is less well known and is not exploited.

Common Pattern in Data Modeling

The motto of the book is:

Behind every data modeling problems there is a (hidden) low rank approximation problem: the model imposes relations on the data which render a matrix constructed from exact data rank deficient.

Although an exact data matrix is low rank, a matrix constructed from observed data is generically full rank due to measurement noise, unaccounted effects, and assumptions about the data generating system that are not satisfied in practice. Therefore, generically, the observed data do not have an exact low-complexity model. This leads to the problem of approximate modeling, which can be formulated as a low rank approximation problem as follows. Modify the data as little as possible, so that the matrix constructed from the modified data has a specified low rank. The modified data matrix being low rank implies that there is an exact model for the modified data. This model is by definition an approximate model for the given data. The transition from exact to approximate modeling is an important step in building a coherent theory for data modeling and is emphasized in this book.

In all applications, the exact modeling problem is discussed before the practically more important approximate modeling problem. This is done because (1) exact modeling is simpler than approximate modeling, so that it is the right starting place, and (2) exact modeling is a part of optimal approximate modeling and suggests ways of solving such problems suboptimally. Indeed, small modifications of exact modeling algorithms lead to effective approximate modeling algorithms. Well known examples of the transition from exact to approximate modeling in systems theory are the progressions from realization theory to model reduction and from deterministic subspace identification to approximate and stochastic subspace identification.

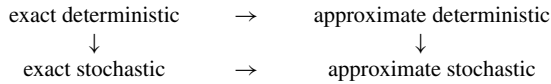


Fig. 1.2 Transitions among exact deterministic, approximate deterministic, exact stochastic, and approximate stochastic modeling problems. The arrows show progression from simple to complex

The estimator consistency question in stochastic estimation problems corresponds to exact data modeling because asymptotically the true data generating system is recovered from observed data. Estimation with finite sample size, however, necessarily involves approximation. Thus in stochastic estimation theory there is also a step of transition from exact to approximate, see Fig. 1.2.

The applications can be read in any order or skipped without loss of continuity.

Applications in Systems and Control

Deterministic System Realization and Model Reduction

Realization theory addresses the problem of finding a state representation of a linear time-invariant dynamical system defined by a transfer function or impulse response representation. The key result in realization theory is that a sequence

$$H = (H(0), H(1), \dots, H(t), \dots)$$

is an impulse response of a discrete-time linear time-invariant system of order n if and only if the two sided infinite Hankel matrix

$$\mathcal{H}(H) := \begin{bmatrix} H(1) & H(2) & H(3) & \dots \\ H(2) & H(3) & \dots & \\ H(3) & \dots & & \\ \vdots & & & \end{bmatrix},$$

constructed from H has rank n , i.e.,

$$\text{rank}(\mathcal{H}(H)) = \text{order of a minimal realization of } H.$$

Therefore, existence of a finite dimensional realization of H (exact low-complexity linear time-invariant model for H) is equivalent to rank deficiency of a Hankel matrix constructed from the data. A minimal state representation can be obtained from a rank revealing factorization of $\mathcal{H}(H)$.

When there is no exact finite dimensional realization of the data or the exact realization is of high order, one may want to find an approximate realization of a specified low order n . These, respectively, approximate realization and model reduction problems naturally lead to *Hankel structured low rank approximation*.

The deterministic system realization and model reduction problems are further considered in Sects. 2.2, 3.1, and 4.2.

Stochastic System Realization

Let y be the output of an n th order linear time-invariant system, driven by white noise (a stochastic system) and let \mathbf{E} be the expectation operator. The sequence

$$R = (R(0), R(1), \dots, R(t), \dots)$$

defined by

$$R(\tau) := \mathbf{E}(y(t)y^\top(t - \tau))$$

is called the *autocorrelation* sequence of y . Stochastic realization theory is concerned with the problem of finding a state representation of a stochastic system that could have generated the observed output y , i.e., a linear time-invariant system driven by white noise, whose output correlation sequence is equal to R .

An important result in stochastic realization theory is that R is the output correlation sequence of an n th order stochastic system if and only if the Hankel matrix $\mathcal{H}(R)$ constructed from R has rank n , i.e.,

$$\text{rank}(\mathcal{H}(R)) = \text{order of a minimal stochastic realization of } R.$$

Therefore, stochastic realization of a random process y is equivalent to deterministic realization of its autocorrelation sequence R . When it exists, the finite dimensional stochastic realizations can be obtained from a rank revealing factorization of the matrix $\mathcal{H}(R)$.

In practice, only a finite number of finite length realizations of the output y are available, so that the autocorrelation sequence is *estimated* from y . With an estimate \widehat{R} of the autocorrelation R , the Hankel matrix $\mathcal{H}(\widehat{R})$ is almost certainly full rank, which implies that a finite dimensional stochastic realization cannot be found. Therefore, the problem of finding an approximate stochastic realization occurs. This problem is again Hankel structured low rank approximation.

System Identification

Realization theory considers a *system representation problem*: pass from one representation of a system to another. Alternatively, it can be viewed as a special exact identification problem: find from impulse response data (a special trajectory of the

system) a state space representation of the data generating system. The exact identification problem (also called deterministic identification problem) is to find from a general response of a system, a representation of that system. Let

$$w = \text{col}(u, y), \quad \text{where } u = (u(1), \dots, u(T)) \quad \text{and} \quad y = (y(1), \dots, y(T))$$

be an input/output trajectory of a discrete-time linear time-invariant system of order n with m inputs and p outputs and let n_{\max} be a given upper bound on n . Then the Hankel matrix

$$\mathcal{H}_{n_{\max}+1}(w) := \begin{bmatrix} w(1) & w(2) & \cdots & w(T - n_{\max}) \\ w(2) & w(3) & \cdots & w(T - n_{\max} + 1) \\ \vdots & \vdots & \ddots & \vdots \\ w(n_{\max} + 1) & w(n_{\max} + 1) & \cdots & w(T) \end{bmatrix} \quad (\mathcal{H}_i)$$

with $n_{\max} + 1$ block rows, constructed from the trajectory w , is rank deficient:

$$\text{rank}(\mathcal{H}_{n_{\max}+1}(w)) \leq \text{rank}(\mathcal{H}_{n_{\max}+1}(u)) + \text{order of the system}. \quad (\text{SYSID})$$

Conversely, if the Hankel matrix $\mathcal{H}_{n_{\max}+1}(w)$ has rank $(n_{\max} + 1)m + n$ and the matrix $\mathcal{H}_{2n_{\max}+1}(u)$ is full row rank (*persistence of excitation* of u), then w is a trajectory of a controllable linear time-invariant system of order n . Under the above assumptions, the data generating system can be identified from a rank revealing factorization of the matrix $\mathcal{H}_{n_{\max}+1}(w)$.

When there are measurement errors or the data generating system is not a low-complexity linear time-invariant system, the data matrix $\mathcal{H}_{n_{\max}+1}(w)$ is generically full rank. In such cases, an approximate low-complexity linear time-invariant model for w can be derived by finding a Hankel structured low rank approximation of $\mathcal{H}_{n_{\max}+1}(w)$. Therefore, the Hankel structured low rank approximation problem can be applied also for approximate system identification. Linear time-invariant system identification is a main topic of the book and appears frequently in the following chapters.

Similarly, to the analogy between deterministic and stochastic system realization, there is an analogy between deterministic and stochastic system identification. The latter analogy suggests an application of Hankel structured low rank approximation to stochastic system identification.

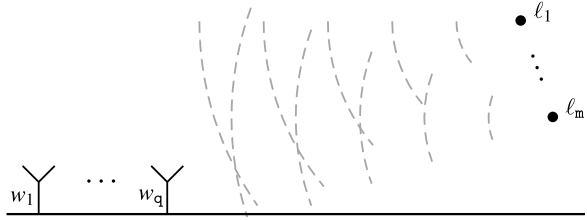
Applications in Computer Algebra

Greatest Common Divisor of Two Polynomials

The greatest common divisor of the polynomials

$$p(z) = p_0 + p_1z + \cdots + p_nz^n \quad \text{and} \quad q(z) = q_0 + q_1z + \cdots + q_mz^m$$

Fig. 1.3 Antenna array processing setup



Consider, first, the case of a single source. The source intensity ℓ_1 (the signal) is a function of time. Let $w(t) \in \mathbb{R}^q$ be the response of the array at time t (w_i being the response of the i th antenna). Assuming that the source is far from the array (relative to the array's length), the array's response is proportional to the source intensity

$$w(t) = p_1 \ell_1(t - \tau_1),$$

where τ_1 is the time needed for the wave to travel from the source to the array and $p_1 \in \mathbb{R}^q$ is the array's response to the source emitting at a unit intensity. The vector p_1 depends only on the array geometry and the source location and is therefore constant in time. Measurements of the antenna at time instants $t = 1, \dots, T$ give a data matrix

$$D := [w(1) \cdots w(T)] = p_1 \underbrace{[\ell_1(1 - \tau) \cdots \ell_1(T - \tau)]}_{\ell_1} = p_1 \ell_1,$$

which has rank equal to one.

Consider now $m < q$ distant sources emitting with intensities ℓ_1, \dots, ℓ_m . Let p_k be the response of the array to the k th source emitting alone with unit intensity. Assuming that the array responds linearly to a mixture of sources, we have

$$D = [w(1) \cdots w(T)] = \sum_{k=1}^m p_k \underbrace{[\ell_k(1 - \tau_k) \cdots \ell_k(T - \tau_k)]}_{\ell_k} = PL,$$

where $P := [p_1 \cdots p_m]$, $L := \text{col}(\ell_1, \dots, \ell_m)$, and τ_k is the delay of the wave coming from the k th source. This shows that the rank of D is less than or equal to the number of sources m . If the number of sources m is less than the number of antennas q and m is less than the number of samples T , the sources intensities ℓ_1, \dots, ℓ_m are linearly independent, and the unit intensity array patterns p_1, \dots, p_m are linearly independent, then we have

$$\text{rank}(D) = \text{the number of sources transmitting to the array.}$$

Moreover, the factors P and L in a rank revealing factorization PL of D carry information about the source locations.

With noisy observations, the matrix D is generically a full rank matrix. Then, assuming that the array's geometry is known, low rank approximation can be used to estimate the number of sources and their locations.

Applications in Chemometrics

Multivariate Calibration

A basic problem in chemometrics, called multivariate calibration, is identification of the number and type of chemical components from spectral measurements of mixtures of these components. Let $p_k \in \mathbb{R}^q$ be the spectrum of the k th component at q predefined frequencies. Under a linearity assumption, the spectrum of a mixture of m components with concentrations ℓ_1, \dots, ℓ_m is $d = P\ell$, where $P := [p_1 \cdots p_m]$ and $\ell = \text{col}(\ell_1, \dots, \ell_m)$. Given N mixtures of the components with vectors of concentrations $\ell^{(1)}, \dots, \ell^{(N)}$, the matrix of the corresponding spectra d_1, \dots, d_N is

$$D := [d_1 \cdots d_N] = \sum_{k=1}^m p_k \underbrace{\left[\ell_k^{(1)} \cdots \ell_k^{(N)} \right]}_{\ell_k} = PL. \quad (\text{RRF})$$

Therefore, the rank of D is less than or equal to the number of components m . Assuming that $q > m$, $N > m$, the spectral responses p_1, \dots, p_m of the components are linearly independent, and the concentration vectors ℓ_1, \dots, ℓ_m are linearly independent, we have

$$\text{rank}(D) = \text{the number of chemical components in the mixtures.}$$

The factors P and L in a rank revealing factorization PL of D carry information about the components' spectra and the concentrations of the components in the mixtures. Noisy spectral observations lead to a full rank matrix D , so that low rank approximation can be used to estimate the number of chemical components, their concentrations, and spectra.

Applications in Psychometrics

Factor Analysis

The psychometric data are test scores and biometrics of a group of people. The test scores can be organized in a data matrix D , whose rows correspond to the scores and the columns correspond to the group members. Factor analysis is a popular method that explains the data as a linear combination of a small number of abilities of the group members. These abilities are called *factors* and the weights by which they are combined in order to reproduce the data are called *loadings*. Factor analysis is based on the assumption that the exact data matrix is low rank with rank being equal to the number of factors. Indeed, the factor model can be written as $D = PL$, where the columns of P correspond to the factors and the rows of L correspond to the loadings. In practice, the data matrix is full rank because the factor model is an

idealization of the way test data are generated. Despite the fact that the factor model is a simplification of the reality, it can be used as an approximation of the way humans perform on tests. Low rank approximation then is a method for deriving optimal in a specified sense approximate psychometric factor models.

The factor model, explained above, is used to assess candidates at the US universities. An important element of the acceptance decision in US universities for undergraduate study is the Scholastic Aptitude Test, and for postgraduate study, the Graduate Record Examination. These tests report three independent scores: writing, mathematics, and critical reading for the Scholastic Aptitude Test; and verbal, quantitative, and analytical for the Graduate Record Examination. The three scores assess what are believed to be the three major factors for, respectively, undergraduate and postgraduate academic performance. In other words, the premise on which the tests are based is that the ability of a prospective student to do undergraduate and postgraduate study is predicted well by a combination of the three factors. Of course, in different areas of study, the weights by which the factors are taken into consideration are different. Even in pure subjects, such as mathematics, however, the verbal as well as quantitative and analytical ability play a role.

Many graduate-school advisors have noted that an applicant for a mathematics fellowship with a high score on the verbal part of the Graduate Record Examination is a better bet as a Ph.D. candidate than one who did well on the quantitative part but badly on the verbal.

Halmos (1985, p. 5)

Applications in Machine Learning

Natural Language Processing

Latent semantic analysis is a method in natural language processing for document classification, search by keywords, synonymy and polysemy detection, etc. Latent semantic analysis is based on low rank approximation and fits into the pattern of the other methods reviewed here:

1. An exact data matrix is rank deficient with rank related to the complexity of the data generating model.
2. A noisy data matrix is full rank and, for the purpose of approximate modeling, it is approximated by a low rank matrix.

Consider N documents, involving q terms and m concepts. If a document belongs to the k th concept only, it contains the i th term with frequency p_{ik} , resulting in the vector of term frequencies $p_k := \text{col}(p_{1k}, \dots, p_{qk})$, related to the k th concept. The latent semantic analysis model assumes that if a document involves a mixture of the concepts with weights ℓ_1, \dots, ℓ_m (ℓ_k indicates the relevance of the k th concept to the document), then the vector of term frequencies for that document is

$$d = P\ell, \quad \text{where } P := [p_1 \cdots p_m] \text{ and } \ell = \text{col}(\ell_1, \dots, \ell_m).$$

Let d_j be the vector of term frequencies, related to the j th document and let $\ell_k^{(j)}$ be the relevance of the k th concept to the j th document. Then, according to the latent semantic analysis model, the term–document frequencies for the N documents form a data matrix, satisfying (RRF). Therefore, the rank of the data matrix is less than or equal to the number of concepts m . Assuming that m is smaller than the number of terms q , m is smaller than the number of documents N , the term frequencies p_1, \dots, p_m are linearly independent, and the relevance of concepts ℓ_1, \dots, ℓ_m are linearly independent, we have

$$\text{rank}(D) = \text{the number of concepts related to the documents.}$$

The factors P and L in a rank revealing factorization PL of D carry information about the relevance of the concepts to the documents and the term frequencies related to the concepts.

The latent semantic analysis model is not satisfied exactly in practice because the notion of (small number of) concepts related to (many) documents is an idealization. Also the linearity assumption is not likely to hold in practice. In reality the term–document frequencies matrix D is full rank indicating that the number of concepts is equal to either the number of terms or the number of documents. Low rank approximation, however, can be used to find a small number of concepts that explain approximately the term–documents frequencies via the model (RRF). Subsequently, similarity of documents can be evaluated in the concepts space, which is a low dimensional vector space. For example, the j_1 th and j_2 th documents are related if they have close relevance $\ell_k^{(j_1)}$ and $\ell_k^{(j_2)}$ to all concepts $k = 1, \dots, m$. This gives a way to classify the documents. Similarly, terms can be clustered in the concepts space by looking at the rows of the P matrix. Nearby rows of P correspond to terms that are related to the same concepts. (Such terms are likely to be synonymous.) Finally, a search for documents by keywords can be done by first translating the keywords to a vector in the concepts space and then finding a nearby cluster of documents to this vector. For example, if there is a single keyword, which is the i th term, then the i th row of the P matrix shows the relevant combination of concepts for this search.

Recommender System

The main issue underlying the abstract low rank approximation problem and the applications reviewed up to now is data approximation. In the recommender system problem, the main issue is the one of *missing data*: given ratings of some items by some users, infer the missing ratings. Unique recovery of the missing data is impossible without additional assumptions. The underlying assumption in many recommender system problems is that the complete matrix of the ratings is of low rank.

Consider q items and N users and let d_{ij} be the rating of the i th item by the j th user. As in the psychometrics example, it is assumed that there is a “small” number m of “typical” (or characteristic, or factor) users, such that all user ratings can

be obtained as linear combinations of the ratings of the typical users. This implies that the complete matrix $D = [d_{ij}]$ of the ratings has rank m , i.e.,

$$\text{rank}(D) = \text{number of "typical" users.}$$

Then exploiting the prior knowledge that the number of "typical" users is small, the missing data recovery problem can be posed as the following matrix completion problem

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{D} && \text{rank}(\widehat{D}) \\ & \text{subject to} && \widehat{D}_{ij} = D_{ij} && \text{for all } (i, j), \text{ where } D_{ij} \text{ is given.} \end{aligned} \tag{MC}$$

This gives a procedure for solving the exact modeling problem (the given elements of D are assumed to be exact). The corresponding solution method can be viewed as the equivalent of the rank revealing factorization problem in exact modeling problems, for the case of complete data.

Of course, the rank minimization problem (MC) is much harder to solve than the rank revealing factorization problem. Moreover, theoretical justification and additional assumptions (about the number and distribution of the given elements of D) are needed for a solution \widehat{D} of (MC) to be unique and to coincide with the complete true matrix D . It turns out, however, that under certain specified assumptions exact recovery is possible by solving the convex optimization problem obtained by replacing $\text{rank}(\widehat{D})$ in (MC) with the nuclear norm

$$\|\widehat{D}\|_* := \text{sum of the singular values of } \widehat{D}.$$

The importance of the result is that under the specified assumptions the hard problem (MC) can be solved efficiently and reliably by convex optimization methods.

In real-life application of recommender systems, however, the additional problem of data approximation occurs. In this case the constraint $\widehat{D}_{ij} = D_{ij}$ of (MC) has to be relaxed, e.g., replacing it by

$$\widehat{D}_{ij} = D_{ij} + \Delta D_{ij},$$

where ΔD_{ij} are corrections, accounting for the data uncertainty. The corrections are additional optimization variables. Taking into account the prior knowledge that the corrections are small, a term $\lambda \|\Delta D\|_F$ is added in the cost function. The resulting matrix approximation problem is

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{D} \text{ and } \Delta D && \text{rank}(\widehat{D}) + \lambda \|\Delta D\|_F \\ & \text{subject to} && \widehat{D}_{ij} = D_{ij} + \Delta D_{ij} && \text{for all } (i, j), \text{ where } D_{ij} \text{ is given.} \end{aligned} \tag{AMC}$$

In a stochastic setting the term $\lambda \|\Delta D\|_F$ corresponds to the assumption that the true data D is perturbed with noise that is zero mean, Gaussian, independent, and with equal variance.

Again the problem can be relaxed to a convex optimization problem by replacing rank with nuclear norm. The choice of the λ parameter reflects the trade-off

between complexity (number of identified “typical” users) and accuracy (size of the correction ΔD) and depends in the stochastic setting on the noise variance.

Nuclear norm and low rank approximation methods for estimation of missing values are developed in Sects. 3.3 and 5.1.

Multidimensional Scaling

Consider a set of N points in the plane

$$\mathcal{X} := \{x_1, \dots, x_N\} \subset \mathbb{R}^2$$

and let d_{ij} be the squared distance from x_i to x_j , i.e.,

$$d_{ij} := \|x_i - x_j\|_2^2.$$

The $N \times N$ matrix $D = [d_{ij}]$ of the pair-wise distances, called in what follows the distance matrix (for the set of points \mathcal{X}), has rank at most 4. Indeed,

$$d_{ij} = (x_i - x_j)^\top (x_i - x_j) = x_i^\top x_i - 2x_i^\top x_j + x_j^\top x_j,$$

so that

$$D = \underbrace{\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} x_1^\top x_1 & \dots & x_N^\top x_N \end{bmatrix}}_{\text{rank} \leq 1} - 2 \underbrace{\begin{bmatrix} x_1^\top \\ \vdots \\ x_N^\top \end{bmatrix} \begin{bmatrix} x_1 & \dots & x_N \end{bmatrix}}_{\text{rank} \leq 2} + \underbrace{\begin{bmatrix} x_1^\top x_1 \\ \vdots \\ x_N^\top x_N \end{bmatrix} \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}}_{\text{rank} \leq 1}. \quad (*)$$

The localization problem from pair-wise distances is: given the distance matrix D , find the locations $\{x_1, \dots, x_N\}$ of the points up to a rigid transformation, i.e., up to translation, rotation, and reflection of the points. Note that rigid transformations preserve the pair-wise distances, so that the distance matrix D alone is not sufficient to locate the points uniquely.

With exact data, the problem can be posed and solved as a rank revealing factorization problem (*). With noisy measurements, however, the matrix D is generically full rank. In this case, the relative (up to rigid transformation) point locations can be *estimated* by approximating D by a rank-4 matrix \widehat{D} . In order to be a valid distance matrix, however, \widehat{D} must have the structure

$$\widehat{D} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} \widehat{x}_1^\top \widehat{x}_1 & \dots & \widehat{x}_N^\top \widehat{x}_N \end{bmatrix} - 2 \widehat{X}^\top \widehat{X} + \begin{bmatrix} \widehat{x}_1^\top \widehat{x}_1 \\ \vdots \\ \widehat{x}_N^\top \widehat{x}_N \end{bmatrix} \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}, \quad (\text{MDS})$$

for some $\widehat{X} = [\widehat{x}_1 \ \dots \ \widehat{x}_N]$, i.e., the estimation problem is a *bilinearly structured low rank approximation problem*.

Microarray Data Analysis

The measurements of a microarray experiment are collected in a $q \times N$ real matrix D —rows correspond to genes and columns correspond to time instances. The element d_{ij} is the *expression level* of the i th gene at the j th moment of time. The rank of D is equal to the number of *transcription factors* that regulate the gene expression levels

$$\text{rank}(D) = \text{number of transcription factors.}$$

In a rank revealing factorization $D = PL$, the j th column of L is a vector of intensities of the transcription factors at time j , and the i th row of P is a vector of sensitivities of the i th gene to the transcription factors. For example, p_{ij} equal to zero means that the j th transcription factor does not regulate the i th gene.

An important problem in bioinformatics is to discover what transcription factors regulate a particular gene and what their time evaluations are. This problem amounts to computing an (approximate) factorization PL of the matrix D . The need of approximation comes from:

1. inability to account for all relevant transcription factors (therefore accounting only for a few dominant ones), and
2. measurement errors occurring in the collection of the data.

Often it is known a priori that certain transcription factors do not regulate certain genes. This implies that certain elements of the sensitivity matrix P are known to be zeros. In addition, the transcription factor activities are modeled to be nonnegative, smooth, and periodic functions of time. Where transcription factors down regulate a gene, the elements of P have to be negative to account for this. In Sect. 5.4, this prior knowledge is formalized as constraints in a low rank matrix approximation problem and optimization methods for solving the problem are developed.

Applications in Computer Vision

Conic Section Fitting

In the applications reviewed so far, the low rank approximation problem was applied to *linear* data modeling. Nonlinear data modeling, however, can also be formulated as a low rank approximation problem. The key step—“linearizing” the problem— involves preprocessing the data by a nonlinear function defining the model structure. In the machine learning literature, where nonlinear data modeling is a common practice, the nonlinear function is called the *feature map* and the resulting modeling methods are referred to as *kernel methods*.

As a specific example, consider the problem of fitting data by a conic section, i.e., given a set of points in the plane

$$\{d_1, \dots, d_N\} \subset \mathbb{R}^2, \quad \text{where } d_j = \text{col}(x_j, y_j),$$

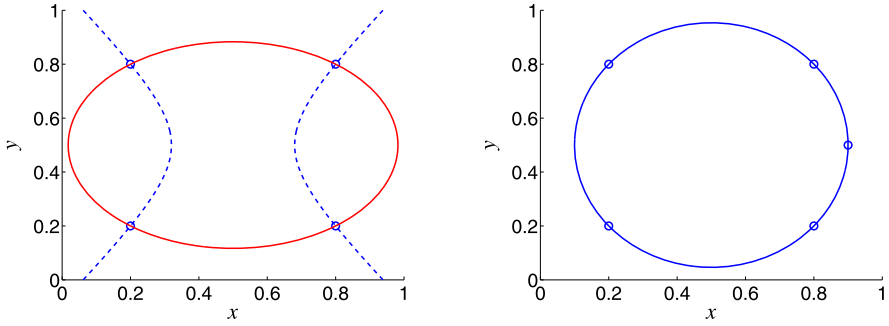


Fig. 1.4 Conic section fitting. *Left:* $N = 4$ points (circles) have nonunique fit (two fits are shown in the figure with solid and dashed lines). *Right:* $N = 5$ different points have a unique fit (solid line)

find a conic section

$$\mathcal{B}(A, b, c) := \{d \in \mathbb{R}^2 \mid d^\top A d + b^\top d + c = 0\} \quad (\mathcal{B}(A, b, c))$$

that fits them. Here A is a 2×2 symmetric matrix, b is a 2×1 vector, and c is a scalar. A , b , and c are parameters defining the conic section. In order to avoid a trivial case $\mathcal{B} = \mathbb{R}^2$, it is assumed that at least one of the parameters A , b , or c is nonzero. The representation $(\mathcal{B}(A, b, c))$ is called implicit representation, because it imposes a relation (implicit function) on the elements x and y of d .

Defining the parameter vector

$$\theta := [a_{11} \ 2a_{12} \ b_1 \ a_{22} \ b_2 \ c],$$

and the extended data vector

$$d_{\text{ext}} := \text{col}(x^2, xy, x, y^2, y, 1), \quad (d_{\text{ext}})$$

we have

$$d \in \mathcal{B}(\theta) = \mathcal{B}(A, b, c) \iff \theta d_{\text{ext}} = 0.$$

(In the machine learning terminology, the map $d \mapsto d_{\text{ext}}$, defined by (d_{ext}) , is the feature map for the conic section model.) Consequently, all data points d_1, \dots, d_N are fitted by the model if

$$\theta \underbrace{[d_{\text{ext},1} \ \cdots \ d_{\text{ext},N}]}_{D_{\text{ext}}} = 0 \iff \text{rank}(D_{\text{ext}}) \leq 5. \quad (\text{CSF})$$

Therefore, for $N > 5$ data points, exact fitting is equivalent to rank deficiency of the extended data matrix D_{ext} . For $N < 5$ data points, there is nonunique exact fit independently of the data. For $N = 5$ different points, the exact fitting conic section is unique, see Fig. 1.4.

With $N > 5$ noisy data points, the extended data matrix D_{ext} is generically full rank, so that an exact fit does not exist. A problem called geometric fitting is to minimize the sum of squared distances from the data points to the conic section. The problem is equivalent to *quadratically structured low rank approximation*.

Generalization of the conic section fitting problem to algebraic curve fitting and solution methods for the latter are presented in Chap. 6.

Exercise 1.1 Find and plot another conic section that fits the points

$$d_1 = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix}, \quad d_2 = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}, \quad d_3 = \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}, \quad \text{and} \quad d_4 = \begin{bmatrix} 0.8 \\ 0.8 \end{bmatrix},$$

in Fig. 1.4, left.

Exercise 1.2 Find the parameters $(A, b, 1)$ in the representation $(\mathcal{B}(A, b, c))$ of the ellipse in Fig. 1.4, right. The data points are:

$$d_1 = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix}, \quad d_2 = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}, \quad d_3 = \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}, \quad d_4 = \begin{bmatrix} 0.8 \\ 0.8 \end{bmatrix}, \quad \text{and} \quad d_5 = \begin{bmatrix} 0.9 \\ 0.5 \end{bmatrix}.$$

$$(\text{Answer: } A = \begin{bmatrix} 3.5156 & 0 \\ 0 & 2.7344 \end{bmatrix}, b = \begin{bmatrix} -3.5156 \\ -2.7344 \end{bmatrix})$$

Fundamental Matrix Estimation

A scene is captured by two cameras at fixed locations (stereo vision) and N matching pairs of points

$$\{u_1, \dots, u_N\} \subset \mathbb{R}^2 \quad \text{and} \quad \{v_1, \dots, v_N\} \subset \mathbb{R}^2$$

are located in the resulting images. Corresponding points u and v of the two images satisfy what is called an *epipolar constraint*

$$\begin{bmatrix} v^\top & 1 \end{bmatrix} F \begin{bmatrix} u \\ 1 \end{bmatrix} = 0, \quad \text{for some } F \in \mathbb{R}^{3 \times 3}, \text{ with } \text{rank}(F) = 2. \quad (\text{EPI})$$

The 3×3 matrix $F \neq 0$, called the *fundamental matrix*, characterizes the relative position and orientation of the cameras and does not depend on the selected pairs of points. Estimation of F from data is a necessary calibration step in many computer vision methods.

The epipolar constraint (EPI) is linear in F . Indeed, defining

$$d_{\text{ext}} := \begin{bmatrix} u_x v_x & u_x v_y & u_x & u_y v_x & u_y v_y & u_y & v_x & v_y & 1 \end{bmatrix}^\top \in \mathbb{R}^9, \quad (d'_{\text{ext}})$$

where $u = \text{col}(u_x, u_y)$ and $v = \text{col}(v_x, v_y)$, (EPI) can be written as

$$\text{vec}^\top(F) d_{\text{ext}} = 0.$$

Note that, as in the application for conic section fitting, the original data (u, v) are mapped to an extended data vector d_{ext} via a nonlinear function (a feature map). In this case, however, the function is *bilinear*.

Taking into account the epipolar constraints for all data points, we obtain the matrix equation

$$\text{vec}^\top(F)D_{\text{ext}} = 0, \quad \text{where } D_{\text{ext}} := [d_{\text{ext},1} \cdots d_{\text{ext},N}]. \quad (\text{FME})$$

The rank constraint imposed on F implies that F is a nonzero matrix. Therefore, by (FME), for $N \geq 8$ data points, the extended data matrix D_{ext} is rank deficient. Moreover, the fundamental matrix F can be reconstructed up to a scaling factor from a vector in the left kernel of D_{ext} .

Noisy data with $N \geq 8$ data points generically give rise to a full rank extended data matrix D_{ext} . The estimation problem is a *bilinearly structured low rank approximation* problem with an additional constraint that $\text{rank}(F) = 2$.

Summary of Applications

Table 1.1 summarized the reviewed applications: given data, data matrix constructed from the original data, structure of the data matrix, and meaning of the rank of the data matrix in the context of the application. More applications are mentioned in the notes and references section in the end of the chapter.

1.4 Overview of Algorithms

The rank constraint in the low rank approximation problem corresponds to the constraint in the data modeling problem that the data are fitted exactly by a linear model of bounded complexity. Therefore, the question of representing the rank constraint in the low rank approximation problem corresponds to the question of choosing the model representation in the data fitting problem. Different representations lead to

- *optimization problems*, the relation among which may not be obvious;
- *algorithms*, which may have different convergence properties and efficiency; and
- *numerical software*, which may have different numerical robustness.

The virtues of the abstract, representation free, low rank approximation problem formulation, are both *conceptual*: it clarifies the equivalence among different parameter optimization problems, and *practical*: it shows various ways of formulating one and the same high level data modeling problem as parameter optimization problems. On the conceptual level, the low rank approximation problem formulation shows what one aims to achieve without a reference to implementation details. In particular, the model representation is such a detail, which is not needed for a high level formulation of data modeling problems. As discussed next, however, the representation is unavoidable when one solves the problem analytically or numerically.

Table 1.1 Meaning of the rank of the data matrix in the applications

Application	Data	Data matrix	Structure	Rank =	Ref.
approximate realization	impulse response H	$\mathcal{H}(H)$	Hankel	system's order	Sect. 2.2 Sect. 3.1 Sect. 4.2
stochastic realization	autocorrelation function R	$\mathcal{H}(R)$	Hankel	system's order	–
system identification	trajectory w of the system	$\mathcal{H}_{n_{\max}+1}(w)$	Hankel	(SYSID)	Sect. 2.3 Sect. 4.3
approximate GCD	polynomials p_d and q_d	$\mathcal{R}(p_d, q_d)$	Sylvester	(GCD)	Sect. 3.2
array processing	array response $(w(1), \dots, w(T))$	$[w(1) \cdots w(T)]$	unstructured	# of signal sources	–
multivariate calibration	spectral responses $\{d_1, \dots, d_N\} \subset \mathbb{R}^q$	$[d_1 \cdots d_N]$	unstructured	# of chemical components	–
factor analysis	test scores d_{ij}	$[d_{ij}]$	unstructured	# of factors	–
natural language processing	term–document frequencies d_{ij}	$[d_{ij}]$	unstructured	# of concepts	–
recommender system	some ratings d_{ij}	$[d_{ij}]$	unstructured	# of tastes missing data	Sect. 3.3 Sect. 5.1
multidimensional scaling	pair-wise distances d_{ij}	$[d_{ij}]$	(MDS)	$\dim(x) + 2$	–
microarray data analysis	gene expression levels d_{ij}	$[d_{ij}]$	unstructured	# of transcript. factors	Sect. 5.4
conic section fitting	points $\{d_1, \dots, d_N\} \subset \mathbb{R}^2$	(d_{ext}) , (CSF)	quadratic	5	Chap. 6
fundamental matrix estimation	points $u_j, v_j \in \mathbb{R}^2$	(d'_{ext}) , (FME)	bilinear	6	–

On the practical level, the low rank problem formulation allows one to translate the abstract data modeling problem to different concrete parametrized problems by choosing the model representation. Different representations naturally lend themselves to different analytical and numerical methods. For example, a controllable linear time-invariant system can be represented by a transfer function, state space, convolution, etc. representations. The analysis tools related to these representations are rather different and, consequently, the obtained solutions differ despite of the fact that they solve the same abstract problem. Moreover, the parameter optimization problems, resulting from different model representations, lead to algorithms and numerical implementations, whose robustness properties and computational efficiency differ. Although, often in practice, there is no universally “best” algorithm or software implementation, having a wider set of available options is an advantage.

Independent of the choice of the rank representation only a few special low rank approximation problems have analytic solutions. So far, the most important special case with an analytic solution is the unstructured low rank approximation in

the Frobenius norm. The solution in this case can be obtained from the singular value decomposition of the data matrix (Eckart–Young–Mirsky theorem). Extensions of this basic solution are problems known as generalized and restricted low rank approximation, where some columns or, more generally submatrices of the approximation, are constrained to be equal to given matrices. The solutions to these problems are given by, respectively, the generalized and restricted singular value decompositions. Another example of low rank approximation problem with analytic solution is the circulant structured low rank approximation, where the solution is expressed in terms of the discrete Fourier transform of the data.

In general, low rank approximation problems are NP-hard. There are three fundamentally different solution approaches for the general low rank approximation problem:

- heuristic methods based on convex relaxations,
- local optimization methods, and
- global optimization methods.

From the class of heuristic methods the most popular ones are the subspace methods. The approach used in the subspace type methods is to relax the difficult low rank approximation problem to a problem with an analytic solution in terms of the singular value decomposition, e.g., ignore the structure constraint of a structured low rank approximation problem. The subspace methods are found to be very effective in model reduction, system identification, and signal processing. The class of the subspace system identification methods is based on the unstructured low rank approximation in the Frobenius norm (i.e., singular value decomposition) while the original problems are Hankel structured low rank approximation.

The methods based on local optimization split into two main categories:

- *alternating projections* and
- *variable projections*

type algorithms. Both alternating projections and variable projections exploit the bilinear structure of the low rank approximation problems.

In order to explain the ideas underlining the alternating projections and variable projections methods, consider the optimization problem

$$\text{minimize over } P \in \mathbb{R}^{q \times m} \text{ and } L \in \mathbb{R}^{m \times N} \quad \|D - PL\|_F^2 \quad (\text{LRA}_P)$$

corresponding to low rank approximation with an image representation of the rank constraint. The term PL is bilinear in the optimization variables P and L , so that for a fixed P , (LRA_P) becomes a linear least squares problem in L and vice versa, for a fixed L , (LRA_P) becomes a linear least squares problem in P . This suggests an iterative algorithm starting from an initial guess for P and L and alternatively solves the problem with one of the variables fixed. Since each step is a projection operation the method has the name alternating projections. It is globally convergent to a locally optimal solution of (LRA_P) with a linear convergence rate.

The bilinear nature of (LRA_P) implies that for a fixed P the problem can be solved in closed form with respect to L . This gives us an equivalent cost function

depending on P . Subsequently, the original problem can be solved by minimizing the equivalent cost function over P . Of course, the latter problem is a nonlinear optimization problem. Standard local optimization methods can be used for this purpose. The elimination of the L variable from the problem has the advantage of reducing the number of optimization variables, thus simplifying the problem. Evaluation of the cost function for a given P is a projection operation. In the course of the nonlinear minimization over P , this variable changes, thus the name of the method—variable projections.

In the statistical literature, the alternating projections algorithm is given the interpretation of *expectation maximization*. The problem of computing the optimal approximation $\widehat{D} = PL$, given P is the expectation step and the problem of computing P , given L is the maximization step of the expectation maximization procedure.

1.5 Literate Programming

At first, I thought programming was primarily analogous to musical composition—to the creation of intricate patterns, which are meant to be performed. But lately I have come to realize that a far better analogy is available: Programming is best regarded as the process of creating *works of literature*, which are meant to be read.

Knuth (1992, p. ix)

The ideas presented in the book are best expressed as algorithms for solving data modeling problems. The algorithms, in turn, are practically useful when implemented in ready-to-use software. The gap between the theoretical discussion of data modeling methods and the practical implementation of these methods is bridged by using a literate programming style. The software implementation (MATLAB code) is interwoven in the text, so that the full implementation details are available in a human readable format and they come in the appropriate context of the presentation.

A literate program is composed of interleaved code segments, called chunks, and text. The program can be split into chunks in any way and the chunks can be presented in any order, deemed helpful for the understanding of the program. This allows us to focus on the logical structure of the program rather than the way a computer executes it. The actual computer executable code is *tangled* from a *web* of the code chunks by skipping the text and putting the chunks in the right order. In addition, literate programming allows us to use a powerful typesetting system such as \LaTeX (rather than plain text) for the documentation of the code.

The `noweb` system for literate programming is used. Its main advantage over alternative systems is independence of the programming language being used.

Next, some typographic conventions are explained. The code is typeset in `small true type` font and consists of a number of code chunks. The code chunks begin with tags enclosed in angle brackets (e.g., `<code tag>`) and are sequentially numbered by the page number and a letter identifying them on the page. Thus the chunk's

identification number (given to the left of the tag) is also used to locate the chunk in the text. For example,

```
25a <Print a figure 25a>≡
function print_fig(file_name)
    xlabel('x'), ylabel('y'), title('t'),
    set(gca, 'fontsize', 25)
    eval(sprintf('print -depsc %s.eps', file_name));
```

Defines:

print_fig, used in chunks 101b, 114b, 126b, 128, 163c, 192f, 213, and 222b.

(a function exporting the current figure to an encapsulated postscript file with a specified name, using default labels and font size) has identification number 25a, locating the code as being on p. 25.

If a chunk is included in, is a continuation of, or is continued by other chunk(s), its definition has references to the related chunk(s). The syntax convention for doing this is best explained by an example.

Example: Block Hankel Matrix Constructor

Consider the implementation of the (block) Hankel matrix constructor

$$\mathcal{H}_{i,j}(w) := \begin{bmatrix} w(1) & w(2) & \cdots & w(j) \\ w(2) & w(3) & \cdots & w(j+1) \\ \vdots & \vdots & \ddots & \vdots \\ w(i) & w(i+1) & \cdots & w(j+i-1) \end{bmatrix}, \quad (\mathcal{H}_{i,j})$$

where

$$w = (w(1), \dots, w(T)), \quad \text{with } w(t) \in \mathbb{R}^{q \times N}.$$

The definition of the function, showing its input and output arguments, is

```
25b <Hankel matrix constructor 25b>≡ 26a>
function H = blkhank(w, i, j)
```

Defines:

blkhank, used in chunks 76a, 79, 88c, 109a, 114a, 116a, 117b, 120b, 125b, 127b, and 211.

(The reference to the right of the identification tag shows that the definition is continued in chunk number 26a.) The third input argument of blkhank—the number of block columns j is optional. Its default value is maximal number of block columns

$$j = T - i + 1.$$

```
25c <optional number of (block) columns 25c>≡ 26)
    if nargin < 3 | isempty(j), j = T - i + 1; end
    if j <= 0, error('Not enough data.'), end
```

(The reference to the right of the identification tag now shows that this chunk is included in other chunks.)

Two cases are distinguished, depending on whether w is a vector ($N = 1$) or matrix ($N > 1$) valued trajectory.

```
26a <Hankel matrix constructor 25b>+≡ <25b
    if length(size(w)) == 3
        <matrix valued trajectory w 26b>
    else
        <vector valued trajectory w 26e>
    end
```

(The reference to the right of the identification tag shows that this chunk is a continuation of chunk 25b and is not followed by other chunks. Its body includes chunks 26b and 26e.)

- If w is a matrix valued trajectory, the input argument w should be a 3 dimensional tensor, constructed as follows:

$$w(:, :, t) = w(t) \quad (w)$$

```
26b <matrix valued trajectory w 26b>≡ (26a) 26c>
    [q, N, T] = size(w);
    <optional number of (block) columns 25c>
```

(This chunk is both included and followed by other chunks.) In this case, the construction of the block Hankel matrix $H_{i,j}(w)$ is done explicitly by a double loop:

```
26c <matrix valued trajectory w 26b>+≡ (26a) <26b
    H = zeros(i * q, j * N);
    for ii = 1:i
        for jj = 1:j
            H((ii - 1) * q + 1):(ii * q), ...
              ((jj - 1) * N + 1):(jj * N)) = w(:, :, ii + jj - 1);
        end
    end
```

- If w is a vector valued trajectory, the input argument w should be a matrix formed as $w(:, t) = w(t)$, however, since T must be greater than or equal to the number of variables $q := \dim(w(t))$, when w has more rows than columns, the input is treated as $w(t, :) = w^\top(t)$.

```
26d <reshape w and define q, T 26d>≡ (26e 80 87b 115b 117a 119b 120a)
    [q, T] = size(w); if T < q, w = w'; [q, T] = size(w); end
```

```
26e <vector valued trajectory w 26e>≡ (26a) 27>
    <reshape w and define q, T 26d>
    <optional number of (block) columns 25c>
```

The reason to consider the case of a vector valued w separately is that in this case the construction of the Hankel matrix $\mathcal{H}_{i,j}(w)$ can be done with a single loop along the block rows.

27 \langle vector valued trajectory w 26e $\rangle + \equiv$ (26a) \triangleleft 26e

```
H = zeros(i * q, j);
for ii = 1:i
    H((ii - 1) * q + 1):(ii * q), :)
      = w(:, ii:(ii + j - 1));
end
```

Since in typical situations when `blkhank` is used (system identification problems), $i \ll j$ and MATLAB, being an interpreted language, executes `for` loops slowly, the reduction to a single `for` loop along the block rows of the matrix leads to significant decrease of the execution time compared to the implementation with two nested `for` loops in the general case.

Exercise 1.3 Download and install `noweb` from

www.cs.tufts.edu/~nr/noweb/.

Exercise 1.4 Write a literate program for constructing the Sylvester matrix $\mathcal{R}(p, q)$, defined in (\mathcal{R}) on p. 11. Use your program to find the degree d of the greatest common divisor of the polynomials

$$p(z) = 1 + 3z + 5z^2 + 3z^3 \quad \text{and} \quad q(z) = 3 + 5z + 3z^2 + z^3.$$

(Answer: $d = 1$)

1.6 Notes

Classical and Behavioral Paradigms for Data Modeling

Methods for solving overdetermined systems of linear equations (i.e., data modeling methods using the classical input/output representation paradigm) are reviewed in Appendix A. The behavioral paradigm for data modeling was put forward by Jan C. Willems in the early 1980s. It became firmly established with the publication of the three part paper Willems (1986a, 1986b, 1987). Other landmark publications on the behavioral paradigm are Willems (1989, 1991, 2007), and the book Polderman and Willems (1998).

The numerical linear algebra problem of low rank approximation is a computational tool for data modeling, which fits the behavioral paradigm as “a hand fits a glove”. Historically the low rank approximation problem is closely related to the singular value decomposition, which is a method for computing low rank approximations and is a main tool in many algorithms for data modeling. A historical account of the development of the singular value decomposition is given in Stewart (1993). The Eckart–Young–Mirsky matrix low rank approximation theorem is proven in Eckart and Young (1936).

Applications

For details about the realization and system identification problems, see Sects. 2.2, 3.1, and 4.3. Direction of arrival and adaptive beamforming problems are discussed in Kumaresan and Tufts (1983), Krim and Viberg (1996). Low rank approximation methods (alternating least squares) for estimation of mixture concentrations in chemometrics are proposed in Wentzell et al. (1997). An early reference on the approximate greatest common divisor problem is Karmarkar and Lakshman (1998). Efficient optimization based methods for approximate greatest common divisor computation are discussed in Sect. 3.2. Other computer algebra problems that reduce to structured low rank approximation are discussed in Botting (2004).

Many problems for information retrieval in machine learning, see, e.g., Shawe-Taylor and Cristianini (2004), Bishop (2006), Fierro and Jiang (2005), are low rank approximation problems and the corresponding solution techniques developed in the machine learning community are methods for solving low rank approximation problems. For example, clustering problems have been related to low rank approximation problems in Ding and He (2004), Kiers (2002), Vichia and Saporta (2009). Machine learning problems, however, are often posed in a stochastic estimation setting which obscures their deterministic approximation interpretation. For example, principal component analysis (Jolliffe 2002; Jackson 2003) and unstructured low rank approximation with Frobenius norm are equivalent optimization problems. The principal component analysis problem, however, is motivated in a statistical setting and for this reason may be considered as a different problem. In fact, principal component analysis provides another (statistical) interpretation of the low rank approximation problem.

The conic section fitting problem has extensive literature, see Chap. 6 and the tutorial paper (Zhang 1997). The kernel principal component analysis method is developed in the machine learning and statistics literature (Schölkopf et al. 1999). Despite of the close relation between kernel principal component analysis and conic section fitting, the corresponding literature are disjoint.

Closely related to the estimation of the fundamental matrix problem in two-view computer vision is the shape from motion problem (Tomasi and Kanade 1993; Ma et al. 2004).

Matrix factorization techniques have been used in the analysis of microarray data in Alter and Golub (2006) and Kim and Park (2007). Alter and Golub (2006) propose a principal component projection to visualize high dimensional gene expression data and show that some known biological aspects of the data are visible in a two dimensional subspace defined by the first two principal components.

Distance Problems

The low rank approximation problem aims at finding the “smallest” correction of a given matrix that makes the corrected matrix rank deficient. This is a special case of

a distance problems: find the “nearest” matrix with a specified property to a given matrix. For an overview of distance problems, see Higham (1989). In Byers (1988), an algorithm for computing the distance of a stable matrix (Hurwitz matrix in the case of continuous-time and Schur matrix in the case of discrete-time linear time-invariant dynamical system) to the set of unstable matrices is presented. Stability radius for structured perturbations and its relation to the algebraic Riccati equation is presented in Hinrichsen and Pritchard (1986).

Structured Linear Algebra

Related to the topic of distance problems is the grand idea that the whole linear algebra (solution of systems of equations, matrix factorization, etc.) can be generalized to uncertain data. The uncertainty is described as structured perturbation on the data and a solution of the problem is obtained by correcting the data with a correction of the smallest size that renders the problem solvable for the corrected data. Two of the first references on the topic of structured linear algebra are El Ghaoui and Lebret (1997), Chandrasekaran et al. (1998).

Structured Pseudospectra

Let $\lambda(A)$ be the set of eigenvalues of $A \in \mathbb{C}^{n \times n}$ and \mathcal{M} be a set of structured matrices

$$\mathcal{M} := \{ \mathcal{S}(p) \mid p \in \mathbb{R}^{np} \},$$

with a given structure specification \mathcal{S} . The ε -structured pseudospectrum (Graillat 2006; Trefethen and Embree 1999) of A is defined as the set

$$\lambda_\varepsilon(A) := \{ z \in \mathbb{C} \mid z \in \lambda(\widehat{A}), \widehat{A} \in \mathcal{M}, \text{ and } \|A - \widehat{A}\|_2 \leq \varepsilon \}.$$

Using the structured pseudospectra, one can determine the structured distance of A to singularity as the minimum of the following optimization problem:

$$\text{minimize over } \widehat{A} \quad \|A - \widehat{A}\|_2 \quad \text{subject to } \widehat{A} \text{ is singular and } \widehat{A} \in \mathcal{M}.$$

This is a special structured low rank approximation problem for squared data matrix and rank reduction by one. Related to structured pseudospectra is the structured condition number problem for a system of linear equations, see Rump (2003).

Statistical Properties

Related to low rank approximation are the *orthogonal regression* (Gander et al. 1994), *errors-in-variables* (Gleser 1981), and *measurement errors* methods in the

statistical literature (Carroll et al. 1995; Cheng and Van Ness 1999). Classic papers on the univariate errors-in-variables problem are Adcock (1877, 1878), Pearson (1901), Koopmans (1937), Madansky (1959), York (1966). Closely related to the errors-in-variables framework for low rank approximation is the probabilistic principal component analysis framework of Tipping and Bishop (1999).

Reproducible Research

An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.

Buckheit and Donoho (1995)

The reproducible research concept is at the core of all sciences. In applied fields such as data modeling, however, algorithms' implementation, availability of data, and reproducibility of the results obtained by the algorithms on data are often neglected. This leads to a situation, described in Buckheit and Donoho (1995) as a scandal. See also Kovacevic (2007).

A quick and easy way of making computational results obtained in MATLAB reproducible is to use the function `publish`. Better still, the code and the obtained results can be presented in a literate programming style.

Literate Programming

The creation of the literate programming is a byproduct of the \TeX project, see Knuth (1992, 1984). The original system, called `web` is used for documentation of the \TeX program (Knuth 1986) and is for the Pascal language. Later a version `cweb` for the C language was developed. The `web` and `cweb` systems are followed by many other systems for literate programming that target specific languages. Unfortunately this leads to numerous literate programming dialects.

The `noweb` system for literate programming, created by N. Ramsey in the mid 90's, is not bound to any specific programming language and text processing system. A tutorial introduction is given in Ramsey (1994). The `noweb` syntax is also adopted in the `babel` part of Emacs `org-mode` (Dominik 2010)—a package for keeping structured notes that includes support for organization and automatic evaluation of computer code.

References

- Adcock R (1877) Note on the method of least squares. *The Analyst* 4:183–184
 Adcock R (1878) A problem in least squares. *The Analyst* 5:53–54

- Alter O, Golub GH (2006) Singular value decomposition of genome-scale mRNA lengths distribution reveals asymmetry in RNA gel electrophoresis band broadening. *Proc Natl Acad Sci* 103:11828–11833
- Bishop C (2006) *Pattern recognition and machine learning*. Springer, Berlin
- Botting B (2004) *Structured total least squares for approximate polynomial operations*. Master's thesis, School of Computer Science, University of Waterloo
- Buckheit J, Donoho D (1995) *Wavelab and reproducible research*. In: *Wavelets and statistics*. Springer, Berlin/New York
- Byers R (1988) A bisection method for measuring the distance of a stable matrix to the unstable matrices. *SIAM J Sci Stat Comput* 9(5):875–881
- Carroll R, Ruppert D, Stefanski L (1995) *Measurement error in nonlinear models*. Chapman & Hall/CRC, London
- Chandrasekaran S, Golub G, Gu M, Sayed A (1998) Parameter estimation in the presence of bounded data uncertainties. *SIAM J Matrix Anal Appl* 19:235–252
- Cheng C, Van Ness JW (1999) *Statistical regression with measurement error*. Arnold, London
- Ding C, He X (2004) K-means clustering via principal component analysis. In: *Proc int conf machine learning*, pp 225–232
- Dominik C (2010) *The org mode 7 reference manual*. Network theory ltd, URL <http://orgmode.org/>
- Eckart G, Young G (1936) The approximation of one matrix by another of lower rank. *Psychometrika* 1:211–218
- El Ghaoui L, Lebret H (1997) Robust solutions to least-squares problems with uncertain data. *SIAM J Matrix Anal Appl* 18:1035–1064
- Fierro R, Jiang E (2005) Lanczos and the Riemannian SVD in information retrieval applications. *Numer Linear Algebra Appl* 12:355–372
- Gander W, Golub G, Strelb R (1994) Fitting of circles and ellipses: least squares solution. *BIT* 34:558–578
- Gleser L (1981) Estimation in a multivariate “errors in variables” regression model: large sample results. *Ann Stat* 9(1):24–44
- Graillat S (2006) A note on structured pseudospectra. *J Comput Appl Math* 191:68–76
- Halmos P (1985) *I want to be a mathematician: an automathography*. Springer, Berlin
- Higham N (1989) Matrix nearness problems and applications. In: Gover M, Barnett S (eds) *Applications of matrix theory*. Oxford University Press, Oxford, pp 1–27
- Hinrichsen D, Pritchard AJ (1986) Stability radius for structured perturbations and the algebraic Riccati equation. *Control Lett* 8:105–113
- Jackson J (2003) *A user's guide to principal components*. Wiley, New York
- Jolliffe I (2002) *Principal component analysis*. Springer, Berlin
- Karmarkar N, Lakshman Y (1998) On approximate GCDs of univariate polynomials. *J Symb Comput* 26:653–666
- Kiers H (2002) Setting up alternating least squares and iterative majorization algorithms for solving various matrix optimization problems. *Comput Stat Data Anal* 41:157–170
- Kim H, Park H (2007) Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics* 23:1495–1502
- Knuth D (1984) Literate programming. *Comput J* 27(2):97–111
- Knuth D (1986) *Computers & typesetting, Volume B: TeX: The program*. Addison-Wesley, Reading
- Knuth D (1992) *Literate programming*. Cambridge University Press, Cambridge
- Koopmans T (1937) *Linear regression analysis of economic time series*. DeErven F Bohn
- Kovacevic J (2007) How to encourage and publish reproducible research. In: *Proc IEEE int conf acoustics, speech signal proc*, pp 1273–1276
- Krim H, Viberg M (1996) Two decades of array signal processing research. *IEEE Signal Process Mag* 13:67–94
- Kumaresan R, Tufts D (1983) Estimating the angles of arrival of multiple plane waves. *IEEE Trans Aerosp Electron Syst* 19(1):134–139

- Ma Y, Soatto S, Kosecká J, Sastry S (2004) An invitation to 3-D vision. Interdisciplinary applied mathematics, vol 26. Springer, Berlin
- Madansky A (1959) The fitting of straight lines when both variables are subject to error. *J Am Stat Assoc* 54:173–205
- Pearson K (1901) On lines and planes of closest fit to points in space. *Philos Mag* 2:559–572
- Polderman J, Willems JC (1998) Introduction to mathematical systems theory. Springer, New York
- Ramsey N (1994) Literate programming simplified. *IEEE Softw* 11:97–105
- Rump S (2003) Structured perturbations, Part I: Normwise distances. *SIAM J Matrix Anal Appl* 25:1–30
- Schölkopf B, Smola A, Müller K (1999) Kernel principal component analysis. MIT Press, Cambridge, pp 327–352
- Shawe-Taylor J, Cristianini N (2004) Kernel methods for pattern analysis. Cambridge University Press, Cambridge
- Stewart GW (1993) On the early history of the singular value decomposition. *SIAM Rev* 35(4):551–566
- Tipping M, Bishop C (1999) Probabilistic principal component analysis. *J R Stat Soc B* 61(3):611–622
- Tomasi C, Kanade T (1993) Shape and motion from image streamers: a factorization method. *Proc Natl Acad Sci USA* 90:9795–9802
- Trefethen LN, Embree M (1999) Spectra and pseudospectra: the behavior of nonnormal matrices and operators. Princeton University Press, Princeton
- Vichia M, Saporta G (2009) Clustering and disjoint principal component analysis. *Comput Stat Data Anal* 53:3194–3208
- Wentzell P, Andrews D, Hamilton D, Faber K, Kowalski B (1997) Maximum likelihood principal component analysis. *J Chemom* 11:339–366
- Willems JC (1986a) From time series to linear system—Part I. Finite dimensional linear time invariant systems. *Automatica* 22:561–580
- Willems JC (1986b) From time series to linear system—Part II. Exact modelling. *Automatica* 22:675–694
- Willems JC (1987) From time series to linear system—Part III. Approximate modelling. *Automatica* 23:87–115
- Willems JC (1989) Models for dynamics. *Dyn Rep* 2:171–269
- Willems JC (1991) Paradigms and puzzles in the theory of dynamical systems. *IEEE Trans Autom Control* 36(3):259–294
- Willems JC (2007) The behavioral approach to open and interconnected systems: modeling by tearing, zooming, and linking. *IEEE Control Syst Mag* 27:46–99
- York D (1966) Least squares fitting of a straight line. *Can J Phys* 44:1079–1086
- Zhang Z (1997) Parameter estimation techniques: a tutorial with application to conic fitting. *Image Vis Comput* 15(1):59–76

Chapter 2

From Data to Models

... whenever we have two different representations of the same thing we can learn a great deal by comparing representations and translating descriptions from one representation into the other. Shifting descriptions back and forth between representations can often lead to insights that are not inherent in either of the representations alone.

Abelson and diSessa (1986, p. 105)

2.1 Linear Static Model Representations

A linear static model with q variables is a subspace of \mathbb{R}^q . We denote the set of linear static models with q variables by \mathcal{L}_0^q . Three basic representations of a linear static model $\mathcal{B} \subseteq \mathbb{R}^q$ are the kernel, image, and input/output ones:

- kernel representation

$$\mathcal{B} = \ker(R) := \{d \in \mathbb{R}^q \mid Rd = 0\}, \tag{KER}_0$$

with parameter $R \in \mathbb{R}^{p \times q}$,

- image representation

$$\mathcal{B} = \text{image}(P) := \{d = P\ell \in \mathbb{R}^q \mid \ell \in \mathbb{R}^m\}, \tag{IMAGE}_0$$

with parameter $P \in \mathbb{R}^{q \times m}$, and

- input/output representation

$$\mathcal{B}_{i/o}(X, \Pi) := \{d = \Pi \text{col}(u, y) \in \mathbb{R}^q \mid u \in \mathbb{R}^m, y = X^T u\}, \tag{I/O}_0$$

with parameters $X \in \mathbb{R}^{m \times p}$ and a $q \times q$ permutation matrix Π .

If the parameter Π in an input/output representation is not specified, then by default it is assumed to be the identity matrix $\Pi = I_q$, i.e., the first m variables are assumed inputs and the other $p := q - m$ variables are outputs.

In the representation (**IMAGE**₀), the columns of P are *generators* of the model \mathcal{B} , i.e., they span or generate the model. In the representation (**KER**₀), the rows of R are *annihilators* of \mathcal{B} , i.e., they are orthogonal to the elements of \mathcal{B} . The parameters R and P are not unique, because

1. linearly dependent generators and annihilators can be added, respectively, to an existing set of annihilators and generators of \mathcal{B} keeping the model the same, and
2. a set of generators or annihilators can be changed by an invertible transformation, without changing the model, i.e.,

$$\ker(R) = \ker(UR), \quad \text{for all } U \in \mathbb{R}^{p \times p}, \quad \text{such that } \det(U) \neq 0;$$

and

$$\text{image}(P) = \text{image}(PV), \quad \text{for all } V \in \mathbb{R}^{m \times m}, \quad \text{such that } \det(V) \neq 0.$$

The smallest possible number of generators, i.e., $\text{col dim}(P)$, such that (**IMAGE**₀) holds is invariant of the representation and is equal to $m := \dim(\mathcal{B})$ —the dimension of \mathcal{B} . Integers, such as m and $p := q - m$ that are invariant of the representation, characterize properties of the model (rather than of the representation) and are called *model invariants*. The integers m and p have data modeling interpretation as number of inputs and number of outputs, respectively. Indeed, m variables are free (unassigned by the model) and the other p variables are determined by the model and the inputs. The number of inputs and outputs of the model \mathcal{B} are denoted by $m(\mathcal{B})$ and $p(\mathcal{B})$, respectively.

The model class of linear static models with q variables, at most m of which are inputs is denoted by $\mathcal{L}_{m,0}^q$. With $\text{col dim}(P) = m$, the columns of P form a basis for \mathcal{B} . The smallest possible $\text{row dim}(R)$, such that $\ker(R) = \mathcal{B}$ is invariant of the representation and is equal to the number of outputs of \mathcal{B} . With $\text{row dim}(R) = p$, the rows of R form a basis for the orthogonal complement \mathcal{B}^\perp of \mathcal{B} . Therefore, without loss of generality we can assume that $P \in \mathbb{R}^{q \times m}$ and $R \in \mathbb{R}^{p \times q}$.

Exercise 2.1 Show that for $\mathcal{B} \in \mathcal{L}_0^q$,

- $\min_{P, \text{image}(P)=\mathcal{B}} \text{col dim}(P)$ is equal to $\dim(\mathcal{B})$ and
- $\min_{R, \ker(R)=\mathcal{B}} \text{row dim}(R)$ is equal to $q - \dim(\mathcal{B})$.

In general, many input/output partitions of the variables d are possible. Choosing an input/output partition amounts to choosing a full rank $p \times p$ submatrix of R or a full rank $m \times m$ submatrix of P . In some data modeling problems, there is no a priori reason to prefer one partition of the variables over another. For such problems, the classical setting posing the problem as an overdetermined system of linear equations $AX \approx B$ is not a natural starting point.

Transition Among Input/Output, Kernel, and Image Representations

The transition from one model representation to another gives insight into the properties of the model. These *analysis problems* need to be solved before the more complicated modeling problems are considered. The latter can be viewed as *synthesis problems* since the model is created or synthesized from data and prior knowledge.

If the parameters R , P , and (X, Π) describe the same system \mathcal{B} , then they are related. We show the relations that the parameters must satisfy as well as code that does the transition from one representation to another. Before we describe the transition among the parameters, however, we need an efficient way to store and multiply by permutation matrices and a tolerance for computing rank numerically.

Input/Output Partition of the Variables

In the input/output model representation $\mathcal{B}_{i/o}(X, \Pi)$, the partitioning of the variables $d \in \mathbb{R}^q$ into inputs $u \in \mathbb{R}^m$ and outputs $y \in \mathbb{R}^p$ is specified by a permutation matrix Π ,

$$d \begin{array}{c} \xrightarrow{\Pi^{-1}=\Pi^T} \\ \xleftarrow{\Pi} \end{array} \text{col}(u, y), \quad d = \Pi \begin{bmatrix} u \\ y \end{bmatrix}, \quad \begin{bmatrix} u \\ y \end{bmatrix} = \Pi^T d.$$

In the software implementation, however, it is more convenient (as well as more memory and computation efficient) to specify the partitioning by a vector

$$\pi := \Pi \text{col}(1, \dots, q) \in \{1, \dots, q\}^q.$$

Clearly, the vector π contains the same information as the matrix Π and one can reconstruct Π from π by permuting the rows of the identity matrix. (In the code `io` is the variable corresponding to the vector π and `Pi` is the variable corresponding to the matrix Π .)

$$37a \quad \langle \pi \mapsto \Pi \ 37a \rangle \equiv \quad (38a)$$

```
Pi = eye(length(io)); Pi = Pi(io, :);
```

Permuting the elements of a vector is done more efficiently by direct reordering of the elements, instead of a matrix-vector multiplication. If d is a variable corresponding to a vector d , then `d(io)` corresponds to the vector Πd .

The default value for Π is the identity matrix I , corresponding to first m variables of d being inputs and the remaining p variables outputs, $d = \begin{bmatrix} u \\ y \end{bmatrix}$.

$$37b \quad \langle \text{default input/output partition } 37b \rangle \equiv \quad (39 \ 40a \ 42a)$$

```
if ~exist('io') || isempty(io), io = 1:q; end
```

In case the inverse permutation $\Pi^{-1} = \Pi^T$ is needed, the corresponding “permutation vector” is

$$\pi' = \Pi^T \text{col}(1, \dots, q) \in \{1, \dots, q\}^q.$$

In the code `inv_io` is the variable corresponding to the vector π' and the transition from the original variables `d` to the partitioned variables $uy = [u; y]$ via `io` and `inv_io` is done by the following indexing operations:

$$d \begin{array}{c} \xrightarrow{\text{io_inv}} \\ \xleftarrow{\text{io}} \end{array} uy, \quad d = uy(\text{io}), \quad uy = d(\text{io_inv}).$$

38a $\langle \text{inverse permutation 38a} \rangle \equiv$ (42b)
 $\langle \pi \mapsto \Pi \text{ 37a} \rangle, \quad \text{inv_io} = (1:\text{length}(\text{io})) * \text{Pi};$

Tolerance for Rank Computation

In the computation of an input/output representation from a given kernel or image representation of a model (as well as in the computation of the models' complexity), we need to find the rank of a matrix. Numerically this is an ill-posed problem because arbitrary small perturbations of the matrix's elements may (generically will) change the rank. A solution to this problem is to replace rank with "numerical rank" defined as follows

$$\text{num rank}(A, \varepsilon) := \text{number of singular values of } A \text{ greater than } \varepsilon, \quad (\text{num rank})$$

where $\varepsilon \in \mathbb{R}_+$ is a user defined tolerance. Note that

$$\text{num rank}(A, 0) = \text{rank}(A),$$

i.e., by taking the tolerance to be equal to zero, the numerical rank reduces to the theoretical rank. A nonzero tolerance ε makes the numerical rank robust to perturbations of size (measured in the induced 2-norm) less than ε . Therefore, ε reflects the size of the expected errors in the matrix. The default tolerance is set to a small value, which corresponds to numerical errors due to a double precision arithmetic. (In the code `tol` is the variable corresponding to ε .)

38b $\langle \text{default tolerance tol 38b} \rangle \equiv$ (41–43 76b)
`if ~exist('tol') || isempty(tol), tol = 1e-12; end`

Note 2.2 The numerical rank definition ([num rank](#)) is the solution of an unstructured rank minimization problem: find a matrix \hat{A} of minimal rank, such that $\|A - \hat{A}\|_2 < \varepsilon$.

From Input/Output Representation to Kernel or Image Representations

Consider a linear static model $\mathcal{B} \in \mathcal{L}_{m,0}^q$, defined by an input/output representation $\mathcal{B}_{\text{io}}(X, \Pi)$. From $y = X^\top u$, we have

$$[X^\top \ -I] \text{col}(u, y) = 0$$

or since $d = \Pi \begin{bmatrix} u \\ y \end{bmatrix}$,

$$\underbrace{[X^\top - I] \Pi^\top}_R \underbrace{\Pi \begin{bmatrix} u \\ y \end{bmatrix}}_d = 0.$$

Therefore, the matrix

$$R = [X^\top - I] \Pi^\top \quad ((X, \Pi) \mapsto R)$$

is a parameter of a kernel representations of \mathcal{B} , i.e.,

$$\mathcal{B}_{i/o}(X, \Pi) = \ker([X^\top - I] \Pi^\top) = \mathcal{B}.$$

Moreover, the representation is minimal because R is full row rank.

Similarly, a minimal image representation is derived from the input/output representation as follows. From $y = X^\top u$,

$$\begin{bmatrix} u \\ y \end{bmatrix} = \begin{bmatrix} I \\ X^\top \end{bmatrix} u,$$

so that, using $d = \Pi \begin{bmatrix} u \\ y \end{bmatrix}$,

$$d = \Pi \begin{bmatrix} u \\ y \end{bmatrix} = \Pi \begin{bmatrix} I \\ X^\top \end{bmatrix} u =: Pu.$$

Therefore, the matrix

$$P = \Pi \begin{bmatrix} I \\ X^\top \end{bmatrix} \quad ((X, \Pi) \mapsto P)$$

is a parameter of an image representations of \mathcal{B} , i.e.,

$$\mathcal{B}_{i/o}(X, \Pi) = \text{image} \left(\Pi \begin{bmatrix} I \\ X^\top \end{bmatrix} \right) = \mathcal{B}.$$

The representation is minimal because P is full column rank.

Formulas $((X, \Pi) \mapsto R)$ and $((X, \Pi) \mapsto P)$ give us a straight forward way of transforming a given input/output representation to minimal kernel and minimal image representations.

39 $((X, \Pi) \mapsto R)$ 39) \equiv
 function r = xio2r(x, io)
 r = [x', -eye(size(x, 2))]; q = size(r, 2);
 (default input/output partition 37b), r = r(:, io);

Defines:

xio2r, used in chunk 45a.

40a $\langle (X, \Pi) \mapsto P \text{ 40a} \rangle \equiv$

```
function p = xio2p(x, io)
    p = [eye(size(x, 1)); x']; q = size(p, 1);
    (default input/output partition 37b), p = p(io, :);
```

Defines:

xio2p, used in chunk 45a.

From Image to Minimal Kernel and from Kernel to Minimal Image Representation

The relation

$$\ker(R) = \text{image}(P) = \mathcal{B} \in \mathcal{L}_{m,0}^q \implies RP = 0 \quad (R \leftrightarrow P)$$

gives a link between the parameters P and R . In particular, a minimal image representation $\text{image}(P) = \mathcal{B}$ can be obtained from a given kernel representation $\ker(R) = \mathcal{B}$ by computing a basis for the null space of R . Conversely, a minimal kernel representation $\ker(R) = \mathcal{B}$ can be obtained from a given image representation $\text{image}(P) = \mathcal{B}$ by computing a basis for the left null space of P .

40b $\langle R \mapsto P \text{ 40b} \rangle \equiv$

```
function p = r2p(r), p = null(r);
```

Defines:

r2p, used in chunks 44 and 45a.

40c $\langle P \mapsto R \text{ 40c} \rangle \equiv$

```
function r = p2r(p), r = null(p)';
```

Defines:

p2r, used in chunk 45a.

Converting an Image or Kernel Representation to a Minimal One

The kernel and image representations obtained from `xio2r`, `xio2p`, `p2r`, and `r2p` are minimal. In general, however, a given kernel or image representations can be non-minimal, i.e., R may have redundant rows and P may have redundant columns. The kernel representation, defined by R , is minimal if and only if R is full row rank. Similarly, the image representation, defined by P , is minimal if and only if P is full column rank.

The problems of converting kernel and image representations to minimal ones are equivalent to the problem of finding a full rank matrix that has the same kernel or image as a given matrix. A numerically reliable way to solve this problem is to use the singular value decomposition.

Consider a model $\mathcal{B} \in \mathcal{L}_{m,0}^q$ with parameters $R \in \mathbb{R}^{g \times q}$ and $P \in \mathbb{R}^{q \times g}$ of, respectively, kernel and image representations. Let

$$R = U \Sigma V^\top$$

be the singular value decomposition of R and let p be the rank of R . With the partitioning,

$$V =: [V_1 \ V_2], \quad \text{where } V_1 \in \mathbb{R}^{q \times p},$$

we have

$$\text{image}(R^\top) = \text{image}(V_1).$$

Therefore,

$$\ker(R) = \ker(V_1^\top) \quad \text{and} \quad V_1 \text{ is full rank,}$$

so that V_1^\top is a parameter of a minimal kernel representation of \mathcal{B} .

Similarly, let

$$P = U \Sigma V^\top$$

be the singular value decomposition of P . With the partitioning,

$$U =: [U_1 \ U_2], \quad \text{where } U_1 \in \mathbb{R}^{q \times m},$$

we have

$$\text{image}(P) = \text{image}(U_1).$$

Since U_1 is full rank, it is a parameter of a minimal image representation of \mathcal{B} .

In the numerical implementation, the rank is replaced by the numerical rank with respect to a user defined tolerance `tol`.

```
41 <R ↦ minimal R 41> ≡
    function r = minr(r, tol)
        [p, q] = size(r); <default tolerance tol 38b>
        [u, s, v] = svd(r, 'econ'); pmin = sum(diag(s) > tol);
        if pmin < p, r = v(:, 1:pmin)'; end
```

Defines:

`minr`, used in chunks 42b and 43c.

Exercise 2.3 Write a function `minp` that implements the transition $P \mapsto \text{minimal } P$.

From Kernel or Image to Input/Output Representation

The transformations from kernel to input/output and from image to input/output representations are closely related. They involve as a sub-problem the problem of finding input/output partitions of the variables in the model. Because of this, they are more complicated than the inverse transformations, considered above.

Assume first that the input/output partition is given. This amounts to knowing the permutation matrix $\Pi \in \mathbb{R}^{q \times q}$ in $(I/O)_0$.

42a $\langle\langle R, \Pi \rangle \mapsto X \text{ 42a} \rangle \equiv$ 42b
`function x = rio2x(r, io, tol)`
`q = size(r, 2); <default input/output partition 37b>, <default tolerance tol 38b>`

Defines:

`rio2x`, used in chunks 43c and 45.

Consider given parameters $R \in \mathbb{R}^{p \times q}$ of *minimal* kernel representation of a linear static system $\mathcal{B} \in \mathcal{L}_{m,0}^q$ and define the partitioning

$$R\Pi =: [R_u \ R_y], \quad \text{where } R_y \in \mathbb{R}^{p \times p}.$$

42b $\langle\langle R, \Pi \rangle \mapsto X \text{ 42a} \rangle + \equiv$ <42a 42c>
`r = minr(r, tol); p = size(r, 1); m = q - p;`
`<inverse permutation 38a>, rpi = r(:, inv_io);`
`ru = rpi(:, 1:m); ry = rpi(:, (m + 1):end);`

Uses `minr` 41.

Similarly, for a parameter $P \in \mathbb{R}^{q \times m}$ of *minimal* image representation of \mathcal{B} , define

$$\Pi^\top P =: \begin{bmatrix} P_u \\ P_y \end{bmatrix}, \quad \text{where } P_u \in \mathbb{R}^{m \times m}.$$

If R_y and P_u are non-singular, it follows from $\langle\langle X, \Pi \rangle \mapsto R \rangle$ and $\langle\langle X, \Pi \rangle \mapsto P \rangle$ that

$$X = -(R_y^{-1} R_u)^\top \quad \langle\langle R, \Pi \rangle \mapsto X \rangle$$

and

$$X = (P_y P_u^{-1})^\top \quad \langle\langle P, \Pi \rangle \mapsto X \rangle$$

is the parameter of the input/output representation $\mathcal{B}_{i/o}(X, \Pi)$ of \mathcal{B} , i.e.,

$$\ker \underbrace{\begin{pmatrix} m & p \\ [R_u & R_y] \Pi^\top \end{pmatrix}}_R = \mathcal{B}_{i/o} \left(-(R_y^{-1} R_u)^\top, \Pi \right)$$

and

$$\text{image} \left(\underbrace{\Pi \begin{pmatrix} P_u \\ P_y \end{pmatrix}}_P \right) = \mathcal{B}_{i/o} \left((P_y P_u^{-1})^\top, \Pi \right).$$

42c $\langle\langle R, \Pi \rangle \mapsto X \text{ 42a} \rangle + \equiv$ <42b
`[u, s, v] = svd(ry); s = diag(s);`
`if s(end) < tol`
`warning('Computation of X is ill conditioned.');`
`x = NaN;`
`else`
`x = -(v * diag(1 ./ s) * u' * ru)';`
`end`

Singularity of the blocks R_y and P_u implies that the input/output representation with a permutation matrix Π is not possible. In such cases, the function `rio2x` issues a warning message and returns NaN value for X .

The function `r2io` uses `rio2x` in order to find all possible input/output partitions for a model specified by a kernel representation.

```
43a <R ↦ Π 43a>≡ 43b>
    function IO = r2io(r, tol)
        q = size(r, 2); <default tolerance tol 38b>
```

Defines:

`r2io`, used in chunk 43d.

The search is exhaustive over all input/output partitionings of the variables (i.e., all choices of m elements of the set of variable indices $\{1, \dots, q\}$), so that the computation is feasible only for a small number of variables (say less than 6).

```
43b <R ↦ Π 43a>+≡ <43a 43c>
    IO = perms(1:q); nio = size(IO, 1);
```

The parameter X for each candidate partition is computed. If the computation of X is ill conditioned, the corresponding partition is not consistent with the model and is discarded.

```
43c <R ↦ Π 43a>+≡ <43b
    not_possible = []; warning_state = warning('off');
    r = minr(r, tol);
    for i = 1:nio
        x = rio2x(r, IO(i, :), tol);
        if isnan(x), not_possible = [not_possible, i]; end
    end
    warning(warning_state); IO(not_possible, :) = [];
```

Uses `minr` 41 and `rio2x` 42a.

Example 2.4 Consider the linear static model with three variables and one input

$$\mathcal{B} = \text{image} \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right) = \ker \left(\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right).$$

Clearly, this model has only two input/output partitions:

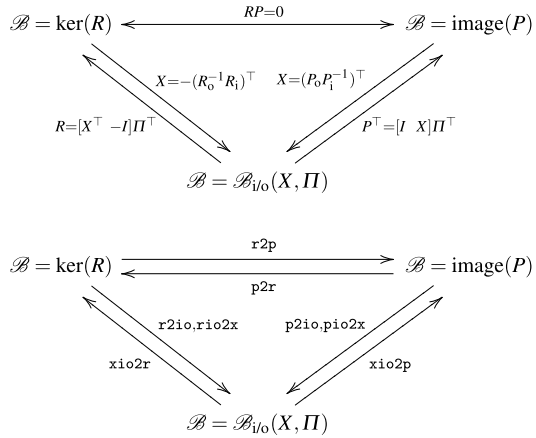
$$u = w_1, y = \begin{bmatrix} w_2 \\ w_3 \end{bmatrix} \quad \text{and} \quad u = w_1, y = \begin{bmatrix} w_3 \\ w_2 \end{bmatrix}.$$

Indeed, the function `r2io`

```
43d <Test r2io 43d>≡
    r2io([0 0 1; 0 1 0])
```

Uses `r2io` 43a.

Fig. 2.1 Relations and functions for the transitions among linear static model parameters



correctly computes the input output partitionings from the parameter R in a kernel representation of the model

ans =

```

1     2     3
1     3     2

```

Exercise 2.5 Write functions `pio2x` and `p2io` that implement the transitions

$$(P, \Pi) \mapsto X \quad \text{and} \quad P \mapsto \Pi.$$

Exercise 2.6 Explain how to check that two models, specified by kernel or image representation, are equivalent.

Summary of Transitions among Representations

Figure 2.1 summarizes the links among the parameters R , P , and (X, Π) of a linear static model \mathcal{B} and the functions that implement the transitions.

Numerical Example

In order to test the functions for transition among the kernel, image, and input/output model representations, we choose, a random linear static model, specified by a kernel representation,

```

44 <Test model transitions 44>≡ 45a>
    m = 2; p = 3; R = rand(p, m + p); P = r2p(R);
    Uses r2p 40b.

```

and traverse the diagram in Fig. 2.1 clock-wise and anti clock-wise

45a `(Test model transitions 44) +≡` ◁44 45b▷

```
R_ = xio2r(pio2x(r2p(R))); P_ = xio2p(rio2x(p2r(P)));
```

Uses p2r 40c, r2p 40b, rio2x 42a, xio2p 40a, and xio2r 39.

As a verification of the software, we check that after traversing the loop, equivalent models are obtained.

45b `(Test model transitions 44) +≡` ◁45a

```
norm(rio2x(R) - rio2x(R_)), norm(pio2x(P) - pio2x(P_))
```

Uses rio2x 42a.

The answers are around the machine precision, which confirms that the models are the same.

Linear Static Model Complexity

A linear static model is a finite dimensional subspace. The dimension m of the subspace is equal to the number of inputs and is invariant of the model representation. The integer constant m quantifies the *model complexity*: the model is more complex when it has more inputs. The rationale for this definition of model complexity is that inputs are “unexplained” variables by the model, so the more inputs the model has, the less it “explains” the modeled phenomenon. In data modeling the aim is to obtain low-complexity models, a principle generally referred to as *Occam’s razor*.

Note 2.7 (Computing the model complexity is a rank estimation problem) Computing the model complexity, implicitly specified by (exact) data, or by a non-minimal kernel or image representation is a rank computation problem; see Sect. 2.3 and the function `minr`.

2.2 Linear Time-Invariant Model Representations

An observation d_j of a static model is a vector of variables. In the dynamic case, the observations depend on time, so that apart from the multivariable aspect, there is also a time evaluation aspect. In the dynamic case, an observation is referred to as a *trajectory*, i.e., it is a vector valued *function* of a scalar argument. The time variable takes its values in the set of integers \mathbb{Z} (*discrete-time* model) or in the set of real numbers \mathbb{R} (*continuous-time* model). We denote the time axis by \mathcal{T} .

A dynamic model \mathcal{B} with q variables is a subset of the trajectory space $(\mathbb{R}^q)^{\mathcal{T}}$ —the set of all functions from the time axis \mathcal{T} to the variable space \mathbb{R}^q .

In this book, we consider the special class of finite dimensional linear time-invariant dynamical models. By definition, a model \mathcal{B} is *linear* if it is a subspace of the data space $(\mathbb{R}^q)^{\mathcal{T}}$. In order to define the *time-invariance* property, we introduce the *shift operator* σ^τ . Acting on a signal w , σ^τ produces a signal $\sigma^\tau w$, which is the backwards shifted version of w by τ time units, i.e.,

$$(\sigma^\tau w)(t) := w(t + \tau), \quad \text{for all } t \in \mathcal{T}.$$

Acting on a set of trajectories, σ^τ shifts all trajectories in the set, i.e.,

$$\sigma^\tau \mathcal{B} := \{\sigma^\tau w \mid w \in \mathcal{B}\}.$$

A model \mathcal{B} is shift-invariant if it is invariant under any shift in time, i.e.,

$$\sigma^\tau \mathcal{B} = \mathcal{B}, \quad \text{for all } \tau \in \mathcal{T}.$$

The model \mathcal{B} is *finite dimensional* if it is a closed subset (in the topology of point-wise convergence). Finite dimensionality is equivalent to the property that at any time t the future behavior of the model is deterministically independent of the past behavior, given a finite dimensional vector, called a *state* of the model. Intuitively, the state is the information (or memory) of the past that is needed in order to predict the future. The smallest state dimension is an invariant of the system, called the *order*. We denote the set of finite dimensional linear time-invariant models with q variables and order at most n by $\mathcal{L}^{q,n}$ and the *order* of \mathcal{B} by $n(\mathcal{B})$.

A finite dimensional linear time-invariant model $\mathcal{B} \in \mathcal{L}^{q,n}$ admits a representation by a difference or differential equation

$$\begin{aligned} R_0 w + R_1 \lambda w + \cdots + R_1 \lambda^1 w &= \underbrace{(R_0 + R_1 \lambda + \cdots + R_1 \lambda^1)}_{R(\lambda)} w \\ &= R(\lambda) w = 0, \end{aligned} \tag{DE}$$

where λ is the unit shift operator σ in the discrete-time case and the differential operator $\frac{d}{dt}$ in the continuous-time case. Therefore, the model \mathcal{B} is the kernel

$$\mathcal{B} := \ker(R(\lambda)) = \{w \mid \text{(DE) holds}\}, \tag{KER}$$

of the difference or differential operator $R(\lambda)$. The smallest degree $\mathbb{1}$ of a polynomial matrix

$$R(z) := R_0 + R_1 z + \cdots + R_1 z^1 \in \mathbb{R}^{q \times q}[z],$$

in a kernel representation (KER) of \mathcal{B} , is invariant of the representation and is called the *lag* $\mathbb{1}(\mathcal{B})$ of \mathcal{B} .

The order of the system is the total degree of the polynomial matrix R in a kernel representation of the system. Therefore, we have the following link between the order and the lag of a linear time-invariant model:

$$n(\mathcal{B}) \leq p(\mathcal{B}) \mathbb{1}(\mathcal{B}).$$

As in the static case, the smallest possible number of rows g of the polynomial matrix R in a kernel representation (KER) of a finite dimensional linear time-invariant system \mathcal{B} is the invariant $\mathfrak{p}(\mathcal{B})$ —number of outputs of \mathcal{B} . Finding an input/output partitioning for a model specified by a kernel representation amounts to selection of a non-singular $\mathfrak{p} \times \mathfrak{p}$ submatrix of R . The resulting input/output representation is:

$$\mathcal{B} = \mathcal{B}_{i/o}(P, Q, \Pi) := \ker(\Pi [Q(\lambda) \ P(\lambda)]), \quad (\text{I/O})$$

with parameters the polynomial matrices

$$Q \in \mathbb{R}^{\mathfrak{p} \times \mathfrak{m}}[z] \quad \text{and} \quad P \in \mathbb{R}^{\mathfrak{p} \times \mathfrak{p}}[z], \quad \text{such that} \quad \det(P) \neq 0,$$

and the permutation matrix Π .

In general, the representation (I/O) involves higher order shifts or derivatives. A first order representation

$$\mathcal{B} = \mathcal{B}_{i/s/o}(A, B, C, D, \Pi) := \{w = \Pi \operatorname{col}(u, y) \mid \text{there is } x, \text{ such that} \\ \lambda x = Ax + Bu \text{ and } y = Cx + Du\}, \quad (\text{I/S/O})$$

with an auxiliary variable x , however, is always possible. The representation (I/S/O) displays not only the input/output structure of the model but also its state structure and is referred to as an input/state/output representation of the model. The parameters of an input/state/output representation are the matrices

$$A \in \mathbb{R}^{n \times n}, \quad B \in \mathbb{R}^{n \times m}, \quad C \in \mathbb{R}^{\mathfrak{p} \times n}, \quad D \in \mathbb{R}^{\mathfrak{p} \times m},$$

and a permutation matrix Π . The parameters are nonunique due to

- nonuniqueness in the choice of the input/output partition,
- existence of redundant states (non-minimality of the representation), and
- change of state space basis

$$\mathcal{B}_{i/s/o}(A, B, C, D) = \mathcal{B}_{i/s/o}(T^{-1}AT, T^{-1}B, CT, D), \\ \text{for any non-singular matrix } T \in \mathbb{R}^{n \times n}. \quad (\text{CB})$$

An input/state/output representation $\mathcal{B}_{i/s/o}(A, B, C, D)$ is called *minimal* when the state dimension n is as small as possible. The minimal state dimension is an invariant of the system and is equal to the order $n(\mathcal{B})$ of the system.

A system \mathcal{B} is *autonomous* if for any trajectory $w \in \mathcal{B}$ the “past”

$$w_- := (\dots, w(-2), w(-1))$$

of w completely determines its “future”

$$w_+ := (w(0), w(1), \dots).$$

Table 2.1 Summary of model $\mathcal{B} \subset (\mathbb{R}^q)^{\mathcal{T}}$ properties

Property	Definition
linearity	$w, v \in \mathcal{B} \implies \alpha w + \beta v \in \mathcal{B}$, for all $\alpha, \beta \in \mathbb{R}$
time-invariance	$\sigma^\tau \mathcal{B} = \mathcal{B}$, for all $\tau \in \mathcal{T}$
finite dimensionality	\mathcal{B} is a closed set; equivalently $n(\mathcal{B}) < \infty$
autonomy	the past of any trajectory completely determines its future; equivalently $m(\mathcal{B}) = 0$
controllability	the past of any trajectory can be concatenated to the future of any other trajectory by a third trajectory if a transition period is allowed

It can be shown that a system \mathcal{B} is autonomous if and only if it has no inputs. An autonomous finite dimensional linear time-invariant system is parametrized by the pair of matrices A and C via the state space representation

$$\mathcal{B}_{i/s/o}(A, C) := \{w = y \mid \text{there is } x, \text{ such that } \sigma x = Ax \text{ and } y = Cx\}.$$

The dimension $\dim(\mathcal{B})$ of an autonomous linear model \mathcal{B} is equal to the order $n(\mathcal{B})$.

In a way the opposite of an autonomous model is a controllable system. The model \mathcal{B} is *controllable* if for any trajectories w_p and w_f of \mathcal{B} , there is $\tau > 0$ and a third trajectory $w \in \mathcal{B}$, which coincides with w_p in the past, i.e., $w(t) = w_p(t)$, for all $t < 0$, and coincides with w_f in the future, i.e., $w(t) = w_f(t)$, for all $t \geq \tau$. The subset of controllable systems of the set of linear time-invariant systems \mathcal{L}^q is denoted by $\mathcal{L}_{\text{ctrb}}^q$. A summary of properties of a dynamical system is given in Table 2.1.

Apart from the kernel, input/output, and input/state/output representation, a controllable finite dimensional linear time-invariant model admits the following representations:

- *image representation*

$$\mathcal{B} = \text{image}(P(\lambda)) := \{w \mid w = P(\lambda)\ell, \text{ for some } \ell\}, \quad (\text{IMAGE})$$

with parameter the polynomial matrix $P(z) \in \mathbb{R}^{q \times g}[z]$,

- *convolution representation*

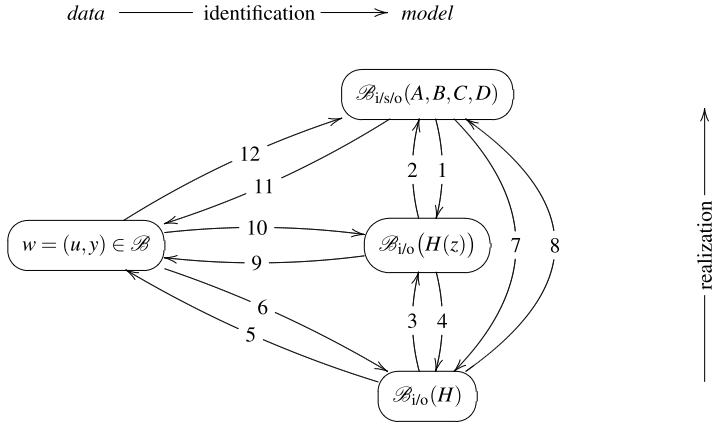
$$\mathcal{B} = \mathcal{B}_{i/o}(H, \Pi) := \{w = \Pi \text{col}(u, y) \mid y = H \star u\}, \quad (\text{CONV})$$

where \star is the convolution operator

$$y(t) = (H \star u)(t) := \sum_{\tau=0}^{\infty} H(\tau)u(t - \tau), \quad \text{in discrete-time, or}$$

$$y(t) = (H \star u)(t) := \int_0^{\infty} H(\tau)u(t - \tau) d\tau, \quad \text{in continuous-time,}$$

with parameters the signal $H : \mathcal{T} \rightarrow \mathbb{R}^{p \times m}$ and a permutation matrix Π ; and



- | | |
|--|---|
| <ol style="list-style-type: none"> 1. $H(z) = C(Iz - A)^{-1}B + D$ 2. realization of a transfer function 3. Z or Laplace transform of $H(t)$ 4. inverse transform of $H(z)$ 5. convolution $y_d = H \star u_d$ 6. exact identification | <ol style="list-style-type: none"> 7. $H(0) = D, H(t) = CA^{t-1}B$ (discrete-time),
$H(t) = Ce^{At}B$ (continuous-time), for $t > 0$ 8. realization of an impulse response 9. simulation with input u_d and $x(0) = 0$ 10. exact identification 11. simulation with input u_d and $x(0) = x_{ini}$ 12. exact identification |
|--|---|

Fig. 2.2 Data, input/output model representations, and links among them

- *transfer function,*

$$\mathcal{B} = \mathcal{B}_{io}(H, \Pi) := \{w = \Pi \text{col}(u, y) \mid \mathcal{F}(y) = H(z)\mathcal{F}(u)\}, \quad (\text{TF})$$

where \mathcal{F} is the Z-transform in discrete-time and the Laplace transform in continuous-time, with parameters the rational matrix $H \in \mathbb{R}^{p \times m}(s)$ (the transfer function) and a permutation matrix Π .

Transitions among the parameters $H, H(z)$, and (A, B, C, D) are classical problems, see Fig. 2.2. Next, we review the transition from impulse response H to parameters (A, B, C, D) of an input/state/output representation, which plays an important role in deriving methods for Hankel structured low rank approximation.

System Realization

The problem of passing from a convolution representation to an input/state/output representation is called (impulse response) realization.

Definition 2.8 (Realization) A linear time-invariant system \mathcal{B} with m inputs and p outputs and an input/output partition, specified by a permutation matrix Π , is a *realization* of (or realizes) an impulse response $H : \mathcal{T} \rightarrow \mathbb{R}^{p \times m}$ if \mathcal{B} has a convolution

representation $\mathcal{B} = \mathcal{B}_{i/o}(H, \Pi)$. A realization \mathcal{B} of H is *minimal* if its order $n(\mathcal{B})$ is the smallest over all realization of H .

In what follows, we fix the input/output partition Π to the default one

$$w = \text{col}(u, y)$$

and use the notation

$$H := [h_1 \cdots h_m] \quad \text{and} \quad I_m := [e_1 \cdots e_m].$$

An equivalent definition of impulse response realization that makes explicit the link of the realization problem to data modeling is the following one.

Definition 2.9 (Realization, as a data modeling problem) The system \mathcal{B} realizes H if the set of input/output trajectories (e_i, h_i) , for $i = 1, \dots, m$ are impulse responses of \mathcal{B} , i.e.,

$$(e_i \delta, 0 \wedge h_i) \in \mathcal{B}, \quad \text{for } i = 1, \dots, m,$$

where δ is the delta function and \wedge is the concatenation map (at time 0)

$$w = w_p \wedge w_f, \quad w(t) := \begin{cases} w_p(t), & \text{if } t < 0 \\ w_f(t), & \text{if } t \geq 0 \end{cases}$$

Note 2.10 (Discrete-time vs. continuous-time system realization) There are some differences between the discrete and continuous-time realization theory. Next, we consider the discrete-time case. It turns out, however, that the discrete-time algorithms can be used for realization of continuous-time systems by applying them on the sequence of the Markov parameter $(H(0), \frac{d}{dt}H(0), \dots)$ of the system.

The sequence

$$H = (H(0), H(1), H(2), \dots, H(t), \dots), \quad \text{where } H(t) \in \mathbb{R}^{p \times m}$$

is a one sided infinite matrix-values time series. Acting on H , the shift operator σ , removes the first sample, i.e.,

$$\sigma H = (H(1), H(2), \dots, H(t), \dots).$$

A sequence H might not be realizable by a *finite dimensional* linear time-invariant system, but if it is realizable, a minimal realization is unique.

Theorem 2.11 (Test for realizability) *The sequence $H : \mathbb{Z}_+ \rightarrow \mathbb{R}^{p \times m}$ is realizable by a finite dimensional linear time-invariant system with m inputs if and only if the two-sided infinite Hankel matrix $\mathcal{H}(\sigma H)$ has a finite rank n . Moreover, the order of a minimal realization is equal to n , and there is a unique system \mathcal{B} in $\mathcal{L}_m^{q,n}$ that realizes H .*

Proof (\implies) Let H be realizable by a system $\mathcal{B} \in \mathcal{L}_m^{q,n}$ with a minimal input/state/output representation $\mathcal{B} = \mathcal{B}_{i/s/o}(A, B, C, D)$. Then

$$H(0) = D \quad \text{and} \quad H(t) = CA^{t-1}B, \quad \text{for } t > 0.$$

The (i, j) block element of the Hankel matrix $\mathcal{H}(\sigma H)$ is

$$H(i + j - 1) = CA^{i+j-2}B = CA^{i-1}A^{j-1}B.$$

Let

$$\mathcal{O}_t(A, C) := \text{col}(C, CA, \dots, CA^{t-1}) \quad (\mathcal{O})$$

be the extended observability matrix of the pair (A, C) and

$$\mathcal{C}_t(A, B) := [B \ AB \ \dots \ A^{t-1}B] \quad (\mathcal{C})$$

be the extended controllability matrix of the pair (A, B) . With $\mathcal{O}(A, C)$ and $\mathcal{C}(A, B)$ being the infinite observability and controllability matrices, we have

$$\mathcal{H}(\sigma H) = \mathcal{O}(A, C)\mathcal{C}(A, B) \quad (\mathcal{OC})$$

Since the representation $\mathcal{B}_{i/s/o}(A, B, C, D)$ is assumed to be minimal, $\mathcal{C}(A, B)$ is full row rank and $\mathcal{O}(A, C)$ is full column rank. Therefore, (\mathcal{OC}) is a rank revealing factorization of $\mathcal{H}(\sigma H)$ and

$$\text{rank}(\mathcal{H}(\sigma H)) = n(\mathcal{B}).$$

(\impliedby) In this direction, the proof is constructive and results in an *algorithm* for computation of the minimal realization of H in $\mathcal{L}_m^{q,n}$, where $n = \text{rank}(\mathcal{H}(\sigma H))$. A realization algorithm is presented in Sect. 3.1. \square

Theorem 2.11 shows that

$$\text{rank}(\mathcal{H}_{i,j}(\sigma H)) = n(\mathcal{B}), \quad \text{for } pi \geq n(\mathcal{B}) \text{ and } mj \geq n(\mathcal{B}).$$

This suggests a method to find the order $n(\mathcal{B})$ of the minimal realization of H : compute the rank of the *finite* Hankel matrix $\mathcal{H}_{i,j}(\sigma H)$, where $n_{\max} := \min(pi, mj)$ is an upper bound of the order. Algorithms for computing the order and parameters of the minimal realization are presented in Chap. 3.1.

Linear Time-Invariant Model Complexity

Associate with a linear time-invariant dynamical system \mathcal{B} , we have defined the following system invariants:

$$\begin{aligned} m(\mathcal{B}) & \text{ number of inputs,} & n(\mathcal{B}) & \text{ order,} \\ p(\mathcal{B}) & \text{ number of outputs} & \text{and} & \ 1(\mathcal{B}) & \text{ lag.} \end{aligned}$$

The complexity of a linear static model \mathcal{B} is the number of inputs $m(\mathcal{B})$ of \mathcal{B} or, equivalently, the dimension $\dim(\mathcal{B})$ of \mathcal{B} . Except for the class of autonomous systems, however, the dimension of a dynamical model is infinite. We define the restriction $\mathcal{B}|_{[1, T]}$ of \mathcal{B} to the interval $[1, T]$,

$$\mathcal{B}|_{[1, T]} := \{w \in \mathbb{R}^T \mid \text{there exist } w_p \text{ and } w_f, \text{ such that } (w_p, w, w_f) \in \mathcal{B}\}. \quad (\mathcal{B}|_{[1, T]})$$

For a linear time-invariant model \mathcal{B} and for $T > n(\mathcal{B})$,

$$\dim(\mathcal{B}|_{[1, T]}) = m(\mathcal{B})T + n(\mathcal{B}) \leq m(\mathcal{B})T + \mathbb{1}(\mathcal{B})p(\mathcal{B}), \quad (\dim \mathcal{B})$$

which shows that the pairs of natural numbers

$$(m(\mathcal{B}), n(\mathcal{B})) \quad \text{and} \quad (m(\mathcal{B}), \mathbb{1}(\mathcal{B}))$$

characterize the model's complexity. The elements of the model class $\mathcal{L}_{m, \mathbb{1}}^q$ are linear time-invariant systems of complexity bounded by the pair $(m, \mathbb{1})$ and, similarly, the elements of the model class $\mathcal{L}_m^{q, n}$ are linear time-invariant systems of complexity bounded by the pair (m, n) . A static model is a special case of a dynamic model when the lag (or the order) is zero. This is reflected in the notation $\mathcal{L}_{m, \mathbb{1}}^{q, n}$: the linear static model class $\mathcal{L}_{m, 0}$ corresponds to the linear time-invariant model class $\mathcal{L}_{m, \mathbb{1}}$ with $\mathbb{1} = 0$.

Note that in the autonomous case, i.e., with $m(\mathcal{B}) = 0$, $\dim(\mathcal{B}) = n$. The dimension of the system corresponds to the number of degrees of freedom in selecting a trajectory. In the case of an autonomous system, the trajectory depends only on the initial condition (an $n(\mathcal{B})$ dimensional vector). In the presence of inputs, the number of degrees of freedom due to the initial condition is increased on each time step by the number of inputs, due to the free variables. Asymptotically as $T \rightarrow \infty$, the term mT in $(\dim \mathcal{B})$ dominates the term n . Therefore, in comparing linear time-invariant system's complexities, by convention, a system with more inputs is more complex than a system with less inputs, irrespective of their state dimensions.

2.3 Exact and Approximate Data Modeling

General Setting for Data Modeling

In order to treat static, dynamic, linear, and nonlinear modeling problems with unified terminology and notation, we need an abstract setting that is general enough to accommodate all envisaged applications. Such a setting is described in this section.

The data \mathcal{D} and a model \mathcal{B} for the data are subsets of a *universal set* \mathcal{U} of possible observations. In static modeling problems, \mathcal{U} is a real q -dimensional vector space \mathbb{R}^q , i.e., the observations are real valued vectors. In dynamic modeling problems, \mathcal{U} is a function space $(\mathbb{R}^q)^{\mathcal{T}}$, with \mathcal{T} being \mathbb{Z} in the discrete-time case and \mathbb{R} in the continuous-time case.

Note 2.12 (Categorical data and finite automata) In modeling problems with categorical data and finite automata, the universal set \mathcal{U} is discrete and may be finite.

We consider data sets \mathcal{D} consisting of a finite number of observations

$$\mathcal{D} = \{w_{d,1}, \dots, w_{d,N}\} \subset \mathcal{U}.$$

In discrete-time dynamic modeling problems, the $w_{d,j}$'s are trajectories, i.e.,

$$w_{d,j} = (w_{d,j}(1), \dots, w_{d,j}(T_j)), \quad \text{with } w_{d,j}(t) \in \mathbb{R}^q \text{ for all } t.$$

In dynamic problems, the data \mathcal{D} often consists of a single trajectory $w_{d,1}$, in which case the subscript index 1 is skipped and \mathcal{D} is identified with w_d . In static modeling problems, an observation $w_{d,j}$ is a vector and the alternative notation $d_j = w_{d,j}$ is used in order to emphasize the fact that the observations do not depend on time.

Note 2.13 (Given data vs. general trajectory) In order to distinguish a general trajectory w of the system from the given data w_d (a specific trajectory) we use the subscript "d" in the notation of the given data.

A *model class* \mathcal{M} is a set of sets of \mathcal{U} , i.e., \mathcal{M} is an element of the power set $2^{\mathcal{U}}$ of \mathcal{U} . We consider the generic model classes of

- linear static models \mathcal{L}_0 ,
- linear time-invariant models \mathcal{L} , and
- polynomial static models \mathcal{P} (see Chap. 6).

In some cases, however, subclasses of the generic classes above are of interest. For examples, the controllable and finite impulse response model subclasses of the class of linear time-invariant models, and the ellipsoids subclass of the class of second order polynomial models (conic sections).

The complexity c of a model \mathcal{B} is a vector of positive integers

$$c(\mathcal{B}) := \begin{cases} m(\mathcal{B}) = \dim(\mathcal{B}), & \text{if } \mathcal{B} \in \mathcal{L}_0, \\ (m(\mathcal{B}), l(\mathcal{B})) \text{ or } (m(\mathcal{B}), n(\mathcal{B})), & \text{if } \mathcal{B} \in \mathcal{L}, \\ (m(\mathcal{B}), \deg(R)), \text{ where } \mathcal{B} = \ker(R), & \text{if } \mathcal{B} \in \mathcal{P}. \end{cases} \quad (c(\mathcal{B}))$$

Complexities are compared in this book by the lexicographic ordering, i.e., two complexities are compared by comparing their corresponding elements in the increasing order of the indices. The first time an index is larger, the corresponding complexity is declared larger. For linear time-invariant dynamic models, this convention and the ordering of the elements in $c(\mathcal{B})$ imply that a model with more inputs is always more complex than a model with less inputs irrespective of their orders.

The complexity c of a model class \mathcal{M} is the largest complexity of a model in the model class. Of interest is the restriction of the generic model classes \mathcal{M} to subclasses $\mathcal{M}_{c_{\max}}$ of models with bounded complexity, e.g., $\mathcal{L}_{m,1_{\max}}^q$, with $m < q$.

Exact Data Modeling

A model \mathcal{B} is an *exact model* for the data \mathcal{D} if $\mathcal{D} \subset \mathcal{B}$. Otherwise, it is an *approximate model*. An exact model for the data may not exist in a model class $\mathcal{M}_{c_{\max}}$ of bounded complexity. This is generically the case when the data are noisy and the data set \mathcal{D} is large enough (relative to the model complexity). A practical data modeling problem must involve approximation. Our starting point, however, is the simpler problem of exact data modeling.

Problem 2.14 (Exact data modeling) Given data $\mathcal{D} \subset \mathcal{U}$ and a model class $\mathcal{M}_{c_{\max}} \in 2^{\mathcal{U}}$, find a model $\hat{\mathcal{B}}$ in $\mathcal{M}_{c_{\max}}$ that contains the data and has minimal (in the lexicographic ordering) complexity or assert that such a model does not exist, i.e.,

$$\text{minimize over } \mathcal{B} \in \mathcal{M}_{c_{\max}} \quad c(\mathcal{B}) \quad \text{subject to } \mathcal{D} \subset \mathcal{B} \quad (\text{EM})$$

The question occurs:

(Existence of exact model) Under what conditions on the data \mathcal{D} and the model class $\mathcal{M}_{c_{\max}}$ does a solution to problem (EM) exist?

If a solution exists, it is unique. This unique solution is called the *most powerful unfalsified model* for the data \mathcal{D} in the model class $\mathcal{M}_{c_{\max}}$ and is denoted by $\mathcal{B}_{\text{mpum}}(\mathcal{D})$. (The model class $\mathcal{M}_{c_{\max}}$ is not a part of the notation $\mathcal{B}_{\text{mpum}}(\mathcal{D})$ and is understood from the context.)

Suppose that the data \mathcal{D} are generated by a model \mathcal{B}_0 in the model class $\mathcal{M}_{c_{\max}}$, i.e.,

$$\mathcal{D} \subset \mathcal{B}_0 \in \mathcal{M}_{c_{\max}}.$$

Then, the exact modeling problem has a solution in the model class $\mathcal{M}_{c_{\max}}$, however, the solution $\mathcal{B}_{\text{mpum}}(\mathcal{D})$ may not be the data generating model \mathcal{B}_0 . The question occurs:

(Identifiability) Under what conditions on the data \mathcal{D} , the data generating model \mathcal{B}_0 , and the model class $\mathcal{M}_{c_{\max}}$, the most powerful unfalsified model $\mathcal{B}_{\text{mpum}}(\mathcal{D})$ in $\mathcal{M}_{c_{\max}}$ coincides with the data generating model \mathcal{B}_0 ?

Example 2.15 (Exact data fitting by a linear static model) Existence of a linear static model $\hat{\mathcal{B}}$ of bounded complexity m for the data \mathcal{D} is equivalent to rank deficiency of the matrix

$$\Phi(\mathcal{D}) := [d_1 \cdots d_N] \in \mathbb{R}^{q \times N},$$

composed of the data. (Show this.) Moreover, the rank of the matrix $\Phi(\mathcal{D})$ is equal to the *minimal* dimension of an exact model for \mathcal{D}

$$\text{existence of } \widehat{\mathcal{B}} \in \mathcal{L}_{m,0}^{\mathcal{A}}, \quad \text{such that } \mathcal{D} \subset \widehat{\mathcal{B}} \iff \text{rank}(\Phi(\mathcal{D})) \leq m. \quad (*)$$

The exact model

$$\widehat{\mathcal{B}} = \text{image}(\Phi(\mathcal{D})) \quad (**)$$

of minimal dimension

$$c(\mathcal{B}) = \text{rank}(\Phi(\mathcal{D}))$$

always exists and is unique.

The equivalence (*) between data modeling and the concept of rank is the basis for application of linear algebra and matrix computations to linear data modeling. Indeed, (**) provides an *algorithm* for exact linear static data modeling. As shown next, exact data modeling has also direct relevance to approximate data modeling.

Approximate Data Modeling

When an exact model does not exist in the considered model class, an approximate model that is in some sense “close” to the data is aimed at instead. Closeness is measured by a suitably defined criterion. This leads to the following approximate data modeling problem.

Problem 2.16 (Approximate data modeling) Given data $\mathcal{D} \subset \mathcal{U}$, a model class $\mathcal{M}_{c_{\max}} \in 2^{\mathcal{U}}$, and a measure $f(\mathcal{D}, \mathcal{B})$ for the lack of fit of the data \mathcal{D} by a model \mathcal{B} , find a model $\widehat{\mathcal{B}}$ in the model class $\mathcal{M}_{c_{\max}}$ that minimizes the lack of fit, i.e.,

$$\text{minimize over } \mathcal{B} \in \mathcal{M}_{c_{\max}} \quad f(\mathcal{D}, \mathcal{B}). \quad (\text{AM})$$

Since an observation w is a point in and the model \mathcal{B} is a subset of the data space \mathcal{U} , it is natural to measure the lack of fit between w and \mathcal{B} by the *geometric distance*

$$\text{dist}(w, \mathcal{B}) := \min_{\widehat{w} \in \mathcal{B}} \|w - \widehat{w}\|_2. \quad (\text{dist}(w, \mathcal{B}))$$

The auxiliary variable \widehat{w} is the best approximation of w in \mathcal{B} . Geometrically, it is the orthogonal projection of w on \mathcal{B} .

For the set of observations \mathcal{D} , we define the distance from \mathcal{D} to \mathcal{B} as

$$\text{dist}(\mathcal{D}, \mathcal{B}) := \min_{\widehat{w}_1, \dots, \widehat{w}_N \in \mathcal{B}} \sqrt{\sum_{j=1}^N \|w_{d,j} - \widehat{w}_j\|_2^2}. \quad (\text{dist})$$

The set of points

$$\widehat{\mathcal{D}} = \{\widehat{w}_1, \dots, \widehat{w}_N\}$$

in the definition of **(dist)** is an approximation of the data \mathcal{D} in the model \mathcal{B} . Note that problem **(dist)** is separable, i.e., it decouples into N independent problems **(dist(w, \mathcal{B}))**.

Algorithms for computing the geometric distance are discussed in Sect. 3.2, in the case of linear models, and in Chap. 6, in the case of polynomial models.

Note 2.17 (Invariance of **(dist)** to rigid transformation) The geometric distance $\text{dist}(\mathcal{D}, \mathcal{B})$ is invariant to a rigid transformation, i.e., translation, rotation, and reflection of the data points and the model.

An alternative distance measure, called *algebraic distance*, is based on a kernel representation $\mathcal{B} = \ker(R)$ of the model \mathcal{B} . Since R is a mapping from \mathcal{U} to \mathbb{R}^k , such that

$$w \in \mathcal{B} \iff R(w) = 0,$$

we have

$$\|R(w)\|_{\mathbb{F}} > 0 \iff w \notin \mathcal{B}.$$

The algebraic “distance” measures the lack of fit between w and \mathcal{B} by the “size” $\|R(w)\|_{\mathbb{F}}$ of the residual $R(w)$. For a data set \mathcal{D} , we define

$$\text{dist}'(\mathcal{D}, \mathcal{B}) := \sqrt{\sum_{j=1}^N \|R(w_{d,j})\|_{\mathbb{F}}^2}. \quad (\text{dist}')$$

The algebraic distance depends on the choice of the parameter R in a kernel representation of the model, while the geometric distance is representation invariant. In addition, the algebraic distance is not invariant to a rigid transformation. However, a modification of the algebraic distance that is invariant to a rigid transformation is presented in Sect. 6.3.

Example 2.18 (Geometric distance for linear and quadratic models) The two plots in Fig. 2.3 illustrate the geometric distance **(dist)** from a set of eight data points

$$\mathcal{D} = \{d_i = (x_i, y_i) \mid i = 1, \dots, 8\}$$

in the plane to, respectively, linear \mathcal{B}_1 and quadratic \mathcal{B}_2 models. As its name suggests, $\text{dist}(\mathcal{D}, \mathcal{B})$ has geometric interpretation—in order to compute the geometric distance, we project the data points on the models. This is a simple task (linear least squares problem) for linear models but a nontrivial task (nonconvex optimization problem) for nonlinear models. In contrast, the algebraic “distance” (not visualized in the figure) has no simple geometrical interpretation but is easy to compute for linear and nonlinear models alike.

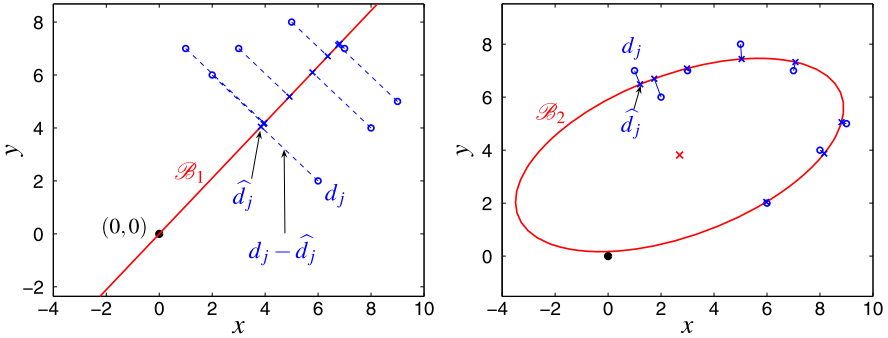


Fig. 2.3 Geometric distance from eight data points to a linear (*left*) and quadratic (*right*) models

Note 2.19 (Approximate modeling in the case of exact data) If an exact model \mathcal{B} exists in the model class $\mathcal{M}_{c_{\max}}$, then \mathcal{B} is a global optimum point of the approximate modeling problem (AM) (irrespective of the approximation criterion f being used). Indeed,

$$\mathcal{D} \subset \mathcal{B} \iff \text{dist}(\mathcal{D}, \mathcal{B}) = \text{dist}'(\mathcal{D}, \mathcal{B}) = 0.$$

An optimal approximate model, i.e., a solution of (AM), however, need not be unique. In contrast, the most powerful unfalsified model is unique. This is due to the fact that (AM) imposes an upper bound but does not minimize the model complexity, while (EM) minimizes the model complexity. As a result, when

$$c(\mathcal{B}_{\text{mpum}}(\mathcal{D})) < c_{\max},$$

(AM) has a nonunique solution. In the next section, we present a more general approximate data modeling problem formulation that minimizes simultaneously the complexity as well as the fitting error.

The terminology “geometric” and “algebraic” distance comes from the computer vision application of the methods for fitting curves and surfaces to data. In the system identification community, the geometric fitting method is related to the *misfit* approach and the algebraic fitting criterion is related to the *latency* approach. Misfit and latency computation are data smoothing operations. For linear time-invariant systems, the misfit and latency can be computed efficiently by Riccati type recursions. In the statistics literature, the geometric fitting is related to errors-in-variable estimation and the algebraic fitting is related to classical regression estimation, see Table 2.2.

Example 2.20 (Algebraic fit and errors-in-variables modeling) From a statistical point of view, the approximate data modeling problem (AM) with the geometric fitting criterion (dist) yields a maximum likelihood estimator for the true model \mathcal{B}_0 in the errors-in-variables setup

$$w_{d,j} = w_{0,j} + \tilde{w}_j, \tag{EIV}$$

Table 2.2 Correspondence among terms for data fitting criteria in different fields

Computer vision	System identification	Statistics	Mathematics
geometric fitting	misfit	errors-in-variables	implicit function
algebraic fitting	latency	regression	function

where

$$\mathcal{D}_0 := \{w_{0,1}, \dots, w_{0,N}\} \subset \mathcal{B}_0$$

is the true data and

$$\tilde{\mathcal{D}} := \{\tilde{w}_1, \dots, \tilde{w}_N\}$$

is the measurement noise, which is assumed to be a set of independent, zero mean, Gaussian random vectors, with covariance matrix $\sigma^2 I$.

Example 2.21 (Algebraic fit by a linear model and regression) A linear model class, defined by the input/output representation $\mathcal{B}_{i/o}(\Theta)$ and algebraic fitting criterion (dist'), where

$$w := \text{col}(u, y) \quad \text{and} \quad R(w) := \Theta^\top u - y$$

lead to the ordinary linear least squares problem

$$\text{minimize over } \Theta \in \mathbb{R}^{m \times p} \quad \|\Theta^\top \Phi(u_d) - \Phi(y_d)\|_F. \quad (\text{LS})$$

The statistical setting for the least squares approximation problem (LS) is the classical regression model

$$R(w_{d,j}) = e_j, \quad (\text{REG})$$

where e_1, \dots, e_N are zero mean independent and identically distributed random variables. Gauss–Markov’s theorem states that the least squares approximate solution is the best linear unbiased estimator for the regression model (REG).

Complexity–Accuracy Trade-off

Data modeling is a mapping from a given data set \mathcal{D} , to a model \mathcal{B} in a given model class \mathcal{M} :

$$\text{data set } \mathcal{D} \subset \mathcal{U} \xrightarrow{\text{data modeling problem}} \text{model } \mathcal{B} \in \mathcal{M} \in 2^{\mathcal{U}}.$$

A data modeling problem is defined by specifying the model class \mathcal{M} and one or more modeling criteria. Basic criteria in any data modeling problem are:

- “simple” model, measured by the model complexity $c(\mathcal{B})$, and
- “good” fit of the data by the model, measured by (vector) cost function $F(\mathcal{D}, \mathcal{B})$.

Small complexity $c(\mathcal{B})$ and small fitting error $F(\mathcal{D}, \mathcal{B})$, however, are contradicting objectives, so that a core issue throughout data modeling is the complexity–accuracy trade-off. A generic data modeling problem is:

Given a data set $\mathcal{D} \in \mathcal{U}$ and a measure F for the fitting error, solve the multi-objective optimization problem:

$$\text{minimize over } \mathcal{B} \in \mathcal{M} \quad \begin{bmatrix} c(\mathcal{B}) \\ F(\mathcal{D}, \mathcal{B}) \end{bmatrix}. \quad (\text{DM})$$

Next we consider the special cases of linear static models and linear time-invariant dynamic models with $F(\mathcal{D}, \mathcal{B})$ being $\text{dist}(\mathcal{D}, \mathcal{B})$ or $\text{dist}'(\mathcal{D}, \mathcal{B})$. The model class assumption implies that $\text{dim}(\mathcal{B})$ is a complexity measure in both static and dynamic cases.

Two Possible Scalarizations: Low Rank Approximation and Rank Minimization

The data set \mathcal{D} , can be parametrized by a real vector $p \in \mathbb{R}^{n_p}$. (Think of the vector p as a representation of the data in the computer memory.) For a linear model \mathcal{B} and exact data \mathcal{D} , there is a relation between the model complexity and the rank of a data matrix $\mathcal{S}(p)$:

$$c(\mathcal{B}_{\text{mpum}}(\mathcal{D})) = \text{rank}(\mathcal{S}(p)). \quad (*)$$

The mapping

$$\mathcal{S} : \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{m \times n}$$

from the data parameter vector p to the data matrix $\mathcal{S}(p)$ depends on the application. For example, $\mathcal{S}(p) = \Phi(\mathcal{D})$ is unstructured in the case of linear static modeling (see Example 2.15) and $\mathcal{S}(p) = \mathcal{H}(w_d)$ is Hankel structured in the case of autonomous linear time-invariant dynamic model identification,

Let p be the parameter vector for the data \mathcal{D} and \hat{p} be the parameter vector for the data approximation $\hat{\mathcal{D}}$. The geometric distance $\text{dist}(\mathcal{D}, \mathcal{B})$ can be expressed in terms of the parameter vectors p and \hat{p} as

$$\text{minimize over } \hat{p} \quad \|p - \hat{p}\|_2 \quad \text{subject to} \quad \hat{\mathcal{D}} \subset \mathcal{B}.$$

Moreover, the norm in the parameter space \mathbb{R}^{n_p} can be chosen as weighted 1-, 2-, and ∞ -(semi)norms:

$$\begin{aligned} \|\tilde{p}\|_{w,1} &:= \|w \odot \tilde{p}\|_1 := \sum_{i=1}^{n_p} |w_i \tilde{p}_i|, \\ \|\tilde{p}\|_{w,2} &:= \|w \odot \tilde{p}\|_2 := \sqrt{\sum_{i=1}^{n_p} (w_i \tilde{p}_i)^2}, \\ \|\tilde{p}\|_{w,\infty} &:= \|w \odot \tilde{p}\|_\infty := \max_{i=1,\dots,n_p} |w_i \tilde{p}_i|, \end{aligned} \quad (\|\cdot\|_w)$$

where w is a vector with nonnegative elements, specifying the weights, and \odot is the element-wise (Hadamard) product.

Using the data parametrization $(*)$ and one of the distance measures $(\|\cdot\|_w)$, the data modeling problem **(DM)** becomes the biobjective matrix approximation problem:

$$\text{minimize over } \hat{p} \quad \begin{bmatrix} \text{rank}(\mathcal{S}(\hat{p})) \\ \|p - \hat{p}\| \end{bmatrix}. \quad (\text{DM}')$$

Two possible ways to scalarize the biobjective problem **(DM')** are:

1. Misfit minimization subject to a bound r on the model complexity

$$\text{minimize over } \hat{p} \quad \|p - \hat{p}\| \quad \text{subject to} \quad \text{rank}(\mathcal{S}(\hat{p})) \leq r. \quad (\text{SLRA})$$

2. Model complexity minimization subject to a bound ε on the fitting error

$$\text{minimize over } \hat{p} \quad \text{rank}(\mathcal{S}(\hat{p})) \quad \text{subject to} \quad \|p - \hat{p}\| \leq \varepsilon. \quad (\text{RM})$$

Problem **(SLRA)** is a structured low rank approximation problem and **(RM)** is a rank minimization problem.

By varying the parameters r and ε from zero to infinity, both problems sweep the trade-off curve (set of Pareto optimal solutions) of **(DM')**. Note, however, that r ranges over the natural numbers and only small values are of practical interest. In addition, in applications often a “suitable” value for r can be chosen a priori or is even a part of the problem specification. In contrast, ε is a positive real number and is data dependent, so that a “suitable” value is not readily available. These considerations, suggest that the structured low rank approximation problem is a more convenient scalarization of **(DM')** for solving practical data modeling problems.

Convex relaxation algorithms for solving **(DM')** are presented in Sect. 3.3.

2.4 Unstructured Low Rank Approximation

Linear static data modeling leads to unstructured low rank approximation. Vice versa, unstructured low rank approximation problems can be given the interpretation (or motivation) of linear static data modeling problems. As argued in Sect. 1.1, these are equivalent problems. The data modeling view of the problem makes link

to applications. The low rank approximation view of the problem makes link to computational algorithms for solving the problem.

Problem 2.22 (Unstructured low rank approximation) Given a matrix $D \in \mathbb{R}^{\mathfrak{q} \times N}$, with $\mathfrak{q} \leq N$, a matrix norm $\|\cdot\|$, and an integer m , $0 < m < \mathfrak{q}$, find a matrix

$$\widehat{D}^* := \arg \min_{\widehat{D}} \|D - \widehat{D}\| \quad \text{subject to} \quad \text{rank}(\widehat{D}) \leq m. \quad (\text{LRA})$$

The matrix \widehat{D}^* is an optimal rank- m (or less) approximation of D with respect to the given norm $\|\cdot\|$.

The special case of (LRA) with $\|\cdot\|$ being the weighted 2-norm

$$\|\Delta D\|_W := \sqrt{\text{vec}^\top(\Delta D) W \text{vec}(\Delta D)}, \quad \text{for all } \Delta D \quad (\|\cdot\|_W)$$

where $W \in \mathbb{R}^{\mathfrak{q}N \times \mathfrak{q}N}$ is a positive definite matrix, is called *weighted low rank approximation problem*. In turn, special cases of the weighted low rank approximation problem are obtained when the weight matrix W has diagonal, block diagonal, or some other structure.

- *Element-wise weighting:*

$$W = \text{diag}(w_1, \dots, w_{\mathfrak{q}N}), \quad \text{where } w_i > 0, \text{ for } i = 1, \dots, \mathfrak{q}N.$$

- *Column-wise weighting:*

$$W = \text{diag}(W_1, \dots, W_N), \quad \text{where } W_j \in \mathbb{R}^{\mathfrak{q} \times \mathfrak{q}}, W_j > 0, \text{ for } j = 1, \dots, N.$$

- *Column-wise weighting with equal weight matrix for all columns:*

$$W = \text{diag}(W_1, \dots, W_1), \quad \text{where } W_1 \in \mathbb{R}^{\mathfrak{q} \times \mathfrak{q}}, W_1 > 0.$$

- *Row-wise weighting:*

$$\overline{W} = \text{diag}(W_1, \dots, W_{\mathfrak{q}}), \quad \text{where } W_i \in \mathbb{R}^{N \times N}, W_i > 0, \text{ for } i = 1, \dots, \mathfrak{q},$$

and \overline{W} is a matrix, such that

$$\|\Delta D\|_W = \|\Delta D^\top\|_{\overline{W}}, \quad \text{for all } \Delta D.$$

- *Row-wise weighting with equal weight matrix for all rows:*

$$\overline{W} = \text{diag}(\underbrace{W_{\mathfrak{r}}, \dots, W_{\mathfrak{r}}}_{\mathfrak{q}}), \quad \text{where } W_{\mathfrak{r}} \in \mathbb{R}^{N \times N}, W_{\mathfrak{r}} > 0.$$

Figure 2.4 shows the hierarchy of weighted low rank approximation problems, according to the structure of the weight matrix W . Exploiting the structure of the weight matrix allows more efficient solution of the corresponding weighted low rank approximation problems compared to the general problem with unstructured W .

As shown in the next section left/right weighting with equal matrix for all rows/columns corresponds to the approximation criteria $\|\sqrt{W_l}\Delta D\|_F$ and $\|\Delta D\sqrt{W_r}\|_F$. The approximation problem with criterion $\|\sqrt{W_l}\Delta D\sqrt{W_r}\|_F$ is called *two-sided weighted* and is also known as the generalized low rank approximation problem. This latter problem allows analytic solution in terms of the singular value decomposition of the data matrix.

Special Cases with Known Analytic Solutions

An extreme special case of the weighted low rank approximation problem is the “unweighted” case, i.e., weight matrix a multiple of the identity $W = v^{-1}I$, for some $v > 0$. Then, $\|\cdot\|_W$ is proportional to the Frobenius norm $\|\cdot\|_F$ and the low rank approximation problem has an analytic solution in terms of the singular value decomposition of D . The results is known as the Eckart–Young–Mirsky theorem or the matrix approximation lemma. In view of its importance, we refer to this case as the *basic low rank approximation problem*.

Theorem 2.23 (Eckart–Young–Mirsky) *Let*

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

be the singular value decomposition of D and partition U , $\Sigma =: \text{diag}(\sigma_1, \dots, \sigma_q)$, and V as follows:

$$U =: \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{matrix} m & q-m \\ \text{---} & \text{---} \end{matrix}, \quad \Sigma =: \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{matrix} m & & \\ & q-m & \\ \text{---} & & \text{---} \end{matrix} \quad \text{and} \quad V =: \begin{bmatrix} V_1 & V_2 \end{bmatrix} \begin{matrix} m & q-m \\ \text{---} & \text{---} \end{matrix} N.$$

Then the rank- m matrix, obtained from the truncated singular value decomposition

$$\widehat{D}^* = U_1 \Sigma_1 V_1^T,$$

is such that

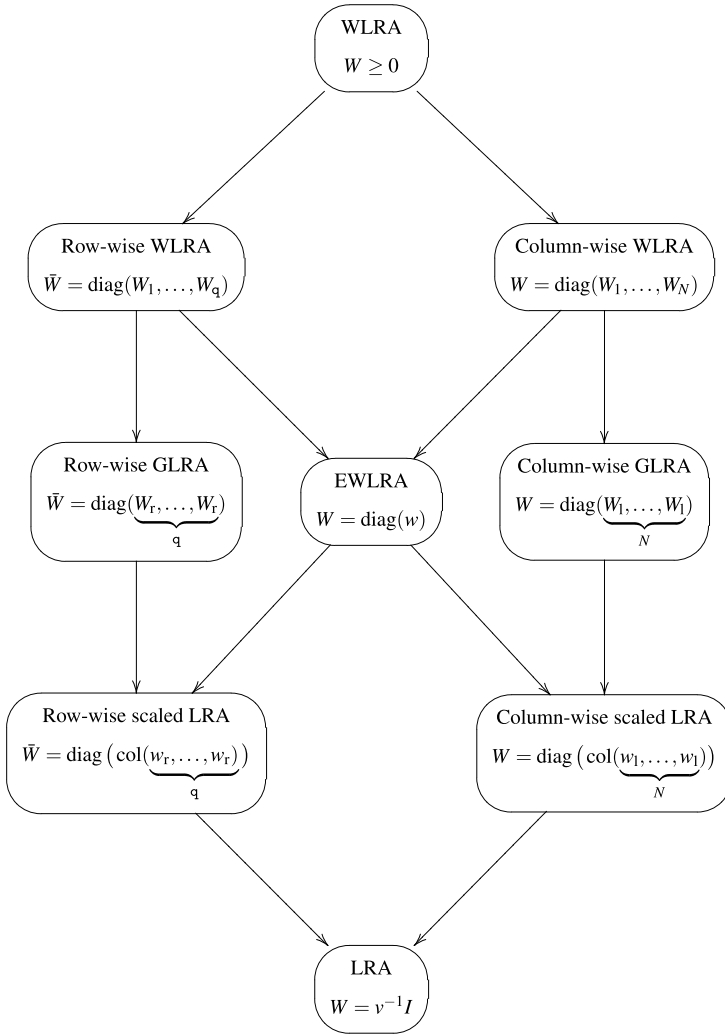
$$\|D - \widehat{D}^*\|_F = \min_{\text{rank}(\widehat{D}) \leq m} \|D - \widehat{D}\|_F = \sqrt{\sigma_{m+1}^2 + \dots + \sigma_q^2}.$$

The minimizer \widehat{D}^ is unique if and only if $\sigma_{m+1} \neq \sigma_m$.*

The proof is given in Appendix B.

Note 2.24 (Unitarily invariant norms) Theorem 2.23 holds for any norm $\|\cdot\|$ that is invariant under orthogonal transformations, i.e., satisfying the relation

$$\|U \Delta D V\| = \|\Delta D\|, \quad \text{for any } \Delta D \text{ and for any orthogonal matrices } U \text{ and } V.$$



LRA — low rank approximation GLRA — generalized low rank approximation
 WLRA — weighted low rank approximation EWLRA — element-wise weighted LRA

Fig. 2.4 Hierarchy of weighted low rank approximation problems according to the structure of the weight matrix W . On the *left side* are weighted low rank approximation problems with row-wise weighting and on the *right side* are weighted low rank approximation problems with column-wise weighting. The generality of the problem reduces from *top* to *bottom*

Note 2.25 (Approximation in the spectral norm) For a matrix ΔD , let $\|\Delta D\|_2$ be the spectral (2-norm induced) matrix norm

$$\|\Delta D\|_2 = \sigma_{\max}(\Delta D).$$

Then

$$\min_{\text{rank}(\widehat{D})=m} \|D - \widehat{D}\|_2 = \sigma_{m+1},$$

i.e., the optimal rank- m spectral norm approximation error is equal to the first neglected singular value. The truncated singular value decomposition yields an optimal approximation with respect to the spectral norm, however, in this case a minimizer is not unique even when the singular values σ_m and σ_{m+1} are different.

As defined, the low rank approximation problem aims at a matrix \widehat{D} that is a solution to the optimization problem (LRA). In data modeling problems, however, of primary interest is the optimal model, i.e., the most powerful unfalsified model for \widehat{D}^* . Theorem 2.23 gives the optimal approximating matrix \widehat{D}^* in terms of the singular value decomposition of the data matrix D . Minimal parameters of kernel and image representations of the corresponding optimal model are directly available from the factors of the singular value decomposition of D .

Corollary 2.26 *An optimal in the Frobenius norm approximate model for the data D in the model class $\mathcal{L}_{m,0}$, i.e., $\widehat{\mathcal{B}}^* := \mathcal{B}_{\text{mpum}}(\widehat{D}^*)$ is unique if and only if the singular values σ_m and σ_{m+1} of D are different, in which case*

$$\widehat{\mathcal{B}}^* = \ker(U_2^\top) = \text{image}(U_1).$$

The proof is left as an exercise.

64 (Low rank approximation 64) \equiv

```
function [R, P, dh] = lra(d, r)
[u, s, v] = svd(d, 0); R = u(:, (r + 1):end)';
P = u(:, 1:r);
if nargout > 2, dh = u(:, 1:r) * s(1:r, 1:r) * v(:, 1:r)';
end
```

Defines:

lra, used in chunks 85d, 88c, 101, 191b, 192e, and 229a.

Corollary 2.27 (Nested approximations) *The optimal in the Frobenius norm approximate models $\widehat{\mathcal{B}}_m^*$ for the data D in the model classes $\mathcal{L}_{m,0}$, where $m = 1, \dots, \mathfrak{q}$ are nested, i.e.,*

$$\widehat{\mathcal{B}}_{\mathfrak{q}} \subseteq \widehat{\mathcal{B}}_{\mathfrak{q}-1} \subseteq \dots \subseteq \widehat{\mathcal{B}}_1.$$

The proof is left as an exercise.

Note 2.28 (Efficient computation using QR factorization when $N \gg \mathfrak{q}$) An optimal model $\widehat{\mathcal{B}}^*$ for D depends only on the left singular vectors of D . Since post multiplication of D by an orthogonal matrix Q does not change the left singular vectors $\widehat{\mathcal{B}}^*$ is an optimal model for the data matrix DQ .

For $N \gg q$, computing the QR factorization

$$D^\top = \begin{bmatrix} R_1 \\ 0 \end{bmatrix} Q^\top, \quad \text{where } R_1 \text{ is upper triangular,} \quad (\text{QR})$$

and the singular value decomposition of R_1 is a more efficient alternative for finding $\hat{\mathcal{B}}$ than computing the singular value decomposition of D .

An analytic solution in terms of the singular value decomposition, similar to the one in Theorem 2.23 is not known for the general weighted low rank approximation problem. Presently the largest class of weighted low rank approximation problems with analytic solution are those with a weight matrix of the form

$$W = W_r \otimes W_l, \quad \text{where } W_l \in \mathbb{R}^{q \times q} \text{ and } W_r \in \mathbb{R}^{N \times N} \quad (W_r \otimes W_l)$$

are positive definite matrices and \otimes is the Kronecker product.

Using the identities

$$\text{vec}(AXB) = (B^\top \otimes A) \text{vec}(X)$$

and

$$(A_1 \otimes B_1)(A_2 \otimes B_2) = (A_1 A_2) \otimes (B_1 B_2),$$

we have

$$\begin{aligned} \|D - \hat{D}\|_{W_r \otimes W_l} &= \sqrt{\text{vec}^\top(\Delta D)(W_r \otimes W_l) \text{vec}(\Delta D)} \\ &= \|(\sqrt{W_r} \otimes \sqrt{W_l}) \text{vec}(\Delta D)\|_2 \\ &= \|\text{vec}(\sqrt{W_l} \Delta D \sqrt{W_r})\|_2 \\ &= \|\sqrt{W_l} \Delta D \sqrt{W_r}\|_F. \end{aligned}$$

Therefore, the low rank approximation problem (LRA) with norm $(\|\cdot\|_W)$ and weight matrix $(W_r \otimes W_l)$ is equivalent to the two-sided weighted (or generalized) low rank approximation problem

$$\begin{aligned} &\text{minimize} \quad \text{over } \hat{D} \quad \|\sqrt{W_l}(D - \hat{D})\sqrt{W_r}\|_F \\ &\text{subject to} \quad \text{rank}(\hat{D}) \leq m, \end{aligned} \quad (\text{WLRA2})$$

which has an analytic solution.

Theorem 2.29 (Two-sided weighted low rank approximation) *Define the modified data matrix*

$$D_m := \sqrt{W_l} D \sqrt{W_r},$$

and let \widehat{D}_m^* be the optimal (unweighted) low rank approximation of D_m . Then

$$\widehat{D}^* := (\sqrt{W_l})^{-1} \widehat{D}_m^* (\sqrt{W_r})^{-1},$$

is a solution of the following two-sided weighted low rank approximation problem (WLRA2). A solution always exists. It is unique if and only if \widehat{D}_m^* is unique.

The proof is left as an exercise (Problem P.14).

Exercise 2.30 Using the result in Theorem 2.29, write a function that solves the two-sided weighted low rank approximation problem (WLRA2).

Data Modeling via (LRA)

The following problem is the approximate modeling problem (AM) for the model class of linear static models, i.e., $\mathcal{M}_{C_{\max}} = \mathcal{L}_{m,0}$, with the orthogonal distance approximation criterion, i.e., $f(\mathcal{D}, \mathcal{B}) = \text{dist}(\mathcal{D}, \mathcal{B})$. The norm $\|\cdot\|$ in the definition of dist , however, in the present context is a general vector norm, rather than the 2-norm.

Problem 2.31 (Static data modeling) Given N , α -variable observations

$$\{d_1, \dots, d_N\} \subset \mathbb{R}^\alpha,$$

a matrix norm $\|\cdot\|$, and model complexity m , $0 < m < \alpha$,

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{\mathcal{B}} \text{ and } \widehat{D} && \|D - \widehat{D}\| \\ & \text{subject to} && \text{image}(\widehat{D}) \subseteq \widehat{\mathcal{B}} && \text{and } \dim(\widehat{\mathcal{B}}) \leq m, \end{aligned} \quad (\text{AM } \mathcal{L}_{m,0})$$

where $D \in \mathbb{R}^{\alpha \times N}$ is the data matrix $D := [d_1 \ \dots \ d_N]$.

A solution $\widehat{\mathcal{B}}^*$ to (AM $\mathcal{L}_{m,0}$) is an optimal approximate model for the data D with complexity bounded by m . Of course, $\widehat{\mathcal{B}}^*$ depends on the approximation criterion, specified by the given norm $\|\cdot\|$. A justification for the choice of the norm $\|\cdot\|$ is provided in the errors-in-variables setting (see Example 2.20), i.e., the data matrix D is assumed to be a noisy measurement of a true matrix D_0

$$\begin{aligned} D &= D_0 + \widetilde{D}, & \text{image}(D_0) &= \mathcal{B}_0, & \dim(\mathcal{B}_0) &\leq m, & \text{and} \\ \text{vec}(\widetilde{D}) &\sim N(0, \sigma^2 W^{-1}), & \text{where } W &> 0 \end{aligned} \quad (\text{EIV}_0)$$

and \widetilde{D} is the measurement error that is assumed to be a random matrix with zero mean and normal distribution. The true matrix D_0 is “generated” by a model \mathcal{B}_0 , with a known complexity bound m . The model \mathcal{B}_0 is the object to be estimated in the errors-in-variables setting.

Proposition 2.32 (Maximum likelihood property of optimal static model $\widehat{\mathcal{B}}^*$) *Assume that the data are generated in the errors-in-variables setting (EIV₀), where the matrix $W \succ 0$ is known and the scalar σ^2 is unknown. Then a solution $\widehat{\mathcal{B}}^*$ to Problem 2.31 with weighted 2-norm ($\|\cdot\|_W$) is a maximum likelihood estimator for the true model \mathcal{B}_0 .*

The proof is given in Appendix B.

The main assumption of Proposition 2.32 is

$$\text{cov}(\text{vec}(\widetilde{D})) = \sigma^2 W^{-1}, \quad \text{with } W \text{ given.}$$

Note, however, that σ^2 is not given, so that the probability density function of \widetilde{D} is not completely specified. Proposition 2.32 shows that the problem of computing the maximum likelihood estimator in the errors-in-variables setting is equivalent to Problem 2.22 with the weighted norm $\|\cdot\|_W$. Maximum likelihood estimation for density functions other than normal leads to low rank approximation with norms other than the weighted 2-norm.

2.5 Structured Low Rank Approximation

Structured low rank approximation is a low rank approximation, in which the approximating matrix \widehat{D} is constrained to have some a priori specified structure; typically, the same structure as the one of the data matrix D . Common structures encountered in applications are Hankel, Toeplitz, Sylvester, and circulant as well as their block versions. In order to state the problem in its full generality, we first define a structured matrix. Consider a mapping \mathcal{S} from a parameter space \mathbb{R}^{n_p} to a set of matrices $\mathbb{R}^{m \times n}$. A matrix $\widehat{D} \in \mathbb{R}^{m \times n}$ is called \mathcal{S} -structured if it is in the image of \mathcal{S} , i.e., if there exists a parameter $\widehat{p} \in \mathbb{R}^{n_p}$, such that $\widehat{D} = \mathcal{S}(\widehat{p})$.

Problem SLRA (Structured low rank approximation) Given a structure specification

$$\mathcal{S} : \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{m \times n}, \quad \text{with } m \leq n,$$

a parameter vector $p \in \mathbb{R}^{n_p}$, a vector norm $\|\cdot\|$, and an integer r , $0 < r < \min(m, n)$,

$$\text{minimize over } \widehat{p} \quad \|\widehat{p} - p\| \quad \text{subject to} \quad \text{rank}(\mathcal{S}(\widehat{p})) \leq r. \quad (\text{SLRA})$$

The matrix $\widehat{D}^* := \mathcal{S}(\widehat{p}^*)$ is an optimal rank- r (or less) approximation of the matrix $D := \mathcal{S}(p)$, within the class of matrices with the same structure as D . Problem SLRA is a generalization of Problem 2.22. Indeed, choosing \mathcal{S} to be vec^{-1} (an operation reconstructing a matrix M from the vector $\text{vec}(M)$) and norm $\|\Delta p\|$ to be such that

$$\|\Delta p\| = \|\mathcal{S}(\Delta p)\|, \quad \text{for all } \Delta p,$$

Problem SLRA is equivalent to Problem 2.22.

Special Cases with Known Analytic Solutions

We showed that some weighted unstructured low rank approximation problems have global analytic solution in terms of the singular value decomposition. Similar result exists for circulant structured low rank approximation. If the approximation criterion is a unitarily invariant matrix norm, the unstructured low rank approximation (obtained for example from the truncated singular value decomposition) is unique. In the case of a circulant structure, it turns out that this unique minimizer also has circulant structure, so the structure constraint is satisfied without explicitly enforcing it in the approximation problem.

An efficient computational way of obtaining the circulant structured low rank approximation is the fast Fourier transform. Consider the scalar case and let

$$P_k := \sum_{j=1}^{n_p} p_j e^{-i\frac{2\pi}{n_p}kj}$$

be the discrete Fourier transform of p . Denote with \mathcal{K} the subset of $\{1, \dots, n_p\}$ consisting of the indices of the m largest elements of $\{|P_1|, \dots, |P_{n_p}|\}$. Assuming that \mathcal{K} is uniquely defined by the above condition, i.e., assuming that

$$k \in \mathcal{K} \quad \text{and} \quad k' \notin \mathcal{K} \quad \implies \quad |P_k| > |P_{k'}|,$$

the solution \hat{p}^* of the structured low rank approximation problem with \mathcal{S} a circulant matrix is unique and is given by

$$\hat{p}^* = \frac{1}{n_p} \sum_{k \in \mathcal{K}} P_k e^{i\frac{2\pi}{n_p}kj}.$$

Data Modeling via (SLRA)

The reason to consider the more general structured low rank approximation is that $D = \mathcal{S}(p)$ being low rank and Hankel structured is equivalent to p being generated by a linear time-invariant dynamic model. To show this, consider first the special case of a scalar Hankel structure

$$\mathcal{H}_{1+1}(p) := \begin{bmatrix} p_1 & p_2 & \cdots & p_{n_p-1} \\ p_2 & p_3 & \cdots & p_{n_p-1+1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1+1} & p_{1+2} & \cdots & p_{n_p} \end{bmatrix}.$$

The approximation matrix

$$\hat{D} = \mathcal{H}_{1+1}(\hat{p})$$

being rank deficient implies that there is a nonzero vector $R = [R_0 \ R_1 \ \cdots \ R_1]$, such that

$$R\mathcal{H}_{1+1}(\widehat{p}) = 0.$$

Due to the Hankel structure, this system of equations can be written as

$$R_0\widehat{p}_t + R_1\widehat{p}_{t+1} + \cdots + R_1\widehat{p}_{t+1} = 0, \quad \text{for } t = 1, \dots, n_p - 1,$$

i.e., a homogeneous constant coefficients difference equation. Therefore, \widehat{p} is a trajectory of an autonomous linear time-invariant system, defined by (KER). Recall that for an autonomous system \mathcal{B} ,

$$\dim(\mathcal{B}|_{[1,T]}) = n(\mathcal{B}), \quad \text{for } T \geq n(\mathcal{B}).$$

The scalar Hankel low rank approximation problem is then equivalent to the following dynamic modeling problem. Given T samples of a scalar signal $w_d \in \mathbb{R}^T$, a signal norm $\|\cdot\|$, and a model complexity n ,

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{\mathcal{B}} \text{ and } \widehat{w} && \|w_d - \widehat{w}\| \\ & \text{subject to} && \widehat{w} \in \widehat{\mathcal{B}}|_{[1,T]} && \text{and } \dim(\widehat{\mathcal{B}}) \leq n. \end{aligned} \quad (\text{AM } \mathcal{L}_{0,1})$$

A solution $\widehat{\mathcal{B}}^*$ is an optimal approximate model for the signal w_d with bounded complexity: order at most n .

In the general case when the data are a vector valued signal with \mathfrak{q} variables, the model \mathcal{B} can be represented by a kernel representation, where the parameters R_i are $\mathfrak{p} \times \mathfrak{q}$ matrices. The block-Hankel structured low rank approximation problem is equivalent to the approximate linear time-invariant dynamic modeling problem (AM) with model class $\mathcal{M}_{c_{\max}} = \mathcal{L}_{m,1}$ and orthogonal distance fitting criterion.

Problem 2.33 (Linear time-invariant dynamic modeling) Given T samples, \mathfrak{q} variables, vector signal $w_d \in (\mathbb{R}^{\mathfrak{q}})^T$, a signal norm $\|\cdot\|$, and a model complexity $(m, 1)$,

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{\mathcal{B}} \text{ and } \widehat{w} && \|w_d - \widehat{w}\| \\ & \text{subject to} && \widehat{w} \in \widehat{\mathcal{B}}|_{[1,T]} && \text{and } \widehat{\mathcal{B}} \in \mathcal{L}_{m,1}^{\mathfrak{q}}. \end{aligned} \quad (\text{AM } \mathcal{L}_{m,1})$$

The solution $\widehat{\mathcal{B}}^*$ is an optimal approximate model for the signal w_d with complexity bounded by $(m, 1)$. Note that problem (AM $\mathcal{L}_{m,1}$) reduces to

- (AM $\mathcal{L}_{0,1}$) when $m = 0$, i.e., when the model is autonomous, and
- (AM $\mathcal{L}_{m,0}$) when $1 = 0$, i.e., when the model is static.

Therefore, (AM $\mathcal{L}_{m,1}$) is a proper generalization of linear static and dynamic autonomous data modeling problems.

Computing the optimal approximate model $\widehat{\mathcal{B}}^*$ from the solution \widehat{p}^* to Problem SLRA is an exact identification problem. As in the static approximation problem, however, the parameter of a model representation is an optimization variable

of the optimization problem, used for Problem SLRA, so that a representation of the model is actually obtained directly from the optimization solver.

Similarly to the static modeling problem, the dynamic modeling problem has a maximum likelihood interpretation in the errors-in-variables setting.

Proposition 2.34 (Maximum likelihood property of an optimal dynamic model)
Assume that the data w_d are generated in the errors-in-variables setting

$$w_d = w_0 + \tilde{w}, \quad \text{where } w_0 \in \mathcal{B}_0|_{[1,T]} \in \mathcal{L}_{m,1}^q \text{ and } \tilde{w} \sim \mathcal{N}(0, vI). \quad (\text{EIV})$$

Then an optimal approximate model $\hat{\mathcal{B}}^$, solving (AM $\mathcal{L}_{m,1}$) with $\|\cdot\| = \|\cdot\|_2$ is a maximum likelihood estimator for the true model \mathcal{B}_0 .*

The proof is analogous to the proof of Proposition 2.32 and is skipped.

In Chap. 3, we describe local optimization methods for general affinely structured low rank approximation problems and show how in the case of Hankel, Toeplitz, and Sylvester structured problem, the matrix structure can be exploited for efficient cost function evaluation.

2.6 Notes

The concept of the most powerful unfalsified model is introduced in Willems (1986, Definition 4). See also Antoulas and Willems (1993), Kuijper and Willems (1997), Kuijper (1997) and Willems (1997). Kung's method for approximate system realization is presented in Kung (1978).

Modeling by the orthogonal distance fitting criterion (misfit approach) is initiated in Willems (1987) and further on developed in Roorda and Heij (1995), Roorda (1995a, 1995b), Markovsky et al. (2005b), where algorithms for solving the problems are developed. A proposal for combination of misfit and latency for linear time-invariant system identification is made in Lemmerling and De Moor (2001).

Weighted low rank approximation methods are developed in Gabriel and Zamir (1979), De Moor (1993), Wentzell et al. (1997), Markovsky et al. (2005a), Manton et al. (2003), Srebro (2004), Markovsky and Van Huffel (2007). The analytic solution of circulant structured low rank approximation problem is derived independently in the optimization community (Beck and Ben-Tal 2006) and in the systems and control community (Vanluyten et al. 2005).

Equivalence of Low Rank Approximation and Principal Component Analysis

The principal component analysis method for dimensionality reduction is usually introduced in a stochastic setting as maximization of the variance of the projected

data on a subspace. Computationally, however, the problem of finding the principal components and the corresponding principal vectors is an eigenvalue/eigenvector decomposition problem for the sample covariance matrix

$$\Psi(\mathcal{D}) := \Phi(\mathcal{D})\Phi^\top(\mathcal{D}), \quad \text{where } \Phi(\mathcal{D}) := [d_1 \cdots d_N].$$

From this algorithmic point view, the equivalence of principal component analysis and low rank approximation problem is a basic linear algebra fact: the space spanned by the first m principal vectors of \mathcal{D} coincides with the model $\widehat{\mathcal{B}} = \text{image}(\Phi(\widehat{\mathcal{D}}))$, where $\widehat{\mathcal{D}}$ is a solution of the low rank approximation problem (LRA).

References

- Abelson H, diSessa A (1986) *Turtle geometry*. MIT Press, New York
- Antoulas A, Willems JC (1993) A behavioral approach to linear exact modeling. *IEEE Trans Automat Control* 38(12):1776–1802
- Beck A, Ben-Tal A (2006) A global solution for the structured total least squares problem with block circulant matrices. *SIAM J Matrix Anal Appl* 27(1):238–255
- De Moor B (1993) Structured total least squares and L_2 approximation problems. *Linear Algebra Appl* 188–189:163–207
- Gabriel K, Zamir S (1979) Lower rank approximation of matrices by least squares with any choice of weights. *Technometrics* 21:489–498
- Kuijper M (1997) An algorithm for constructing a minimal partial realization in the multivariable case. *Control Lett* 31(4):225–233
- Kuijper M, Willems JC (1997) On constructing a shortest linear recurrence relation. *IEEE Trans Automat Control* 42(11):1554–1558
- Kung S (1978) A new identification method and model reduction algorithm via singular value decomposition. In: *Proc 12th asilomar conf circuits, systems, computers*, Pacific Grove, pp 705–714
- Lemmerling P, De Moor B (2001) Misfit versus latency. *Automatica* 37:2057–2067
- Manton J, Mahony R, Hua Y (2003) The geometry of weighted low-rank approximations. *IEEE Trans Signal Process* 51(2):500–514
- Markovsky I, Van Huffel S (2007) Left vs right representations for solving weighted low rank approximation problems. *Linear Algebra Appl* 422:540–552
- Markovsky I, Rastello ML, Premoli A, Kukush A, Van Huffel S (2005a) The element-wise weighted total least squares problem. *Comput Stat Data Anal* 50(1):181–209
- Markovsky I, Willems JC, Van Huffel S, Moor BD, Pintelon R (2005b) Application of structured total least squares for system identification and model reduction. *IEEE Trans Automat Control* 50(10):1490–1500
- Roorda B (1995a) Algorithms for global total least squares modelling of finite multivariable time series. *Automatica* 31(3):391–404
- Roorda B (1995b) *Global total least squares—a method for the construction of open approximate models from vector time series*. PhD thesis, Tinbergen Institute
- Roorda B, Heij C (1995) Global total least squares modeling of multivariate time series. *IEEE Trans Automat Control* 40(1):50–63
- Srebro N (2004) *Learning with matrix factorizations*. PhD thesis, MIT
- Vanluyten B, Willems JC, De Moor B (2005) Model reduction of systems with symmetries. In: *Proc 44th IEEE conf dec control*, Seville, Spain, pp 826–831
- Wentzell P, Andrews D, Hamilton D, Faber K, Kowalski B (1997) Maximum likelihood principal component analysis. *J Chemom* 11:339–366

- Willems JC (1986) From time series to linear system—Part II. Exact modelling. *Automatica* 22(6):675–694
- Willems JC (1987) From time series to linear system—Part III. Approximate modelling. *Automatica* 23(1):87–115
- Willems JC (1997) On interconnections, control, and feedback. *IEEE Trans Automat Control* 42:326–339

Chapter 3

Algorithms

Writing a book is a little more difficult than writing a technical paper, but writing software is a lot more difficult than writing a book.

D. Knuth

3.1 Subspace Methods

The singular value decomposition is at the core of many algorithms for approximate modeling, most notably the methods based on balanced model reduction, the subspace identification methods, and the MUSIC and ESPRIT methods in signal processing. The reason for this is that the singular value decomposition is a robust and efficient way of computing unstructured low rank approximation of a matrix in the Frobenius norm. In system identification, signal processing, and computer algebra, however, the low rank approximation is restricted to the class of matrices with specific (Hankel, Toeplitz, Sylvester) structure. Ignoring the structure constraint renders the singular value decomposition-based methods suboptimal with respect to a desired optimality criterion.

Except for the few special cases, described in Sect. 2.5, there are no global solution methods for general structured and weighted low rank approximation problems. The singular value decomposition-based methods can be seen as relaxations of the original NP-hard structured weighted low rank approximation problem, obtained by removing the structure constraint and using the Frobenius norm in the approximation criterion. Another approach is taken in Sect. 3.3, where convex relaxations of the related rank minimization problem are proposed. Convex relaxation methods give polynomial time suboptimal solutions and are shown to provide globally optimal solutions in certain cases.

Presently, there is no uniformly best method for computing suboptimal structured low rank approximation. In the context of system identification (i.e., block-Hankel structured low rank approximation), subspace and local optimization-based methods have been compared on practical data sets. In general, the heuristic methods are faster but less accurate than the methods based on local optimization. It is a common

practice to use a suboptimal solution obtained by a heuristic method as an initial approximation for an optimization-based method. Therefore, the two approaches complement each other.

Realization Algorithms

The aim of the realization algorithms is to compute a state space representation $\mathcal{B}_{i/s/o}(A, B, C, D)$ of the minimal realization $\mathcal{B}_{\text{mpum}}(H)$ of H , see Sect. 2.2. Finding the model parameters A, B, C, D can be done by computing a rank revealing factorization of a Hankel matrix constructed from the data. Let

$$\mathcal{H}_{n+1, n+1}(\sigma H) = \Gamma \Delta, \quad \text{where } \Gamma \in \mathbb{R}^{\mathfrak{p}(n+1) \times n} \text{ and } \Delta \in \mathbb{R}^{n \times \mathfrak{m}(n+1)}$$

be a rank revealing factorization of the finite Hankel matrix $\mathcal{H}_{n+1, n+1}(\sigma H)$. The Hankel structure implies that the factors Γ and Δ are observability and controllability matrices, i.e., there are matrices $A \in \mathbb{R}^{n \times n}$, $C \in \mathbb{R}^{\mathfrak{p} \times n}$, and $B \in \mathbb{R}^{n \times \mathfrak{m}}$, such that

$$\Gamma = \mathcal{O}_{n+1}(A, C) \quad \text{and} \quad \Delta = \mathcal{C}_{n+1}(A, B).$$

Then, $\mathcal{B}_{i/s/o}(A, B, C, D)$ is the minimal realization of H .

A rank revealing factorization is not unique. For any $n \times n$ nonsingular matrix T , a new factorization

$$\mathcal{H}_{n+1, n+1}(\sigma H) = \Gamma \Delta = \underbrace{\Gamma T}_{\Gamma'} \underbrace{T^{-1} \Delta}_{\Delta'}$$

is obtained with the same inner dimension. The nonuniqueness of the factorization corresponds to the nonuniqueness of the input/state/output representation of the minimal realization due to a change of the state space bases:

$$\mathcal{B}_{i/s/o}(A, B, C, D) = \mathcal{B}_{i/s/o}(T^{-1}AT, T^{-1}B, CT, D).$$

The structure of the observability and controllability matrices is referred to as the *shift structure*. The parameters B and C of an input/state/output representation of the realization are directly available from the first block elements of Γ and Δ , respectively.

$$\begin{aligned} 74a \quad \langle \Gamma, \Delta \rangle &\mapsto (A, B, C) \quad 74a) \equiv & (76d) \quad 74b) > \\ & \mathfrak{b} = C(:, 1:\mathfrak{m}); \quad \mathfrak{c} = \mathcal{O}(1:\mathfrak{p}, :); \end{aligned}$$

The parameter A is computed from the overdetermined system of linear equations

$$\sigma^{-1} \Gamma A = \sigma \Gamma, \quad (\text{SE}_1)$$

where, acting on a block matrix, σ and σ^{-1} remove, respectively, the first and the last block elements.

$$\begin{aligned} 74b \quad \langle \Gamma, \Delta \rangle &\mapsto (A, B, C) \quad 74a) + \equiv & (76d) \quad < 74a \\ & \mathfrak{a} = \mathcal{O}(1:\text{end} - \mathfrak{p}, :) \setminus \mathcal{O}((\mathfrak{p} + 1):\text{end}, :); \end{aligned}$$

Note 3.1 (Solution of the shift equation) When a unique solution exists, the code in chunk 74a computes the exact solution. When a solution A of (SE₁) does not exist, the same code computes a least squares approximate solution.

Equivalently (in the case of exact data), A can be computed from the Δ factor

$$A\sigma^{-1}\Delta = \sigma\Delta. \quad (\text{SE}_2)$$

In the case of noisy data (approximate realization problem) or data from a high order system (model reduction problem), (SE₁) and (SE₂) generically have no exact solutions and their least squares approximate solutions are different.

Implementation

As square as possible Hankel matrix $\mathcal{H}_{i,j}(\sigma H)$ is formed, using all data points, i.e.,

$$i = \left\lceil \frac{Tm}{m+p} \right\rceil \quad \text{and} \quad j = T - i. \quad (i, j)$$

75a *(dimension of the Hankel matrix 75a)* \equiv (76d 81)

```

if ~exist('i', 'var') | ~isreal(i) | isempty(i)
    i = ceil(T * m / (m + p));
end
if ~exist('j', 'var') | ~isreal(j) | isempty(j)
    j = T - i;
elseif j > T - i
    error('Not enough data.')
end

```

The choice (i, j) for the dimension of the Hankel matrix maximizes the order of the realization that can be computed. Indeed, a realization of order n can be computed from the matrix $\mathcal{H}_{i,j}(\sigma H)$ provided

$$n \leq n_{\max} := \min(pi - 1, mj).$$

75b *(check $n < \min(pi - 1, mj)$ 75b)* \equiv (76d)

```

if n > min(i * p - 1, j * m), error('Not enough data'), end

```

The minimal number of samples T of the impulse response that allows identification of a system of order n is

$$T_{\min} := \left\lceil \frac{n}{p} \right\rceil + \left\lceil \frac{n}{m} \right\rceil + 1. \quad (T_{\min})$$

The key computational step of the realization algorithm is the factorization of the Hankel matrix. In particular, this step involves rank determination. In finite precision arithmetic, however, rank determination is a nontrivial problem. A numerically reliable way of computing rank is the singular value decomposition

$$\mathcal{H}_{i,j}(\sigma H) = U \Sigma V^T.$$

76a *(singular value decomposition of $\mathcal{H}_{i,j}(\sigma H)$ 76a)* \equiv (76d)
`[U, S, V] = svd(blkhank(h(:, :, 2:end), i, j), 0);`
`s = diag(S);`

Uses blkhank 25b.

The order n of the realization is theoretically equal to the rank of the Hankel matrix, which is equal to the number of nonzero singular values $\sigma_1, \dots, \sigma_{\min(i,j)}$. In practice, the system's order is estimated as the numerical rank of the Hankel matrix, i.e., the number of singular values greater than a user specified tolerance.

76b *(order selection 76b)* \equiv (76d)
`(default tolerance tol 38b), n = sum(s > tol);`
 Defining the partitioning

$$U =: [U_1 \ U_2], \quad \Sigma =: \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}^n, \quad \text{and} \quad V =: [V_1 \ V_2],$$

the factors Γ and Δ of the rank revealing factorization are chosen as follows

$$\Gamma := U_1 \sqrt{\Sigma_1} \quad \text{and} \quad \Delta := \sqrt{\Sigma_1} V_1^\top. \quad (\Gamma, \Delta)$$

76c *(define Δ and Γ 76c)* \equiv (76d)
`sqrt_s = sqrt(s(1:n))';`
`O = sqrt_s(ones(size(U, 1), 1), :) .* U(:, 1:n);`
`C = (sqrt_s(ones(size(V, 1), 1), :) .* V(:, 1:n))';`

This choice leads to a *finite-time balanced* realization of $\mathcal{B}_{i/s/o}(A, B, C, D)$, i.e., the finite time controllability and observability Gramians

$$\mathcal{O}_i^\top(A, C) \mathcal{O}_i(A, C) = \Gamma^\top \Gamma \quad \text{and} \quad \mathcal{C}_j(A, B) \mathcal{C}_j^\top(A, B) = \Delta \Delta^\top$$

are equal,

$$\Gamma^\top \Gamma = \Delta \Delta^\top = \Sigma.$$

Note 3.2 (Kung's algorithm) The combination of the described realization algorithm with the singular value decomposition-based rank revealing factorization (Γ, Δ) , i.e., unstructured low rank approximation, is referred to as *Kung's algorithm*.

76d *($H \mapsto \mathcal{B}_{i/s/o}(A, B, C, D)$ 76d)* \equiv
`function [sys, hh] = h2ss(h, n, tol, i, j)`
`(reshape H and define m, p, T 77)`
`(dimension of the Hankel matrix 75a)`
`(singular value decomposition of $\mathcal{H}_{i,j}(\sigma H)$ 76a)`
`if ~exist('n', 'var') | isempty(n)`
`(order selection 76b)`
`else`
`(check $n < \min(pi - 1, mj)$ 75b)`
`end`

```

(define  $\Delta$  and  $\Gamma$  76c)
( $\Gamma, \Delta$ )  $\mapsto$  (A, B, C) 74a, sys = ss(a, b, c, h(:, :, 1), -1);
if nargout > 1, hh = shiftdim(impulse(sys, T), 1); end

```

Defines:

h2ss, used in chunks 81, 100e, 101b, 109d, 110b, 113, and 125c.

Similarly, to the convention (w) on p. 26 for representing vector and matrix valued trajectories in MATLAB, the impulse response H is stored as an $p \times m \times T$ tensor

$$h(:, :, t) = H(t),$$

or, in the case of a single input system, as a $p \times T$ matrix.

```

77 (reshape H and define m, p, T 77)  $\equiv$  (76d 108b)
if length(size(h)) == 2
    [p, T] = size(h); if p > T, h = h'; [p, T] = size(h); end
    h = reshape(h, p, 1, T);
end
[p, m, T] = size(h);

```

Note 3.3 (Approximate realization by Kung's algorithm) When H is not realizable by a linear time-invariant system of order less than or equal to n_{\max} , i.e., when $\mathcal{H}_{i,j}(\sigma H)$ is full rank, h2ss computes an approximate realization $\mathcal{B}_{i/s/o}(A, B, C, D)$ of order $n \leq n_{\max}$. The link between the realization problem and Hankel structured low rank approximation implies that

Kung's algorithm, implemented in the function h2ss, is a method for Hankel structured low rank approximation. The structured low rank approximation of $\mathcal{H}_{i,j}(\sigma H)$ is $\mathcal{H}_{i,j}(\sigma \hat{H})$, where \hat{H} is the impulse response of the approximate realization $\mathcal{B}_{i/s/o}(A, B, C, D)$.

Note 3.4 (Suboptimality of Kung's algorithm) Used as a method for Hankel structured low rank approximation, Kung's algorithm is suboptimal. The reason for this is that the factorization

$$\mathcal{H}_{i,j}(\sigma H) \approx \Gamma \Delta,$$

performed by the singular value decomposition is unstructured low rank approximation and unless the data are exact, Γ and Δ are not extended observability and controllability matrices, respectively. As a result, the shift equations (SE₁) and (SE₂) do not have solutions and Kung's algorithm computes an approximate solution in the least squares sense.

Note 3.5 (Unstructured vs. structure enforcing methods) The two levels of approximation:

1. approximation of the Hankel matrix, constructed from the data, by unstructured low rank matrix, and
2. computation of an approximate solution of a system of linear equations for the parameter estimate by the ordinary least squares method

are common to the subspace methods. In contrast, methods based on Hankel structured low rank approximation do not involve approximation on the second stage (model parameter computation). By construction, the approximation computed on the first step is guaranteed to have an exact solution on the second step.

Note 3.6 (Interpretation of Kung’s algorithm as a finite-time balanced model reduction) Kung’s algorithm computes a realization in a finite time $\min(i, j)$ balanced bases. In the case of noisy data or data obtained from a high order model, the computed realization is obtain by truncation of the realization, using the user specified state dimension n or tolerance tol . This justifies Kung’s algorithm as a data-driven method for finite-time balanced model reduction. (The term data-driven refers to the fact that a model of the full order system is not used.) The link between Kung’s algorithm and model reduction further justifies the choice (i, j) on p. 75 for the shape of the Hankel matrix—the choice (i, j) maximizes the horizon for the finite-time balanced realization.

Computation of the Impulse Response from General Trajectory

In the realization problem the given data are a special trajectory—the impulse response of the model. Therefore, the realization problem is a special exact identification problem. In this section, the general exact identification problem:

$$w_d \xrightarrow{\text{exact identification}} \mathcal{B}_{\text{mpum}}(w_d)$$

is solved by reducing it to the realization problem:

$$w_d \xrightarrow{\text{impulse response computation (w2h)}} H_{\text{mpum}} \xrightarrow{\text{realization (h2ss)}} \mathcal{B}_{\text{mpum}}(w_d). \quad (w_d \mapsto H \mapsto \hat{\mathcal{B}})$$

First, the impulse response H_{mpum} of the most powerful unfalsified model $\mathcal{B}_{\text{mpum}}(w_d)$ is computed from the given general trajectory w_d . Then, an input/state/output representation of $\mathcal{B}_{\text{mpum}}(w_d)$ is computed from H_{mpum} by a realization algorithm, e.g., Kung’s algorithm.

The key observation in finding an algorithm for the computation of the impulse response is that the image of the Hankel matrix $\mathcal{H}_{i,j}(w_d)$, with $j > qi$, constructed

from the data w_d is the restriction $\mathcal{B}_{\text{mpum}}(w_d)|_{[1,i]}$ of the most powerful unfalsified model on the interval $[1, i]$, i.e.,

$$\mathcal{B}_{\text{mpum}}(w_d)|_{[1,i]} = \text{span}(\mathcal{H}_i(w_d)). \quad (\text{DD})$$

Therefore, any i -samples long trajectory w of $\mathcal{B}_{\text{mpum}}(w_d)$ can be constructed as a linear combination of the columns of the Hankel matrix

$$w = \mathcal{H}_i(w_d)g$$

for some vector g .

As in the realization problem, in what follows, the default input/output partitioning $w = \text{col}(u, y)$ is assumed. Let e_k be the k th column of the $m \times m$ identity matrix and δ be the unit pulse function. The impulse response is a matrix valued trajectory, the columns of which are the m trajectories corresponding to zero initial conditions and inputs $e_1\delta, \dots, e_m\delta$. Therefore, the problem of computing the impulse response is reduced to the problem of finding a vector g_k , such that $\mathcal{H}_i(w_d)g_k$ is of the form $(e_k\delta, h_k)$, where $h_k(\tau) = 0$, for all $\tau < 0$. The identically zero input/output trajectory for negative time (in the past) implies that the response from time zero on (in the future) is the impulse response.

Let 1 be the lag of the most powerful unfalsified model $\mathcal{B}_{\text{mpum}}(w_d)$. In order to describe the construction of a vector g_k that achieves the impulse response h_k , define the “past” Hankel matrix

$$\mathbf{H}_p := \mathcal{H}_{1,j}(w_d), \quad \text{where } j := T - (1 + i)$$

and the “future” input and output Hankel matrices

$$\mathbf{H}_{f,u} := \mathcal{H}_{i,j}(\sigma^1 u_d) \quad \text{and} \quad \mathbf{H}_{f,y} := \mathcal{H}_{i,j}(\sigma^1 y_d).$$

79 *(define \mathbf{H}_p , $\mathbf{H}_{f,u}$, and $\mathbf{H}_{f,y}$ 79)* (80b)

```

j = T - (1 + i);
Hp = blkhank(w, 1, j);
Hfu = blkhank(u(:, (1 + 1):end), i, j);
Hfy = blkhank(y(:, (1 + 1):end), i, j);

```

Uses blkhank 25b.

With these definitions, a vector g_k that achieves the impulse response h_k must satisfy the system of linear equations

$$\begin{bmatrix} \mathbf{H}_p \\ \mathbf{H}_{f,u} \end{bmatrix} g_k = \begin{bmatrix} 0_{q \times 1} \\ e_k \\ 0_{(t-1)m \times 1} \end{bmatrix}. \quad (*)$$

By construction, any solution g_k of $(*)$ is such that

$$\mathbf{H}_{f,y} g_k = h_k.$$

Choosing the least norm solution as a particular solution and using matrix notation, the first i samples of the impulse response are given by

$$H = \mathbf{H}_{f,y} \begin{bmatrix} \mathbf{H}_p \\ \mathbf{H}_{f,u} \end{bmatrix}^+ \begin{bmatrix} 0_{q1 \times m} \\ I_m \\ 0_{(i-1)m \times m} \end{bmatrix},$$

where A^+ is the pseudo-inverse of A .

80a \langle data driven computation of the impulse response 80a $\rangle \equiv$ (80b)
`wini_uf = [zeros(1 * q, m); eye(m); zeros((i - 1) * m, m)];`
`h_ = Hfy * pinv([Hp; Hfu]) * wini_uf;`

We have the following function for computation of the impulse response of the most powerful unfalsified model from data:

80b $\langle w \mapsto H$ 80b $\rangle \equiv$
`function h = w2h(w, m, n, i)`
`reshape w and define q, T 26d), p = q - m; l = ceil(n / p);`
`u = w(1:m, :); y = w((m + 1):q, :);`
`define Hp, Hf,u, and Hf,y 79)`
`data driven computation of the impulse response 80a)`
`for ii = 1:i`
`h(:, :, ii) = h_(((ii - 1) * p + 1):(ii * p), :);`
`end`

Defines:

w2h, used in chunk 81.

As in the case of the realization problem, when the data are noisy or generated by a high order model (higher than the specified order n), `w2h` computes a (suboptimal) approximation.

Using the functions `w2h` and `h2ss`, we obtain a method for exact identification ($w_d \mapsto H \mapsto \mathcal{B}$)

80c \langle Most powerful unfalsified model in $\mathcal{L}_m^{q,n}$ 80c $\rangle \equiv$ 80d \triangleright
`function sys = w2h2ss(w, m, n, Th, i, j)`
`reshape w and define q, T 26d), p = q - m;`

Defines:

w2h2ss, used in chunks 116 and 118.

The optional parameter `Th` specifies the number of samples of the impulse response, to be computed on the first step $w_d \mapsto H$. The default value is the minimum number (T_{\min}), defined on p. 75.

80d \langle Most powerful unfalsified model in $\mathcal{L}_m^{q,n}$ 80c $\rangle + \equiv$ \langle 80c 81 \rangle
`if ~exist('Th') | isempty(Th)`
`Th = ceil(n / p) + ceil(n / m) + 1;`
`end`

The dimensions i and j of the Hankel matrix $\mathcal{H}_{i,j}(\sigma H)$, to be used at the realization step $w_d \mapsto H$, are also optional input parameters. With exact data, their values are irrelevant as long as the Hankel matrix has the minimum number n of rows and columns. However, with noisy data, the values of `Th`, `i`, and `j` affect the

approximation. Empirical results suggest that the default choice (i, j) gives best approximation.

81 $\langle \text{Most powerful unfalsified model in } \mathcal{L}_m^{q,n} \text{ 80c} \rangle + \equiv \quad \triangleleft 80d$
 $T = \text{Th}; \langle \text{dimension of the Hankel matrix 75a} \rangle$
 $\text{sys} = \text{h2ss}(\text{w2h}(w, m, n, \text{Th}), n, [], i, j);$
 Uses `h2ss` 76d and `w2h` 80b.

3.2 Algorithms Based on Local Optimization

Consider the structured low rank approximation problem

$$\text{minimize over } \hat{p} \quad \|p - \hat{p}\|_W \quad \text{subject to} \quad \text{rank}(\mathcal{S}(\hat{p})) \leq r. \quad (\text{SLRA})$$

As discussed in Sect. 1.4, different methods for solving the problem are obtained by choosing different combinations of

- rank parametrization, and
- optimization method.

In this section, we choose the kernel representation for the rank constraint

$$\text{rank}(\mathcal{S}(\hat{p})) \leq r \iff \text{there is } R \in \mathbb{R}^{(m-r) \times m}, \quad \text{such that}$$

$$R\mathcal{S}(\hat{p}) = 0 \text{ and } RR^\top = I_{m-r}, \quad (\text{rank}_R)$$

and the variable projections approach (in combination with standard methods for nonlinear least squares optimization) for solving the resulting parameter optimization problem.

The developed method is applicable for the general affinely structured and weighted low rank approximation problem (SLRA). The price paid for the generality, however, is lack of efficiency compared to specialized methods exploiting the structure of the data matrix $\mathcal{S}(p)$ and the weight matrix W .

In two special cases—single input single output linear time-invariant system identification and computation of approximate greatest common divisor—efficient methods are described later in the chapter. The notes and references section links the presented material to state-of-the-art methods for structured low rank approximation. Efficient methods for weighted low rank approximation problems are developed in Chap. 5.

Representing the constraint of (SLRA) in the kernel form (rank_R), leads to the double minimization problem

$$\text{minimize over } R \quad f(R) \quad \text{subject to} \quad RR^\top = I_{m-r}, \quad (\text{SLRA}_R)$$

$$\text{where } f(R) := \min_{\hat{p}} \|p - \hat{p}\| \quad \text{subject to} \quad R\mathcal{S}(\hat{p}) = 0.$$

The inner minimization (computation of $f(R)$) is over the correction \widehat{p} and the outer minimization is over the model parameter $R \in \mathbb{R}^{(m-r) \times m}$. The inner minimization problem can be given the interpretation of projecting the columns of $\mathcal{S}(p)$ onto the model $\mathcal{B} := \ker(R)$, for a given matrix R . However, the projection depends on the parameter R , which is the variable in the outer minimization problem.

For affine structures \mathcal{S} , the constraint $R\mathcal{S}(\widehat{p}) = 0$ is bilinear in the optimization variables R and \widehat{p} . Then, the evaluation of the cost function f for the outer minimization problem is a linear least norm problem. Direct solution has computational complexity $O(n_p^3)$, where n_p is the number of structure parameters. Exploiting the structure of the problem (inherited from \mathcal{S}), results in computational methods with cost $O(n_p^2)$ or $O(n_p)$, depending on the type of structure. For a class of structures, which includes block Hankel, block Toeplitz, and block Sylvester ones, efficient $O(n_p)$ cost function evaluation can be done by Cholesky factorization of a block-Toeplitz banded matrix.

A Method for Affinely Structured Problems

Structure Specification

The general affine structure

$$\mathcal{S}(\widehat{p}) = S_0 + \sum_{k=1}^{n_p} S_k \widehat{p}_k \quad (\mathcal{S}(\widehat{p}))$$

is specified by the matrices $S_0, S_1, \dots, S_{n_p} \in \mathbb{R}^{m \times n}$. These data are represented in the code by an $m \times n$ matrix variable `s0`, corresponding to the matrix S_0 , and an $mn \times n_p$ matrix variable `bfs`, corresponding to the matrix

$$\mathbf{S} := [\text{vec}(S_1) \cdots \text{vec}(S_{n_p})] \in \mathbb{R}^{mn \times n_p}.$$

With this representation, the sum in $(\mathcal{S}(\widehat{p}))$ is implemented as a matrix–vector product:

$$\text{vec}(\mathcal{S}(\widehat{p})) = \text{vec}(S_0) + \mathbf{S}\widehat{p}, \quad \text{or} \quad \mathcal{S}(\widehat{p}) = S_0 + \text{vec}^{-1}(\mathbf{S}\widehat{p}). \quad (\mathbf{S})$$

82 `((S0, S, p̂) ↦ D̂ = S(p̂) 82) ≡
dh = s0 + reshape(bfs * ph, m, n);`

Note 3.7 In many applications the matrices S_k are sparse, so that, for efficiency, they can be stored and manipulated as sparse matrices.

A commonly encountered special case of an affine structure is

$$[\mathcal{S}(\widehat{p})]_{ij} = \begin{cases} S_{0,ij}, & \text{if } S_{ij} = 0 \\ \widehat{p}_{S_{ij}} & \text{otherwise} \end{cases} \quad \text{for some } S_{ij} \in \{0, 1, \dots, n_p\}^{m \times n}, \quad (\mathbf{S})$$

or, written more compactly,

$$[\mathcal{S}(\widehat{p})]_{ij} = S_{0,ij} + \widehat{p}_{\text{ext},S_{ij}}, \quad \text{where } \widehat{p}_{\text{ext}} := \begin{bmatrix} 0 \\ \widehat{p} \end{bmatrix}.$$

In (S), each element of the structured matrix $\mathcal{S}(p)$ is equal to the corresponding element of the matrix S_0 or to the S_{ij} th element of the parameter vector p . The structure is then specified by the matrices S_0 and S . Although (S) is a special case of the general affine structure ($\mathcal{S}(\widehat{p})$), it covers all linear modeling problems considered in this book and will therefore be used in the implementation of the solution method.

In the implementation of the algorithm, the matrix S corresponds to a variable `tts` and the extended parameter vector p_{ext} corresponds to a variable `pext`. Since in MATLAB indices are positive integers (zero index is not allowed), in all indexing operations of `pext`, the index is incremented by one. Given the matrices S_0 and S , specifying the structure, and a structure parameter vector \widehat{p} , the structured matrix $\mathcal{S}(\widehat{p})$ is constructed by

$$\begin{aligned} \langle (S_0, S, \widehat{p}) \mapsto \widehat{D} = \mathcal{S}(\widehat{p}) \text{ 83a} \rangle \equiv & \quad (85d \text{ 98a}) \\ \text{pext} = [0; \text{ph}(:)]; \text{dh} = \text{s0} + \text{pext}(\text{tts} + 1); \end{aligned}$$

The matrix dimensions m , n , and the number of parameters n_p are obtained from S as follows:

$$\begin{aligned} \langle S \mapsto (m, n, n_p) \text{ 83b} \rangle \equiv & \quad (84c \text{ 85e } 97 \text{ 98c}) \\ [m, n] = \text{size}(\text{tts}); n_p = \max(\max(\text{tts})); \end{aligned}$$

The transition from the specification of (S) to the specification in the general affine case ($\mathcal{S}(\widehat{p})$) is done by

$$\begin{aligned} \langle S \mapsto \mathbf{S} \text{ 83c} \rangle \equiv & \quad (84c \text{ 85e } 98c) \\ \text{vec_tts} = \text{tts}(:); \text{NP} = 1:n_p; \\ \text{bfs} = \text{vec_tts}(:, \text{ones}(1, n_p)}) == \text{NP}(\text{ones}(m * n, 1), :); \end{aligned}$$

Conversely, for a linear structure of the type (S), defined by \mathbf{S} (and m, n), the matrix S is constructed by

$$\langle \mathbf{S} \mapsto S \text{ 83d} \rangle \equiv \text{tts} = \text{reshape}(\text{bfs} * (1:n_p)', m, n);$$

In most applications that we consider, the structure \mathcal{S} is linear, so that `s0` is an optional input argument to the solvers with default value the zero matrix.

$$\langle \text{default } s_0 \text{ 83e} \rangle \equiv \quad (84c \text{ 85e } 97 \text{ 98c})$$

```
if ~exist('s0', 'var') | isempty(s0), s0 = zeros(m, n); end
```

The default weight matrix W in the approximation criterion is the identity matrix.

$$\langle \text{default weight matrix 83f} \rangle \equiv \quad (84c \text{ 85e } 98c)$$

```
if ~exist('w', 'var') | isempty(w), w = eye(n_p); end
```

Minimization over \widehat{p}

In order to solve the optimization problem (SLRA), we change variables

$$\widehat{p} \mapsto \Delta p = p - \widehat{p}.$$

Then, the constraint is written as a system of linear equations with unknown Δp :

$$\begin{aligned}
 R\mathcal{S}(\hat{p}) = 0 &\iff R\mathcal{S}(p - \Delta p) = 0 \\
 &\iff R\mathcal{S}(p) - R\mathcal{S}(\Delta p) + RS_0 = 0 \\
 &\iff \text{vec}(R\mathcal{S}(\Delta p)) = \text{vec}(R\mathcal{S}(p)) + \text{vec}(RS_0) \\
 &\iff \underbrace{[\text{vec}(RS_1) \cdots \text{vec}(RS_{n_p})]}_{G(R)} \Delta p = \underbrace{G(R)p + \text{vec}(RS_0)}_{h(R)} \\
 &\iff G(R)\Delta p = h(R).
 \end{aligned}$$

84a *(form $G(R)$ and $h(R)$ 84a)* \equiv (84c)

```

g = reshape(R * reshape(bfs, m, n * np),
             size(R, 1) * n, np);
h = g * p + vec(R * s0);

```

The inner minimization in (SLRA)_R with respect to the new variable Δp is a linear least norm problem

$$\text{minimize over } \Delta p \quad \|\Delta p\|_W \quad \text{subject to} \quad G(R)\Delta p = h(R) \quad (\text{LNP})$$

and has the analytic solution

$$\Delta p^*(R) = W^{-1}G^\top(R)(G(R)W^{-1}G^\top(R))^{-1}h(R).$$

84b *(solve the least-norm problem 84b)* \equiv (84c)

```

dp = inv_w * g' * (pinv(g * inv_w * g') * h);

```

Finally, the cost function to be minimized over the parameter R is

$$f(R) = \|\Delta p^*(R)\|_W = \sqrt{\Delta p^{*\top}(R)W\Delta p^*(R)}.$$

The function f corresponds to the data–model misfit function in data modeling problems and will be referred to as the structured low rank approximation misfit.

84c *(Structured low rank approximation misfit 84c)* \equiv

```

function [M, ph] = misfit_slra(R, tts, p, w, s0, bfs, inv_w)
⟨S ↦ (m, n, n_p) 83b⟩
⟨default s0 83e⟩, ⟨default weight matrix 83f⟩
if ~exist('bfs'), ⟨S ↦ S 83c⟩, end
⟨form G(R) and h(R) 84a⟩
if ~exist('inv_w'), inv_w = inv(w); end
⟨solve the least-norm problem 84b⟩
M = sqrt(dp' * w * dp); ph = p - dp;

```

Defines:

`misfit_slra`, used in chunk 85.

Minimization over R

General purpose constrained optimization methods are used for the outer minimization problem in (SLRA $_R$), i.e., the minimization of f over R , subject to the constraint $RR^T = I$. This is a non-convex optimization problem, so that there is no guarantee that a globally optimal solution is found.

```
85a <set optimization solver and options 85a>≡ (85c 88b 191d)
    prob = optimset();
    prob.solver = 'fmincon';
    prob.options = optimset('disp', 'off');
```

```
85b <call optimization solver 85b>≡ (85c 88b 191d)
    [x, fval, flag, info] = fmincon(prob); info.M = fval;
```

```
85c <nonlinear optimization over R 85c>≡ (85e)
    <set optimization solver and options 85a>
    prob.x0 = Rini; inv_w = inv(w);
    prob.objective = ...
        @(R) misfit_slra(R, tts, p, w, s0, bfs, inv_w);
    prob.nonlcon = @(R) deal([], [R * R' - eye(size(R, 1))]);
    <call optimization solver 85b>, R = x;
```

Uses `misfit_slra` 84c.

If not specified, the initial approximation is computed from a heuristic that ignores the structure and replaces the weighted norm by the Frobenius norm, so that the resulting problem can be solved by the singular value decomposition (function `lra`).

```
85d <default initial approximation 85d>≡ (85e)
    if ~exist('Rini') | isempty(Rini)
        ph = p; <(S0, S,  $\hat{p}$ ) $\mapsto$   $\hat{D} = \mathcal{S}(\hat{p})$  83a>, Rini = lra(dh, r);
    end
```

Uses `lra` 64.

The resulting function is:

```
85e <Structured low rank approximation 85e>≡
    function [R, ph, info] = slra(tts, p, r, w, s0, Rini)
    <(S $\mapsto$  (m, n, np) 83b>, <(S $\mapsto$   $\mathbf{S}$  83c>
    <default s0 83e>, <default weight matrix 83f>
    <default initial approximation 85d>
    <nonlinear optimization over R 85c>
    if nargin > 1,
        [M, ph] = misfit_slra(R, tts, p, w, s0, bfs, inv_w);
    end
```

Defines:

`slra`, used in chunks 100e and 108a.

Uses `misfit_slra` 84c.

Exercise 3.8 Use `slra` and `r2x` to solve approximately an overdetermined system of linear equations $AX \approx B$ in the least squares sense. Check the accuracy of the answer by using the analytical expression.

Exercise 3.9 Use `slra` to solve the basic low rank approximation problem

$$\text{minimize over } \widehat{D} \quad \|D - \widehat{D}\|_F \quad \text{subject to} \quad \text{rank}(\widehat{D}) \leq m.$$

(Unstructured approximation in the Frobenius norm.) Check the accuracy of the answer by the Eckart–Young–Mirsky theorem (`lra`).

Exercise 3.10 Use `slra` to solve the weighted low rank approximation problem (unstructured approximation in the weighted norm ($\|\cdot\|_W$), defined on p. 61). Check the accuracy of the answer in the special case of two-sided weighted low rank approximation, using Theorem 2.29.

Algorithms for Linear System Identification

Approximate linear system identification problems can be solved as equivalent Hankel structured low rank approximation problems. Therefore, the function `slra`, implemented in the previous section can be used for linear time-invariant system identification. This approach is developed in Sect. 4.3.

In this section an alternative approach for approximate system identification that is motivated from a system theoretic view of the problem is used. The Hankel structure in the problem is exploited, which results in efficient computational methods.

Misfit Computation

Consider the misfit between the data w_d and a model \mathcal{B}

$$\text{misfit}(w_d, \mathcal{B}) := \min_{\widehat{w}} \|w_d - \widehat{w}\|_2 \quad \text{subject to} \quad \widehat{w} \in \mathcal{B}. \quad (\text{misfit } \mathcal{L}_{m,1})$$

Geometrically, $\text{misfit}(w_d, \mathcal{B})$ is the orthogonal projection of w_d on \mathcal{B} . Assuming that \mathcal{B} is controllable, \mathcal{B} has a minimal image representation $\text{image}(P(\sigma))$. In terms of the parameter P , the constraint $\widehat{w} \in \mathcal{B}$ becomes $\widehat{w} = P(\sigma)\ell$, for some latent variable ℓ . In a matrix form,

$$\widehat{w} = \mathcal{T}_T(P)\ell,$$

where

$$\mathcal{T}_T(P) := \begin{bmatrix} P_0 & P_1 & \cdots & P_1 & & & \\ & P_0 & P_1 & \cdots & P_1 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & P_0 & P_1 & \cdots & P_1 \end{bmatrix} \in \mathbb{R}^{q \times (T+1)}. \quad (\mathcal{T})$$

87a *(Toeplitz matrix constructor 87a)*≡

```
function TP = blktoep(P, T)
[q, l1] = size(P); l = l1 - 1; TP = zeros(T * q, T + 1);
ind = 1 + (0:T - 1) * q * (T + 1);
for i = 1:q
    for j = 1:l1
        TP(ind + (i - 1) + (j - 1) * (T * q)) = P(i, j);
    end
end
```

Defines:

blktoep, used in chunk 87b.

The misfit computation problem (misfit $\mathcal{L}_{m,1}$) is equivalent to the standard linear least squares problem

$$\text{minimize over } \ell \quad \|w_d - \mathcal{T}_T(P)\ell\|, \quad (\text{misfit}_P)$$

so that the solution, implemented in the functions `misfit_asiso`, is

$$\hat{w} = \mathcal{T}_T(P)(\mathcal{T}_T^\top(P)\mathcal{T}_T(P))^{-1}\mathcal{T}_T^\top(P)w_d.$$

87b *(dist(w_d, \mathcal{B}) 87b)*≡

```
[M, wh] = misfit_asiso(w, P)
try, [M, wh] = misfit_asiso_efficient(w, P);
catch
    (reshape w and define q, T 26d)
    TP = blktoep(P, T); wh = reshape(TP * (TP \ w(:)), 2, T);
    M = norm(w - wh, 'fro');
end
```

Defines:

misfit_asiso, used in chunks 88b, 116c, 118, and 127c.

Uses blktoep 87a.

First, an efficient implementation (`misfit_asiso_efficient`) of the misfit computation function, exploiting the banded Toeplitz structure of the matrix $\mathcal{T}_T(P)$, is attempted. `misfit_asiso_efficient` calls a function of the SLICOT library to carry out the computation and requires a mex file, which is platform dependent. If a mex file is not available, the computation is reverted to solution of (`misfitP`) without exploiting the structure of $\mathcal{T}_T(P)$ (MATLAB backslash operator).

Note 3.11 (Misfit computation by Kalman smoothing) The efficient implementation in `misfit_asiso_efficient` is based on structured linear algebra computations (Cholesky factorization of positive definite banded Toeplitz matrix). The computational methods implemented in the SLICOT library use the generalized Schur algorithm and have computational complexity $O(n_p)$.

An alternative approach, which also results in $O(n_p)$ methods, is based on the system theoretic interpretation of the problem: equivalence between misfit computation and Kalman smoothing. In this latter approach, the computation is done by a Riccati type recursion.

Misfit Minimization

Consider the misfit minimization problem

$$\widehat{\mathcal{B}}^* := \arg \min_{\mathcal{B}} \text{misfit}(w_d, \mathcal{B}) \quad \text{subject to} \quad \widehat{\mathcal{B}} \in \mathcal{L}_{m,1}^q. \quad (\text{SYSID})$$

Using the representation $\mathcal{B} = \text{image}(P)$, (SYSID) is equivalent to

$$\begin{aligned} & \text{minimize} \quad \text{over } P \in \mathbb{R}^{q(1+1) \times m} \quad \text{misfit}(w_d, \text{image}(P(\sigma))) \\ & \text{subject to} \quad P^\top P = I_m, \end{aligned} \quad (\text{SYSID}_P)$$

which is a constrained nonlinear least squares problem.

88a *(Single input single output system identification 88a)*≡

```
function [sysh, wh, info] = ident_asiso(w, n, sys)
if ~exist('sys', 'var')
    (suboptimal approximate single input single output system identification 88c)
else
    (TF) ↦ P 88e
end
(misfit minimization 88b)
```

Defines:

ident_asiso, used in chunks 89g and 127d.

Optimization Toolbox is used for performing the misfit minimization.

88b *(misfit minimization 88b)*≡ (88a)

```
    (set optimization solver and options 85a)
    prob.x0 = P;
    prob.objective = @(P) misfit_asiso(w, P);
    prob.nonlcon = @(P) deal([], [P(1, :) * P(1, :)' - 1]);
    (call optimization solver 85b), P = x;
    (P ↦ (TF) 88d) sysh = sys;
    if nargout > 1, [M, wh] = misfit_asiso(w, P); end
```

Uses misfit_asiso 87b.

The initial approximation is computed from a relaxation ignoring the structure constraint:

88c *(suboptimal approximate single input single output system identification 88c)*≡ (88a 127c)

```
R = lra(blkhank(w, n + 1), 2 * n + 1); (R ↦ P 89a)
```

Uses blkhank 25b and lra 64.

The solution obtained by the optimization solver is an image representation of a (locally) optimal approximate model $\widehat{\mathcal{B}}^*$. In the function `ident_asiso`, the image representation is converted to a transfer function representation as follows:

88d *(P ↦ (TF) 88d)*≡ (88b 89c)

```
p = fliplr(P(1, :)); q = fliplr(P(2, :)); sys = tf(q, p, -1);
```

The reverse transformation

88e *((TF) ↦ P 88e)*≡ (88a 89a 116c 118)

```
[q, p] = tfdata(tf(sys), 'v'); P = zeros(2, length(p));
P(1, :) = fliplr(p);
P(2, :) = fliplr([q zeros(length(p) - length(q))]);
```

is used when an initial approximation is specified by a transfer function representation. When no initial approximation is supplied, a default one is computed by unstructured low rank approximation, which produces a kernel representation of the model. Transition from kernel to image representation is done indirectly by passing through a transfer function representation:

$$\begin{aligned} 89a \quad \langle R \mapsto P \ 89a \rangle &\equiv & (88c) \\ \langle R \mapsto (TF) \ 89b \rangle &\langle (TF) \mapsto P \ 88e \rangle \end{aligned}$$

where

$$\begin{aligned} 89b \quad \langle R \mapsto (TF) \ 89b \rangle &\equiv & (89a) \\ q &= - \text{fliplr}(R(1:2:\text{end})); \ p = \text{fliplr}(R(2:2:\text{end})); \\ \text{sys} &= \text{tf}(q, \ p, \ -1); \end{aligned}$$

For later use, next, we define also the reverse mapping:

$$\begin{aligned} 89c \quad \langle P \mapsto R \ 89c \rangle &\equiv \\ \langle P \mapsto (TF) \ 88d \rangle &\langle (TF) \mapsto R \ 89d \rangle \end{aligned}$$

where

$$\begin{aligned} 89d \quad \langle (TF) \mapsto R \ 89d \rangle &\equiv & (89c) \\ [q, \ p] &= \text{tfdata}(\text{tf}(\text{sys}), \ 'v'); \ R = \text{zeros}(1, \ \text{length}(p) * 2); \\ R(1:2:\text{end}) &= - \text{fliplr}([q \ \text{zeros}(\text{length}(p) - \text{length}(q))]); \\ R(2:2:\text{end}) &= \text{fliplr}(p); \end{aligned}$$

Numerical Example

The function `ident_asiso` is applied on data obtained in the errors-in-variables setup (EIV) on p. 70. The true data generating model is a random single input single output linear time-invariant system and the true data w_0 are a random trajectory of that system. In order to make the results reproducible, in all simulations, we first initialize the random number generator:

$$\begin{aligned} 89e \quad \langle \text{initialize the random number generator } 89e \rangle &\equiv & (89f \ 100a \ 103a \ 110a \ 114b \ 116c \ 118 \ 121a) \\ \text{randn}('seed', \ 0); \ \text{rand}('seed', \ 0); & & (125a \ 127a \ 142c \ 162a \ 192a \ 212a) \end{aligned}$$

$$\begin{aligned} 89f \quad \langle \text{Test ident_asiso } 89f \rangle &\equiv & 89g \triangleright \\ \langle \text{initialize the random number generator } 89e \rangle & & \\ \text{sys0} &= \text{drss}(n); \ \text{xini0} = \text{rand}(n, \ 1); \\ \text{u0} &= \text{rand}(T, \ 1); \ \text{y0} = \text{lsim}(\text{sys0}, \ \text{u0}, \ 1:T, \ \text{xini0}); \\ \text{w} &= [\text{u0} \ \text{y0}] + s * \text{randn}(T, \ 2); \end{aligned}$$

Defines:

`test_ident_asiso`, used in chunk 90b.

The optimal approximation obtained by `ident_asiso`

$$\begin{aligned} 89g \quad \langle \text{Test ident_asiso } 89f \rangle &+ \equiv & \triangleleft 89f \ 90a \triangleright \\ [\text{sysh}, \ \text{wh}, \ \text{info}] &= \text{ident_asiso}(w, \ n); \ \text{info} \\ \text{Uses ident_asiso } &88a. \end{aligned}$$

is verified by comparing it with the approximation obtained by the function `slra` (called by the wrapper function `ident_eiv`, see Sect. 4.3)

```
90a (Test ident_siso 89f) ≡ <89g
    [sysh_, wh_, info_] = ident_eiv(w, 1, n); info_
    norm(sysh - sysh_)
```

Uses `ident_eiv` 115b.

In a specific example

```
90b (Compare ident_siso and ident_eiv 90b) ≡
    n = 2; T = 20; s = 0.1; test_ident_siso
```

Uses `test_ident_siso` 89f.

the norm of the difference between the two computed approximations is of the order of magnitude of the convergence tolerance used by the optimization solvers. This shows that the methods converged to the same locally optimal solution.

Computation of an Approximate Greatest Common Divisor

Associated with a polynomial

$$p(z) := p_0 + p_1z + \cdots + p_nz^n$$

of degree at most n is an $(n + 1)$ -dimensional coefficients vector

$$p := \text{col}(p_0, p_1, \dots, p_n) \in \mathbb{R}^{n+1}$$

and vice versa a vector $p \in \mathbb{R}^{n+1}$ corresponds to a polynomial p with degree at most n . With some abuse of notation, we denote the polynomial and its coefficients vector with the same letter. The intended meaning is understood from the context.

The coefficients vector of a polynomial, however, is not unique. (Scaling of the coefficients vector by a nonzero number result in the same polynomial.) In order to remove this nonuniqueness, we scale the coefficients, so that the highest power coefficient is equal to one (monic polynomial). In what follows, it is assumed that the coefficients are always scaled in this way.

The polynomials p and \hat{p} of degree n are “close” to each other if the distance measure

$$\text{dist}(p, \hat{p}) := \|p - \hat{p}\|_2$$

is “small”, i.e., if the norm of the coefficients vector of the error polynomial $\Delta p := p - \hat{p}$ is small.

Note 3.12 The distance $\text{dist}(p, \hat{p})$, defined above, might not be an appropriate distance measure in applications where the polynomial roots rather than coefficients are of primary interest. Polynomial roots might be sensitive (especially for high order polynomials) to perturbations in the coefficients, so that closeness of coefficients does not necessarily imply closeness of roots. Using the quadratic distance

measure in terms of the polynomial coefficients, however, simplifies the solution of the approximate common divisor problem defined next.

Problem 3.13 (Approximate common divisor) Given polynomials p and q , and a natural number d , smaller than the degrees of p and q , find polynomials \hat{p} and \hat{q} that have a common divisor c of degree d and minimize the approximation error

$$\text{dist}(\text{col}(p, q), \text{col}(\hat{p}, \hat{q})).$$

The polynomial c is an optimal (in the specified sense) approximate common divisor of p and q .

Note 3.14 The object of interest in solving Problem 3.13 is the approximate common divisor c . The approximating polynomials \hat{p} and \hat{q} are auxiliary variables introduced for the purpose of defining c .

Note 3.15 Problem 3.13 has the following system theoretic interpretation. Consider the single input single output linear time-invariant system $\mathcal{B} = \mathcal{B}_{i/o}(p, q)$. The system \mathcal{B} is controllable if and only if p and q have no common factor. Therefore, Problem 3.13 finds the nearest uncontrollable system $\hat{\mathcal{B}} = \mathcal{B}_{i/o}(\hat{p}, \hat{q})$ to the given system \mathcal{B} . The bigger the approximation error is, the more robust the controllability property of \mathcal{B} is. In particular, with zero approximation error, \mathcal{B} is uncontrollable.

Equivalent Optimization Problem

By definition, the polynomial c is a common divisor of \hat{p} and \hat{q} if there are polynomials u and v , such that

$$\hat{p} = uc \quad \text{and} \quad \hat{q} = vc. \quad (\text{GCD})$$

With the auxiliary variables u and v , Problem 3.13 becomes the following optimization problem:

$$\begin{aligned} & \text{minimize} \quad \text{over } \hat{p}, \hat{q}, u, v, \text{ and } c \quad \text{dist}(\text{col}(p, q), \text{col}(\hat{p}, \hat{q})) \\ & \text{subject to} \quad \hat{p} = uc, \quad \hat{q} = vc, \quad \text{and} \quad \text{degree}(c) = d. \end{aligned} \quad (\text{AGCD})$$

Theorem 3.16 *The optimization problem (AGCD) is equivalent to*

$$\text{minimize} \quad \text{over } c_0, \dots, c_{d-1} \in \mathbb{R} \quad f(c), \quad (\text{AGCD}')$$

where

$$f(c) := \text{trace}([p \ q]^T (I - \mathcal{T}_{n+1}(c)(\mathcal{T}_{n+1}^T(c)\mathcal{T}_{n+1}(c))^{-1}\mathcal{T}_{n+1}^T(c))[p \ q]),$$

and $\mathcal{T}_{n+1}(c)$ is an upper triangular Toeplitz matrix, defined in (\mathcal{T}) on p. 86.

The proof is given in Appendix B.

Compared with the original optimization problem (AGCD), in (AGCD'), the constraint and the auxiliary decision variables \widehat{p} , \widehat{q} , u , and v are eliminated. This achieves significant simplification from a numerical optimization point of view. The equivalent problem (AGCD') is a nonlinear least squares problem and can be solved by standard local optimization methods, see Algorithm 1.

Algorithm 1 Optimal approximate common divisor computation

Input: Polynomials p and q and a positive integer d .

- 1: Compute an initial approximation $c_{\text{ini}} \in \mathbb{R}^{d+1}$.
- 2: Execute a standard optimization algorithm for the minimization (AGCD') with initial approximation c_{ini} .
- 3: **if** \widehat{p} and \widehat{q} have to be displayed **then**
- 4: Solve for u and v the linear least squares problem in u and v

$$\begin{bmatrix} p \\ q \end{bmatrix} = \mathcal{T}_{n-d+1}(c) \begin{bmatrix} u \\ v \end{bmatrix}.$$

- 5: Define $\widehat{p} = u \star c$ and $\widehat{q} = v \star c$, where \star denotes discrete convolution.
- 6: **end if**

Output: The approximation $c \in \mathbb{R}^{d+1}$ found by the optimization algorithm upon convergence, the value of the cost function $f(c)$ at the optimal solution, and if computed \widehat{p} and \widehat{q} .

Since

$$f(c) = \text{dist}(\text{col}(p, q), \text{col}(\widehat{p}, \widehat{q}))$$

the value of the cost function $f(c)$ shows the approximation errors in taking c as an approximate common divisor of p and q . Optionally, Algorithm 1 returns a ‘‘certificate’’ \widehat{p} and \widehat{q} for c being an approximate common divisor of p and q with approximation error $f(c)$.

In order to complete Algorithm 1, next, we specify the computation of the initial approximation c_{ini} . Also, the fact that the analytic expression for $f(c)$ involves the highly structured matrix $\mathcal{T}_{n-d+1}(c)$ suggests that $f(c)$ (and its derivatives) can be evaluated efficiently.

Efficient Cost Function Evaluation

The most expensive operation in the cost function evaluation is solving the least squares problem

$$\begin{bmatrix} p \\ q \end{bmatrix} = \mathcal{T}_{n-d+1}(c) \begin{bmatrix} u \\ v \end{bmatrix}.$$

Since $\mathcal{T}_{n-d+1}(c)$ is an upper triangular, banded, Toeplitz matrix, this operation can be done efficiently. One approach is to compute efficiently the QR factorization of $\mathcal{T}_{n-d+1}(c)$, e.g., via the *generalized Schur algorithm*. Another approach is to solve the normal system of equations

$$\mathcal{T}_{n+1}^{\top}(c) \begin{bmatrix} p \\ q \end{bmatrix} = \mathcal{T}_{n-d+1}^{\top}(c) \mathcal{T}_{n-d+1}(c) \begin{bmatrix} u \\ v \end{bmatrix},$$

exploiting the fact that $\mathcal{T}_{n-\bar{d}+1}^\top(c)\mathcal{T}_{n-\bar{d}+1}(c)$ is banded and Toeplitz structured. The first approach is implemented in the function `MB02ID` from the `SLICOT` library.

Once the least squares problem is solved, the product

$$\mathcal{T}_{n-\bar{d}+1}(c) [u \ v] = [c \star u \ c \star v]$$

is computed efficiently by the *fast Fourier transform*. The resulting algorithm has computational complexity $O(n)$ operations. The first derivative $f'(c)$ can be evaluated also in $O(n)$ operations, so assuming that $\bar{d} \ll n$, the overall cost per iteration for Algorithm 1 is $O(n)$.

Initial Approximation

Suboptimal initial approximation can be computed by the singular value decomposition of the Sylvester (sub)matrix

$$\begin{aligned} \mathcal{R}_{\bar{d}}(p, q) &= [\mathcal{T}_{n-\bar{d}+1}^\top(p) \ \mathcal{T}_{n-\bar{d}+1}^\top(q)] \\ &= \begin{bmatrix} p_0 & & & & q_0 & & & & \\ p_1 & p_0 & & & q_1 & q_0 & & & \\ \vdots & p_1 & \ddots & & \vdots & q_1 & \ddots & & \\ p_n & \vdots & \ddots & p_0 & q_n & \vdots & \ddots & q_0 & \\ & p_n & & p_1 & q_n & q_1 & & & \\ & & \ddots & \vdots & & \ddots & \vdots & & \\ & & & p_n & & & q_n & & \end{bmatrix} \in \mathbb{R}^{(2n-\bar{d}+1) \times (2n-2\bar{d}+2)}. \end{aligned}$$

Since, the approximate greatest common divisor problem ([AGCD](#)) is a structured low rank approximation problem, ignoring the Sylvester structure constraint results a suboptimal solution method—unstructured low rank approximation. A suboptimal solution is therefore computable by the singular value decomposition.

The polynomial c is a common divisor of \hat{p} and \hat{q} if and only if there are polynomials u and v , such that

$$\hat{p}v = \hat{q}u. \tag{*}$$

With $\text{degree}(c) = \bar{d}$, the polynomial equation (*) is equivalent to the system of algebraic equations

$$\mathcal{R}_{\bar{d}}(\hat{p}, \hat{q}) \begin{bmatrix} v \\ -u \end{bmatrix} = 0.$$

The degree constraint for c is equivalent to

$$\text{degree}(u) = n - \bar{d},$$

or equivalently $u_{n-d+1} \neq 0$. Since u is defined up to a scaling factor, we impose the normalization $u_{n-d+1} = 1$. This shows that problem (AGCD) is equivalent to

$$\begin{aligned} & \text{minimize} \quad \text{over } \widehat{p}, \widehat{q} \in \mathbb{R}^{n+1} \text{ and } u, v \in \mathbb{R}^{n-d+1} \quad \|[p \ q] - [\widehat{p} \ \widehat{q}]\|_F \\ & \text{subject to} \quad \mathcal{R}_d(\widehat{p}, \widehat{q}) \begin{bmatrix} v \\ -u \end{bmatrix} = 0 \quad \text{and} \quad u_{n-d+1} = 1. \end{aligned} \quad (\text{AGCD}^*)$$

The approximate common factor c is not explicitly computed in (AGCD^{*}). Once the optimal u and v are known, however, c can be found from (GCD). (By construction these equations have unique solution). Alternatively, without using the auxiliary variables \widehat{p} and \widehat{q} , c can be computed from the least squares problem

$$\begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} c,$$

or in linear algebra notation

$$\begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} \mathcal{F}_{d+1}^\top(u) \\ \mathcal{F}_{d+1}^\top(v) \end{bmatrix} c. \quad (3.16)$$

Problem (AGCD^{*}) is a structured low rank approximation problem: it aims to find a Sylvester rank deficient matrix $\mathcal{R}_d(\widehat{p}, \widehat{q})$ as close as possible to a given matrix $\mathcal{R}_d(p, q)$ with the same structure. If p and q have no common divisor of degree d , $\mathcal{R}_d(p, q)$ is full rank so that an approximation is needed.

The (unstructured) low rank approximation problem

$$\begin{aligned} & \text{minimize} \quad \text{over } \widehat{D} \text{ and } r \quad \|\mathcal{R}_d(p, q) - \widehat{D}\|_F^2 \\ & \text{subject to} \quad \widehat{D}r = 0 \quad \text{and} \quad r^\top r = 1 \end{aligned} \quad (\text{LRA}_r)$$

has an analytic solution in terms of the singular value decomposition of $\mathcal{R}_d(p, q)$. The vector $r \in \mathbb{R}^{2(n-d+1)}$ corresponding to the optimal solution of (LRA_r) is equal to the right singular vector of $\mathcal{R}_d(p, q)$ corresponding to the smallest singular value. The vector $\text{col}(v, -u)$ composed of the coefficients of the approximate divisors v and $-u$ is up to a scaling factor (that enforces the normalization constraint $u_{n-d+1} = 1$) equal to r . This gives Algorithm 2 as a method for computing a suboptimal initial approximation.

Algorithm 2 Suboptimal approximate common divisor computation

Input: Polynomials p and q and an integer d .

- 1: Solve the unstructured low rank approximation problem (LRA_r).
- 2: Let $\text{col}(v, -u) := r$, where $u, v \in \mathbb{R}^{n-d+1}$.
- 3: Solve the least squares problem (3.16).

Output: The solution c of the least squares problem.

Numerical Examples

Implementation of the method for computation of approximate common divisor, described in this section, is available from the book's web page. We verify the results obtained by Algorithm 1 on examples from Zhi and Yang (2004) and Karmarkar and Lakshman (1998). Up to the number of digits shown the results match the ones reported in the literature.

Example 4.1 from Zhi and Yang (2004)

The given polynomials are

$$p(z) = (4 + 2z + z^2)(5 + 2z) + 0.05 + 0.03z + 0.04z^2$$

$$q(z) = (4 + 2z + z^2)(5 + z) + 0.04 + 0.02z + 0.01z^2$$

and an approximate common divisor c of degree $d = 2$ is sought. Algorithm 1 converges in 4 iteration steps with the following answer

$$c(z) = 3.9830 + 1.9998z + 1.0000z^2.$$

To this approximate common divisor correspond approximating polynomials

$$\widehat{p}(z) = 20.0500 + 18.0332z + 9.0337z^2 + 2.0001z^3$$

$$\widehat{q}(z) = 20.0392 + 14.0178z + 7.0176z^2 + 0.9933z^3$$

and the approximation error is

$$f(c) = \text{dist}^2(p, \widehat{p}) + \text{dist}^2(q, \widehat{q}) = 1.5831 \times 10^{-4}.$$

Example 4.2, Case 1, from Zhi and Yang (2004), originally given in Karmarkar and Lakshman (1998)

The given polynomials are

$$p(z) = (1 - z)(5 - z) = 5 - 6z + z^2$$

$$q(z) = (1.1 - z)(5.2 - z) = 5.72 - 6.3z + z^2$$

and an approximate common divisor c of degree $d = 1$ (a common root) is sought. Algorithm 1 converges in 6 iteration steps with the following answer

$$c(z) = -5.0989 + 1.0000z.$$

The corresponding approximating polynomials are

$$\widehat{p}(z) = 4.9994 - 6.0029z + 0.9850z^2$$

$$\widehat{q}(z) = 5.7206 - 6.2971z + 1.0150z^2$$

and the approximation error is $f(c) = 4.6630 \times 10^{-4}$.

3.3 Data Modeling Using the Nuclear Norm Heuristic

The nuclear norm heuristic leads to a semidefinite optimization problem, which can be solved by existing algorithms with provable convergence properties and readily available high quality software packages. Apart from theoretical justification and easy implementation in practice, formulating the problem as a semidefinite program has the advantage of flexibility. For example, adding regularization and affine inequality constraints in the data modeling problem still leads to semidefinite optimization problems that can be solved by the same algorithms and software as the original problem.

A disadvantage of using the nuclear norm heuristic is the fact that the number of optimization variables in the semidefinite optimization problem depends quadratically on the number of data points in the data modeling problem. This makes methods based on the nuclear norm heuristic impractical for problems with more than a few hundreds of data points. Such problems are “small size” data modeling problem.

Nuclear Norm Heuristics for Structured Low Rank Approximation

Regularized Nuclear Norm Minimization

The nuclear norm of a matrix is the sum of the matrix’s singular values

$$\|M\|_* = \text{sum of the singular values of } M. \quad (\text{NN})$$

Recall the mapping \mathcal{S} , see $(\mathcal{S}(\widehat{p}))$, on p. 82, from a structure parameter space \mathbb{R}^{2p} to the set of matrices $\mathbb{R}^{m \times n}$. Regularized nuclear norm minimization

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{p} && \|\mathcal{S}(\widehat{p})\|_* + \gamma \|p - \widehat{p}\| \\ & \text{subject to} && && G\widehat{p} \leq h \end{aligned} \quad (\text{NNM})$$

is a convex optimization problem and can be solved globally and efficiently. Using the fact

$$\|\widehat{D}\|_* < \mu \iff \frac{1}{2}(\text{trace}(U) + \text{trace}(V)) < \mu \quad \text{and} \quad \begin{bmatrix} U & \widehat{D}^\top \\ \widehat{D} & V \end{bmatrix} \succeq 0,$$

we obtain an equivalent problem

$$\begin{aligned} & \text{minimize} && \text{over } \hat{p}, U, V, \text{ and } \nu && \frac{1}{2}(\text{trace}(U) + \text{trace}(V)) + \gamma\nu \\ & \text{subject to} && \begin{bmatrix} U & \mathcal{S}(\hat{p})^\top \\ \mathcal{S}(\hat{p}) & V \end{bmatrix} \geq 0, && \text{(NNM')} \\ & && \|p - \hat{p}\| < \nu, \quad \text{and} \quad G\hat{p} \leq h, \end{aligned}$$

which can further be reduced to a semidefinite programming problem and solved by standard methods.

Structured Low Rank Approximation

Consider the affine structured low rank approximation problem (SLRA). Due to the rank constraint, this problem is non-convex. Replacing the rank constraint by a constraint on the nuclear norm of the affine structured matrix, however, results in a convex relaxation of (SLRA)

$$\text{minimize over } \hat{p} \quad \|p - \hat{p}\| \quad \text{subject to} \quad \|\mathcal{S}(\hat{p})\|_* \leq \mu. \quad \text{(RLRA)}$$

The motivation for this heuristic of solving (SLRA) is that approximation with an appropriately chosen bound on the nuclear norm tends to give solutions $\mathcal{S}(\hat{p})$ of low (but nonzero) rank. Moreover, the nuclear norm is the tightest relaxation of the rank function, in the same way ℓ_1 -norm is the tightest relaxation of the function mapping a vector to the number of its nonzero entries.

Problem (RLRA) can also be written in the equivalent unconstrained form

$$\text{minimize over } \hat{p} \quad \|\mathcal{S}(\hat{p})\|_* + \gamma\|p - \hat{p}\|. \quad \text{(RLRA')}$$

Here γ is a regularization parameter that is related to the parameter μ in (RLRA). The latter formulation of the relaxed affine structured low rank approximation problem is a regularized nuclear norm minimization problem (NNM').

Literate Programs

Regularized Nuclear Norm Minimization

The CVX package is used in order to automatically translate problem (NNM') into a standard convex optimization problem and solve it by existing optimization solvers.

97 *(Regularized nuclear norm minimization 97)* \equiv 98a \triangleright
`function [ph, info] = nucnrm(tts, p, gamma, nrm, w, s0, g, h)`
`⟨S ↦ (m, n, np) 83b⟩, ⟨default s0 83e⟩`

Defines:

`nucnrm`, used in chunks 98d and 104.

The code consists of definition of the optimization variables:

```
98a (Regularized nuclear norm minimization 97)+≡ <97 98b>
    cvx_begin sdp; cvx_quiet(true);
        variable U(n, n) symmetric;
        variable V(m, m) symmetric;
        variables ph(np) nu;
        ((S0, S,  $\hat{p}$ ) ↦  $\hat{D} = \mathcal{L}(\hat{p})$  83a)
```

and direct rewriting of the cost function and constraints of (NNM') in CVX syntax:

```
98b (Regularized nuclear norm minimization 97)+≡ <98a
    minimize( trace(U) / 2 + trace(V) / 2 + gamma * nu );
    subject to
        [U dh'; dh V] > 0;
        norm(w * (p - ph), nrm) < nu;
        if (nargin > 6) & ~isempty(g), g * ph < h; end
    cvx_end
```

The w argument specifies the norm ($\|\cdot\|_w$) and is equal to 1, 2, inf, or (in the case of a weighted 2-norm) a $n_p \times n_p$ positive semidefinite matrix. The info output variable is a structure with fields `optval` (the optimal value) and `status` (a string indicating the convergence status).

Structured Low Rank Approximation

The following function finds suboptimal solution of the structured low rank approximation problem by solving the relaxation problem (RLRA'). Affine structures of the type (S) are considered.

```
98c (Structured low rank approximation using the nuclear norm 98c)≡ 99a>
    function [ph, gamma] = slra_nn(tts, p, r, gamma, nrm, w, s0)
    (S ↦ (m, n, np) 83b), (S ↦ S 83c), (default s0 83e), (default weight matrix 83f)
    if ~exist('gamma', 'var'), gamma = []; end % default gamma
    if ~exist('nrm', 'var'), nrm = 2; end % default norm
```

Defines:

`slra_nn`, used in chunks 100d, 101b, 125b, and 127b.

If a parameter γ is supplied, the convex relaxation (RLRA') is completely specified and can be solved by a call to `nucnrm`.

```
98d (solve the convex relaxation (RLRA') for given  $\gamma$  parameter 98d)≡ (99)
    ph = nucnrm(tts, p, gamma, nrm, w, s0);
```

Uses `nucnrm` 97.

Large values of γ lead to solutions \hat{p} with small approximation error $\|p - \hat{p}\|_w$, but potentially high rank. Vice versa, small values of γ lead to solutions \hat{p} with low rank, but potentially high approximation error $\|p - \hat{p}\|_w$. If not given as an input argument, a value of γ which gives an approximation matrix $\mathcal{L}(\hat{p})$ with numerical

rank r can be found by bisection on an a priori given interval $[\gamma_{\min}, \gamma_{\max}]$. The interval can be supplied via the input argument `gamma`, in which case it is a vector $[\gamma_{\min}, \gamma_{\max}]$.

```

99a (Structured low rank approximation using the nuclear norm 98c) += <98c
    if ~isempty(gamma) & isscalar(gamma)
        <solve the convex relaxation (RLRA') for given  $\gamma$  parameter 98d>
    else
        if ~isempty(gamma)
            gamma_min = gamma(1); gamma_max = gamma(2);
        else
            gamma_min = 0; gamma_max = 100;
        end
        <parameters of the bisection algorithm 99c>
        <bisection on  $\gamma$  99b>
    end

```

On each iteration of the bisection algorithm, the convex relaxation (RLRA') is solved for γ equal to the mid point $(\gamma_{\min} + \gamma_{\max})/2$ of the interval and the numerical rank of the approximation $\mathcal{S}(\hat{p})$ is checked by computing the singular values of the approximation. If the numerical rank is higher than r , γ_{\max} is redefined to the mid point, so that the search continues on smaller values of γ (which have the potential of decreasing the rank). Otherwise, γ_{\min} is redefined to the mid point, so that the search continues on higher values of γ (which have the potential of increasing the rank). The search continues till the interval $[\gamma_{\min}, \gamma_{\max}]$ is sufficiently small or a maximum number of iterations is exceeded.

```

99b (bisection on  $\gamma$  99b) = (99a)
    iter = 0;
    while ((gamma_max - gamma_min)
           / gamma_max > rel_gamma_tol) ...
        & (iter < maxiter)
        gamma = (gamma_min + gamma_max) / 2;
        <solve the convex relaxation (RLRA') for given  $\gamma$  parameter 98d>
        sv = svd(ph(tts));
        if (sv(r + 1) / sv(1) > rel_rank_tol) ...
            & (sv(1) > abs_rank_tol)
            gamma_max = gamma;
        else
            gamma_min = gamma;
        end
        iter = iter + 1;
    end

```

The rank test and the interval width test involve a priori set tolerances.

```

99c (parameters of the bisection algorithm 99c) = (99a)
    rel_rank_tol = 1e-6; abs_rank_tol = 1e-6;
    rel_gamma_tol = 1e-5; maxiter = 20;

```

Examples

Unstructured and Hankel Structured Problems

The function `slra_nn` for affine structured low rank approximation by the nuclear norm heuristic is tested on randomly generated Hankel structured and unstructured examples. A rank deficient “true” data matrix is constructed, where the rank r_0 is a simulation parameter. In the case of a Hankel structure, the “true” structure parameter vector p_0 is generated as the impulse response (skipping the first sample) of a discrete-time random linear time-invariant system of order r_0 . This ensures that the “true” Hankel structured data matrix $\mathcal{S}(p_0)$ has the desired rank r_0 .

```
100a <Test slra_nn 100a>≡ 100b>
      (initialize the random number generator 89e)
      if strcmp(structure, 'hankel')
          np = m + n - 1; tts = hankel(1:m, m:np);
          p0 = impulse(drss(r0), np + 1); p0 = p0(2:end);
```

Defines:

`test_slra_nn`, used in chunk 102.

In the unstructured case, the data matrix is generated by multiplication of random $m \times r_0$ and $r_0 \times n$ factors of a rank revealing factorization of the data matrix $\mathcal{S}(p_0)$.

```
100b <Test slra_nn 100a>+≡ <100a 100c>
      else % unstructured
          np = m * n; tts = reshape(1:np, m, n);
          p0 = rand(m, r0) * rand(r0, n); p0 = p0(:);
      end
```

The data parameter p , passed to the low rank approximation function, is a noisy version of the true data parameter p_0 , where the additive noise’s standard deviation is a simulation parameter.

```
100c <Test slra_nn 100a>+≡ <100b 100d>
      e = randn(np, 1); p = p0 + nl * e / norm(e) * norm(p0);
```

The results obtained by `slra_nn`

```
100d <Test slra_nn 100a>+≡ <100c 100e>
      [ph, gamma] = slra_nn(tts, p, r0);
```

Uses `slra_nn` 98c.

are compared with the ones of alternative methods by checking the singular values of $\mathcal{S}(\hat{p})$, indicating the numerical rank, and the fitting error $\|p - \hat{p}\|$.

In the case of a Hankel structure, the alternative methods, being used, is Kung’s method (implemented in the function `h2ss`) and the method based on local optimization (implemented in the function `slra`).

```
100e <Test slra_nn 100a>+≡ <100d 101a>
      if strcmp(structure, 'hankel')
          sysh = h2ss([0; p], r0);
          ph2 = impulse(sysh, np + 1); ph2 = ph2(2:end);
          tts_ = hankel(1:(r0 + 1), (r0 + 1):np);
```

```

[Rh, ph3] = slra(tts_, p, r0);
sv = [svd(p(tts)) svd(ph(tts)) svd(ph2(tts))
      svd(ph3(tts))]
cost = [norm(p - p) norm(p - ph) ...
       norm(p - ph2) norm(p - ph3)]

```

Uses `h2ss` 76d and `slra` 85e.

In the unstructured case, the alternative method is basic low rank approximation (`lra`), which gives globally optimal result in this setup.

```

101a (Test slra_nn 100a)+≡ <100e
else % unstructured
    [Rh, Ph, dh] = lra(p(tts)', r0); dh = dh';
    sv = [svd(p(tts)) svd(ph(tts)) svd(dh)]
    cost = [norm(p - p) norm(p - ph) norm(p - dh(:))]
end
(trade-off curve 101b)

```

Uses `lra` 64.

Finally, the numerical rank vs. fitting error trade-off curves are computed and plotted for the methods based on nuclear norm minimization and unstructured low rank approximation.

```

101b (trade-off curve 101b)≡ (101a)
E = []; E_ = [];
for rr = 1:min(m,n) - 1
    ph = slra_nn(tts, p, rr); E = [E norm(p - ph)];
    if strcmp(structure, 'hankel')
        sysh = h2ss([0;p], rr);
        ph = impulse(sysh, np + 1); ph = ph(2:end);
    else % unstructured
        if m > n, [Rh, Ph, dh] = lra(p(tts)', rr); dh = dh';
        else [Rh, Ph, dh] = lra(p(tts), rr); end
        ph = dh(:);
    end
    E_ = [E_ norm(p - ph)];
end
plot(E , 1:min(m,n)-1, 'bx', 'markersize', 8), hold on
plot(E_ , 1:min(m,n)-1, 'ro', 'markersize', 8)
legend('slra_nn', 'lra')
plot(E , 1:min(m,n)-1, 'b-', 'linewidth', 2,
      'markersize', 8)
plot(E_ , 1:min(m,n)-1, 'r-.', 'linewidth', 2,
      'markersize', 8)
print_fig(structure)

```

Uses `h2ss` 76d, `lra` 64, `print_fig` 25a, and `slra_nn` 98c.

The first test example is a 5×5 , unstructured matrix, whose true value has rank 3. Note that in this case the method `lra`, which is based on the singular value decomposition gives an optimal approximation.

Table 3.1 Output of `test_slra_nn` in unstructured problem

sv =			
	4.8486	3.7087	4.8486
	2.1489	1.0090	2.1489
	1.5281	0.3882	1.5281
	1.1354	0.0000	0.0000
	0.5185	0.0000	0.0000

cost =			
	0	2.3359	1.2482

Table 3.2 Output of `test_slra_nn` in Hankel structured problem

sv =				
	0.8606	0.8352	0.8667	0.8642
	0.1347	0.1270	0.1376	0.1376
	0.0185	0.0011	0.0131	0.0118
	0.0153	0.0000	0.0000	0.0000
	0.0037	0.0000	0.0000	0.0000

cost =				
	0	0.0246	0.0130	0.0127

102a `(Test slra_nn on unstructured problem 102a)≡`
`m = 5; n = 5; r0 = 3; n1 = 1; structure = 'unstructured';`
`test_slra_nn`

Uses `test_slra_nn 100a`.

The output of `test_slra_nn` is given in Table 3.1. It shows that the numerical rank (with tolerance 10^{-5}) of both approximations is equal to the specified rank but the approximation error achieved by the nuclear norm heuristic is about two times the approximation error of the optimal approximation. The trade-off curve is shown in Fig. 3.1, left plot.

The second test example is a 5×5 , Hankel structured matrix, whose true value has rank 3. In this case the singular value decomposition-based method `h2ss` and the local optimization-based method `slra` are also heuristics for solving the Hankel structured low rank approximation problem and give, respectively, suboptimal and locally optimal results.

102b `(Test slra_nn on Hankel structured problem 102b)≡`
`m = 5; n = 5; r0 = 3; n1 = 0.05; structure = 'hankel';`
`test_slra_nn`

Uses `test_slra_nn 100a`.

The output of the `test_slra_nn` is given in Table 3.2. It again shows that the approximations have numerical rank matching the specification but the nuclear norm

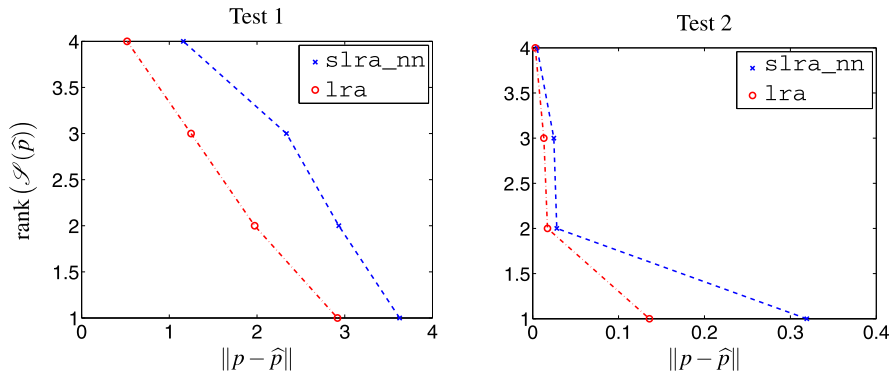


Fig. 3.1 Rank vs. approximation error trade-off curve

heuristic gives more than two times larger approximation error. The corresponding trade-off curve is shown in Fig. 3.1, right plot.

Missing Data Estimation

A random rank deficient “true” data matrix is constructed, where the matrix dimensions $m \times n$ and its rank r_0 are simulation parameters.

```
103a <Test missing data 103a>≡ 103b>
    <initialize the random number generator 89e>
    p0 = rand(m, r0) * rand(r0, n); p0 = p0(:);
```

Defines:
test_missing_data, used in chunk 104e.

The true matrix is unstructured:

```
103b <Test missing data 103a>+≡ <103a 103c>
    np = m * n; tts = reshape(1:np, m, n);
```

and is perturbed by a sparse matrix with sparsity level od , where od is a simulation parameter. The perturbation has constant values nl (a simulation parameter) in order to simulate outliers.

```
103c <Test missing data 103a>+≡ <103b 104a>
    pt = zeros(np, 1); no = round(od * np);
    I = randperm(np); I = I(1:no);
    pt(I) = nl * ones(no, 1); p = p0 + pt;
```

Under certain conditions, derived in Candés et al. (2009), the problem of recovering the true values from the perturbed data can be solved *exactly* by the regularized nuclear norm heuristic with 1-norm regularization and regularization parameter

$$\gamma = \sqrt{\frac{1}{\min(m, n)}}.$$

104a `<Test missing data 103a>+≡` <103c 104b>
`ph1 = nucnrm(tts, p, 1 / sqrt(max(m, n)), 1, eye(np));`
 Uses `nucnrm` 97.

In addition, the method can deal with missing data at known locations by setting the corresponding entries of the weight vector w to zero. In order to test this feature, we simulate the first element of the data matrix as missing and recover it by the approximation problem

104b `<Test missing data 103a>+≡` <104a 104c>
`p(1) = 0;`
`ph2 = nucnrm(tts, p, 1 / sqrt(max(m, n)), ...`
`1, diag([0; ones(np - 1, 1)]));`

Uses `nucnrm` 97.

We check the approximation errors $\|p_0 - \hat{p}\|_2$, were p_0 is the true value of the data parameter and \hat{p} is its approximation, obtained by the nuclear norm heuristic method. For comparison, we print also the perturbation size $\|p_0 - p\|_2$.

104c `<Test missing data 103a>+≡` <104b 104d>
`[norm(p0 - ph1) norm(p0 - ph2) norm(p0 - p)]`

Finally, we check the error of the recovered missing value.

104d `<Test missing data 103a>+≡` <104c
`[abs(p0(1) - ph1(1)) abs(p0(1) - ph2(1))]`

The result of a particular test

104e `<Test slra_nn on small problem with missing data 104e>≡`
`m = 50; n = 100; r0 = 5; nl = 0.4; od = 0.15;`
`test_missing_data`

Uses `test_missing_data` 103a.

is

ans =

```
0.0000    0.0000    10.9969
```

ans =

```
1.0e-09 *
```

```
0.4362    0.5492
```

showing that the method indeed recovers the missing data exactly.

3.4 Notes

Efficient Software for Structured Low Rank Approximation

The SLICOT library includes high quality FORTRAN implementation of algorithms for Cholesky factorization of positive definite Toeplitz banded matrices. The library

is used in a software package (Markovsky et al. 2005; Markovsky and Van Huffel 2005) for solving structured low rank approximation problems, based on the variable projections approach (Golub and Pereyra 2003) and Levenberg–Marquardt’s algorithm, implemented in MINPACK. This algorithm is globally convergent with a superlinear convergence rate.

Approximate Greatest Common Divisor

An alternative method for solving structured low rank approximation problems, called structured total least norm, has been modified for Sylvester structured matrices and applied to computation of approximate common divisor in Zhi and Yang (2004). The structured total least norm approach is different from the approach present in this chapter because it solves directly problem (AGCD) and does not use the elimination step leading to the equivalent problem (AGCD’).

Nuclear Norm Heuristic

The nuclear norm relaxation for solving rank minimization problems (RM) was proposed in Fazel (2002). It is a generalization of the ℓ_1 -norm heuristic from sparse vector approximation problems to low rank matrix approximation problems. The CVX package is developed and maintained by Grant and Boyd (2008a), see also Grant and Boyd (2008b). A Python version is also available (Dahl and Vandenberghe 2010).

The computational engines of CVX are SDPT3 and SeDuMi. These solvers can deal with a few tens of parameters ($n_p < 100$). An efficient interior point method for solving (NNM), which can deal with up to 500 parameters, is presented in Liu and Vandenberghe (2009). The method is implemented in Python.

References

- Candés E, Li X, Ma Y, Wright J (2009) Robust principal component analysis? www-stat.stanford.edu/~candes/papers/RobustPCA.pdf
- Dahl J, Vandenberghe L (2010) CVXOPT: Python software for convex optimization. abel.ee.ucla.edu/cvxopt
- Fazel M (2002) Matrix rank minimization with applications. PhD thesis, Elec. Eng. Dept., Stanford University
- Golub G, Pereyra V (2003) Separable nonlinear least squares: the variable projection method and its applications. *Inst. Phys., Inverse Probl.* 19:1–26
- Grant M, Boyd S (2008a) CVX: Matlab software for disciplined convex programming. stanford.edu/~boyd/cvx
- Grant M, Boyd S (2008b) Graph implementations for nonsmooth convex programs. In: Blondel V, Boyd S, Kimura H (eds) *Recent advances in learning and control*. Springer, Berlin, pp 95–110. stanford.edu/~boyd/graph_dcp.html

- Karmarkar N, Lakshman Y (1998) On approximate GCDs of univariate polynomials. *J Symb Comput* 26:653–666. Special issue on Symbolic Numeric Algebra for Polynomials
- Liu Z, Vandenberghe L (2009) Interior-point method for nuclear norm approximation with application to system identification. *SIAM J Matrix Anal Appl* 31(3):1235–1256
- Markovsky I, Van Huffel S (2005) High-performance numerical algorithms and software for structured total least squares. *J Comput Appl Math* 180(2):311–331
- Markovsky I, Van Huffel S, Pintelon R (2005) Block-Toeplitz/Hankel structured total least squares. *SIAM J Matrix Anal Appl* 26(4):1083–1099
- Zhi L, Yang Z (2004) Computing approximate GCD of univariate polynomials by structure total least norm. In: *MM research preprints*, vol 24, pp 375–387. Academia Sinica

Chapter 4

Applications in System, Control, and Signal Processing

4.1 Introduction

Structured low rank approximation is defined as deterministic data approximation.

Problem SLRA 1 (Structured low rank approximation) Given a structure specification

$$\mathcal{S} : \mathbb{R}^n \rightarrow \mathbb{R}^{m \times n}, \quad \text{with } m \leq n,$$

a parameter vector $p \in \mathbb{R}^n$, a vector norm $\|\cdot\|$, and an integer r , $0 < r < \min(m, n)$,

$$\text{minimize over } \hat{p} \quad \|p - \hat{p}\| \quad \text{subject to} \quad \text{rank}(\mathcal{S}(\hat{p})) \leq r. \quad (\text{SLRA})$$

Equivalently, the problem can be posed as (maximum likelihood) parameter estimation in the errors-in-variables setting:

$$p = p_0 + \tilde{p}, \quad \text{where } \text{rank}(\mathcal{S}(p_0)) \leq r \quad \text{and} \quad \tilde{p} \sim N(0, \sigma^2 V).$$

The statistical setting gives a recipe for choosing the norm ($\|\cdot\| = \|\cdot\|_{V^{-1}}$) and a “quality certificate” for the approximation method (SLRA): the method works “well” (consistency) and is optimal in a statistical sense (minimal asymptotic error covariance) under certain specified conditions.

The assumptions underlying the statistical setting are that the data p are generated by a true model that is in the considered model class with additive noise \tilde{p} that is a stochastic process satisfying certain additional assumptions. Model–data mismatch, however, is often due to a restrictive linear time-invariant model class being used and not (only) due to measurement noise. This implies that the approximation aspect of the method is often more important than the stochastic estimation one. The problems reviewed in this chapter are stated as deterministic approximation

problems although they can be given also the interpretation of defining maximum likelihood estimators under appropriate stochastic assumptions.

All problems are special cases of Problem SLRA for certain specified choices of the norm $\|\cdot\|$, structure \mathcal{S} , and rank r . In all problems the structure is affine, so that the algorithm and corresponding function `slra`, developed in Chap. 3, for solving affine structured low rank approximation problems can be used for solving the problems in this chapter.

108a `(solve Problem SLRA 108a)≡` (108b 113 115b 117a 120a)
`[R, ph, info] = slra(tts, par, r, [], s0);`
 Uses `slra` 85e.

4.2 Model Reduction

Approximate Realization

Define the 2-norm $\|\Delta H\|_2$ of a matrix-valued signal $\Delta H \in (\mathbb{R}^{p \times m})^{T+1}$ as

$$\|\Delta H\|_2 := \sqrt{\sum_{t=0}^T \|\Delta H(t)\|_F^2},$$

and let σ be the shift operator

$$\sigma(H)(t) = H(t+1).$$

Acting on a finite time series $(H(0), H(1), \dots, H(T))$, σ removes the first sample $H(0)$.

Problem 4.1 (Approximate realization) Given $H_d \in (\mathbb{R}^{p \times m})^T$ and a complexity specification \mathfrak{l} , find an optimal approximate model for H_d of a bounded complexity (m, \mathfrak{l}) , such that

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{H} \text{ and } \widehat{\mathcal{B}} && \|\widehat{H}_d - \widehat{H}\|_2 \\ & \text{subject to} && \widehat{H} \text{ is the impulse response of } \widehat{\mathcal{B}} \in \mathcal{L}_{m, \mathfrak{l}}^{m+p}. \end{aligned}$$

108b `(2-norm optimal approximate realization 108b)≡`
`function [sysh, hh, info] = h2ss_opt(h, n)`
`(reshape H and define m, p, T 77)`
`(approximate realization structure 109a)`
`(solve Problem SLRA 108a)`
`($\widehat{p} \mapsto \widehat{H}$ 109c), ($\widehat{H} \mapsto \widehat{\mathcal{B}}$ 109d)`

Defines:

`h2ss_opt`, used in chunks 110–12.

Problem 4.1 is equivalent to Problem SLRA, with

- norm $\| \cdot \| = \| \cdot \|_2$,
- Hankel structured data matrix $\mathcal{S}(p) = \mathcal{H}_{l+1}(\sigma H_d)$, and
- rank reduction by the number of outputs p .

109a *(approximate realization structure 109a)* \equiv (108b)

```
par = vec(h(:, :, 2:end)); s0 = []; r = n; (define l1 109b)
tts = blkhank(reshape(1:length(par), p, m, T - 1), l1);
```

Uses blkhank 25b.

The statement follows from the equivalence

$$\widehat{H} \text{ is the impulse response of } \widehat{\mathcal{B}} \in \mathcal{L}_{m,1} \iff \text{rank}(\mathcal{H}_{l+1}(\sigma \widehat{H})) \leq pl.$$

The optimal approximate model $\widehat{\mathcal{B}}^*$ does not depend on the shape of the Hankel matrix as long as the Hankel matrix dimensions are sufficiently large: at least $p \lceil (n+1)/p \rceil$ rows and at least $m \lceil (n+1)/m \rceil$ columns. However, solving the low rank approximation problem for a data matrix $\mathcal{H}_{l'+1}(\sigma H_d)$, where $l' > l$, one needs to achieve rank reduction by $p(l' - l + 1)$ instead of by p . Larger rank reduction leads to more difficult computational problems. On one hand, the cost per iteration gets higher and on another hand, the search space gets higher dimensional, which makes the optimization algorithm more susceptible to local minima. The variable $l1$ corresponds to $l + 1$. It is computed from the specified order n .

109b *(define l1 109b)* \equiv (109a 116a 117b)

```
l1 = ceil((n + 1) / p);
```

The mapping $\widehat{p} \mapsto \widehat{H}$ from the solution \widehat{p} of the structured low rank approximation problem to the optimal approximation \widehat{H} of the noisy impulse response H is reshaping the vector \widehat{p} as a $m \times p \times T$ tensor hh , representing the sequence $\widehat{H}(1), \dots, \widehat{H}(T-1)$ and setting $\widehat{H}(0) = H(0)$ (since $\widehat{D} = H(0)$).

109c $(\widehat{p} \mapsto \widehat{H} \text{ 109c}) \equiv$ (108b)

```
hh = zeros(p, m, T); hh(:, :, 1) = h(:, :, 1);
hh(:, :, 2:end) = reshape(ph(:), p, m, T - 1);
```

The mapping $\widehat{H} \mapsto \widehat{\mathcal{B}}$ is system realization, i.e., the 2-norm (locally) optimal realization $\widehat{\mathcal{B}}^*$ is obtained by exact realization of the approximation \widehat{H} , computed by the structured low rank approximation method.

109d $(\widehat{H} \mapsto \widehat{\mathcal{B}} \text{ 109d}) \equiv$ (108b)

```
sysh = h2ss(hh, n);
```

Uses h2ss 76d.

By construction, the approximation \widehat{H} of H has an exact realization in the model class $\mathcal{L}_{m,1}^{m+p}$.

Note 4.2 In the numerical solution of the structured low rank approximation problem, the kernel representation (rank_R) on p. 81 of the rank constraint is used. The parameter R , computed by the solver, gives a kernel representation of the optimal approximate model $\widehat{\mathcal{B}}^*$. The kernel representation can subsequently be converted

to a state space representation. This gives an alternative way of implementing the function `h2ss_opt` to the one using system realization ($\widehat{H} \mapsto \widehat{\mathcal{B}}$). The same note applies to the other problems, reviewed in the chapter.

Example

The following script verifies that the local optimization-based method `h2ss_opt` improves the suboptimal approximation computed by Kung's method `h2ss`. The data are a noisy impulse response of a random stable linear time-invariant system. The number of inputs m and outputs p , the order n of the system, the number of data points T , and the noise standard deviation s are simulation parameters.

```
110a (Test h2ss_opt 110a)≡ 110b>
      (initialize the random number generator 89e)
      sys0 = drss(n, p, m);
      h0 = reshape(shiftdim(impulse(sys0, T), 1), p, m, T);
      h = h0 + s * randn(size(h0));
```

Defines:

`test_h2ss_opt`, used in chunk 110c.

The solutions, obtained by the unstructured and Hankel structured low rank approximation methods, are computed and the relative approximation errors are printed.

```
110b (Test h2ss_opt 110a)+≡ <110a
      [sysh, hh] = h2ss(h, n); norm(h(:) - hh(:)) / norm(h(:))
      [sysh_, hh_, info] = h2ss_opt(h, n);
      norm(h(:) - hh_(:)) / norm(h(:))
```

Uses `h2ss` 76d and `h2ss_opt` 108b.

The optimization-based method improves the suboptimal results of the singular value decomposition-based method at the price of extra computation, a process referred to as *iterative refinement* of the solution.

```
110c (Compare h2ss and h2ss_opt 110c)≡
      m = 2; p = 3; n = 4; T = 15; s = 0.1; test_h2ss_opt
```

Uses `test_h2ss_opt` 110a.

\rightsquigarrow 0.1088 for `h2ss` and 0.0735 for `h2ss_opt`

Model Reduction

The finite time- T H_2 norm $\|\Delta\mathcal{B}\|_{2,T}$ of a linear time-invariant system $\Delta\mathcal{B}$ is defined as the 2-norm of the sequence of its first T Markov parameters, i.e., if ΔH is the impulse response of $\Delta\mathcal{B}$, $\|\Delta\mathcal{B}\|_{2,T} := \|\Delta H\|_2$.

Problem 4.3 (Finite time H_2 model reduction) Given a linear time-invariant system $\mathcal{B}_d \in \mathcal{L}_{m,1}^q$ and a complexity specification $l_{\text{red}} < 1$, find an optimal approximation of \mathcal{B}_d with bounded complexity (m, l_{red}) , such that

$$\text{minimize over } \widehat{\mathcal{B}} \quad \|\mathcal{B}_d - \widehat{\mathcal{B}}\|_{2,T} \quad \text{subject to } \widehat{\mathcal{B}} \in \mathcal{L}_{m,1_{\text{red}}}^q.$$

Problem 4.3 is equivalent to Problem SLRA with

- norm $\|\cdot\| = \|\cdot\|_2$,
- Hankel structured data matrix $\mathcal{S}(p) = \mathcal{H}_{1+1}(\sigma H_d)$, where H_d is the impulse response of \mathcal{B}_d , and
- rank reduction by the number of outputs $p := q - m$,

which shows that finite time H_2 model reduction is equivalent to the approximate realization problem with H_d being the impulse response of \mathcal{B}_d . In practice, \mathcal{B}_d need not be linear time-invariant system since in the model reduction problem only the knowledge of its impulse response H_d is used.

111 *(Finite time H_2 model reduction 111)* \equiv

```
function [sysh, hh, info] = mod_red(sys, T, r)
    [sysh, hh, info]
    = h2ss_opt(shiftdim(impulse(sys, T), 1), r);
```

Defines:

mod_red, used in chunk 125d.

Uses h2ss_opt 108b.

Exercise 4.4 Compare the results of `mod_red` with the ones obtained by using unstructured low rank approximation (finite-time balanced model reduction) on simulation examples.

Output only Identification

Excluding the cases of multiple poles, the model class of autonomous linear time-invariant systems $\mathcal{L}_{0,1}^p$ is equivalent to the *sum-of-damped exponentials model* class, i.e., signals y that can be represented in the form

$$y(t) = \sum_{k=1}^1 \alpha_k e^{\beta_k t} e^{i(\omega_k t + \phi_k)} \quad (\mathbf{i} = \sqrt{-1}).$$

The parameters $\{\alpha_k, \beta_k, \omega_k, \phi_k\}_{j=1}^1$ of the sum-of-damped exponentials model have the following meaning: α_k are amplitudes, β_k damping factors, ω_k frequencies, and ϕ_k initial phases.

Problem 4.5 (Output only identification) Given a signal $y_d \in (\mathbb{R}^p)^T$ and a complexity specification l , find an optimal approximate model for y_d of bounded complexity $(0, l)$, such that

$$\begin{aligned} & \text{minimize} \quad \text{over } \widehat{\mathcal{B}} \text{ and } \widehat{y} \quad \|y_d - \widehat{y}\|_2 \\ & \text{subject to} \quad \widehat{y} \in \widehat{\mathcal{B}}|_{[1,T]} \quad \text{and} \quad \widehat{\mathcal{B}} \in \mathcal{L}_{0,1}^p. \end{aligned}$$

Problem 4.5 is equivalent to Problem SLRA with

- norm $\| \cdot \| = \| \cdot \|_2$,
- Hankel structured data matrix $\mathcal{S}(p) = \mathcal{H}_{1+1}(y_d)$, and
- rank reduction by the number of outputs p ,

which shows that output only identification is equivalent to approximate realization and is, therefore, also equivalent to finite time H_2 model reduction. In the signal processing literature, the problem is known as *linear prediction*.

112a *(Output only identification 112a)* \equiv

```
function [sysh, yh, xinih, info] = ident_aut(y, n)
[sysh, yh, info] = h2ss_opt(y, n);
(impulse response realization  $\mapsto$  autonomous system realization 112b)
```

Defines:

ident_aut, used in chunk 209c.

Uses h2ss_opt 108b.

Let $\mathcal{B}_{i/s/o}(A, b, C, d)$ be a realization of y . Then the response of the autonomous system $\mathcal{B}_{ss}(A, C)$ to initial condition $x(0) = b$ is y . This gives a link between impulse response realization and autonomous system realization:

112b *(impulse response realization \mapsto autonomous system realization 112b)* \equiv (112a 113)

```
xinih = sysh.b; sysh = ss(sysh.a, [], sysh.c, [], -1);
```

Exercise 4.6 Compare the results of `ident_aut` with the ones obtained by using unstructured low rank approximation on simulation examples.

Harmonic Retrieval

The aim of the harmonic retrieval problem is to approximate the data by a sum of sinusoids. From a system theoretic point of view, harmonic retrieval aims to approximate the data by a marginally stable linear time-invariant autonomous system.¹

Problem 4.7 (Harmonic retrieval) Given a signal $y_d \in (\mathbb{R}^p)^T$ and a complexity specification 1 , find an optimal approximate model for y_d that is in the model class $\mathcal{L}_{0,1}^p$ and is marginally stable, i.e.,

$$\begin{aligned} & \text{minimize} \quad \text{over } \widehat{\mathcal{B}} \text{ and } \widehat{y} \quad \|y_d - \widehat{y}\|_2 \\ & \text{subject to} \quad \widehat{y} \in \widehat{\mathcal{B}}|_{[1,T]}, \quad \widehat{\mathcal{B}} \in \mathcal{L}_{0,1}^p, \quad \text{and} \quad \widehat{\mathcal{B}} \text{ is marginally stable.} \end{aligned}$$

Due to the stability constraint, Problem 4.7 is not a special case of problem SLRA. In the univariate case $p = 1$, however, a necessary condition for an

¹A linear time-invariant autonomous system is marginally stable if all its trajectories, except for the $y = 0$ trajectory, are bounded and do not converge to zero.

autonomous model \mathcal{B} to be marginally stable is that a kernel representation $\ker(R)$ of \mathcal{B} is either palindromic,

$$R(z) := \sum_{i=0}^1 z^i R_i \text{ is palindromic: } \iff R_{\ell-i} = R_i, \quad \text{for } i = 0, 1, \dots, 1$$

or antipalindromic,

$$R(z) \text{ is antipalindromic: } \iff R_{\ell-i} = -R_i, \quad \text{for } i = 0, 1, \dots, 1.$$

The antipalindromic case is nongeneric in the space of the marginally stable systems, so as relaxation of the stability constraint, we can use the constraint that the kernel representation is palindromic.

Problem 4.8 (Harmonic retrieval, relaxed version, scalar case) Given a signal $y_d \in (\mathbb{R})^T$ and a complexity specification 1 , find an optimal approximate model for y_d that is in the model class $\mathcal{L}_{0,1}^1$ and has a palindromic kernel representation, such that

$$\text{minimize over } \widehat{\mathcal{B}} \text{ and } \widehat{y} \quad \|y_d - \widehat{y}\|_2$$

$$\text{subject to } \widehat{y} \in \widehat{\mathcal{B}}|_{[1,T]}, \quad \widehat{\mathcal{B}} \in \mathcal{L}_{0,1}^1 \quad \text{and} \quad \ker(\widehat{R}) = \widehat{\mathcal{B}}, \quad \text{with } R \text{ palindromic.}$$

113 *(Harmonic retrieval 113)* \equiv

```
function [sysh, yh, xinih, info] = harmonic_retrieval(y, n)
    (harmonic retrieval structure 114a)
    (solve Problem SLRA 108a), yh = ph; sysh = h2ss(yh, n);
    (impulse response realization  $\mapsto$  autonomous system realization 112b)
```

Defines:

harmonic_retrieval, used in chunk 114b.

Uses h2ss 76d.

The constraint “ R palindromic” can be expressed as a structural constraint on the data matrix, which reduces the relaxed harmonic retrieval problem to the structured low rank approximation problem. Problem 4.8 is equivalent to Problem SLRA with

- norm $\|\cdot\| = \|\cdot\|_2$,
- structured data matrix composed of a Hankel matrix next to a Toeplitz matrix:

$$\mathcal{S}(p) = [\mathcal{H}_{1+1}(y) \updownarrow \mathcal{H}_{1+1}(y)],$$

where

$$\updownarrow \mathcal{H}_{1+1}(y) := \begin{bmatrix} y_{1+1} & y_{1+2} & \cdots & y_T \\ \vdots & \vdots & \ddots & \vdots \\ y_2 & y_3 & \cdots & y_{T-1+1} \\ y_1 & y_2 & \cdots & y_{T-1} \end{bmatrix},$$

and

- rank reduction by one.

114a *(harmonic retrieval structure 114a)* \equiv (113)

```
par = y(:); np = length(par); r = n; s0 = [];
tts = [blkhank(1:np, n + 1) flipud(blkhank(1:np, n + 1))];
```

Uses blkhank 25b.

The statement follows from the equivalence

$$\hat{y} \in \widehat{\mathcal{B}}|_{[1,T]}, \widehat{\mathcal{B}} \in \mathcal{L}_{0,1}^1 \text{ and } \ker(\widehat{R}) = \widehat{\mathcal{B}} \text{ is palindromic}$$

$$\iff \text{rank}([\mathcal{H}_{1+1}(\hat{y}) \downarrow \mathcal{H}_{1+1}(\hat{y})]) \leq 1.$$

In order to show it, let $\ker(R)$, with $R(z) = \sum_{i=0}^1 z^i R_i$ full row rank, be a kernel representation of $\mathcal{B} \in \mathcal{L}_{0,1}^1$. Then $\hat{y} \in \widehat{\mathcal{B}}|_{[1,T]}$ is equivalent to

$$[R_0 \ R_1 \ \cdots \ R_1] \mathcal{H}_{1+1}(\hat{y}) = 0.$$

If, in addition, R is palindromic, then

$$[R_1 \ \cdots \ R_1 \ R_0] \mathcal{H}_{1+1}(\hat{y}) = 0 \iff [R_0 \ R_1 \ \cdots \ R_1] \downarrow \mathcal{H}_{1+1}(\hat{y}) = 0.$$

We have

$$[R_0 \ R_1 \ \cdots \ R_1] [\mathcal{H}_{1+1}(\hat{y}) \downarrow \mathcal{H}_{1+1}(\hat{y})] = 0 \quad (*)$$

which is equivalent to

$$\text{rank}([\mathcal{H}_{1+1}(\hat{y}) \downarrow \mathcal{H}_{1+1}(\hat{y})]) \leq 1.$$

Conversely, (*) implies $\hat{y} \in \widehat{\mathcal{B}}|_{[1,T]}$ and R palindromic.

Example

The data are generated as a sum of random sinusoids with additive noise. The number hn of sinusoids, the number of samples T , and the noise standard deviation s are simulation parameters.

114b *(Test harmonic_retrieval 114b)* \equiv

```
(initialize the random number generator 89e)
t = 1:T; f = 1 * pi * rand(hn, 1);
phi = 2 * pi * rand(hn, 1);
y0 = sum(sin(f * t + phi(:, ones(1, T)))));
yt = randn(size(y0)); y = y0 + s * norm(y0) * yt / norm(yt);
[sysh, yh, xinih, info] = harmonic_retrieval(y, hn * 2);
plot(t, y0, 'k-', t, y, 'k:', t, vec(yh), 'b-');
ax = axis; axis([1 T ax(3:4)])
print_fig('test_harmonic_retrieval')
```

Defines:

test_harmonic_retrieval, used in chunk 115a.

Uses harmonic_retrieval 113 and print_fig 25a.

Fig. 4.1 Results of `harmonic_retrieval`:
solid line—true system’s trajectory y_0 , *dotted*
dotted—noisy data y_d ,
dashed line—best approximation \hat{y}

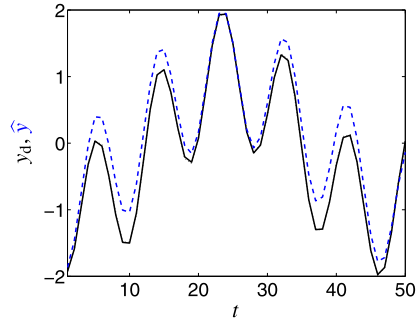


Figure 4.1 shows the true signal, the noisy signal, and the estimate obtained with `harmonic_retrieval` in the following simulation example:

```
115a <Example of harmonic retrieval 115a>≡
      clear all, T = 50; hn = 2; s = 0.015;
      test_harmonic_retrieval
      Uses test_harmonic_retrieval 114b.
```

4.3 System Identification

Errors-in-Variables Identification

In errors-in-variables data modeling problems, the observed variables are a priori known (or assumed) to be noisy. This prior knowledge is used to correct the data, so that the corrected data are consistent with a model in the model class. The resulting problem is equivalent of the misfit minimization problem (see, Problem 2.33), considered in Chap. 2.

Problem 4.9 (Errors-in-variables identification) Given T samples, q variables, vector signal $w_d \in (\mathbb{R}^q)^T$, a signal norm $\|\cdot\|$, and a model complexity $(m, 1)$,

$$\begin{aligned} &\text{minimize} && \text{over } \hat{\mathcal{B}} \text{ and } \hat{w} && \|w_d - \hat{w}\| \\ &\text{subject to} && \hat{w} \in \hat{\mathcal{B}}|_{[1,T]} && \text{and } \hat{\mathcal{B}} \in \mathcal{L}_{m,1}^q. \end{aligned}$$

```
115b <Errors-in-variables identification 115b>≡
      function [sysh, wh, info] = ident_eiv(w, m, n)
      <reshape w and define q, T 26d>
      <errors-in-variables identification structure 116a>
      <solve Problem SLRA 108a>, wh = reshape(ph(:), q, T);
      <exact identification: w-hat to B-hat 116b>
```

Defines:
`ident_eiv`, used in chunks 90a and 116c.

Problem 4.9 is equivalent to Problem SLRA with

- Hankel structured data matrix $\mathcal{S}(p) = \mathcal{H}_{l+1}(w_d)$ and
- rank reduction with the number of outputs p

116a *(errors-in-variables identification structure 116a)* \equiv (115b)

```
par = w(:); np = length(par);
p = q - m; (define l1 109b), r = m * l1 + n;
tts = blkhank(reshape(1:np, q, T), l1); s0 = [];
```

Uses blkhank 25b.

The identified system is recovered from the optimal approximating trajectory \hat{w} by exact identification.

116b *(exact identification: $\hat{w} \mapsto \hat{\mathcal{B}}$ 116b)* \equiv (115b 117a)

```
sysh = w2h2ss(wh, m, n);
```

Uses w2h2ss 80c.

Example

In this example, the approximate model computed by the function `ident_eiv` is compared with the model obtained by the function `w2h2ss`. Although `w2h2ss` is an exact identification method, it can be used as a heuristic for approximate identification. The data are generated in the errors-in-variables setup (see, (EIV) on p. 70). The true system is a random single input single output system.

116c *(Test ident_eiv 116c)* \equiv

```
(initialize the random number generator 89e)
m = 1; p = 1; sys0 = drss(n, p, m);
xini0 = rand(n, 1); u0 = rand(T, m);
y0 = lsim(sys0, u0, 1:T, xini0);
w = [u0'; y0'] + s * randn(m + p, T);
sys = w2h2ss(w, m, n); ((TF)  $\mapsto$  P 88e) misfit_asis(w, P)
[sysh, wh, info] = ident_eiv(w, m, n); info.M
```

Defines:

```
test_ident_eiv, used in chunk 116d.
```

Uses `ident_eiv` 115b, `misfit_asis` 87b, and `w2h2ss` 80c.

In a particular example

116d *(Compare w2h2ss and ident_eiv 116d)* \equiv

```
n = 4; T = 30; s = 0.1; test_ident_eiv
```

Uses `test_ident_eiv` 116c.

the obtained results are `misfit` 1.2113 for `w2h2ss` and 0.2701 for `ident_eiv`.

Output Error Identification

In the errors-in-variables setting, using an input-output partitioning of the variables, both the input and the output are noisy. In some applications, however, the input is not measured; it is designed by the user. Then, it is natural to assume that the input is noise free. This leads to the output error identification problem.

Problem 4.10 (Output error identification) Given a signal

$$w_d = (w_d(1), \dots, w_d(T)), \quad w_d(t) \in \mathbb{R}^q,$$

with an input/output partitioning $w = (u, y)$, $\dim(u) = m$, and a complexity specification l , find an optimal approximate model for w_d of a bounded complexity (m, l) , such that

$$\begin{aligned} & \text{minimize} \quad \text{over } \widehat{\mathcal{B}} \text{ and } \widehat{y} \quad \|y_d - \widehat{y}\|_2 \\ & \text{subject to} \quad (u_d, \widehat{y}) \in \widehat{\mathcal{B}}|_{[1, T]} \quad \text{and} \quad \widehat{\mathcal{B}} \in \mathcal{L}_{m, l}^q. \end{aligned}$$

117a *(Output error identification 117a)* \equiv

```
function [sysh, wh, info] = ident_oe(w, m, n)
    (reshape w and define q, T 26d)
    (output error identification structure 117b)
    (solve Problem SLRA 108a), wh = [w(1:m, :); reshape(ph(:), p, T)];
    (exact identification:  $\widehat{w} \mapsto \widehat{\mathcal{B}}$  116b)
```

Defines:

`ident_oe`, used in chunks 118 and 206a.

Output error identification is a limiting case of the errors-in-variables identification problem when the noise variance tends to zero. Alternatively, output error identification is a special case in the prediction error setting when the noise term is not modeled.

As shown next, Problem 4.10 is equivalent to Problem SLRA with

- norm $\|\cdot\| = \|\cdot\|_2$,
- data matrix

$$\mathcal{S}(p) = \begin{bmatrix} \mathcal{H}_{1+1}(u_d) \\ \mathcal{H}_{1+1}(y_d) \end{bmatrix}$$

composed of a fixed block and a Hankel structured block, and

- rank reduction by the number of outputs p .

117b *(output error identification structure 117b)* \equiv (117a)

```
par = vec(w((m + 1):end, :)); np = length(par); p = q - m;
(define l1 109b), j = T - l1 + 1; r = m * l1 + n;
s0 = [blkhank(w(1:m, :), l1); zeros(l1 * p, j)];
tts = [zeros(l1 * m, j); blkhank(reshape(1:np, p, T), l1)];
```

Uses `blkhank` 25b.

The statement is based on the equivalence

$$\begin{aligned} & (u_d, \widehat{y})|_{[1, T-1]} \in \widehat{\mathcal{B}}|_{[1, T-1]} \quad \text{and} \quad \widehat{\mathcal{B}} \in \mathcal{L}_{m, l} \\ & \iff \text{rank} \left(\begin{bmatrix} \mathcal{H}_{1+1}(u_d) \\ \mathcal{H}_{1+1}(\widehat{y}) \end{bmatrix} \right) \leq m(1 + l) + pl, \end{aligned}$$

which is a corollary of the following lemma.

Lemma 4.11 *The signal w is a trajectory of a linear time-invariant system of complexity bounded by $(m, 1)$, i.e.,*

$$w|_{[1, T-1]} \in \mathcal{B}|_{[1, T-1]} \quad \text{and} \quad \mathcal{B} \in \mathcal{L}_{m,1}^q \quad (*)$$

if and only if

$$\text{rank}(\mathcal{H}_{1+1}(w)) \leq m(1+1) + (q-m)1. \quad (**)$$

Proof (\implies) Assume that $(*)$ holds and let $\ker(R)$, with

$$R(z) = \sum_{i=0}^1 z^i R_i \in \mathbb{R}^{g \times q}[z]$$

full row rank, be a kernel representation of \mathcal{B} . The assumption $\mathcal{B} \in \mathcal{L}_{m,1}$ implies that $g \geq p := q - m$. From $w \in \mathcal{B}|_{[1, T]}$, we have

$$[R_0 \ R_1 \ \cdots \ R_1] \mathcal{H}_{1+1}(w) = 0, \quad (***)$$

which implies that $(**)$ holds.

(\impliedby) Assume that $(**)$ holds. Then, there is a full row rank matrix

$$R := [R_0 \ R_1 \ \cdots \ R_1] \in \mathbb{R}^{p \times q(1+1)},$$

such that $(***)$. Define the polynomial matrix $R(z) = \sum_{i=0}^1 z^i R_i$. Then the system \mathcal{B} induced by $R(z)$ via the kernel representation $\ker(R(\sigma))$ is such that $(*)$ holds. \square

Example

This example is analogous to the example of the errors-in-variables identification method. The approximate model obtained with the function `w2h2ss` is compared with the approximate model obtained with `ident_oe`. In this case, the data are generated in the output error setting, i.e., the input is exact and the output is noisy. In this simulation setup we expect that the optimization-based method `ident_oe` improves the result obtained with the subspace-based method `w2h2ss`.

```
118 <Test ident_oe 118>≡
    <initialize the random number generator 89e>
    m = 1; p = 1; sys0 = drss(n, p, m); xini0 = rand(n, 1);
    u0 = rand(T, m); y0 = lsim(sys0, u0, 1:T, xini0);
    w = [u0'; y0'] + s * [zeros(m, T); randn(p, T)];
    sys = w2h2ss(w, m, n); <(TF) to P 88e> misfit_siso(w, P)
    [sysh, wh, info] = ident_oe(w, m, n); info.M
```

Defines:

`test_ident_oe`, used in chunk 119a.

Uses `ident_oe` 117a, `misfit_siso` 87b, and `w2h2ss` 80c.

In the following simulation example

119a *(Example of output error identification 119a)*≡
`n = 4; T = 30; s = 0.1; test_ident_oe`
 Uses `test_ident_oe` 118.
`w2h2ss` achieves misfit 1.0175 and `ident_oe` achieves misfit 0.2331.

Finite Impulse Response System Identification

Let $\text{FIR}_{m,1}$ be the model class of finite impulse response linear time-invariant systems with m inputs and lag at most 1, i.e.,

$$\text{FIR}_{m,1} := \{ \mathcal{B} \in \mathcal{L}_{m,1} \mid \mathcal{B} \text{ has finite impulse response and } m \text{ inputs} \}.$$

Identification of a finite impulse response model in the output error setting leads to the ordinary linear least squares problem

$$[\widehat{H}(0) \widehat{H}(1) \cdots \widehat{H}(1)] \mathcal{H}_{1+1}(u_d) = [y_d(1) \cdots y_d(T-1)].$$

119b *(Output error finite impulse response identification 119b)*≡
`function [hh, wh] = ident_fit_oe(w, m, l)`
(reshape w and define q, T 26d)
(Finite impulse response identification structure 120b)
`D = par(tts);`
`hh_ = D((m * l1) + 1):end, :) / D(1:(m * l1), :);`
`hh = reshape(hh_, p, m, l1); hh = hh(:, :, end:-1:1);`
`uh = w(1:m, :); yh = [hh_ * D(1:(m * l1), :) zeros(p, l)];`
`wh = [uh; yh];`

Defines:

`ident_fir_oe`, used in chunk 121b.

Next, we define the finite impulse response identification problem in the errors-in-variables setting.

Problem 4.12 (Errors-in-variables finite impulse response identification) Given a signal

$$w_d = (w_d(1), \dots, w_d(T)), \quad w_d(t) \in \mathbb{R}^q,$$

with an input/output partition $w = (u, y)$, with $\dim(u) = m$, and a complexity specification l , find an optimal approximate finite impulse response model for w_d of bounded complexity (m, l) , such that

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{\mathcal{B}} \text{ and } \widehat{w} && \|w_d - \widehat{w}\|_2 \\ & \text{subject to} && \widehat{w} \in \widehat{\mathcal{B}}|_{[1,T]} \text{ and } \widehat{\mathcal{B}} \in \text{FIR}_{m,1}. \end{aligned}$$

120a

(Errors-in-variables finite impulse response identification 120a)≡

```
function [hh, wh, info] = ident_fir_eiv(w, m, 1)
  <reshape w and define q, T 26d>
  <Finite impulse response identification structure 120b>
  <solve Problem SLRA 108a>, hh = r2x(R)';
  hh = reshape(hh, p, m, 1 + 1);
  hh = hh(:, :, end:-1:1);
  uh = reshape(ph(1:(T * m)), m, T);
  yh = reshape(ph((T * m) + 1):end), p, T - 1);
  wh = [uh; [yh zeros(p, 1)]];
```

Defines:

ident_fir_eiv, used in chunk 121b.

Problem 4.12 is equivalent to Problem SLRA with

- norm $\|\cdot\| = \|\cdot\|_2$,
- data matrix

$$\mathcal{S}(p) = \begin{bmatrix} \mathcal{H}_{1+1}(u_d) \\ [y_d(1) \cdots y_d(T-1)] \end{bmatrix},$$

composed of a fixed block and a Hankel structured block, and

- rank reduction by the number of outputs p .

120b

(Finite impulse response identification structure 120b)≡

(119b 120a)

```
p = q - m; l1 = 1 + 1; r = l1 * m;
par = vec([w(1:m, :), w((m + 1):end, 1:(T - 1))]); s0 = [];
tts = [blkhank((1:(T * m)), l1);
       reshape(T * m + (1:((T - 1) * p)), p, T - 1)];
```

Uses blkhank 25b.

The statement follows from the equivalence

$$\begin{aligned} \widehat{w}|_{1,T-1} \in \mathcal{B}|_{[1,T-1]} \quad \text{and} \quad \widehat{\mathcal{B}} \in \text{FIR}_{m,1} \\ \iff \text{rank} \left(\begin{bmatrix} \mathcal{H}_{1+1}(\widehat{u}) \\ [\widehat{y}(1) \cdots \widehat{y}(T-1)] \end{bmatrix} \right) \leq m(1+1). \end{aligned}$$

In order to show it, let

$$H = (H(0), H(1), \dots, H(1), 0, 0, \dots)$$

be the impulse response of $\widehat{\mathcal{B}} \in \text{FIR}_{m,1}$. The signal $\widehat{w} = (\widehat{u}, \widehat{y})$ is a trajectory of \mathcal{B} if and only if

$$[H(1) \cdots H(1) H(0)] \mathcal{H}_{1+1}(\widehat{u}) = [\widehat{y}(1) \cdots \widehat{y}(T-1)].$$

Equivalently, $\widehat{w} = (\widehat{u}, \widehat{y})$ is a trajectory of \mathcal{B} if and only if

$$[H(1) \cdots H(1) H(0) - I_p] \begin{bmatrix} \mathcal{H}_{1+1}(\widehat{u}) \\ [\widehat{y}(1) \cdots \widehat{y}(T-1)] \end{bmatrix} = 0,$$

or,

$$\text{rank} \left(\begin{bmatrix} \mathcal{H}_{l+1}(\hat{u}) \\ [\hat{y}(1) \cdots \hat{y}(T-l)] \end{bmatrix} \right) \leq m(l+1).$$

For exact data, i.e., assuming that

$$y_d(t) = (H \star u_d)(t) := \sum_{\tau=0}^l H(\tau)u_d(t-\tau)$$

the finite impulse response identification problem is equivalent to the deconvolution problem: Given the signals u_d and $y_d := H \star u_d$, find the signal H . For noisy data, the finite impulse response identification problem can be viewed as an *approximate deconvolution problem*. The approximation is in the sense of finding the nearest signals \hat{u} and \hat{y} to the given ones u_d and y_d , such that $\hat{y} := \hat{H} \star \hat{u}$, for a signal \hat{H} with a given length l .

Example

Random data from a moving average finite impulse response system is generated in the errors-in-variables setup. The number of inputs and outputs, the system lag, number of observed data points, and the noise standard deviation are simulation parameters.

```
121a <Test ident_fir 121a>≡ 121b>
    <initialize the random number generator 89e>,
    h0 = ones(m, p, l + 1); u0 = rand(m, T); t = 1:(l + 1);
    y0 = conv(h0(:), u0); y0 = y0(end - T + 1:end);
    w0 = [u0; y0]; w = w0 + s * randn(m + p, T);
```

Defines:

test_ident_fir, used in chunk 121c.

The output error and errors-invariables finite impulse response identification methods `ident_fir_oe` and `ident_fir_eiv` are applied on the data and the relative fitting errors $\|w_d - \hat{w}\|/\|w_d\|$ are computed.

```
121b <Test ident_fir 121a>+≡ <121a
    [hh_oe, wh_oe] = ident_fir_oe(w, m, l);
    e_oe = norm(w(:) - wh_oe(:)) / norm(w(:))
    [hh_eiv, wh_eiv, info] = ident_fir_eiv(w, m, l);
    e_eiv = norm(w(:) - wh_eiv(:)) / norm(w(:))
```

Uses `ident_fir_eiv` 120a and `ident_fir_oe` 119b.

The obtained result in the following example

```
121c <Example of finite impulse response identification 121c>≡
    m = 1; p = 1; l = 10; T = 50; s = 0.5; test_ident_fir
```

Uses `test_ident_fir` 121a.

are: relative error 0.2594 for `ident_fir_oe` and 0.2391 for `ident_fir_eiv`.

4.4 Analysis and Synthesis

Distance to Uncontrollability

Checking controllability of a linear time-invariant system \mathcal{B} is a rank test problem. However, the matrices which rank deficiency indicates lack of controllability for \mathcal{B} are *structured* and, depending on the representation of \mathcal{B} , might be nonlinear transformations of the system parameters. Computing the numerical rank of a structured matrix by the singular value decomposition no longer gives a guarantee that there is a nearby matrix with the specified rank that has the same structure as the given matrix. For checking controllability, this implies that the system might be declared close to uncontrollable, but that there is no nearby system that is uncontrollable. In other words, the standard singular value decomposition test might be pessimistic.

Let $\overline{\mathcal{L}}_{\text{ctrb}}$ be the set of uncontrollable linear time-invariant systems and let $\text{dist}(\mathcal{B}, \widehat{\mathcal{B}})$ be a measure for the distance from \mathcal{B} to $\widehat{\mathcal{B}}$. Consider for simplicity the single input single output case and let $\mathcal{B}_{i/o}(P, Q)$, be an input/output representation of \mathcal{B} . Moreover, without loss of generality, assume that P is monic. With this normalization the parameters P, Q are unique and, therefore, the distance measure

$$\text{dist}(\mathcal{B}, \widehat{\mathcal{B}}) := \sqrt{\|P - \widehat{P}\|_2^2 + \|Q - \widehat{Q}\|_2^2} \quad (\text{dist})$$

is a property of the pair of systems $(\mathcal{B}, \widehat{\mathcal{B}})$.

In terms of the parameters \widehat{P} and \widehat{Q} , the constraint $\widehat{\mathcal{B}} \in \overline{\mathcal{L}}_{\text{ctrb}}$ is equivalent to rank deficiency of the Sylvester matrix $\mathcal{R}(\widehat{P}, \widehat{Q})$ (see \mathcal{R} on p. 11). With respect to the distance measure (dist), the problem of computing the distance from \mathcal{B} to uncontrollability is equivalent to a Sylvester structured low rank approximation problem

$$\begin{aligned} & \text{minimize} \quad \text{over } \widehat{P} \text{ and } \widehat{Q} \quad \|P - \widehat{P}\|_2^2 + \|Q - \widehat{Q}\|_2^2 \\ & \text{subject to} \quad \text{rank}(\mathcal{R}(\widehat{P}, \widehat{Q})) \leq \text{degree}(P) + \text{degree}(Q) - 1, \end{aligned}$$

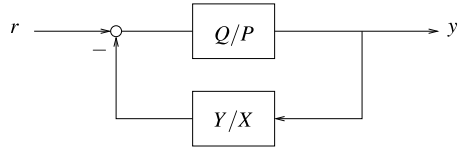
for which numerical algorithms are developed in Sect. 3.2. The implementation details are left as an exercise for the reader (see Note 3.15 and Problem P.20).

Pole Placement by a Low-Order Controller

Consider the single input single output feedback system shown in Fig. 4.2. The polynomials P and Q , define the transfer function Q/P of the plant and are given. They are assumed to be relatively prime and the transfer function Q/P is assumed to satisfy the constraint

$$\text{deg}(Q) \leq \text{deg}(P) =: \mathbb{1}_P,$$

Fig. 4.2 Feedback control system



which ensures that the plant is a causal linear time-invariant system. The polynomials Y and X parameterize the controller $\mathcal{B}_{1/o}(X, Y)$ and are unknowns. The design constraints are that the controller should be causal and have order bounded by a specified integer $\mathbb{1}_X$. These specifications translate to the following constraints on the polynomials Y and X

$$\text{deg}(Y) \leq \text{deg}(X) =: \mathbb{1}_X < \mathbb{1}_P. \tag{deg}$$

The pole placement problem is to determine X and Y , so that the poles of the closed-loop system are as close as possible in some specified sense to desired locations, given by the roots of a polynomial F , where $\text{deg}(F) = \mathbb{1}_X + \mathbb{1}_P$. We consider a modification of the pole placement problem that aims to assign exactly the poles of a plant that is as close to the given plant as possible.

In what follows, we use the correspondence between $\mathbb{1}_P + 1$ dimensional vectors and $\mathbb{1}_P$ th degree polynomials

$$\text{col}(P_0, P_1, \dots, P_{\mathbb{1}_P}) \in \mathbb{R}^{\mathbb{1}_P+1} \leftrightarrow P(z) = P_0 + P_1 z + \dots + P_{\mathbb{1}_P} z^{\mathbb{1}_P} \in \mathbb{R}[z],$$

and (with some abuse of notation) refer to P as both a vector and a polynomial.

Problem 4.13 (Pole placement by low-order controller) Given

1. the transfer function Q/P of a plant,
2. a polynomial F , whose roots are the desired poles of the closed-loop system, and
3. a bound $\mathbb{1}_X < \text{deg}(P)$ on the order of the controller,

find the transfer function Y/X of the controller, such that

1. the degree constraint (deg) is satisfied and
2. the controller assigns the poles of a system whose transfer function \widehat{Q}/\widehat{P} is as close as possible to the transfer function Q/P in the sense that

$$\|\text{col}(P, Q) - \text{col}(\widehat{P}, \widehat{Q})\|_2$$

is minimized.

Next, we write down explicitly the considered optimization problem, which shows its equivalence to a structured low rank approximation problem. The closed-loop transfer function is

$$\frac{QX}{PX + QY},$$

so that a solution to the pole placement problem is given by a solution to the Diophantine equation

$$PX + QY = F.$$

The Diophantine equation can be written as a Sylvester structured system of equations

$$\underbrace{\begin{bmatrix} P_0 & & & Q_0 \\ P_1 & \ddots & & Q_1 & \ddots \\ \vdots & \ddots & P_0 & \vdots & \ddots & Q_0 \\ P_{1_p} & & P_1 & Q_{1_p} & & Q_1 \\ & \ddots & \vdots & & \ddots & \vdots \\ & & P_{1_p} & & & Q_{1_p} \end{bmatrix}}_{\mathcal{R}_{1_X+1}(P, Q)} \begin{bmatrix} X_0 \\ \vdots \\ X_{1_X} \\ Y_0 \\ \vdots \\ Y_{1_X} \end{bmatrix} = \underbrace{\begin{bmatrix} F_0 \\ \vdots \\ F_{1_p} \\ F_{1_p+1} \\ \vdots \\ F_{1_p+1_X} \end{bmatrix}}_F,$$

which is an overdetermined system of equations due to the degree constraint (**deg**). Therefore Problem 4.13 can be written as

$$\begin{aligned} &\text{minimize} \quad \text{over } \hat{P}, \hat{Q} \in \mathbb{R}^{1_p+1} \text{ and } X, Y \in \mathbb{R}^{1_X+1} \quad \left\| \begin{bmatrix} P \\ Q \end{bmatrix} - \begin{bmatrix} \hat{P} \\ \hat{Q} \end{bmatrix} \right\|_2 \\ &\text{subject to} \quad \mathcal{R}_{1_X+1}(\hat{P}, \hat{Q}) \begin{bmatrix} X \\ Y \end{bmatrix} = F. \end{aligned}$$

Problem 4.13 is equivalent to Problem SLRA with

- norm $\|\cdot\| = \|\cdot\|_2$,
- data matrix

$$\mathcal{S}(p) = \begin{bmatrix} [F_0 \ F_1 \ \cdots \ F_{1_p+1_X}] \\ \mathcal{R}_{1_X+1}^\top(P, Q) \end{bmatrix},$$

composed of a fixed block and a Sylvester structured block, and

- rank reduction by one.

Indeed

$$\mathcal{R}_{1_X+1}(\hat{P}, \hat{Q}) \begin{bmatrix} X \\ Y \end{bmatrix} = F \iff \text{rank}(\mathcal{S}(\hat{p})) \leq 21_X + 1.$$

Exercise 4.14 Implement the method for pole placement by low-order controller, outlined in this section, using the function `slra`. Test it on simulation examples and compare the results with the ones obtained with the MATLAB function `place`.

4.5 Simulation Examples

Model Reduction

In this section, we compare the nuclear norm heuristic-based methods, developed in Sect. 3.3, for structured low rank approximation with classical methods for Hankel structured low rank approximation, such as Kung's method and local optimization-based methods, on single input single output model reduction problems. The order n of the system \mathcal{B}_d , the bound n_{red} for the order of the reduced system $\widehat{\mathcal{B}}$, and the approximation horizon T are simulation parameters. The system \mathcal{B} with the specified order n is selected as a random stable single input single output system.

125a *(Test model reduction 125a)* \equiv 125b \triangleright
(initialize the random number generator 89e)
`sys = c2d(rss(n), 1);`
`h = impulse(sys, T); sh = h(2:(T + 1));`

Defines:

`test_mod_red`, used in chunk 126b.

The nuclear norm approximation is computed by the function `slra_nn`.

125b *(Test model reduction 125a)* \equiv \triangleleft 125a 125c \triangleright
`tts = blkhank(1:T, r + 1);`
`hh1 = [h(1); slra_nn(tts, sh, r)];`

Uses `blkhank` 25b and `slra_nn` 98c.

Kung's method is a singular value decomposition-based heuristic and is implemented in the function `h2ss`.

125c *(Test model reduction 125a)* \equiv \triangleleft 125b 125d \triangleright
`[sysh2, hh2] = h2ss(h, r); hh2 = hh2(:);`

Uses `h2ss` 76d.

Model reduction, based on local optimization, is done by solving Hankel structured low rank approximation problem, using the function `mod_red`.

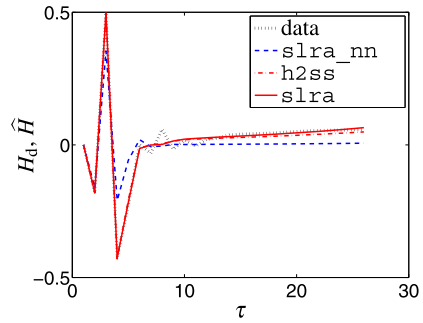
125d *(Test model reduction 125a)* \equiv \triangleleft 125c 125e \triangleright
`[sysh3, hh3] = mod_red(sys, T, r); hh3 = hh3(:);`

Uses `mod_red` 111.

We compare the results by checking the (numerical) rank of $\mathcal{H}_{n_{\text{red}}+1}(\sigma \widehat{H})$ and the approximation error $\|H - \widehat{H}\|_2$.

125e *(Test model reduction 125a)* \equiv \triangleleft 125d 126a \triangleright
`sh = h(2:end); shh1 = hh1(2:end);`
`shh2 = hh2(2:end); shh3 = hh3(2:end);`
`sv = [svd(sh(tts)) svd(shh1(tts))`
`svd(shh2(tts)) svd(shh3(tts))]`
`cost = [norm(h - h) norm(h - hh1)`
`norm(h - hh2) norm(h - hh3)]`

Fig. 4.3 Impulse responses of the high-order system (*dotted line*) and the low-order approximations obtained by the nuclear norm heuristic (*dashed line*), Kung's method (*dashed dotted line*), and a local optimization-based method (*solid line*)



The Markov parameters of the high-order system and the low-order approximations are plotted for visual inspection of the quality of the approximations.

```
126a <Test model reduction 125a>+≡ <125e
    plot(h, ':k', 'linewidth', 4), hold on,
    plot(hh1, 'b-'), plot(hh2, 'r-.'), plot(hh3, 'r-')
    legend('high order sys.', 'nucnrm', 'kung', 'slra')
```

The results of an approximation of a 50th-order system by a 4th-order system with time horizon 25 time steps

```
126b <Test structured low rank approximation methods on a model reduction problem 126b>≡
    n = 50; r = 4; T = 25; test_mod_red,
    print_fig('test_mod_red')
```

Uses `print_fig` 25a and `test_mod_red` 125a.
are

sv =

0.8590	0.5660	0.8559	0.8523
0.6235	0.3071	0.6138	0.6188
0.5030	0.1009	0.4545	0.4963
0.1808	0.0190	0.1462	0.1684
0.1074	0.0000	0.0000	0.0000

cost =

0	0.3570	0.0904	0.0814
---	--------	--------	--------

and Fig. 4.3. The nuclear norm approximation gives worse results than the singular value decomposition-based heuristic, which in turn can be further improved by the local optimization heuristic.

System Identification

Next we compare the nuclear norm heuristic with an alternative heuristic method, based on the singular value decomposition, and a method based on local optimiza-

tion, on single input single output system identification problems. The data are generated according to the errors-in-variables model (EIV). A trajectory w_0 of a linear time-invariant system \mathcal{B}_0 of order n_0 is corrupted by noise, where the noise \tilde{w} is zero mean, white, Gaussian with covariance σ^2 , i.e.,

$$w_d = w_0 + \tilde{w}.$$

The system \mathcal{B}_0 , referred to as the “true system”, is generated as a random stable single input single output system. The trajectory w_0 is then generated as a random trajectory of \mathcal{B}_0 . The order n_0 , the trajectory length T , and the noise standard deviation σ are simulation parameters. The approximation order is set equal to the order of the true system.

```
127a <Test system identification 127a>≡ 127b>
      (initialize the random number generator 89e)
      sys0 = drss(n0); u0 = randn(T, 1); xini0 = randn(n0, 1);
      y0 = lsim(sys0, u0, 1:T, xini0); w0 = [u0'; y0'];
      w = w0 + n1 * randn(size(w0)); n = n0;
```

Defines:

test_sysid, used in chunk 128.

The nuclear norm approximation is computed by

```
127b <Test system identification 127a>+≡ <127a 127c>
      tts = blkhank(reshape(1:(2 * T), 2, T), n + 1);
      wh1 = slra_nn(tts, w(:, 2 * n + 1));
```

Uses blkhank 25b and slra_nn 98c.

Next we apply an identification method obtained by solving suboptimally the Hankel structured low rank approximation problem ignoring the structure:

```
127c <Test system identification 127a>+≡ <127b 127d>
      (suboptimal approximate single input single output system identification 88c)
      [M2, wh2] = misfit_siso(w, P);
```

Uses misfit_siso 87b.

Finally, we apply the optimization-based method `ident_siso`

```
127d <Test system identification 127a>+≡ <127c 127e>
      [sysh3, wh3, info] = ident_siso(w, n); M3 = info.M;
```

Uses `ident_siso` 88a.

The results are compared by checking the rank constraint, the approximation error,

```
127e <Test system identification 127a>+≡ <127d 127f>
      sv = [svd(wh1(tts)) svd(wh2(tts)) svd(wh3(tts))]
      cost = [norm(w(:) - wh1) norm(w(:)
              - wh2(:)) norm(w(:) - wh3(:))]
```

and plotting the approximations on top of the data.

```
127f <Test system identification 127a>+≡ <127e
      plot(w(2, :), 'k'), hold on, plot(wh1(2:2:end), 'r')
      plot(wh2(2, :), 'b-'), plot(wh3(2, :), 'c-.')
      legend('data', 'nucnm', 'lra', 'slra')
```

The results of

```
128 (Test structured low rank approximation methods on system identification 128)≡
    n0 = 2; T = 25; nl = 0.2; test_sysid,
    print_fig('test_sysid')
```

Uses `print_fig` 25a and `test_sysid` 127a.

are

sv =

3.3862	4.6877	4.7428
2.8827	4.2789	4.2817
2.8019	4.1504	4.1427
0.6283	1.2102	1.3054
0.0000	0.3575	0.4813
0.0000	0.0000	0.0000

cost =

1.6780	0.8629	0.6676
--------	--------	--------

They are consistent with previous experiments: the nuclear norm approximation gives worse results than the singular value decomposition-based heuristic, which in turn can be further improved by the local optimization heuristic.

4.6 Notes

System Theory

Survey on applications of structured low rank approximation is given in De Moor (1993) and Markovsky (2008). Approximate realization is a special identification problem (the input is a pulse and the initial conditions are zeros). Nevertheless the exact version of this problem is a well studied problem. The classical references are the Ho-Kalman's realization algorithm (Ho and Kalman 1966), see also Kalman et al. (1969, Chap. 10). Approximate realization methods, based on the singular value decomposition, are proposed in Kung (1978), Zeiger and McEwen (1974).

Comprehensive treatment of model reduction methods is given in Antoulas (2005). The balanced model reduction method is proposed by Moore (1981) and error bounds are derived by Glover (1984). Proper orthogonal decomposition is a popular method for nonlinear model reduction. This method is unstructured low rank approximation of a matrix composed of “snapshots” of the state vector of the system. The method is data-driven in the sense that the method operates on data of the full-order system and a model of that system is not derived.

Errors-in-variables system identification methods are developed in Aoki and Yue (1970), Roorda and Heij (1995), Roorda (1995), Lemmerling and De Moor (2001), Pintelon et al. (1998) and Markovsky et al. (2005). Their consistency properties are

studied in Pintelon and Schoukens (2001) and Kukush et al. (2005). A survey paper on errors-in-variables system identification is Söderström (2007). Most of the work on the subject is presented in the classical input/output setting, i.e., the proposed methods are defined in terms of transfer function, matrix fraction description, or input/state/output representations. The salient feature of the errors-in-variables problems, however, is that all variables are treated on an equal footing as noise corrupted. Therefore, the input/output partitioning implied by the classical model representations is irrelevant in the errors-in-variables problem.

Signal Processing

Linear prediction methods based on optimization techniques are developed in Bresler and Macovski (1986) and Cadzow (1988). Cadzow (1988) proposed a method for Hankel structured low rank approximation that alternates between unstructured low rank approximation and structure approximation. This method, however, does not converge to a locally optimal solution. See De Moor (1994) for a counter example. Application of structured low rank approximation methods for audio processing is described in Lemmerling et al. (2003). The so called shape from moments problem (Milanfar et al. 1995; Golub et al. 1999; Gustafsson et al. 2000) is equivalent to Hankel structured low rank approximation (Schuermans et al. 2006).

Computer Vision

An overview of the application of low rank approximation (total least squares) in motion analysis is given in Mühlich and Mester (1998). Image deblurring applications are presented in Pruessner and O’Leary (2003), Mastronardi et al. (2004) and Fu and Barlow (2004). The image deblurring problem is solved by regularized structured low rank approximation methods in Younan and Fan (1998), Mastronardi et al. (2005), Ng et al. (2000, 2002).

Analysis Problems

The distance to uncontrollability with respect to the distance measure dist is naturally defined as

$$\min_{\widehat{\mathcal{B}} \in \mathcal{L}_{\text{ctrlb}}} \text{dist}(\mathcal{B}, \widehat{\mathcal{B}}).$$

The distance to uncontrollability proposed in (Paige 1981, Sect. X) matches the definition given in this book by taking as distance measure

$$\text{dist}(\mathcal{B}, \widehat{\mathcal{B}}) = \|[A \ B] - [\widehat{A} \ \widehat{B}]\|_2^2, \quad (*)$$

where A , B and \widehat{A} , \widehat{B} are parameters of input/state/output representations

$$\mathcal{B} = \mathcal{B}_{i/s/o}(A, B, C, D) \quad \text{and} \quad \widehat{\mathcal{B}} = \mathcal{B}_{i/s/o}(\widehat{A}, \widehat{B}, \widehat{C}, \widehat{D}),$$

respectively. Computing distance to uncontrollability with respect to $(*)$ received significant attention in the literature, but it has a drawback: $(*)$ depends on the choice of the state space basis. In other words, $(*)$ is representation dependent and, therefore, not a genuine property of the pair of systems $(\mathcal{B}, \widehat{\mathcal{B}})$.

References

- Antoulas A (2005) Approximation of large-scale dynamical systems. SIAM, Philadelphia
- Aoki M, Yue P (1970) On a priori error estimates of some identification methods. IEEE Trans Autom Control 15(5):541–548
- Bresler Y, Macovski A (1986) Exact maximum likelihood parameter estimation of superimposed exponential signals in noise. IEEE Trans Acoust Speech Signal Process 34:1081–1089
- Cadzow J (1988) Signal enhancement—a composite property mapping algorithm. IEEE Trans Signal Process 36:49–62
- De Moor B (1993) Structured total least squares and L_2 approximation problems. Linear Algebra Appl 188–189:163–207
- De Moor B (1994) Total least squares for affinely structured matrices and the noisy realization problem. IEEE Trans Signal Process 42(11):3104–3113
- Fu H, Barlow J (2004) A regularized structured total least squares algorithm for high-resolution image reconstruction. Linear Algebra Appl 391(1):75–98
- Glover K (1984) All optimal Hankel-norm approximations of linear multivariable systems and their l^∞ -error bounds. Int J Control 39(6):1115–1193
- Golub G, Milanfar P, Varah J (1999) A stable numerical method for inverting shape from moments. SIAM J Sci Comput 21:1222–1243
- Gustafsson B, He C, Milanfar P, Putinar M (2000) Reconstructing planar domains from their moments. Inverse Probl 16:1053–1070
- Ho BL, Kalman RE (1966) Effective construction of linear state-variable models from input/output functions. Regelungstechnik 14(12):545–592
- Kalman RE, Falb PL, Arbib MA (1969) Topics in mathematical system theory. McGraw-Hill, New York
- Kukush A, Markovsky I, Van Huffel S (2005) Consistency of the structured total least squares estimator in a multivariate errors-in-variables model. J Stat Plan Inference 133(2):315–358
- Kung S (1978) A new identification method and model reduction algorithm via singular value decomposition. In: Proc 12th asilomar conf circuits, systems, computers, Pacific Grove, pp 705–714
- Lemmerling P, De Moor B (2001) Misfit versus latency. Automatica 37:2057–2067
- Lemmerling P, Mastronardi N, Van Huffel S (2003) Efficient implementation of a structured total least squares based speech compression method. Linear Algebra Appl 366:295–315
- Markovsky I (2008) Structured low-rank approximation and its applications. Automatica 44(4):891–909
- Markovsky I, Willems JC, Van Huffel S, Moor BD, Pintelon R (2005) Application of structured total least squares for system identification and model reduction. IEEE Trans Autom Control 50(10):1490–1500
- Mastronardi N, Lemmerling P, Kalsi A, O’Leary D, Van Huffel S (2004) Implementation of the regularized structured total least squares algorithms for blind image deblurring. Linear Algebra Appl 391:203–221

- Mastronardi N, Lemmerling P, Van Huffel S (2005) Fast regularized structured total least squares algorithm for solving the basic deconvolution problem. *Numer Linear Algebra Appl* 12(2–3):201–209
- Milanfar P, Verghese G, Karl W, Willisky A (1995) Reconstructing polygons from moments with connections to array processing. *IEEE Trans Signal Process* 43:432–443
- Moore B (1981) Principal component analysis in linear systems: controllability, observability and model reduction. *IEEE Trans Autom Control* 26(1):17–31
- Mühlich M, Mester R (1998) The role of total least squares in motion analysis. In: Burkhardt H (ed) *Proc 5th European conf computer vision*. Springer, Berlin, pp 305–321
- Ng M, Plemmons R, Pimentel F (2000) A new approach to constrained total least squares image restoration. *Linear Algebra Appl* 316(1–3):237–258
- Ng M, Koo J, Bose N (2002) Constrained total least squares computations for high resolution image reconstruction with multisensors. *Int J Imaging Syst Technol* 12:35–42
- Paige CC (1981) Properties of numerical algorithms related to computing controllability. *IEEE Trans Autom Control* 26:130–138
- Pintelon R, Schoukens J (2001) *System identification: a frequency domain approach*. IEEE Press, Piscataway
- Pintelon R, Guillaume P, Vandersteen G, Rolain Y (1998) Analyses, development, and applications of TLS algorithms in frequency domain system identification. *SIAM J Matrix Anal Appl* 19(4):983–1004
- Pruessner A, O’Leary D (2003) Blind deconvolution using a regularized structured total least norm algorithm. *SIAM J Matrix Anal Appl* 24(4):1018–1037
- Roorda B (1995) Algorithms for global total least squares modelling of finite multivariable time series. *Automatica* 31(3):391–404
- Roorda B, Heij C (1995) Global total least squares modeling of multivariate time series. *IEEE Trans Autom Control* 40(1):50–63
- Schuermans M, Lemmerling P, Lathauwer LD, Van Huffel S (2006) The use of total least squares data fitting in the shape from moments problem. *Signal Process* 86:1109–1115
- Söderström T (2007) *Errors-in-variables methods in system identification*. *Automatica* 43:939–958
- Younan N, Fan X (1998) Signal restoration via the regularized constrained total least squares. *Signal Process* 71:85–93
- Zeiger H, McEwen A (1974) Approximate linear realizations of given dimension via Ho’s algorithm. *IEEE Trans Autom Control* 19:153

Part II
Miscellaneous Generalizations

Chapter 5

Missing Data, Centering, and Constraints

5.1 Weighted Low Rank Approximation with Missing Data

Modeling problems with missing data occur in

- factor analysis of data from questioners due to questions left unanswered,
- computer vision due to occlusions,
- signal processing due to irregular measurements in time/space, and
- control due to malfunction of measurement devices.

In this section, we present a method for linear static modeling with missing data. The problem is posed as an element-wise weighted low rank approximation problem with nonnegative weights, i.e.,

$$\text{minimize over } \widehat{D} \quad \|D - \widehat{D}\|_{\Sigma} \quad \text{subject to} \quad \text{rank}(\widehat{D}) \leq m, \quad (\text{EWLRA})$$

where

$$\|\Delta D\|_{\Sigma} := \|\Sigma \odot \Delta D\|_F = \sqrt{\sum_{i=1}^q \sum_{j=1}^N \sigma_{ij} e_{ij}}, \quad \Sigma \in \mathbb{R}^{q \times N} \text{ and } \Sigma \geq 0$$

is a seminorm. (The σ_{ij} 's are weights; not noise standard deviations.) In the extreme case of a zero weight, e.g., $\sigma_{ij} = 0$, the corresponding element d_{ij} of D is not taken into account in the approximation and therefore it is treated as a missing value. In this case, the approximation problem is called a singular problem. The algorithms, described in Chap. 3, for the regular weighted low rank approximation problem fail in the singular case. In this section, the methods are extended to solve singular problems and therefore account for missing data.

Exercise 5.1 (Missing rows and columns) Show that in the case of missing rows and/or columns of the data matrix, the singular low rank approximation problem reduces to a smaller dimensional regular problem.

Using the image representation, we have

$$\text{rank}(\widehat{D}) \leq m \iff \text{there are } P \in \mathbb{R}^{q \times m} \text{ and } L \in \mathbb{R}^{m \times N},$$

$$\text{such that } \widehat{D} = PL, \tag{RRF}$$

which turns problem (EWLRA) into the following parameter optimization problem

$$\text{minimize over } P \in \mathbb{R}^{q \times m} \text{ and } L \in \mathbb{R}^{m \times N} \quad \|D - PL\|_{\Sigma}. \tag{EWLRA}_P$$

Unfortunately the problem is nonconvex and there are no efficient methods to solve it. We present local optimization methods based on the alternating projections and variable projection approaches. These methods are initialized by a suboptimal solution of (EWLRA_P) , computed by a direct method.

Algorithms

Direct Method

The initial approximation for the iterative optimization methods is obtained by solving unweighted low rank approximation problem where all missing elements (encoded as NaN's) are filled in by zeros.

136a *(Low rank approximation with missing data 136a)* \equiv
`function [p, l] = lra_md(d, m)`
`d(isnan(d)) = 0; [q, N] = size(d);`
`if nargin == 1, (data compression 137), end`
(matrix approximation 136b)

Defines:

`lra_md`, used in chunks 140a and 143e.

The problem is solved using the singular value decomposition, however, in view of the large scale of the data the function `svds` which computes selected singular values and corresponding singular vectors is used instead of the function `svd`.

136b *(matrix approximation 136b)* \equiv (136a)
`[u, s, v] = svds(d, m); p = u(:, 1:m); l = p' * d;`

The model $\widehat{\mathcal{B}} = (P)$ depends only on the left singular vectors of the data matrix D . Therefore, $\widehat{\mathcal{B}}$ is an optimal model for the data DQ , where Q is any orthogonal matrix. Let

$$D^{\top} = Q \begin{bmatrix} R_1 \\ 0 \end{bmatrix}, \quad \text{where } R_1 \text{ is upper triangular} \tag{QR}$$

be the QR factorization of D^{\top} . For $N \gg q$, computing the QR factorization and the singular value decomposition of R_1 is a more efficient alternative for finding an image representation of the optimal subspace than computing the singular value decomposition of D .

$$\begin{aligned}
 & \text{(data compression 137)} \equiv & (136a) \\
 & \mathbf{d} = \text{triu}(\text{qr}(\mathbf{d}'))'; \mathbf{d} = \mathbf{d}(1:\mathbf{q}, :); \\
 & \text{(After these assignments, the variable } \mathbf{d} \text{ is equal to } R_1.)
 \end{aligned}$$

Alternating Projections

The alternating projections method exploits the fact that problem (EWLRA_P) is a linear least squares problem in either P or L . This suggests a method of alternatively minimizing over P with L fixed to the value computed on the previous iteration step and minimizing over L with P fixed to the value computed on the previous iteration step. A summary of the alternating projections method is given in Algorithm 3. MATLAB-like notation is used for indexing a matrix. For a $\mathbf{q} \times N$ matrix D and subsets \mathcal{I} and \mathcal{J} of the sets of, respectively, row and column indices, $D_{\mathcal{I}, \mathcal{J}}$ denotes the submatrix of D with elements whose indices are in \mathcal{I} and \mathcal{J} . Either of \mathcal{I} and \mathcal{J} can be replaced by “:” in which case all rows/columns are indexed.

On each iteration step of the alternating projections algorithm, the cost function value is guaranteed to be non-increasing and is typically decreasing. It can be shown that the iteration converges and that the local convergence rate is linear.

The quantity $e^{(k)}$, computed on step 9 of the algorithm is the squared approximation error

$$e^{(k)} = \|D - \widehat{D}^{(k)}\|_{\Sigma}^2$$

on the k th iteration step. Convergence of the iteration is judged on the basis of the relative decrease of the error $e^{(k)}$ after an update step. This corresponds to choosing a tolerance on the relative decrease of the cost function value. More expensive alternatives are to check the convergence of the approximation $\widehat{D}^{(k)}$ or the size of the gradient of the cost function with respect to the model parameters.

Variable Projections

In the second solution method, (EWLRA_P) is viewed as a double minimization problem

$$\text{minimize over } P \in \mathbb{R}^{\mathbf{q} \times \mathbf{m}} \quad \underbrace{\min_{L \in \mathbb{R}^{\mathbf{m} \times N}} \|D - PL\|_{\Sigma}^2}_{f(P)}. \quad (\text{EWLRA}'_P)$$

The inner minimization is a weighted least squares problem and therefore can be solved in closed form. Using MATLAB set indexing notation, the solution is

$$f(P) = \sum_{j=1}^N D_{\mathcal{I}, j}^{\top} \text{diag}(\Sigma_{\mathcal{I}, j}^2) P_{\mathcal{I}, :} (P_{\mathcal{I}, :}^{\top} \text{diag}(\Sigma_{\mathcal{I}, j}^2) P_{\mathcal{I}, :})^{-1} P_{\mathcal{I}, :}^{\top} \text{diag}(\Sigma_{\mathcal{I}, j}^2) D_{\mathcal{I}, j}, \quad (5.2)$$

where \mathcal{I} is the set of indices of the non-missing elements in the j th column of D .

Algorithm 3 Alternating projections algorithm for weighted low rank approximation with missing data

Input: Data matrix $D \in \mathbb{R}^{q \times N}$, rank constraint m , element-wise nonnegative weight matrix $\Sigma \in \mathbb{R}^{q \times N}$, and relative convergence tolerance ε .

1: Initial approximation: compute the Frobenius norm low rank approximation of D with missing elements filled in with zeros

$$P^{(0)} := \text{lra_md}(D, m).$$

2: Let $k := 0$.

3: **repeat**

4: Let $e^{(k)} := 0$.

5: **for** $j = 1, \dots, N$ **do**

6: Let \mathcal{J} be the set of indices of the non-missing elements in $D_{:,j}$.

7: Define

$$c := \text{diag}(\Sigma_{\mathcal{J},j}) D_{\mathcal{J},j} = \Sigma_{\mathcal{J},j} \odot D_{\mathcal{J},j}$$

$$P := \text{diag}(\Sigma_{\mathcal{J},j}) P_{\mathcal{J},:}^{(k)} = (\Sigma_{\mathcal{J},j} \mathbf{1}_m^\top) \odot P_{\mathcal{J},:}^{(k)};$$

8: Compute

$$\ell_j^{(k)} := (P^\top P)^{-1} P^\top c.$$

9: Let

$$e^{(k)} := e^{(k)} + \|c - P \ell_j^{(k)}\|^2.$$

10: **end for**

11: Define

$$L^{(k)} = \begin{bmatrix} \ell_1^{(k)} & \dots & \ell_N^{(k)} \end{bmatrix}.$$

12: Let $e^{(k+1)} := 0$.

13: **for** $i = 1, \dots, q$ **do**

14: Let \mathcal{J} be the set of indices of the non-missing elements in the i th row $D_{i,:}$.

15: Define

$$r := D_{i,\mathcal{J}} \text{diag}(\Sigma_{i,\mathcal{J}}) = D_{i,\mathcal{J}} \odot \Sigma_{i,\mathcal{J}}$$

$$L := L_{:, \mathcal{J}}^{(k)} \text{diag}(\Sigma_{i,\mathcal{J}}) = L_{:, \mathcal{J}}^{(k)} \odot (\mathbf{1}_m \Sigma_{i,\mathcal{J}}).$$

16: Compute

$$p_i^{(k+1)} := r L^\top (L L^\top)^{-1}.$$

17: Let

$$e^{(k+1)} := e^{(k+1)} + \|r - p_i^{(k+1)} L\|^2.$$

18: **end for**

19: Define

$$P^{(k+1)} = \begin{bmatrix} p_1^{(k+1)} \\ \vdots \\ p_q^{(k+1)} \end{bmatrix}.$$

20: $k = k + 1$.

21: **until** $|e^{(k)} - e^{(k-1)}|/e^{(k)} < \varepsilon$.

Output: Locally optimal solution $\widehat{D} = \widehat{D}^{(k)} := P^{(k)} L^{(k)}$ of (EWLRA_P).

Exercise 5.2 Derive the expression (5.2) for the function f in (EWLRA' $_p$).

The outer minimization is a nonlinear least squares problem and can be solved by general purpose local optimization methods. The inner minimization is a weighted projection on the subspace spanned by the columns of P . Consequently, $f(P)$ has the geometric interpretation of the sum of squared distances from the data points to the subspace. Since the parameter P is modified by the outer minimization, the projections are on a varying subspace.

Note 5.3 (Gradient and Hessian of f) In the implementation of the method, we are using finite difference numerical computation of the gradient and Hessian of f . (These approximations are computed by the optimization method.) More efficient alternative, however, is to supply to the method analytical expressions for the gradient and the Hessian.

Implementation

Both the alternating projections and the variable projections methods for solving weighted low rank approximation problems with missing data are callable through the function `wlra`.

```
139 (Weighted low rank approximation 139)≡
    function [p, l, info] = wlra(d, m, s, opt)
        tic, (default parameters opt 140a)
        switch lower(opt.Method)
            case {'altpro', 'ap'}
                (alternating projections method 140b)
            case {'varpro', 'vp'}
                (variable projections method 141d)
            otherwise
                error('Unknown method %s', opt.Method)
        end
        info.time = toc;
```

Defines:

`wlra`, used in chunks 143e and 230.

The output parameter `info` gives the

- approximation error $\|D - \widehat{D}\|_{\Sigma}^2$ (`info.err`),
- number of iterations (`info.iter`), and
- execution time (`info.time`) for computing the local approximation \widehat{D} .

The optional parameter `opt` specifies the

- method (`opt.Method`) and, in the case of the variable projections method, algorithm (`opt.alg`) to be used,
- initial approximation (`opt.P`),

- convergence tolerance ε (`opt.TolFun`),
- an upper bound on the number of iterations (`opt.MaxIter`), and
- level of printed information (`opt.Display`).

The initial approximation `opt.P` is a $q \times m$ matrix, such that the columns of $P^{(0)}$ form a basis for the span of the columns of $\widehat{D}^{(0)}$, where $\widehat{D}^{(0)}$ is the initial approximation of D , see step 1 in Algorithm 3. If it is not provided via the parameter `opt`, the default initial approximation is chosen to be the unweighted low rank approximation of the data matrix with all missing elements filled in with zeros.

Note 5.4 (Large scale, sparse data) In an application of (EWLRA) to building recommender systems, the data matrix D is large but only a small fraction of the elements are given. Such problems can be handled efficiently, encoding D and Σ as sparse matrices. The convention in this case is that missing elements are zeros.

```
140a <default parameters opt 140a>≡ (139)
    if ~exist('opt.MaxIter'), opt.MaxIter = 100; end
    if ~exist('opt.TolFun'), opt.TolFun = 1e-5; end
    if ~exist('opt.Display'), opt.Display = 'off'; end
    if ~exist('opt.Method'), opt.Method = 'ap'; end
    if ~exist('opt.alg'), opt.alg = 'lsqnonlin'; end
    if ~exist('opt.P'), p = lra_md(d, m); else p = opt.P, end
```

Uses `lra_md` 136a.

Alternating Projections

The iteration loop for the alternating projections algorithm is:

```
140b <alternating projections method 140b>≡ (139)
    [q, N] = size(d); sd = norm(s .* d, 'fro') ^ 2;
    cont = 1; k = 0;
    while (cont)
        <compute L, given P 140c>
        <compute P, given L 141a>
        <check exit condition 141b>
        <print progress information 141c>
    end
    info.err = e1; info.iter = k;
```

The main computational steps on each iteration of the algorithm are the two weighted least squares problems.

```
140c <compute L, given P 140c>≡ (140b 142)
    dd = []; % vec(D - DH)
    for j = 1:N
        J = find(s(:, j));
        sJj = full(s(J, j));
```

```

c = sJj .* full(d(J, j));
P = sJj(:, ones(1, m)) .* p(J, :);
% = diag(sJj) * p(J, :);
l(:, j) = P \ c; dd = [dd; c - P * l(:, j)];
end
ep = norm(dd) ^ 2;

```

141a *(compute P, given L 141a)*≡ (140b)

```

dd = []; % vec(D - DH)
for i = 1:q
    I = find(s(i, :));
    sIi = full(s(i, I));
    r = sIi .* full(d(i, I));
    L = sIi(ones(m, 1), :) .* l(:, I);
    % = l(:, I) * diag(sIi)
    p(i, :) = r / L; dd = [dd, r - p(i, :) * L];
end
e1 = norm(dd) ^ 2;

```

The convergence is checked by the size of the relative decrease in the approximation error $e^{(k)}$ after one update step.

141b *(check exit condition 141b)*≡ (140b)

```

k = k + 1; re = abs(e1 - ep) / e1;
cont = (k < opt.MaxIter) & (re > opt.TolFun) & (e1 > eps);

```

If the optional parameter `opt.Display` is set to 'iter', `wlra` prints on each iteration step the relative approximation error.

141c *(print progress information 141c)*≡ (140b)

```

switch lower(opt.Display)
    case 'iter',
        fprintf('%2d : relative error = %18.8f\n', k, e1 / sd)
end

```

Variable Projections

Optimization Toolbox is used for performing the outer minimization in $(EWLRA'_p)$, i.e., the nonlinear minimization over the P parameter. The parameter `opt.alg` specifies the algorithm to be used. The available options are

- `fminunc`—a quasi-Newton type algorithm, and
- `lsqnonlin`—a nonlinear least squares algorithm.

Both algorithm allow for numerical approximation of the gradient and Hessian through finite difference computations. In version of the code shown next, the numerical approximation is used.

141d *(variable projections method 141d)*≡ (139)

```

switch lower(opt.alg)
    case {'fminunc'}
        [p, err, f, info] = fminunc(@(p)mwlra(p, d, s), p, opt);

```

```

case {'lsqnonlin'}
    [p, rn, r, f, info] = ...
        lsqnonlin(@(p)mwlra2(p, d, s), p, [], []);
otherwise
    error('Unknown algorithm %s.', opt.alg)
end
[info.err, l] = mwlra(p, d, s); % obtain the L parameter

```

Uses `mwlra` 142a and `mwlra2` 142b.

The inner minimization in $(EWLRA'_p)$ has an analytic solution (5.2). The implementation of (5.2) is the chunk of code for computing the L parameter, given the P parameter, already used in the alternating projections algorithm.

```

142a <dist( $\mathcal{D}, \mathcal{B}$ ) (weighted low rank approximation) 142a>≡
    function [ep, l] = mwlra(p, d, s)
        N = size(d, 2); m = size(p, 2); <compute L, given P 140c>

```

Defines:

`mwlra`, used in chunk 141d.

In the case of using a nonlinear least squares type algorithm, the cost function is not the sum of squares of the errors but the correction matrix ΔD (`dd`).

```

142b <Weighted low rank approximation correction matrix 142b>≡
    function dd = mwlra2(p, d, s)
        N = size(d, 2); m = size(p, 2); <compute L, given P 140c>

```

Defines:

`mwlra2`, used in chunk 141d.

Test on Simulated Data

A “true” random rank- m matrix D_0 is selected by generating randomly its factors P_0 and L_0 in a rank revealing factorization

$$D_0 = P_0 L_0, \quad \text{where } P_0 \in \mathbb{R}^{q \times m} \text{ and } L_0 \in \mathbb{R}^{m \times N}.$$

```

142c <Test missing data 2 142c>≡ 142d>
    <initialize the random number generator 89e>
    p0 = rand(q, m); l0 = rand(m, N);

```

Defines:

`test_missing_data2`, used in chunks 144 and 145.

The location of the given elements is chosen randomly row by row. The number of given elements is such that the sparsity of the resulting matrix, defined as the ratio of the number of missing elements to the total number qN of elements, matches the specification r .

The number of given elements of the data matrix is

```

142d <Test missing data 2 142c>+≡ <142c 143a>
    ne = round((1 - r) * q * N);

```

Then the number of given elements of the data matrix per row is

```
143a <Test missing data 2 142c>+≡ <142d 143b>
    ner = round(ne / q);
```

The variables `I` and `J` contain the row and column indices of the given elements. They are randomly chosen.

```
143b <Test missing data 2 142c>+≡ <143a 143c>
    I = []; J = [];
    for i = 1:q
        I = [I i*ones(1, ner)]; rp = randperm(N);
        J = [J rp(1:ner)];
    end
    ne = length(I);
```

By construction there are `ner` given elements in each row of the data matrix, however, there may be columns with a few (or even zero) given elements. Columns with less than `m` given elements cannot be recovered from the given observations, even when the data are noise-free. Therefore, we remove such columns from the data matrix.

```
143c <Test missing data 2 142c>+≡ <143b 143d>
    tmp = (1:N)';
    J_del = find(sum(J(ones(N, 1), :) ...
        == tmp(:, ones(1, ne)), 2) < m);
    l0(:, J_del) = [];
    tmp = sparse(I, J, ones(ne, 1), q, N); tmp(:, J_del) = [];
    [I, J] = find(tmp); N = size(l0, 2);
```

Next, a noisy data matrix with missing elements is constructed by adding to the true values of the given data elements independent, identically, distributed, zero mean, Gaussian noise, with a specified standard deviation `s`. The weight matrix Σ is binary: $\sigma_{ij} = 1$ if d_{ij} is given and $\sigma_{ij} = 0$ if d_{ij} is missing.

```
143d <Test missing data 2 142c>+≡ <143c 143e>
    d0 = p0 * l0;
    Ie = I + q * (J - 1);
    d = zeros(q * N, 1);
    d(Ie) = d0(Ie) + sigma * randn(size(d0(Ie)));
    d = reshape(d, q, N);
    s = zeros(q, N); s(Ie) = 1;
```

The methods implemented in `lra` and `wlra` are applied on the noisy data matrix D with missing elements and the results are validated against the complete true matrix D_0 .

```
143e <Test missing data 2 142c>+≡ <143d 144a>
    tic, [p0, l0] = lra_md(d, m); t0 = toc;
    err0 = norm(s .* (d - p0 * l0), 'fro') ^ 2;
    e0 = norm(d0 - p0 * l0, 'fro') ^ 2;
    [ph1, lh1, info1] = wlra(d, m, s);
    e1 = norm(d0 - ph1 * lh1, 'fro') ^ 2;
    opt.Method = 'vp'; opt.alg = 'fminunc';
    [ph2, lh2, info2] = wlra(d, m, s, opt);
```

```
e2 = norm(d0 - ph2 * lh2, 'fro') ^ 2;
opt.Method = 'vp'; opt.alg = 'lsqnonlin';
[ph3, lh3, info3] = wlra(d, m, s, opt);
e3 = norm(d0 - ph3 * lh3, 'fro') ^ 2;
```

Uses `lra_md` 136a and `wlra` 139.

For comparison, we use also a method for low rank matrix completion, called *singular value thresholding* (Cai et al. 2009). Although the singular value thresholding method is initially designed for the of exact case, it can cope with noisy data as well, i.e., solve low rank approximation problems with missing data. The method is based on convex relaxation of the rank constraint and does not require an initial approximation.

```
144a (Test missing data 2 142c)+≡ <143e 144b>
tau = 5 * sqrt(q * N); delta = 1.2 / (ne / q / N);
try
    tic, [U, S, V] = SVT([q N], Ie, d(Ie), tau, delta);
    t4 = toc;
    dh4 = U(:, 1:m) * S(1:m, 1:m) * V(:, 1:m)';
catch
    dh4 = NaN; t4 = NaN; % SVT not installed
end
err4 = norm(s .* (d - dh4), 'fro') ^ 2;
e4 = norm(d0 - dh4, 'fro') ^ 2;
```

The final result shows the

- relative approximation error $\|D - \widehat{D}\|_{\Sigma}^2 / \|D\|_{\Sigma}^2$,
- estimation error $\|D_0 - \widehat{D}\|_{\mathbb{F}}^2 / \|D_0\|_{\mathbb{F}}^2$, and
- computation time

for the five methods.

```
144b (Test missing data 2 142c)+≡ <144a>
nd = norm(s .* d, 'fro')^2; nd0 = norm(d0, 'fro') ^ 2;
format long
res =
[err0/nd info1.err/nd info2.err/nd info3.err/nd err4/nd;
 e0/nd0 e1/nd0 e2/nd0 e3/nd0 e4/nd0;
 t0 info1.time info2.time info3.time t4]
```

First, we call the test script with exact (noise-free) data.

```
144c (Missing data experiment 1: small sparsity, exact data 144c)≡
q = 10; N = 100; m = 2; r = 0.1; sigma = 0;
test_missing_data2
```

Uses `test_missing_data2` 142c.

The experiment corresponds to a matrix completion problem (Candés and Recht 2009). The results, summarized in Tables 5.1, show that all methods, except for `lra_md`, complete correctly (up to numerical errors) the missing elements. As proved by Candés and Recht (2009), exact matrix completion is indeed possible in the case of Experiment 1.

Table 5.1 Results for Experiment 1. (SVT—singular value thresholding, VP—variable projections)

	<code>lra_md</code>	<code>ap</code>	VP + <code>fminunc</code>	VP + <code>lsqnonlin</code>	SVT
$\ D - \widehat{D}\ _{\Sigma}^2 / \ D\ _{\Sigma}^2$	0.02	0	0	0	0
$\ D_0 - \widehat{D}\ _{\mathbb{F}}^2 / \ D\ _{\mathbb{F}}^2$	0.03	0	0	0	0
Execution time (sec)	0.04	0.18	0.17	0.18	1.86

Table 5.2 Results for Experiment 2

	<code>lra</code>	<code>ap</code>	VP + <code>fminunc</code>	VP + <code>lsqnonlin</code>	SVT
$\ D - \widehat{D}\ _{\Sigma}^2 / \ D\ _{\Sigma}^2$	0.049	0.0257	0.0257	0.0257	0.025
$\ D_0 - \widehat{D}\ _{\mathbb{F}}^2 / \ D\ _{\mathbb{F}}^2$	0.042	0.007	0.007	0.007	0.007
Execution time (sec)	0.04	0.11	0.11	0.11	1.51

Table 5.3 Results for Experiment 3

	<code>lra_md</code>	<code>ap</code>	VP + <code>fminunc</code>	VP + <code>lsqnonlin</code>	SVT
$\ D - \widehat{D}\ _{\Sigma}^2 / \ D\ _{\Sigma}^2$	0.17	0.02	0.02	0.02	0.02
$\ D_0 - \widehat{D}\ _{\mathbb{F}}^2 / \ D\ _{\mathbb{F}}^2$	0.27	0.018	0.018	0.018	0.17
Execution time (sec)	0.04	0.21	0.21	0.21	1.87

The second experiment is with noisy data.

145a *(Missing data experiment 2: small sparsity, noisy data 145a)*≡
`vq = 10; N = 100; m = 2; r = 0.1; sigma = 0.1;`
`test_missing_data2`

Uses `test_missing_data2 142c`.

The results, shown in Tables 5.2, indicate that the methods implemented in `wlra` converge to the same (locally) optimal solution. The alternating projections method, however, is about 100 times faster than the variable projections methods, using the Optimization Toolbox functions `fminunc` and `lsqnonlin`, and about 10 times faster than the singular value thresholding method. The solution produces by the singular value thresholding method is suboptimal but close to being (locally) optimal.

In the third experiment we keep the noise standard deviation the same as in Experiment 2 but increase the sparsity.

145b *(Missing data experiment 3: bigger sparsity, noisy data 145b)*≡
`q = 10; N = 100; m = 2; r = 0.4; sigma = 0.1;`
`test_missing_data2`

Uses `test_missing_data2 142c`.

The results, shown in Tables 5.3, again indicate that the methods implemented in `wlra` converge to the same (locally) optimal solutions. In this case, the singular

value thresholding method is further away from being (locally) optimal, but is still much better than the solution of `lra_md`—1% vs. 25% relative prediction error.

The three methods based on local optimization (`ap`, `VP + fminunc`, `VP + lsqnonlin`) need not compute the same solution even when started from the same initial approximation. The reason for this is that the methods only guarantee convergence to a locally optimal solution, however, the problem is non convex and may have multiple local minima. Moreover, the trajectories of the three methods in the parameter space are different because the update rules of the methods are different.

The computation times for the three methods are different. The number of floating point operations per iteration can be estimated theoretically which gives an indication which of the methods may be the faster per iteration. Note, however, that the number of iterations, needed for convergence, is not easily predictable, unless the methods are started “close” to a locally optimal solution. The alternating projections method is most efficient per iteration but needs most iteration steps. In the current implementation of the methods, the alternating projections method is still the winner of the three methods for large scale data sets because for q more than a few hundreds optimization methods are too computationally demanding. This situation may be improved by analytically computing the gradient and Hessian.

Test on the MovieLens Data

The MovieLens data sets were collected and published by the GroupLens Research Project at the University of Minnesota in 1998. Currently, they are recognized as a benchmark for predicting missing data in recommender systems. The “100K data set” consists of 100000 ratings of $q = 943$ users’ on $N = 1682$ movies and demographic information for the users. (The ratings are encoded by integers in the range from 1 to 5.) Here, we use only the ratings, which constitute a $q \times N$ matrix with missing elements. The task of a recommender system is to fill in the missing elements.

Assuming that the true complete data matrix is rank deficient, building a recommender system is a problem of low rank approximation with missing elements. The assumption that the true data matrix is low rank is reasonable in practice because user ratings are influenced by a few factors. Thus, we can identify typical users (related to different combinations of factors) and reconstruct the ratings of any user as a linear combination of the ratings of the typical users. As long as the typical users are fewer than the number of users, the data matrix is low rank. In reality, the number of factors is not small but there are a few dominant ones, so that the true data matrix is approximately low rank.

It turns out that two factors allow us to reconstruct the missing elements with 7.1% average error. The reconstruction results are validated by cross validation with 80% identification data and 20% validation data. Five such partitionings of the data are given on the MovieLens web site. The matrix

$$\Sigma_{\text{idt}}^{(k)} \in \{0, 1\}^{q \times N}$$

Table 5.4 Results on the MovieLens data

	lra_md	ap	SVT
Mean identification error e_{idt}	0.100	0.060	0.298
Mean prediction error e_{val}	0.104	0.071	0.307
Mean execution time (sec)	1.4	156	651

indicates the positions of the given elements in the k th partition:

- $\Sigma_{\text{idt},ij}^{(k)} = 1$ means that the element D_{ij} is used for identification and
- $\Sigma_{\text{idt},ij}^{(k)} = 0$ means that D_{ij} is missing.

Similarly, $\Sigma_{\text{val}}^{(k)}$ indicates the validation elements in the k th partition.

Table 5.4 shows the mean relative identification and validation errors

$$e_{\text{idt}} := \frac{1}{5} \sum_{k=1}^5 \|D - \widehat{D}^{(k)}\|_{\Sigma_{\text{idt}}^{(k)}}^2 / \|D\|_{\Sigma_{\text{idt}}^{(k)}}^2$$

and

$$e_{\text{val}} := \frac{1}{5} \sum_{k=1}^5 \|D - \widehat{D}^{(k)}\|_{\Sigma_{\text{val}}^{(k)}}^2 / \|D\|_{\Sigma_{\text{val}}^{(k)}}^2,$$

where $\widehat{D}^{(k)}$ is the reconstructed matrix in the k th partitioning of the data. The singular value thresholding method issues a message “Divergence!”, which explains the poor results obtained by this method.

5.2 Affine Data Modeling

Problem Formulation

Closely related to the linear model is the affine one. The observations

$$\mathcal{D} = \{d_1, \dots, d_N\}$$

satisfy an *affine static model* \mathcal{B} if $\mathcal{D} \subset \mathcal{B}$, where \mathcal{B} is an affine set, i.e., $\mathcal{B} = c + \mathcal{B}'$, with $\mathcal{B}' \subset \mathbb{R}^q$ a linear model and $c \in \mathbb{R}^q$ an offset vector. Obviously, the affine model class contains as a special case the linear model class. The parameter c , however, allows us to account for a constant offset in the data. Consider, for example, the data

$$\mathcal{D} = \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\},$$

which satisfy the affine model

$$\mathcal{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \{d \mid [1 \ 0]d = 0\}$$

but are not fitted by a linear model of dimension one.

Subtracting the offset c from the data vector d , reduces the affine modeling problem with known offset parameter to an equivalent linear modeling problem. In a realistic data modeling setup, however, the offset parameter is unknown and has to be identified together with the linear model \mathcal{B}' . An often used heuristic for solving this problem is to replace the offset c by the mean

$$\mathbf{E}(D) := \frac{1}{N} D \mathbf{1}_N = (d_1 + \cdots + d_N)/N \in \mathbb{R}^q,$$

where

$$D = [d_1 \cdots d_N]$$

is the data matrix and $\mathbf{1}_N$ is the vector in \mathbb{R}^N with all elements equal to one. This leads to the following two-stage procedure for identification of affine models:

1. *preprocessing step*: subtract the mean from the data points,
2. *linear identification step*: identify a linear model for the centered data.

When the aim is to derive an optimal in some specified sense approximate affine model, the two-stage procedure may lead to suboptimal results. Indeed, even if the data centering and linear identification steps are individually optimal with respect to the desired optimality criterion, their composition need not be optimal for the affine modeling problem, i.e., simultaneous subspace fitting and centering.

It is not clear a priori whether the two-stage procedure is optimal when combined with other data modeling approaches. It turns out that, in the case of low rank approximation in the Frobenius norm with no additional constraints, the two-stage procedure is optimal. It follows from the analysis that a solution is not unique. Also, counter examples show that in the more general cases of weighted and Hankel structured low rank approximation problems, the two-stage procedure is suboptimal. Methods based on the alternating projections and variable projections algorithms are developed in these cases.

Matrix Centering

The matrix centering operation is subtraction of the mean $\mathbf{E}(D)$ from all columns of the data matrix D :

$$\mathbf{C}(D) := D - \mathbf{E}(D) \mathbf{1}_N^\top = D \left(I - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^\top \right).$$

The following proposition justifies the name “matrix centering” for $\mathbf{C}(\cdot)$.

Proposition 5.5 (Matrix centering) *The matrix $\mathbf{C}(D)$ is column centered, i.e., its mean is zero:*

$$\mathbf{E}(\mathbf{C}(D)) = 0.$$

The proof is left as an exercise, see Problem P.21.

Next we give an interpretation of the mean computation as a simple optimal modeling problem.

Proposition 5.6 (Mean computation as an optimal modeling) *$\mathbf{E}(D)$ is solution of the following optimization problem:*

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{D} \text{ and } c && \|D - \widehat{D}\|_F \\ & \text{subject to} && && \widehat{D} = c\mathbf{1}_N^\top. \end{aligned}$$

The proof is left as an exercise, see Problem P.22.

Note 5.7 (Intercept) Data fitting with an intercept is a special case of centering when all but one of the row means are set to zero, i.e., centering of one row. Intercept is appropriate when an input/output partition of the variables is imposed and there is a single output that has an offset.

Unweighted Low Rank Approximation with Centering

In this section, we consider the low rank approximation problem in the Frobenius norm with centering:

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{D} \text{ and } c && \|D - c\mathbf{1}_N^\top - \widehat{D}\|_F \\ & \text{subject to} && && \text{rank}(\widehat{D}) \leq m. \end{aligned} \tag{LRA}_c$$

The following theorem shows that the two-stage procedure yields a solution to (LRA_c) .

Theorem 5.8 (Optimality of the two-stage procedure) *A solution to (LRA_c) is the mean of D , $c^* = \mathbf{E}(D)$, and an optimal in a Frobenius norm rank- m approximation \widehat{D}^* of the centered data matrix $\mathbf{C}(D)$.*

Proof Using a kernel representation of the rank constraint

$$\text{rank}(\widehat{D}) \leq m \iff \text{there is full rank matrix } R \in \mathbb{R}^{(q-m) \times q}, \text{ such that } R\widehat{D} = 0,$$

we have the following equivalent problem to (LRA_c)

$$\begin{aligned} & \text{minimize over } \widehat{D}, c, \text{ and } R \in \mathbb{R}^{(q-m) \times q} \quad \|D - c\mathbf{1}_N^\top - \widehat{D}\|_F^2 \\ & \text{subject to } R\widehat{D} = 0 \quad \text{and} \quad RR^\top = I_{q-m}. \end{aligned} \quad (\text{LRA}_{c,R})$$

The Lagrangian of $(\text{LRA}_{c,R})$ is

$$\begin{aligned} L(\widehat{D}, c, R, \Lambda, \mathcal{E}) := & \sum_{i=1}^q \sum_{j=1}^N (d_{ij} - c_i - \widehat{d}_{ij})^2 \\ & + 2\text{trace}(R\widehat{D}\Lambda) + \text{trace}(\mathcal{E}(I - RR^\top)). \end{aligned}$$

Setting the partial derivatives of L to zero, we obtain the necessary optimality conditions

$$\partial L / \partial \widehat{D} = 0 \implies D - c\mathbf{1}_N^\top - \widehat{D} = R^\top \Lambda^\top, \quad (\text{L1})$$

$$\partial L / \partial c = 0 \implies Nc = (D - \widehat{D})\mathbf{1}_N, \quad (\text{L2})$$

$$\partial L / \partial R = 0 \implies \widehat{D}\Lambda = R^\top \mathcal{E}, \quad (\text{L3})$$

$$\partial L / \partial \Lambda = 0 \implies R\widehat{D} = 0, \quad (\text{L4})$$

$$\partial L / \partial \mathcal{E} = 0 \implies RR^\top = I. \quad (\text{L5})$$

The theorem follows from the system of (L1-L5) . Next we list the derivation steps.

From (L3) , (L4) , and (L5) , it follows that $\mathcal{E} = 0$, and from (L1) we obtain

$$D - \widehat{D} = c\mathbf{1}_N^\top + R^\top \Lambda^\top.$$

Substituting the last identity in (L2) , we have

$$\begin{aligned} Nc = (c\mathbf{1}_N^\top + R^\top \Lambda^\top)\mathbf{1}_N = Nc + R^\top \Lambda^\top \mathbf{1}_N & \implies R^\top \Lambda^\top \mathbf{1}_N = 0 \\ & \implies \Lambda^\top \mathbf{1}_N = 0. \end{aligned}$$

Multiplying (L1) from the left by R and using (L4) and (L5) , we have

$$R(D - c\mathbf{1}_N^\top) = \Lambda^\top. \quad (*)$$

Now, multiplication of the last identity from the right by $\mathbf{1}_N$ and use of $\Lambda^\top \mathbf{1}_N = 0$, shows that c is the row mean of the data matrix D ,

$$R(D\mathbf{1}_N - Nc) = 0 \implies c = \frac{1}{N}D\mathbf{1}_N.$$

Next, we show that \widehat{D} is an optimal in a Frobenius norm rank- m approximation of $D - c\mathbf{1}_N^\top$. Multiplying (L1) from the right by Λ and using $\widehat{D}\Lambda = 0$, we have

$$(D - c\mathbf{1}_N^\top)\Lambda = R^\top \Lambda^\top \Lambda. \quad (**)$$

Defining

$$\Sigma := \sqrt{\Lambda^\top \Lambda} \quad \text{and} \quad V := \Lambda \Sigma^{-1},$$

(*) and (**) become

$$\begin{aligned} R(D - c\mathbf{1}_N^\top) &= \Sigma V^\top, & V^\top V &= I \\ (D - c\mathbf{1}_N^\top)V &= R^\top \Sigma, & RR^\top &= I. \end{aligned}$$

The above equations show that the rows of R and the columns of V span, respectively, left and right m -dimensional singular subspaces of the centered data matrix $D - c\mathbf{1}_N^\top$. The optimization criterion is minimization of

$$\|D - \widehat{D} - c\mathbf{1}_N^\top\|_F = \|R^\top \Lambda^\top\|_F = \sqrt{\text{trace}(\Lambda \Lambda^\top)} = \text{trace}(\Sigma).$$

Therefore, a minimum is achieved when the rows of R and the columns of V span the, respectively, left and right m -dimensional singular subspaces of the centered data matrix $D - c\mathbf{1}_N^\top$, corresponding to the m smallest singular values. The solution is unique if and only if the m th singular value is strictly bigger than the $(m + 1)$ st singular value. Therefore, \widehat{D} is a Frobenius norm optimal rank- m approximation of the centered data matrix $D - c\mathbf{1}_N^\top$, where $c = D\mathbf{1}_N/N$. \square

Theorem 5.9 (Nonuniqueness) *Let*

$$\widehat{D} = PL, \quad \text{where } P \in \mathbb{R}^{q \times m} \text{ and } L \in \mathbb{R}^{m \times N}$$

be a rank revealing factorization of an optimal in a Frobenius norm rank- m approximation of the centered data matrix $\mathbf{C}(D)$. The solutions of (LRA)_c are of the form

$$\begin{aligned} c^*(z) &= \mathbf{E}(D) + Pz \\ \widehat{D}^*(z) &= P(L - z\mathbf{1}_N^\top) \end{aligned} \quad \text{for } z \in \mathbb{R}^m.$$

Proof

$$\begin{aligned} c\mathbf{1}_N^\top + \widehat{D} &= c\mathbf{1}_N^\top + PL \\ &= c\mathbf{1}_N^\top + Pz\mathbf{1}_N^\top + PL - Pz\mathbf{1}_N^\top \\ &= \underbrace{(c + Pz)}_{c'} \mathbf{1}_N^\top + P \underbrace{(L - z\mathbf{1}_N^\top)}_{L'} = c'\mathbf{1}_N^\top + \widehat{D}'. \end{aligned}$$

Therefore, if (c, \widehat{D}) is a solution, then (c', \widehat{D}') is also a solution. From Theorem 5.8, it follows that $c = \mathbf{E}(D)$, $\widehat{D} = PL$ is a solution. \square

The same type of nonuniqueness appears in weighted and structured low rank approximation problems with centering. This can cause problems in the optimization algorithms and implies that solutions produced by different methods cannot be compared directly.

Weighted Low Rank Approximation with Centering

Consider the weighted low rank approximation problem with centering:

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{D} \text{ and } c && \|D - \widehat{D} - c\mathbf{1}^\top\|_W \\ & \text{subject to} && \text{rank}(\widehat{D}) \leq m, \end{aligned} \tag{WLRA}_c$$

where W is a symmetric positive definite matrix and $\|\cdot\|_W$ is the weighted norm, defined in $(\|\cdot\|_W)$. The two-stage procedure of computing the mean in a preprocessing step and then the weighted low rank approximation of the centered data matrix, in general, yields a suboptimal solution to (WLRA_c) . We present two algorithms for finding a locally optimal solution to (WLRA_c) . The first one is an alternating projections type method and the second one is a variable projections type method. First, however, we present a special case of (WLRA_c) with analytic solution that is more general than the case $W = \alpha I$, with $\alpha \neq 0$.

Two-Sided Weighted Low Rank Approximation

Theorem 5.10 (Reduction to an unweighted problem) *A solution to (WLRA_c) , in the case*

$$W = W_r \otimes W_1, \quad \text{where } W_1 \in \mathbb{R}^{q \times q} \text{ and } W_r \in \mathbb{R}^{N \times N} \tag{W_r \otimes W_1}$$

with $W_r \mathbf{1}_N = \lambda \mathbf{1}_N$, for some λ , is

$$c^* = \sqrt{W_1^{-1}} c_m^* / \sqrt{\lambda}, \quad \widehat{D}^* = \sqrt{W_1^{-1}} \widehat{D}_m^* \sqrt{W_r^{-1}},$$

where (c_m^*, \widehat{D}_m^*) is a solution to the unweighted low rank approximation problem with centering

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{D}_m \text{ and } c_m && \|D_m - c_m \mathbf{1}_N^\top - \widehat{D}_m\|_F \\ & \text{subject to} && \text{rank}(\widehat{D}_m) \leq m. \end{aligned}$$

for the modified data matrix $D_m := \sqrt{W_1} D \sqrt{W_r}$.

Proof Using the property $W_r \mathbf{1}_N = \lambda \mathbf{1}_N$ of W_r , we have

$$\begin{aligned} \|D - \widehat{D} - c\mathbf{1}^\top\|_W &= \|\sqrt{W_1}(D - \widehat{D} - c\mathbf{1}^\top)\sqrt{W_r}\|_F \\ &= \|D_m - \widehat{D}_m - c_m \mathbf{1}^\top\|_F \end{aligned}$$

where

$$D_m = \sqrt{W_1} D \sqrt{W_r}, \quad \widehat{D}_m = \sqrt{W_1} \widehat{D} \sqrt{W_r}, \quad \text{and} \quad c_m = \sqrt{W_1} c \sqrt{\lambda}.$$

Therefore, the considered problem is equivalent to the low rank approximation problem (LRA_c) for the modified data matrix D_m . \square

Alternating Projections Algorithm

Using the image representation of the rank constraint

$$\text{rank}(\widehat{D}) \leq m \iff \widehat{D} = PL, \quad \text{where } P \in \mathbb{R}^{q \times m} \text{ and } L \in \mathbb{R}^{m \times N},$$

we obtain the following problem equivalent to (WLRA_c)

$$\text{minimize over } P \in \mathbb{R}^{q \times m}, L \in \mathbb{R}^{m \times N}, \text{ and } c \in \mathbb{R}^q \quad \|D - PL - c\mathbf{1}_N^\top\|_W. \quad (\text{WLRA}_{c,P})$$

The method is motivated by the fact that (WLRA_{c,P}) is linear in c and P as well as in c and L . Indeed,

$$\begin{aligned} \|D - c\mathbf{1}_N^\top - PL\|_W &= \left\| \text{vec}(D) - [I_N \otimes P \mathbf{1}_N \otimes I_q] \begin{bmatrix} \text{vec}(L) \\ c \end{bmatrix} \right\|_W \\ &= \left\| \text{vec}(D) - [L^\top \otimes I_q \mathbf{1}_N \otimes I_q] \begin{bmatrix} \text{vec}(P) \\ c \end{bmatrix} \right\|_W. \end{aligned}$$

This suggests an iterative algorithm alternating between minimization over c and P with a fixed L and over c and L with a fixed P , see Algorithm 4. Each iteration step is a weighted least squares problem, which can be solved globally and efficiently. The algorithm starts from an initial approximation $c^{(0)}$, $P^{(0)}$, $L^{(0)}$ and on each iteration step updates the parameters with the newly computed values from the last least squares problem. Since on each iteration the cost function value is guaranteed

Algorithm 4 Alternating projections algorithm for weighted low rank approximation with centering

Input: data matrix $D \in \mathbb{R}^{q \times N}$, rank constraint m , positive definite weight matrix $W \in \mathbb{R}^{Nq \times Nq}$, and relative convergence tolerance ε .

1: Initial approximation: compute the mean $c^{(0)} := \mathbf{E}(D)$ and the rank- m approximation $\widehat{D}^{(0)}$ of the centered matrix $D - c^{(0)}\mathbf{1}_N^\top$. Let $P^{(0)} \in \mathbb{R}^{q \times m}$ and $L^{(0)} \in \mathbb{R}^{m \times N}$ are full rank matrices, such that $\widehat{D}^{(0)} = P^{(0)}L^{(0)}$.

2: $k := 0$.

3: **repeat**

4: Let $\mathbf{P} := [I_N \otimes P^{(k)} \mathbf{1}_N \otimes I_q]$ and

$$\begin{bmatrix} \text{vec}(L^{(k+1)}) \\ \widehat{c}^{(k+1)} \end{bmatrix} := (\mathbf{P}^\top W \mathbf{P})^{-1} \mathbf{P}^\top W \text{vec}(D).$$

5: Let $\mathbf{L} := [L^{(k+1)\top} \otimes I_q \mathbf{1}_N \otimes I_q]$ and

$$\begin{bmatrix} \text{vec}(P^{(k+1)}) \\ \widehat{c}^{(k+1)} \end{bmatrix} := (\mathbf{L}^\top W \mathbf{L})^{-1} \mathbf{L}^\top W \text{vec}(D).$$

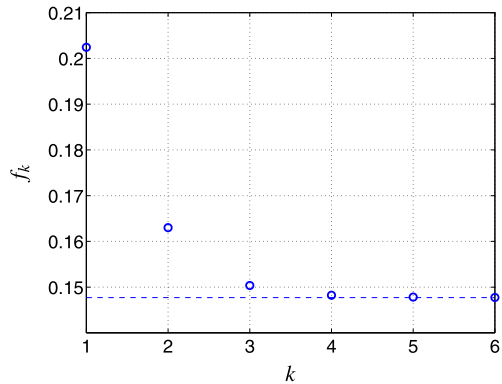
6: Let $\widehat{D}^{(k+1)} := P^{(k+1)}L^{(k+1)}$.

7: $k = k + 1$.

8: **until** $\|\widehat{D}^{(k)} - \widehat{D}^{(k-1)}\|_W / \|\widehat{D}^{(k)}\|_W < \varepsilon$.

Output: Locally optimal solution $\widehat{c} := \widehat{c}^{(k)}$ and $\widehat{D}^* = \widehat{D}^{(k)}$ of (WLRA_{c,P}).

Fig. 5.1 Sequence of cost function values, produced by Algorithm 4



to be non-increasing and the cost function is bounded from below, the sequence of cost function values, generated by the algorithm converges. Moreover, it can be shown that the sequence of parameter approximations $c^{(k)}$, $P^{(k)}$, $L^{(k)}$ converges to a locally optimal solution of (WLRA $_{c,P}$).

Example 5.11 Implementation of the methods for weighted and structured low rank approximation with centering, presented in this section, are available from the book's web page. Figure 5.1 shows the sequence of the cost function values for a randomly generated weighted rank-1 approximation problem with $q = 3$ variables and $N = 6$ data points. The mean of the data matrix and the approximation of the mean, produced by the Algorithm 4 are, respectively

$$c^{(0)} = \begin{bmatrix} 0.5017 \\ 0.7068 \\ 0.3659 \end{bmatrix} \quad \text{and} \quad \hat{c} = \begin{bmatrix} 0.4365 \\ 0.6738 \\ 0.2964 \end{bmatrix}.$$

The weighted rank-1 approximation of the matrix $D - c^{(0)}\mathbf{1}_N^\top$ has approximation error 0.1484, while the weighted rank-1 approximation of the matrix $D - \hat{c}\mathbf{1}_N^\top$ has approximation error 0.1477. This proves the suboptimality of the two-stage procedure—data centering, followed by weighted low rank approximation.

Variable Projections Algorithm

The variable projections approach is based on the observation that (WLRA $_{c,P}$) is a double minimization problem

$$\text{minimize over } P \in \mathbb{R}^{q \times m} \quad f(P)$$

where the inner minimization is a weighted least squares problem

$$f(P) := \min_{L \in \mathbb{R}^{m \times N}, c \in \mathbb{R}^q} \|D - PL - c\mathbf{1}_N^\top\|_W$$

and therefore can be solved analytically. This reduces the original problem to a nonlinear least squares problem over P only. We have

$$f(P) = \sqrt{\text{vec}^\top(D) W \mathbf{P} (\mathbf{P}^\top W \mathbf{P})^{-1} \mathbf{P}^\top W \text{vec}(D)},$$

where

$$\mathbf{P} := [I_N \otimes P \quad \mathbf{1}_N \otimes I_q].$$

For the outer minimization any standard unconstrained nonlinear (least squares) algorithm is used.

Example 5.12 For the same data, initial approximation, and convergence tolerance as in Example 5.11, the variable projections algorithm, using numerical approximation of the derivatives in combination with quasi-Newton method converges to a locally optimal solution with approximation error 0.1477—the same as the one found by the alternating projections algorithm. The optimal parameters found by the two algorithms are equivalent up to the nonuniqueness of a solution (Theorem 5.9).

Hankel Low Rank Approximation with Centering

In the case of data centering we consider the following modified Hankel low rank approximation problem:

$$\begin{aligned} & \text{minimize} \quad \text{over } \widehat{w} \text{ and } c \quad \|w - c - \widehat{w}\|_2 \\ & \text{subject to} \quad \text{rank}(\mathcal{H}_{n+1}(\widehat{w})) \leq r. \end{aligned} \tag{HLRA}_c$$

Algorithm

Consider the kernel representation of the rank constraint

$$\begin{aligned} \text{rank}(\mathcal{H}_{n+1}(\widehat{w})) \leq r \quad & \iff \quad \text{there is full rank matrix } R \in \mathbb{R}^{p \times (n+1)q} \\ & \text{such that } R \mathcal{H}_{n+1}(w_d) = 0. \end{aligned}$$

We have

$$R \mathcal{H}_{n+1}(\widehat{w}) = 0 \quad \iff \quad \mathcal{T}(R) \widehat{w} = 0,$$

where

$$\mathcal{T}(R) = \begin{bmatrix} R_0 & R_1 & \cdots & R_n & & & \\ & R_0 & R_1 & \cdots & R_n & & \\ & & \ddots & \ddots & & \ddots & \\ & & & R_0 & R_1 & \cdots & R_n \end{bmatrix}$$

(all missing elements are zeros). Let P be a full rank matrix, such that

$$(P) = \ker(\mathcal{T}(R)).$$

Then the constraint of (HLRA_c) can be replaced by

$$\text{there is } \ell, \text{ such that } \widehat{w} = P\ell,$$

which leads to the following problem equivalent to (LRA_c)

$$\text{minimize over } R \quad f(R),$$

where

$$f(R) := \min_{c, \ell} \left\| w - [\mathbf{1}_N \otimes I_{\mathcal{Q}} P] \begin{bmatrix} c \\ \ell \end{bmatrix} \right\|_2.$$

The latter is a standard least squares problem, so that the evaluation of f for a given R can be done efficiently. Moreover, one can exploit the Toeplitz structure of the matrix \mathcal{T} in the computation of P and in the solution of the least squares problem.

Example 5.13 The data sequence is

$$w(t) = 0.9^t + 1, \quad t = 1, \dots, 10.$$

The sequence $(0.9^1, \dots, 0.9^{10})$ satisfies a difference equation

$$\sigma w = aw$$

(a first order autonomous linear time-invariant model), however, a shifted sequence $w(t) = 0.9^t + c$, with $c \neq 0$, does not satisfy such an equation. The mean of the data is $\mathbf{E}(w) = 1.5862$, so that the centered data $w(t) - \mathbf{E}(D)$ are not a trajectory of a first order autonomous linear time-invariant model. Solving the Hankel structured low rank approximation problem with centering (HLRA_c) , however, yields the exact solution $\widehat{c} = 1$.

Preprocessing by centering the data is common in system identification. Example 5.13 shows that preprocessing can lead to suboptimal results. Therefore, there is need for methods that combine data preprocessing with the existing identification methods. The algorithm derived in this section is such a method for identification in the errors-in-variables setting. It can be modified for output error identification, i.e., assuming that the input of the system is known exactly.

5.3 Complex Least Squares Problem with Constrained Phase

Problem Formulation

The problem considered in this section is defined as follows.

Problem 5.14 Given a complex valued $m \times n$ matrix A and an $m \times 1$ vector b , find a real valued $n \times 1$ vector x and a number ϕ , such that the equation's error or residual of the overdetermined system of linear equations

$$Ax e^{i\phi} \approx b \quad (\mathbf{i} \text{ is the imaginary unit}),$$

is minimized in the least squares sense, i.e.,

$$\text{minimize over } x \in \mathbb{R}^n \text{ and } \phi \in (-\pi, \pi] \quad \|Ax e^{i\phi} - b\|_2. \quad (\text{CLS})$$

Problem (CLS) is a complex linear least squares problem with constraint that all elements of the solution have the same phase.

As formulated, (CLS) is a nonlinear optimization problem. General purpose local optimization methods can be used for solving it, however, this approach has the usual disadvantages of local optimization methods: need of initial approximation, no guarantee of global optimality, convergence issues, and no insight in the geometry of the solutions set. In Bydler (2010) the following closed-form solution of (CLS) is derived:

$$\hat{x} = (\Re(A^H A))^+ \Re(A^H b e^{-i\phi}), \quad (\text{SOL1 } \hat{x})$$

$$\hat{\phi} = \frac{1}{2} \angle ((A^H b)^\top \Re(A^H A)^+ (A^H b)), \quad (\text{SOL1 } \hat{\phi})$$

where $\Re(A)/\Im(A)$ is the real/imaginary part, $\angle(A)$ is the angle, A^H is the complex conjugate transpose, and A^+ is the pseudoinverse of A . Moreover, in the case when a solution of (CLS) is not unique, (SOL1 \hat{x} , SOL1 $\hat{\phi}$) is a least norm element of the solution set, i.e., a solution (x, ϕ) , such that $\|x\|_2$ is minimized. Expression (SOL1 \hat{x}) is the result of minimizing the cost function $\|Ax e^{i\phi} - b\|_2$ with respect to x , for a fixed ϕ . This is a linear least squares problems (with complex valued data and real valued solution). Then minimization of the cost function with respect to ϕ , for x fixed to its optimal value (SOL1 \hat{x}), leads through a nontrivial chain of steps to (SOL1 $\hat{\phi}$).

Solution

Problem (CLS) is equivalent¹ to the problem

$$\text{minimize over } x \in \mathbb{R}^n \text{ and } \phi' \in (-\pi, \pi] \quad \|Ax - b e^{i\phi'}\|_2, \quad (\text{CLS}')$$

¹Two optimization problems are equivalent if the solution of the first can be obtained from the solution of the second by a one-to-one transformation. Of practical interest are equivalent problems for which the transformation is "simple".

where $\phi' = -\phi$. With

$$y_1 := \Re(e^{i\phi'}) = \cos(\phi') = \cos(\phi) \quad \text{and} \quad y_2 := \Im(e^{i\phi'}) = \sin(\phi') = -\sin(\phi),$$

we have

$$\begin{bmatrix} \Re(be^{i\phi'}) \\ \Im(be^{i\phi'}) \end{bmatrix} = \begin{bmatrix} \Re(b) & -\Im(b) \\ \Im(b) & \Re(b) \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}.$$

Then, (CLS') is furthermore equivalent to the problem

$$\begin{aligned} &\text{minimize} \quad \text{over } x \in \mathbb{R}^n \text{ and } y \in \mathbb{R}^2 \quad \left\| \begin{bmatrix} \Re(A) \\ \Im(A) \end{bmatrix} x - \begin{bmatrix} \Re(b) & -\Im(b) \\ \Im(b) & \Re(b) \end{bmatrix} y \right\| \\ &\text{subject to} \quad \|y\|_2 = 1, \end{aligned}$$

or

$$\text{minimize} \quad \text{over } z \in \mathbb{R}^{n+2} \quad z^\top C^\top C z \quad \text{subject to} \quad z^\top D^\top D z = 1, \quad (\text{CLS''})$$

with

$$C := \begin{bmatrix} \Re(A) & \Re(b) & -\Im(b) \\ \Im(A) & \Im(b) & \Re(b) \end{bmatrix} \in \mathbb{R}^{2m \times (n+2)} \quad \text{and} \quad D := \begin{bmatrix} 0 & 0 \\ 0 & I_2 \end{bmatrix} \in \mathbb{R}^{(n+2) \times (n+2)}. \quad (\text{C}, D)$$

It is well known that a solution of problem (CLS'') can be obtained from the generalized eigenvalue decomposition of the pair of matrices $(C^\top C, D)$. More specifically, the smallest generalized eigenvalue λ_{\min} of $(C^\top C, D)$ is equal to the minimum value of (CLS''), i.e.,

$$\lambda_{\min} = \|A\hat{x}e^{i\hat{\phi}} - b\|_2^2.$$

If λ_{\min} is simple, a corresponding generalized eigenvector z_{\min} is of the form

$$z_{\min} = \alpha \begin{bmatrix} \hat{x} \\ -\cos(\hat{\phi}) \\ \sin(\hat{\phi}) \end{bmatrix},$$

for some $\alpha \in \mathbb{R}$. We have the following result.

Theorem 5.15 *Let λ_{\min} be the smallest generalized eigenvalue of the pair of matrices $(C^\top C, D)$, defined in (C, D), and let z_{\min} be a corresponding generalized eigenvector. Assuming that λ_{\min} is a simple eigenvalue, problem (CLS) has unique solution, given by*

$$\hat{x} = \frac{1}{\|z_2\|_2} z_1, \quad \hat{\phi} = \angle(-z_{2,1} + \mathbf{i}z_{2,2}), \quad \text{where } z_{\min} =: \left\{ \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \right\}_2. \quad (\text{SOL2})$$

Remarks

1. *Generalized eigenvalue decomposition vs. generalized singular value decomposition* Since the original data are the matrix A and the vector b , the generalized singular value decomposition of the pair (C, D) can be used instead of the generalized eigenvalue decomposition of the pair $(C^\top C, D)$. This avoids “squaring” the data and is recommended from a numerical point of view.
2. *Link to low rank approximation and total least squares* Problem (CLS) is equivalent to the generalized low rank approximation problem

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{C} \in \mathbb{R}^{2m \times (n+2)} && \|(C - \widehat{C})D\|_F \\ & \text{subject to} && \text{rank}(\widehat{C}) \leq n + 1 && \text{and } \widehat{C}D^\perp = CD^\perp, \end{aligned} \quad (\text{GLRA})$$

where

$$D^\perp = \begin{bmatrix} I_n & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{(n+2) \times (n+2)}$$

and $\|\cdot\|_F$ is the Frobenius norm. Indeed, the constraints of (GLRA) imply that

$$\|(C - \widehat{C})D\|_F = \|b - \widehat{b}\|_2, \quad \text{where } \widehat{b} = Ax e^{i\phi}.$$

The normalization (SOL2) is reminiscent to the generic solution of the total least squares problems. The solution of total least squares problems, however, involves a normalization by scaling with the last element of a vector z_{\min} in the approximate kernel of the data matrix C , while the solution of (CLS) involves normalization by scaling with the norm of the last two elements of the vector z_{\min} .

3. *Uniqueness of the solution and minimum norm solutions* A solution x of (CLS) is nonunique when A has nontrivial null space. This source of nonuniqueness is fixed in Bydler (2010) by choosing from the solutions set a least norm solution. A least norm solution of (CLS), however, may also be nonunique due to possible nonuniqueness of ϕ . Consider the following example,

$$A = \begin{bmatrix} 1 & \mathbf{i} \\ -\mathbf{i} & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ -\mathbf{i} \end{bmatrix},$$

which has two least norm solutions

$$\widehat{x} e^{i\phi_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad \widehat{x}' e^{i\phi_2} = \begin{bmatrix} 0 \\ -\mathbf{i} \end{bmatrix}.$$

Moreover, there is a trivial source of nonuniqueness in x and ϕ due to

$$x e^{i\phi} = -x e^{i(\phi \pm \pi)}$$

with both ϕ and one of the angles $\phi \pm \pi$ in the interval $(-\pi, \pi]$.

Computational Algorithms

Solution (SOL1 \hat{x} , SOL1 $\hat{\phi}$) gives a straightforward procedure for computing a least norm solution of problem (CLS).

160a *(Complex least squares, solution by (SOL1 \hat{x} , SOL1 $\hat{\phi}$) 160a)≡*

```
function cx = cls1(A, b)
    invM = pinv(real(A' * A)); Atb = A' * b;
    phi = 1 / 2 * angle((Atb).' * invM * Atb);
    x = invM * real(Atb * exp(-i * phi));
    cx = x * exp(i * phi);
```

Defines:

cls1, used in chunk 163b.

The corresponding computational cost is

$$\text{cls1} — O(n^2m + n^3).$$

Theorem 5.15 gives two alternative procedures—one based on the generalized eigenvalue decomposition:

160b *(Complex least squares, solution by generalized eigenvalue decomposition 160b)≡*

```
function cx = cls2(A, b)
    (define C, D, and n 160c)
    [v, l] = eig(C' * C, D); l = diag(l);
    l(find(l < 0)) = inf; % ignore nevatve values
    [ml, mi] = min(l); z = v(:, mi);
    phi = angle(-z(end - 1) + i * z(end));
    x = z(1:(end - 2)) / norm(z((end - 1):end));
    cx = x * exp(i * phi);
```

Defines:

cls2, used in chunk 163b.

160c *(define C, D, and n 160c)≡* (160 161)

```
C = [real(A) real(b) -imag(b);
      imag(A) imag(b) real(b)];
n = size(A, 2); D = diag([zeros(1, n), 1, 1]);
```

and the other one based on the generalized singular value decomposition:

160d *(Complex least squares, solution by generalized singular value decomposition 160d)≡*

```
function cx = cls3(A, b)
    (define C, D, and n 160c)
    [u, v] = gsvd(C, D); z = v(:, 1);
    phi = angle(-z(end - 1) + i * z(end));
    x = pinv(C(:, 1:n)) * [real(b * exp(- i * phi));
                          imag(b * exp(- i * phi))];
    cx = x * exp(i * phi);
```

Defines:

cls3, used in chunk 163b.

The computational costs are

$$\text{cls2} \text{ --- } O((n+2)^2m + (n+2)^3)$$

and

$$\text{cls3} \text{ --- } O(m^3 + (n+2)^2m^2 + (n+2)^2m + (n+2)^3).$$

Note, however, that `cls2` and `cls3` compute the full generalized eigenvalue decomposition and generalized singular value decomposition, respectively, while only the smallest generalized eigenvalue/eigenvector or singular value/singular vector pair is needed for solving (CLS). This suggests a way of reducing the computational complexity by a factor of magnitude.

The equivalence between problem (CLS) and the generalized low rank approximation problem (GLRA), noted in remark 2 above, allows us to use the algorithm from Golub et al. (1987) for solving problem (CLS). The resulting Algorithm 5 is implemented in the function `cls4`.

```
161 (Complex least squares, solution by Algorithm 5 161)≡
function cx = cls4(A, b)
    <define C, D, and n 160c>
    R = triu(qr(C, 0));
    [u, s, v] = svd(R((n+1):(n+2), (n+1):end));
    phi = angle(v(1, 2) - i * v(2, 2));
    x = R(1:n, 1:n)
        \ (R(1:n, (n+1):end) * [v(1, 2); v(2, 2)]);
    cx = x * exp(i * phi);
```

Defines:

`cls4`, used in chunk 163b.

Its computational cost is

$$\text{cls4} \text{ --- } O((n+2)^2m).$$

A summary of the computational costs for the methods, implemented in the functions `cls1-4`, is given in Table 5.5.

Algorithm 5 Solution of problem (CLS) using generalized low rank approximation

Input: $A \in \mathbb{C}^{m \times n}$, $b \in \mathbb{C}^{m \times 1}$

1: QR factorization of C , $QR = C$.

2: Define $R =: \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}^n$, where $R_{11} \in \mathbb{R}^{n \times n}$.

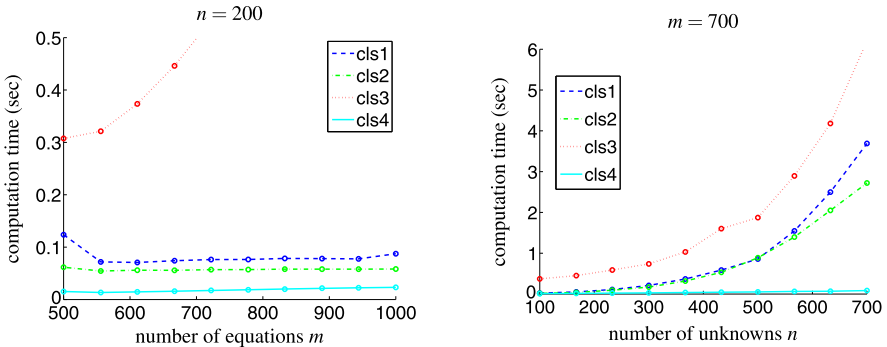
3: Singular value decomposition of R_{22} , $U\Sigma V^T = R_{22}$.

4: Let $\hat{\phi} := \angle(v_{12} - iv_{22})$ and $\hat{x} := R_{11}^{-1}R_{12} \begin{bmatrix} v_{12} \\ v_{22} \end{bmatrix}$.

Output: $\hat{x}e^{i\hat{\phi}}$

Table 5.5 Summary of methods for solving the complex least squares problem (CLS)

Function	Method	Computational cost
cls1	(SOL1 \hat{x} , SOL1 $\hat{\phi}$)	$O(n^2m + n^3)$
cls2	full generalized eigenvalue decomp.	$O((n+2)^2m + (n+2)^3)$
cls3	full generalized singular value decomp.	$O(m^3 + (n+2)^2m^2 + (n+2)^2m + (n+2)^3)$
cls4	Algorithm 5	$O((n+2)^2m)$

**Fig. 5.2** Computation time for the four methods, implemented in the functions `cls1`, ..., `cls4`

Numerical Examples

Generically, the four solution methods implemented in the functions `cls1`, ..., `cls4` compute the same result, which is equal to the unique solution of problem (CLS). As predicted by the theoretical computation costs, the method based on Algorithm 5 is the fastest of the four methods when both the number of equations and the number of unknowns is growing, see Fig. 5.2.

The figures are generated by using random test data

162a `(Computation time for cls1-4 162a)≡` 162b>

```

(initialize the random number generator 89e)
mm = 1000; nm = 1000; s = {'b-' 'g-' 'r:' 'c-'};
Am = rand(mm, nm) + i * rand(mm, nm);
bm = rand(mm, 1) + i * rand(mm, 1);

```

Defines:

```
test_cls, never used.
```

and solving problems with increasing number of equations m

162b `(Computation time for cls1-4 162a)+≡` <162a 163a>

```

Nm = 10; M = round(linspace(500, 1000, Nm)); n = 200;
for j = 1:Nm, m = M(j); <call cls1-4 163b> end
k = 1; x = M; ax = [500 1000 0 0.5]; name = 'cls-f1';
(plot cls results 163c)

```

as well as increasing number of unknowns n

163a `(Computation time for cls1-4 162a)+≡` ◁162b

```
Nn = 10; N = round(linspace(100, 700, Nn)); m = 700;
for j = 1:Nn, n = N(j); (call cls1-4 163b) end
k = 2; x = N; ax = [100 700 0 6]; name = 'cls-f2';
(plot cls results 163c)
```

163b `(call cls1-4 163b)≡` (162b 163a)

```
A = Am(1:m, 1:n); b = bm(1:m);
for i = 1:4 % cls1, cls2, cls3, cls4
    eval(sprintf('tic, x = cls%d(A, b); t(%d) = toc;', i, i))
end
T(:, j) = t';
```

Uses cls1 160a, cls2 160b, cls3 160d, and cls4 161.

163c `(plot cls results 163c)≡` (162b 163a)

```
figure(k), hold on
for i = 1:4 plot(x, T(i, :), s{i}, 'linewidth', 2), end
legend('cls1', 'cls2', 'cls3', 'cls4')
for i = 1:4, plot(x, T(i, :), [s{i}(1) 'o']), end
axis(ax), print_fig(name)
```

Uses print_fig 25a.

5.4 Approximate Low Rank Factorization with Structured Factors

Problem Formulation

Rank Estimation

Consider an $q \times N$ real matrix D_0 with rank $m_0 < q$ and let

$$D_0 = P_0 L_0, \quad \text{where } P_0 \in \mathbb{R}^{q \times m_0} \text{ and } L_0 \in \mathbb{R}^{m_0 \times N}$$

be a *rank revealing* factorization of D_0 . Suppose that instead of D_0 a matrix

$$D := D_0 + \tilde{D}$$

is observed, where \tilde{D} is a perturbation, e.g., \tilde{D} can represent rounding errors in a finite precision arithmetic or measurement errors in data acquisition. The rank of the perturbed matrix D may not be equal to m_0 . If \tilde{D} is random, generically, D is full rank, so that from a practical point of view, a nonzero perturbation \tilde{D} makes the matrix D full rank. If, however, \tilde{D} is “small”, in the sense that its Frobenius norm $\|\tilde{D}\|_F$ is less than a constant ε (defining the perturbation size), then D will be

“close” to a rank- m_0 matrix in the sense that the distance of D to the manifold of rank- m_0 matrices

$$\text{dist}(D, m_0) := \min_{\hat{D}} \|D - \hat{D}\|_F \quad \text{subject to} \quad \text{rank}(\hat{D}) = m_0 \quad (5.12)$$

is less than the perturbation size ε . Therefore, provided that the size ε of the perturbation \tilde{D} is known, the distance measure $\text{dist}(D, m)$, for $m = 1, 2, \dots$, can be used to estimate the rank of the unperturbed matrix as follows

$$\hat{m} = \arg \min \{m \mid \text{dist}(D, m) < \varepsilon\}.$$

Problem (5.12) has analytic solution in terms of the singular values $\sigma_1, \dots, \sigma_q$ of D

$$\text{dist}(D, m_0) := \sqrt{\sigma_{m_0+1}^2 + \dots + \sigma_q^2},$$

and therefore the rank of D_0 can be estimated from the decay of the singular values of D (find the largest singular value that is sufficiently small compared to the perturbation size ε). This is the standard way for rank estimation in numerical linear algebra, where the estimate \hat{m} is called *numerical rank* of D , cf., p. 38. The question occurs:

Given a perturbed matrix $D := D_0 + \tilde{D}$, is the numerical rank of D the “best” estimate for the rank of D_0 , and if so, in what sense?

The answer to the above question depends on the type of the perturbation \tilde{D} . If \tilde{D} is a random matrix with zero mean elements that are normally distributed, independent, and with equal variances, then the estimate \hat{D} , defined by (5.12) is a maximum likelihood estimator of D_0 , i.e., it is statistically optimal. If, however, one or more of the above assumptions are not satisfied, \hat{D} is not optimal and can be improved by modifying problem (5.12). Our objective is to justify this statement in a particular case when there is prior information about the true matrix D_0 in the form of structure in a normalized rank-revealing factorization and the elements of the perturbation \tilde{D} are independent but possibly with different variances.

Prior Knowledge in the Form of Structure

In applications often there is prior knowledge about the unperturbed matrix D_0 , apart from the basic one that D_0 is rank deficient. Whenever available, such prior knowledge is beneficial to use in the computation of the distance measure $\text{dist}(D, m)$. Using the prior knowledge amounts to modification of problem (5.12). For example, common prior information in image and text classification is nonnegativity of the elements of D_0 . In this case, we require the approximation \hat{D} to be

nonnegative and in order to achieve this, we impose nonnegativity of the estimate \widehat{D} as an extra constraint in (5.12). Similarly, in signal processing and system theory the matrix D_0 is Hankel or Toeplitz structured and the relevant modification of (5.12) is to constrain \widehat{D} to have the same structure. In chemometrics, the measurement errors \tilde{d}_{ij} may have different variances $\sigma^2 v_{ij}$, which are known (up to a scaling factor) from the measurement setup or from repeated experiments. Such prior information amounts to changing the cost function $\|D - \widehat{D}\|_F$ to the weighted norm $\|D - \widehat{D}\|_\Sigma$ of the error matrix $D - \widehat{D}$, where the elements of the weight matrix Σ are up to a scaling factor equal to the inverse square root of the error variance $\sigma^2 V$. In general, either the addition of constraints on \widehat{D} or the replacement of the Frobenius norm with a weighted norm, renders the modified distance problem (5.12) difficult to solve. A globally optimal solution can no longer be given in terms of the singular values of D and the resulting optimization problem is nonconvex.

A factorization $D = PL$ is nonunique; for any $r \times r$ nonsingular matrix T , we obtain a new factorization $D = P'L'$, where $P' := PT^{-1}$ and $L' = TL$. Obviously, this imposes a problem in estimating the factors P and L from the data D . In order to resolve the nonuniqueness problem, we assume that

$$P = \begin{bmatrix} I_m \\ P' \end{bmatrix}.$$

Next, we present an algorithm for approximate low rank factorization with structured factors and test its performance on synthetic data. We use the alternating projections approach, because it is easier to modify for constrained optimization problems. Certain constrained problems can be treated also using a modification of the variable projections.

Statistical Model and Maximum Likelihood Estimation Problem

Consider the errors-in-variables model

$$\begin{aligned} D &= D_0 + \tilde{D}, \quad \text{where } D_0 = P_0 L_0, \quad \text{with} \\ P_0 &\in \mathbb{R}^{\mathfrak{q} \times m}, \quad L_0 \in \mathbb{R}^{m \times N}, \quad m < \mathfrak{q} & \text{(EIV}_0\text{)} \\ \text{and } \text{vec}(\tilde{D}) &\sim \text{N}(0, \sigma^2 \text{diag}(v)). \end{aligned}$$

The *true data matrix* D_0 has rank equal to m and the measurement errors \tilde{d}_{ij} are zero mean, normal, and uncorrelated, with covariance $\sigma^2 v_{i+\mathfrak{q}(j-1)}$. The vector $v \in \mathbb{R}^{\mathfrak{q}N}$ specifies the element-wise variances of the measurement error matrix \tilde{D} up to an unknown factor σ^2 .

In order to make the parameters P_0 and L_0 unique, we impose the normalization constraint (or assumption on the “true” parameter values)

$$P_0 = \begin{bmatrix} I_m \\ P'_0 \end{bmatrix}. \quad \text{(A}_1\text{)}$$

In addition, the block P'_0 of P_0 has elements (specified by a selector matrix S) equal to zero

$$S \text{vec}(P'_0) = 0. \quad (\text{A}_2)$$

The parameter L_0 is periodic with a period $l \in \mathbb{Z}_+$

$$L_0 = \mathbf{1}_l^\top \otimes L'_0, \quad (\text{A}_3)$$

nonnegative

$$L'_0 \geq 0, \quad (\text{A}_4)$$

and with smooth rows in the sense that

$$\|L'_0 \mathbf{D}\|_{\text{F}}^2 \leq \delta, \quad (\text{A}_5)$$

where $\delta > 0$ is a smoothness parameter and \mathbf{D} is a finite difference matrix

$$\mathbf{D} := \begin{bmatrix} 1 & & & & -1 \\ -1 & 1 & & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix}.$$

Define the $q \times N$ matrix

$$\Sigma = \text{vec}^{-1} \left(\left[v_1^{-1/2} \dots v_{qN}^{-1/2} \right] \right) := \begin{bmatrix} v_1^{-1/2} & v_{q+1}^{-1/2} & \dots & v_{q(N-1)+1}^{-1/2} \\ v_2^{-1/2} & v_{q+2}^{-1/2} & \dots & v_{q(N-1)+2}^{-1/2} \\ \vdots & \vdots & \ddots & \vdots \\ v_q^{-1/2} & v_{2q}^{-1/2} & \dots & v_{qN}^{-1/2} \end{bmatrix}. \quad (\Sigma)$$

The maximum likelihood estimator for the parameters P_0 and L_0 in (EIV₀) under assumptions (A₁–A₅), with known parameters m , v , S , and δ , is given by the following optimization problem:

$$\text{minimize over } P', L', \text{ and } \widehat{D} \quad \|D - \widehat{D}\|_{\Sigma}^2 \quad (\text{cost function}) \quad (\text{C}_0)$$

$$\text{subject to} \quad \widehat{D} = CP \quad (\text{rank constraint})$$

$$P = \begin{bmatrix} I_m \\ P' \end{bmatrix} \quad (\text{normalization of } P) \quad (\text{C}_1)$$

$$S \text{vec}(P') = 0 \quad (\text{zero elements of } P') \quad (\text{C}_2)$$

$$L = \mathbf{1}_l^\top \otimes L' \quad (\text{periodicity of } L) \quad (\text{C}_3)$$

$$L' \geq 0 \quad (\text{nonnegativity of } L) \quad (\text{C}_4)$$

$$\|L' \mathbf{D}\|_{\text{F}}^2 \leq \delta \quad (\text{smoothness of } L) \quad (\text{C}_5)$$

The rank and measurement errors assumptions in the model (EIV₀) imply the weighted low rank approximation nature of the estimation problem (C₀–C₅) with weight matrix given by (Σ). Furthermore, the assumptions (A₁–A₅) about the true data matrix D₀ correspond to the constraints (C₁–C₅) in the estimation problem.

Computational Algorithm

Algorithm 6 Alternating projections algorithm for solving problem (C₀–C₅)

- Find an initial approximation (P⁽⁰⁾, L⁽⁰⁾).
 - For k = 0, 1, ... till convergence do
 1. P^(k+1) := arg min_{P'} ||D - PL||_Σ² subject to (C₁–C₂) with L' = L^(k)
 2. L^(k+1) := arg min_{L'} ||D - PL||_Σ² subject to (C₃–C₅) with P' = P^(k+1)
-

The alternating projections algorithm, see Algorithm 6, is based on the observation that the cost function (C₀) is quadratic and the constraints (C₁–C₅) are linear in either P or L. Therefore, for a fixed value of P, (C₀–C₅) is a nonnegativity constrained least squares problem in L and vice versa, for a fixed value of L, (C₀–C₅) is a constrained least squares problem in P. These problems correspond to, respectively, steps 1 and 2 of the algorithm. Geometrically they are projections. In the unweighted (i.e., Σ = 1_q1_N^T) and unconstrained case, the problem on step 1 is the orthogonal projection

$$\widehat{D} = DL^T(LL^T)^{-1}L^T = D\Pi_L$$

of the row of D on the span of the rows of L, and problem on step 2 is the orthogonal projection

$$\widehat{D} = P(P^T P)^{-1}P^T D = \Pi_P D$$

of the columns of D on the span of the column of P. The algorithm iterates the two projections.

Note 5.16 (Rank deficient factors P and L) If the factor L is rank deficient, the indicated inverse in the computation of the projected matrix P* does not exist. (This happens when the rank of the approximation \widehat{D} is less than m.) The projection P*, however, is still well defined by the optimization problem on step 1 of the algorithm and can be computed in closed form by replacing the inverse with the pseudo inverses. The same is true when the factor L is rank deficient.

Theorem 5.17 Algorithm 6 is globally and monotonically convergent in the || · ||_Σ norm, i.e., if

$$\widehat{D}^{(k)} := P^{(k)} L^{(k)}$$

is the approximation on the k th step of the algorithm, then

$$f(k) := \|D - \widehat{D}^{(k)}\|_{\Sigma}^2 \rightarrow f^*, \quad \text{as } k \rightarrow \infty. \quad (f(k) \rightarrow f^*)$$

Assuming that there exists a solution to the problem (C₀–C₅) and any (locally optimal) solution is unique (i.e., it is a strict minimum), the sequences $\widehat{D}^{(k)}$, $P^{(k)}$, and $L^{(k)}$ converge element-wise, i.e.,

$$\widehat{D}^{(k)} \rightarrow D^*, \quad P^{(k)} \rightarrow P^*, \quad \text{and } L^{(k)} \rightarrow L^*, \quad \text{as } k \rightarrow \infty, \quad (D^{(k)} \rightarrow D^*)$$

where $\widehat{D}^* := P^*L^*$ is a (locally optimal) solution of (C₀–C₅).

The proof is given in Appendix B.

Simulation Results

In this section, we show empirically that exploiting prior knowledge ((Σ) and assumptions (A₁–A₅)) improves the performance of the estimator. The data matrix D is generated according to the errors-in-variables model (EIV₀) with parameters $N = 100$, $\mathfrak{q} = 6$, and $m = 2$. The true low rank matrix $D_0 = P_0L_0$ is random and the parameters P_0 and L_0 are normalized according to assumption (A₁) (so that they are unique). For the purpose of validating the algorithm, the element $p_{0,\mathfrak{q}N}$ is set to zero but this prior knowledge is not used in the parameter estimation.

The estimation algorithm is applied on $M = 100$ independent noise realizations of the data D . The estimated parameters on the i th repetition are denoted by $P^{(i)}$, $L^{(i)}$ and $\widehat{D}^{(i)} := P^{(i)}L^{(i)}$. The performance of the estimator is measured by the following average relative estimation errors:

$$e_D = \frac{1}{M} \sum_{i=1}^M \frac{\|D_0 - \widehat{D}^{(i)}\|_{\mathbb{F}}^2}{\|D_0\|_{\mathbb{F}}^2}, \quad e_P = \frac{1}{M} \sum_{i=1}^M \frac{\|P_0 - P^{(i)}\|_{\mathbb{F}}^2}{\|P_0\|_{\mathbb{F}}^2},$$

$$e_L = \frac{1}{M} \sum_{i=1}^M \frac{\|L_0 - L^{(i)}\|_{\mathbb{F}}^2}{\|L_0\|_{\mathbb{F}}^2}, \quad \text{and } e_z = \frac{1}{M} \sum_{i=1}^M |p_{\mathfrak{q}N}^{(i)}|.$$

For comparison the estimation errors are reported for the low rank approximation algorithm, using only the normalization constraint (A₁), as well as for the proposed algorithm, exploiting the available prior knowledge. The difference between the two estimation errors is an indication of how important is the prior knowledge in the estimation.

Lack of prior knowledge is reflected by specific choice of the simulation parameters as follows:

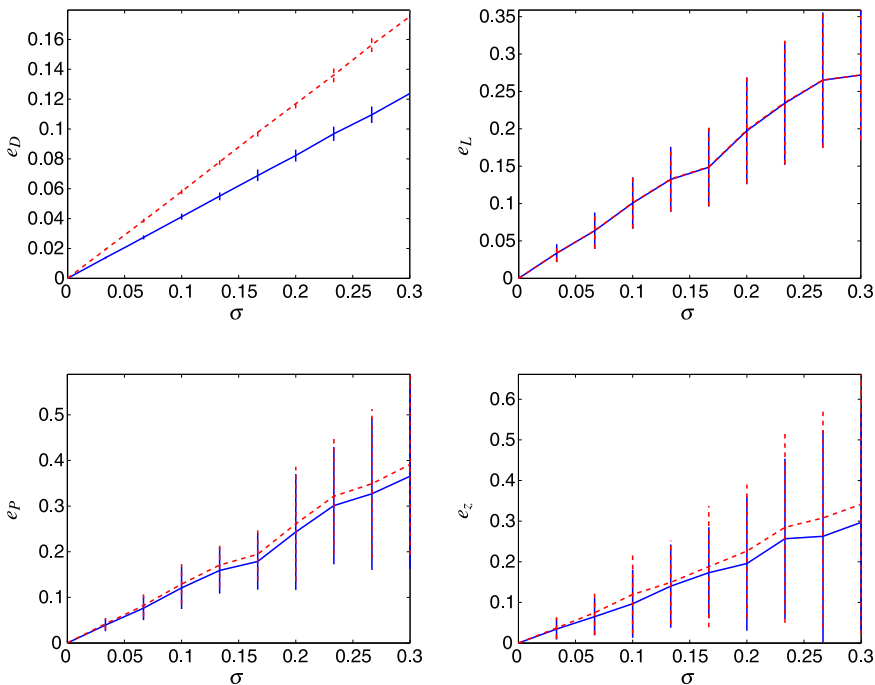


Fig. 5.3 Effect of weighting (solid line—exploiting prior knowledge, dashed line—without exploiting prior knowledge, vertical bars—standard deviations)

homogeneous errors	\leftrightarrow	$\Sigma = \text{ones}(q, N)$
no periodicity	\leftrightarrow	$l = 1$
no zeros in P'	\leftrightarrow	$S = []$
no sign constraint on L'	\leftrightarrow	nonneg = 0

We perform the following experiments:

Σ	l	$S = []$	nonneg
rand(q, N)	1	yes	0
ones(q, N)	3	yes	0
ones(q, N)	1	no	0
ones(q, N)	1	yes	1
rand(q, N)	3	no	1

which test individually the effect of (Σ), assumptions (A_2), (A_3), (A_4), and their combined effect on the estimation error. Figures 5.3–5.7 show the average relative estimation errors (solid line is the estimator that exploits prior knowledge and dashed line is the estimator that does not exploit prior knowledge) versus the measurement noise standard deviation σ , for the five experiments. The vertical bars on

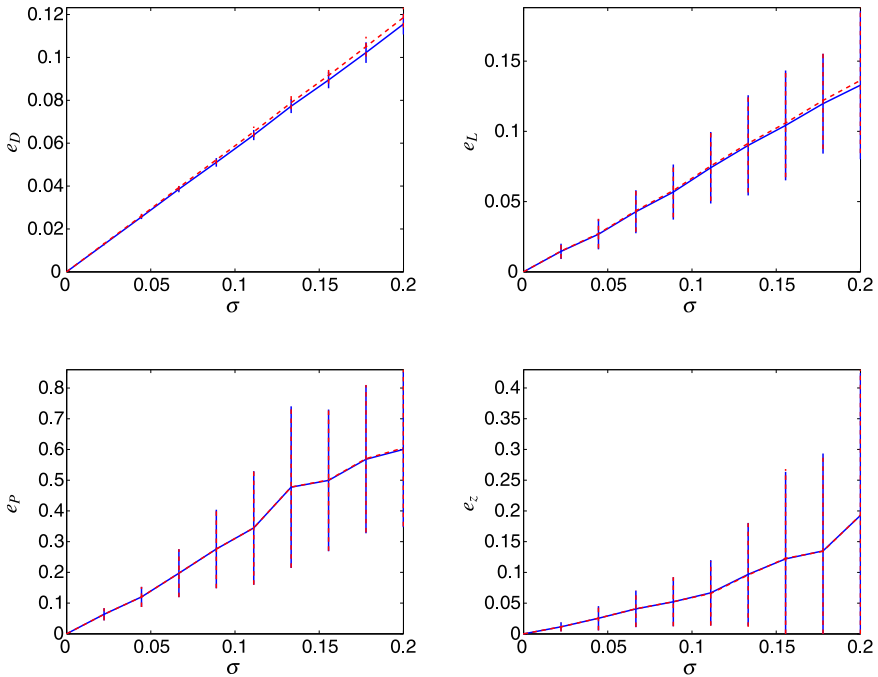


Fig. 5.4 Effect of periodicity of L (*solid line*—exploiting prior knowledge, *dashed line*—without exploiting prior knowledge, *vertical bars*—standard deviations)

the plots visualize the standard deviation of the estimates. The results indicate that main factors for the improved performance of the estimator are:

1. assumption (A_3) —known zeros in the P'_0 and
2. (Σ) —known covariance structure of the measurement noise.

Files reproducing the numerical results and figures presented are available from the book's web page.

Implementation of Algorithm 6

Initial Approximation

For initial approximation $(P^{(0)}, L^{(0)})$ we choose the normalized factors of a rank revealing factorization of the solution \widehat{D} of (5.12). Let

$$D = U \Sigma V^\top$$

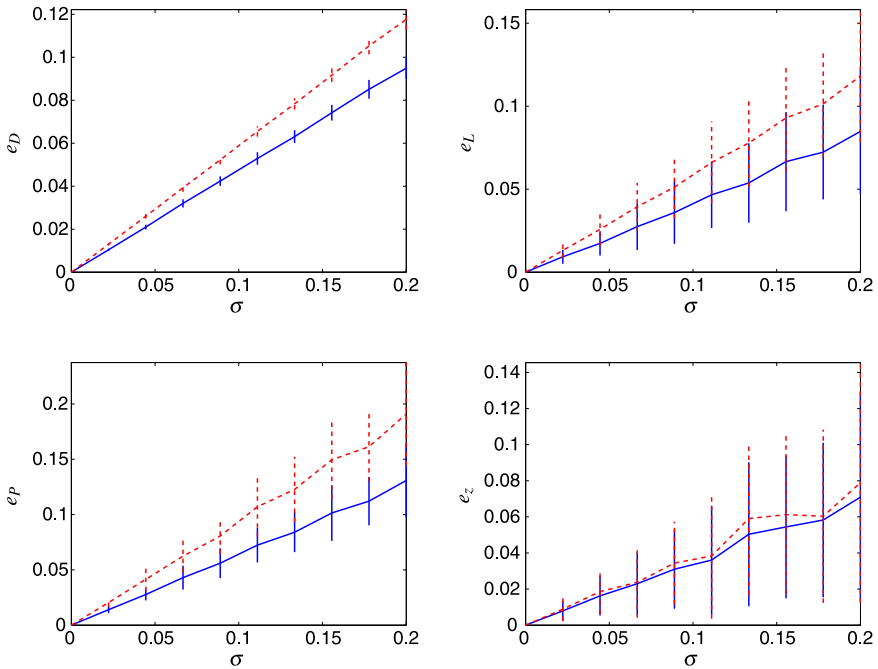


Fig. 5.5 Effect of zero elements in P (solid line—exploiting prior knowledge, dashed line—without exploiting prior knowledge, vertical bars—standard deviations)

be the singular value decomposition of D and define the partitioning

$$U =: \begin{bmatrix} U_1 & U_2 \end{bmatrix}, \quad \Sigma =: \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}_{\substack{m \times (q-m) \\ (q-m) \times (q-m)}}, \quad V =: \begin{bmatrix} V_1 & V_2 \end{bmatrix}_{\substack{m \times (N-m) \\ (N-m) \times (N-m)}}.$$

Furthermore, let

$$\begin{bmatrix} U_{11} \\ U_{21} \end{bmatrix} := U, \quad \text{with } U_{11} \in \mathbb{R}^{m \times m}.$$

Then

$$P^{(0)} := U_{21}U_{11}^{-1} \quad \text{and} \quad L^{(0)} := U_{11}\Sigma V^\top$$

define the Frobenius-norm optimal unweighted and unconstrained low rank approximation

$$\widehat{D}^{(0)} := \begin{bmatrix} I \\ P^{(0)} \end{bmatrix} L^{(0)}.$$

More sophisticated choices for the initial approximation that take into account the weight matrix Σ are described in Sect. 2.4.

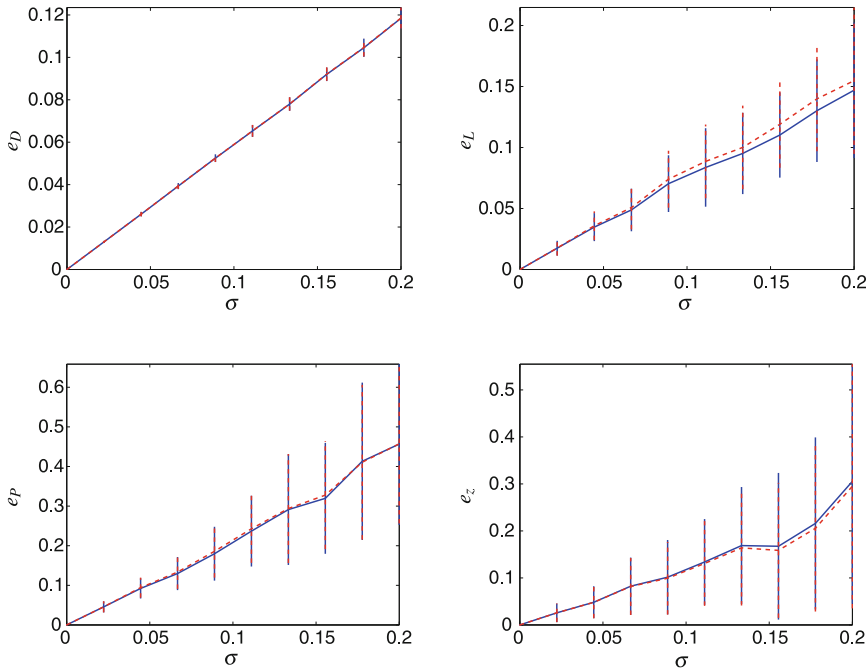


Fig. 5.6 Effect of nonnegativity of L (*solid line*—exploiting prior knowledge, *dashed line*—without exploiting prior knowledge, *vertical bars*—standard deviations)

Separable Least Squares Problem for P

In the weighted case, the projection on step 1 of the algorithm is computed separately for each row p^i of P . Let d^i be the i th row of D and w^i be the i th row of Σ . The problem

$$\text{minimize over } P \quad \|D - PL\|_{\Sigma}^2 \quad \text{subject to } (\mathbf{C}_1\text{--}\mathbf{C}_2)$$

is equivalent to the problem

$$\begin{aligned} & \text{minimize over } p^i \quad \|(d^i - p^i L) \text{diag}(w^i)\|_2^2 \\ & \text{subject to } (\mathbf{C}_1\text{--}\mathbf{C}_2), \quad \text{for } i = 1, \dots, m. \end{aligned} \quad (*)$$

The projection on step 2 of the algorithm is not separable due to constraint (\mathbf{C}_5) .

Taking into Account Constraint (\mathbf{C}_1)

Since the first m rows of P are fixed, we do not solve $(*)$ for $i = 1, \dots, m$, but define

$$p^i := e_i^\top, \quad \text{for } i = 1, \dots, m,$$

where e_i is the i th unit vector (the i th column of the identity matrix I_m).

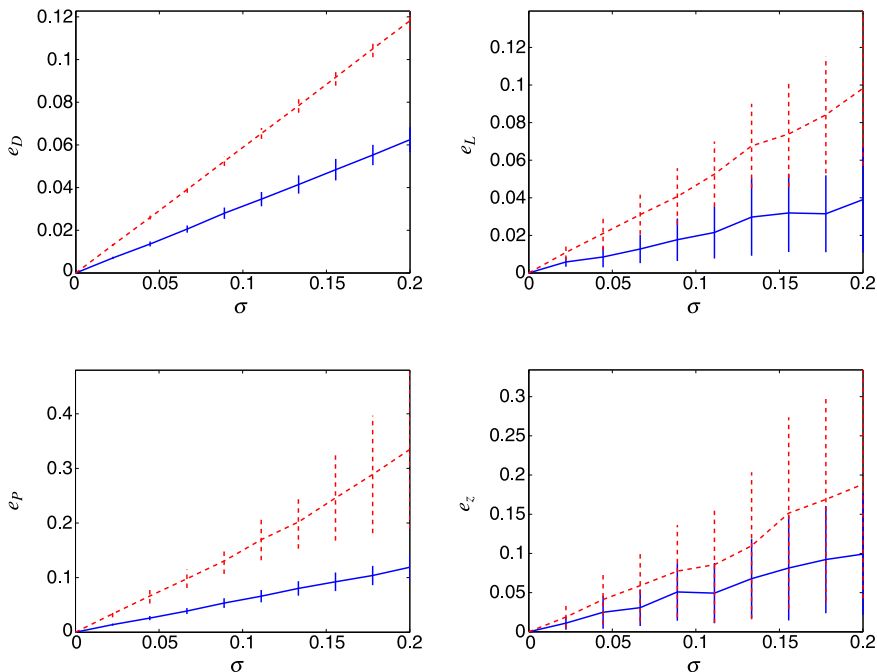


Fig. 5.7 Effect of weighting, periodicity, and nonnegativity of L , and zero elements in P (solid line—exploiting prior knowledge, dashed line—without exploiting prior knowledge, vertical bars—standard deviations)

Taking into Account Constraint (C₂)

Let S_i be a selector matrix for the zeros in the i th row of P

$$S \text{vec}(P') = 0 \iff p^i S_i = 0, \text{ for } i = m + 1, \dots, q.$$

(If there are no zeros in the i th row, then S_i is skipped.) The i th problem in (*) becomes

$$\text{minimize over } p^i \quad \|(d^i - p^i L) \text{diag}(w^i)\|_2^2 \quad \text{subject to } p^i S_i = 0. \quad (**)$$

Let the rows of the matrix N_i form a basis for the left null space of S_i . Then $p^i S_i = 0$ if and only if $p^i = z_i N_i$, for certain z_i , and problem (**) becomes

$$\text{minimize over } z_i \quad \|(d^i - z_i N_i L) \text{diag}(w^i)\|_2^2.$$

Therefore, the solution of (*) is

$$p^{i,*} = d^i L^\top N_i^\top (N_i L L^\top N_i^\top)^{-1} N_i.$$

Note 5.18 It is not necessary to explicitly construct the matrices S_i and compute basis N_i for their left null spaces. Since S_i is a selector matrix, it is a submatrix of the identity matrix I_m . The rows of the complementary submatrix of I_m form a basis for the left null space of S_i . This particular matrix N_i is also a selector matrix, so that the product $N_i L$ need not be computed explicitly.

Taking into Account Constraint (C₃)

We have,

$$\begin{aligned} D - PL &= D - P(\mathbf{1}_l^\top \otimes L') = [D_1 \cdots D_l] - P[L' \cdots L'] \\ &= \begin{bmatrix} D_1 \\ \vdots \\ D_l \end{bmatrix} - \begin{bmatrix} P \\ \vdots \\ P \end{bmatrix} L' =: D' - \underbrace{(\mathbf{1}_l \otimes P)}_{P'} L' = D' - P' L'. \end{aligned}$$

Let

$$\Sigma' := \begin{bmatrix} \Sigma_1 \\ \vdots \\ \Sigma_l \end{bmatrix}, \quad \text{where } \Sigma =: [\Sigma_1 \cdots \Sigma_N].$$

Then the problem

$$\text{minimize over } L \quad \|D - PL\|_{\Sigma}^2 \quad \text{subject to } (\mathbf{C}_3\text{--}\mathbf{C}_5)$$

is equivalent to the problem

$$\text{minimize over } L' \quad \|D' - P' L'\|_{\Sigma'}^2, \quad \text{subject to } (\mathbf{C}_4\text{--}\mathbf{C}_5).$$

Taking into Account Constraint (C₄)

Adding the nonnegativity constraint changes the least squares problem to a nonnegative least squares problem, which is a standard convex optimization problem for which robust and efficient methods and software exist.

Taking into Account Constraint (C₅)

The problem

$$\text{minimize over } L \quad \|D - PL\|_{\Sigma}^2 \quad \text{subject to } \|LD\|_{\mathbb{F}}^2 \leq \delta$$

is equivalent to a regularized least squares problem

$$\text{minimize over } L \quad \|D - PL\|_{\Sigma}^2 + \gamma \|LD\|_{\mathbb{F}}^2$$

for certain regularization parameter γ . The latter problem is equivalent to the standard least squares problem

$$\text{minimize over } L \quad \left\| \begin{bmatrix} \text{diag}(\text{vec}(\Sigma)) \text{vec}(D) \\ 0 \end{bmatrix} - \begin{bmatrix} \text{diag}(\text{vec}(\Sigma))(I \otimes P) \\ \sqrt{\gamma}(\mathbf{D}^\top \otimes I) \end{bmatrix} \text{vec}(L) \right\|_2^2.$$

Stopping Criteria

The iteration is terminated when the following stopping criteria are satisfied

$$\begin{aligned} & \|P^{(k+1)}L^{(k+1)} - P^{(k)}L^{(k)}\|_\Sigma / \|P^{(k+1)}L^{(k+1)}\|_\Sigma < \varepsilon_D, \\ & \|(P^{(k+1)} - P^{(k)})L^{(k+1)}\|_\Sigma / \|P^{(k+1)}L^{(k+1)}\|_\Sigma < \varepsilon_P, \text{ and} \\ & \|L^{(k+1)}(L^{(k+1)} - L^{(k)})\|_\Sigma / \|P^{(k+1)}L^{(k+1)}\|_\Sigma < \varepsilon_L. \end{aligned}$$

Here ε_D , ε_P , and ε_L are user defined relative convergence tolerances for D , P , and L , respectively.

5.5 Notes

Missing Data

Optimization methods for solving weighted low rank approximation problems with nonsingular weight matrix have been considered in the literature under different names:

- criss-cross multiple regression (Gabriel and Zamir 1979),
- Riemannian singular value decomposition (De Moor 1993),
- maximum likelihood principal component analysis (Wentzell et al. 1997),
- weighted low rank approximation (Manton et al. 2003), and
- weighted low rank approximation (Markovsky et al. 2005).

Gabriel and Zamir (1979) consider an element-wise weighted low rank approximation problem with diagonal weight matrix W , and propose an iterative solution method. Their method, however, does not necessarily converge to a minimum point, see the discussion in Gabriel and Zamir (1979, Sect. 6, p. 491). Gabriel and Zamir (1979) proposed an alternating projections algorithm for the case of unweighted approximation with missing values, i.e., $w_{ij} \in \{0, 1\}$. Their method was further generalized by Srebro (2004) for arbitrary weights.

The Riemannian singular value decomposition framework of De Moor (1993) includes the weighted low rank approximation problem with rank specification $r = \min(m, n) - 1$ and a diagonal weight matrix W as a special case. In De Moor (1993), an algorithm resembling the inverse power iteration algorithm is proposed. The method, however, has no proven convergence properties.

Manton et al. (2003) treat the problem as an optimization over a Grassman manifold and propose steepest decent and Newton type algorithms. The least squares nature of the problem is not exploited in this work and the proposed algorithms are not globally convergent.

The maximum likelihood principal component analysis method of Wentzell et al. (1997) is developed for applications in chemometrics, see also (Schuermans et al. 2005). This method is an alternating projections algorithm. It applies to the general weighted low rank approximation problems and is globally convergent. The convergence rate, however, is linear and the method could be rather slow when the $r + 1$ st and the r th singular values of the data matrix D are close to each other. In the unweighted case this situation corresponds to lack of uniqueness of the solution, cf., Theorem 2.23. The convergence properties of alternating projections algorithms are studied in Krijnen (2006), Kiers (2002).

An implementation of the singular value thresholding method in MATLAB is available at <http://svt.caltech.edu/>. Practical methods for solving the recommender system problem are given in Segaran (2007). The MovieLens data set is available from GroupLens (2009).

Nonnegative Low Rank Approximation

The notation $D \geq 0$ is used for a matrix $D \in \mathbb{R}^{q \times N}$ whose elements are nonnegative. A low rank approximation problem with element-wise nonnegativity constraint

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{D} && \|D - \widehat{D}\| \\ & \text{subject to} && \text{rank}(\widehat{D}) \leq m && \text{and } \widehat{D} \geq 0 \end{aligned} \tag{NNLRA}$$

arises in Markov chains (Vanluyten et al. 2006) and image mining (Lee and Seung 1999). Using the image representation, we obtain the following problem

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{D}, P \in \mathbb{R}^{q \times m}, \text{ and } L \in \mathbb{R}^{m \times N} && \|D - \widehat{D}\| \\ & \text{subject to} && \widehat{D} = PL && \text{and } P, L \geq 0, \end{aligned} \tag{NNLRA}_P$$

which is a relaxation of problem (NNLRA). The minimal m , for which (NNLRA)_P has a solution, is called the *positive rank* of \widehat{D} (Berman and Shaked-Monderer 2003). In general, the positive rank is less than or equal to the rank.

Note that due to the nonnegativity constraint on \widehat{D} , the problem cannot be solved using the variable projections method. (There is no closed-form solution for the equivalent problem with \widehat{D} eliminated.) The alternating projections algorithm, however, can be used almost without modification for the solution of the relaxed problem (NNLRA)_P. Let the norm $\|\cdot\|$ in (NNLRA) be the Frobenius norm. (In the context of Markov chains more adequate is the choice of the Kullback–Leibler divergence as a distance measure between D and \widehat{D} .) Then at each iteration step of the algorithm two least squares problems with nonnegativity constraint (i.e., standard

optimization problems) are solved. The resulting alternating least squares algorithm is Algorithm 7.

Algorithm 7 Alternating projections algorithm for nonnegative low rank approximation

Input: Data matrix D , desired rank m , and convergence tolerance ε .

1: Set $k := 0$ and compute an initial approximation $\widehat{D}^{(0)} := P^{(0)}L^{(0)}$ from the singular value decomposition by setting all negative elements to zero.

2: **repeat**

3: $k := k + 1$.

4: Solve: $L^{(k)} := \arg \min_L \|D - P^{(k-1)}L\|$ subject to $L \geq 0$.

5: Solve: $P^{(k)} := \arg \min_P \|D - PL^{(k)}\|$ subject to $P \geq 0$.

6: **until** $\|P^{(k-1)}L^{(k-1)} - P^{(k)}L^{(k)}\| < \varepsilon$

Output: A locally optimal solution $\widehat{D}^* := P^{(k)}L^{(k)}$ to problem (NNLRA $_P$).

References

- Berman A, Shaked-Monderer N (2003) Completely positive matrices. World Scientific, Singapore
- Bydder M (2010) Solution of a complex least squares problem with constrained phase. *Linear Algebra Appl* 433(11–12):1719–1721
- Cai JF, Candés E, Shen Z (2009) A singular value thresholding algorithm for matrix completion. www-stat.stanford.edu/~candes/papers/SVT.pdf
- Candés E, Recht B (2009) Exact matrix completion via convex optimization. *Found Comput Math* 9:717–772
- De Moor B (1993) Structured total least squares and L_2 approximation problems. *Linear Algebra Appl* 188–189:163–207
- Gabriel K, Zamir S (1979) Lower rank approximation of matrices by least squares with any choice of weights. *Technometrics* 21:489–498
- Golub G, Hoffman A, Stewart G (1987) A generalization of the Eckart–Young–Mirsky matrix approximation theorem. *Linear Algebra Appl* 88/89:317–327
- GroupLens (2009) Movielens data sets. www.grouplens.org/node/73
- Kiers H (2002) Setting up alternating least squares and iterative majorization algorithms for solving various matrix optimization problems. *Comput Stat Data Anal* 41:157–170
- Krijnen W (2006) Convergence of the sequence of parameters generated by alternating least squares algorithms. *Comput Stat Data Anal* 51:481–489
- Lee D, Seung H (1999) Learning the parts of objects by non-negative matrix factorization. *Nature* 401:788–791
- Manton J, Mahony R, Hua Y (2003) The geometry of weighted low-rank approximations. *IEEE Trans Signal Process* 51(2):500–514
- Markovsky I, Rastello ML, Premoli A, Kukush A, Van Huffel S (2005) The element-wise weighted total least squares problem. *Comput Stat Data Anal* 50(1):181–209
- Schuermans M, Markovsky I, Wentzell P, Van Huffel S (2005) On the equivalence between total least squares and maximum likelihood PCA. *Anal Chim Acta* 544:254–267
- Segarán T (2007) Programming collective intelligence: building smart Web 2.0 applications. O'Reilly Media
- Srebro N (2004) Learning with matrix factorizations. PhD thesis, MIT
- Vanluyten B, Willems JC, De Moor B (2006) Matrix factorization and stochastic state representations. In: Proc 45th IEEE conf on dec and control, San Diego, California, pp 4188–4193
- Wentzell P, Andrews D, Hamilton D, Faber K, Kowalski B (1997) Maximum likelihood principal component analysis. *J Chemom* 11:339–366

Chapter 6

Nonlinear Static Data Modeling

6.1 A Framework for Nonlinear Static Data Modeling

Introduction

Identifying a curve in a set of curves that best fits given data points is a common problem in computer vision, statistics, and coordinate metrology. More abstractly, approximation by Fourier series, wavelets, splines, and sum-of-exponentials are also curve-fitting problems. In the applications, the fitted curve is a model for the data and, correspondingly, the set of candidate curves is a model class.

Data modeling problems are specified by choosing a model class and a fitting criterion. The fitting criterion is maximization of a measure for fit between the data and a model. Equivalently, the criterion can be formulated as minimization of a measure for lack of fit (misfit) between the data and a model. Data modeling problems can be classified according to the type of model and the type of fitting criterion as follows:

- linear/affine vs. nonlinear model class,
- algebraic vs. geometric fitting criterion.

A model is a subset of the data space. The model is linear/affine if it is a subspace/affine set. Otherwise, it is nonlinear. A geometric fitting criterion minimizes the sum-of-squares of the Euclidean distances from the data points to a model. An algebraic fitting criterion minimizes an equation error (residual) in a representation of the model. In general, the algebraic fitting criterion has no simple geometric interpretation. Problems using linear model classes and algebraic criteria are easier to solve numerically than problems using nonlinear model classes and geometric criteria.

In this chapter, a nonlinear model class of bounded complexity, consisting of affine varieties, i.e., kernels of systems of multivariable polynomials is considered. The complexity of an affine variety is defined as the pair of the variety's dimension and the degree of its polynomial representation. In Sect. 6.2, an equivalence is established between the data modeling problem and low rank approximation of a polynomially structured matrix constructed from the data. Algorithms for solving

nonlinearly structured low rank approximation problems are presented in Sect. 6.3. As illustrated in Sect. 6.4, the low rank approximation setting makes it possible to use a single algorithm and a piece of software for solving a wide variety of curve-fitting problems.

Data, Model Class, and Model Complexity

We consider static multivariate modeling problems. The to-be-modeled data \mathcal{D} are a set of N observations (also called data points)

$$\mathcal{D} = \{d_1, \dots, d_N\} \subset \mathbb{R}^q.$$

The observations d_1, \dots, d_N are real q -dimensional vectors. A model for the data \mathcal{D} is a subset of the data space \mathbb{R}^q and a model class \mathcal{M}^q for \mathcal{D} is a set of subsets of the data space \mathbb{R}^q , i.e., \mathcal{M}^q is an element of the powerset $2^{\mathbb{R}^q}$. For example, the linear model class in \mathbb{R}^q consists of the subspaces of \mathbb{R}^q . An example of a nonlinear model class in \mathbb{R}^2 is the set of the conic sections. When the dimension q of the data space is understood from the context, it is skipped from the notation of the model class.

In nonlinear data modeling problems, the model is usually represented by a function $y = f(u)$, where $d = \Pi \operatorname{col}(u, y)$, with Π a permutation matrix. The corresponding statistical estimation problem is regression. As in the linear case, we call the functional relation $y = f(u)$ among the variables u and y , an input/output representation of the model

$$\mathcal{B} = \{\Pi \operatorname{col}(u, y) \mid y = f(u)\} \quad (\text{I/O})$$

that this relation defines. The input/output representation $y = f(u)$ implies that the variables u are inputs and the variables y are outputs of the model \mathcal{B} .

Input-output representations are appealing because they are *explicit functions*, mapping some variables (inputs) to other variables (outputs) and thus display a causal relation among the variables (the inputs cause the outputs). The alternative kernel representation

$$\mathcal{B} = \ker(R) := \{d \in \mathbb{R}^q \mid R(d) = 0\} \quad (\text{KER})$$

defines the model via an *implicit function* $R(d) = 0$, which does not a priori bound one set of variables as a cause and another set of variables as an effect.

A priori fixed causal relation, imposed on the model by an input/output representation, is restrictive. Consider, for example, data fitting by a model that is a conic section. Only parabolas and lines can be represented by functions. Hyperbolas, ellipses, and the vertical line $\{(u, y) \mid u = 0\}$ are not graphs of a function $y = f(u)$ and therefore cannot be modeled by an input/output representation.

The complexity of a linear static model \mathcal{B} is defined as the dimension of \mathcal{B} , i.e., the smallest number m , such that there is a linear function $P : \mathbb{R}^m \rightarrow \mathbb{R}^q$, for which

$$\mathcal{B} = \text{image}(P) := \{P(\ell) \mid \ell \in \mathbb{R}^m\}. \quad (\text{IMAGE})$$

Similarly, the dimension of a nonlinear model \mathcal{B} is defined as the smallest natural number m , such that there is a (possibly nonlinear) function $P : \mathbb{R}^m \rightarrow \mathbb{R}^q$, for which (IMAGE) holds. In the context of nonlinear models, however, the model dimension alone is not sufficient to define the model complexity. For example, in \mathbb{R}^2 both a linear model (a line passing through the origin) and an ellipse have dimension equal to one, however, it is intuitively clear that the ellipse is a more “complex” model than the line.

The missing element in the definition of the model complexity in the nonlinear case is the “complexity” of the function P . In what follows, we restrict to models that can be represented as kernels of polynomial functions, i.e., we consider models that are *affine varieties*. Complexity of an affine variety (IMAGE) is defined as the pair (m, \bar{d}) , where m is the dimension of \mathcal{B} and \bar{d} is the degree of R . This definition allows us to distinguish a linear or affine model ($\bar{d} = 1$) from a nonlinear model ($\bar{d} > 1$) with the same dimension. For a model \mathcal{B} with complexity (m, \bar{d}) , we call \bar{d} the degree of \mathcal{B} .

The complexity of a model class is the maximal complexity (in a lexicographic ordering of the pairs (m, \bar{d})) over all models in the class. The model class of complexity bounded by (m, \bar{d}) is denoted by $\mathcal{P}_{m, \bar{d}}$.

Special Cases

The model class $\mathcal{P}_{m, \bar{d}}^q$ and the related exact and approximate modeling problems (EM) and (AM) have as an important special case the linear model class and linear data modeling problems.

1. *Linear/affine model class of bounded complexity.* An affine model \mathcal{B} (i.e., an affine set in \mathbb{R}^q) is an affine variety, defined by a first order polynomial through kernel or image representation. The dimension of the affine variety coincides with the dimension of the affine set. Therefore, $\mathcal{P}_{m, 1}^q$ is an affine model class in \mathbb{R}^q with complexity bounded by m . The linear model class in \mathbb{R}^q , with dimension bounded by m , is a subset $\mathcal{L}_{m, 0}^q$ of $\mathcal{P}_{m, 1}^q$.
2. *Geometric fitting by a linear model.* Approximate data modeling using the linear model class \mathcal{L}_m^q and the geometric fitting criterion (dist) is a low rank approximation problem

$$\begin{aligned} & \text{minimize} && \text{over } \hat{\mathcal{D}} && \|\Phi(\mathcal{D}) - \Phi(\hat{\mathcal{D}})\|_F \\ & \text{subject to} && \text{rank}(\Phi(\hat{\mathcal{D}})) \leq m, \end{aligned} \quad (\text{LRA})$$

where

$$\Phi(\mathcal{D}) := [d_1 \cdots d_N].$$

The rank constraint in (LRA) is equivalent to the constraint that the data $\widehat{\mathcal{D}}$ are exact for a linear model of dimension bounded by m . This justifies the statement that exact modeling is an ingredient of approximate modeling.

3. *Algebraic curves.* In the special case of a curve in the plane, we use the notation

$$x := \text{first component } d_1. \text{ of } d \quad \text{and} \quad y := \text{second component } d_2. \text{ of } d.$$

Note that $w = \text{col}(x, y)$ is not necessarily an input/output partitioning of the variables. An affine variety of dimension one is called an algebraic curve.

A second order algebraic curve

$$\mathcal{B} = \{d = \text{col}(x, y) \mid d^\top A d + b^\top d + c = 0\},$$

where $A = A^\top$, b , and c are parameters, is a conic section. Examples of third order algebraic curves, see Fig. 6.1, are the cissoid

$$\mathcal{B} = \{\text{col}(x, y) \mid y^2(1+x) - (1-x)^3 = 0\},$$

and the folium of Descartes

$$\mathcal{B} = \{\text{col}(x, y) \mid x^3 + y^3 - 3xy = 0\}.$$

Examples of 4rd order algebraic curves, see Fig. 6.2, are the eight curve

$$\mathcal{B} = \{\text{col}(x, y) \mid y^2 - x^2 + x^4 = 0\}$$

and the limaçon of Pascal

$$\mathcal{B} = \{\text{col}(x, y) \mid y^2 + x^2 - (4x^2 - 2x + 4y^2)^2 = 0\}.$$

The four-leaved rose, see Fig. 6.3,

$$\mathcal{B} = \{\text{col}(x, y) \mid (x^2 + y^2)^3 - 4x^2y^2 = 0\}$$

is a sixth order algebraic curve.

6.2 Nonlinear Low Rank Approximation

Parametrization of the Kernel Representations

Consider a kernel representation (KER) of an affine variety $\mathcal{B} \in \mathcal{P}_{m,q}^{\mathfrak{q}}$, parametrized by a $p \times 1$ multivariable polynomial R . The number of monomials in q variables

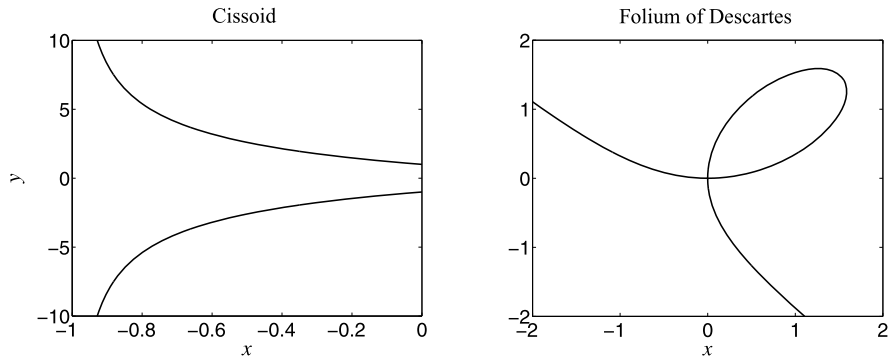


Fig. 6.1 Examples of algebraic curves of third order

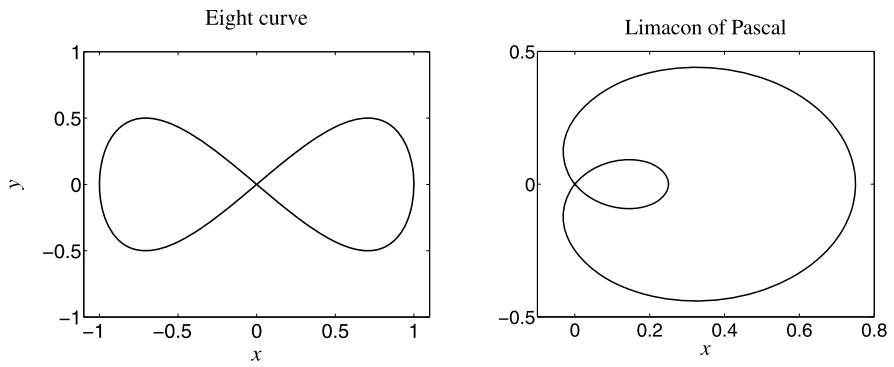
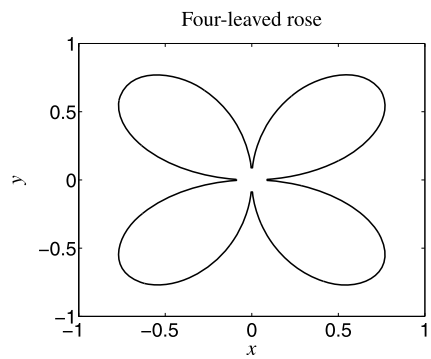


Fig. 6.2 Examples of algebraic curves of fourth order

Fig. 6.3 Example of an algebraic curve of sixth order



with degree d or less is

$$q_{\text{ext}} := \binom{q+d}{d} = \frac{(q+d)!}{d!q!}. \quad (q_{\text{ext}})$$

Define the vector of all such monomials

$$\phi(d) := [\phi_1(d) \cdots \phi_{\mathfrak{q}_{\text{ext}}}(d)]^\top.$$

The polynomial R can be written as

$$R_\Theta(d) = \sum_{k=1}^{\mathfrak{q}_{\text{ext}}} \Theta_k \phi_k(d) = \Theta \phi(d), \quad (R_\Theta)$$

where Θ is an $\mathfrak{p} \times \mathfrak{q}_{\text{ext}}$ parameter matrix.

In what follows, we assume that the monomials are ordered in $\phi(d)$ in decreasing degree according to the lexicographic ordering (with alphabet the indices of d). For example, with $\mathfrak{q} = 2$, $\mathfrak{d} = 2$, and $d = \text{col}(x, y)$,

$$\begin{aligned} \mathfrak{q}_{\text{ext}} = 6 \quad \text{and} \quad \phi^\top(x, y) &= [\phi_1 \ \phi_2 \ \phi_3 \ \phi_4 \ \phi_5 \ \phi_6] \\ &= [x^2 \ xy \ x \ y^2 \ y \ 1]. \end{aligned}$$

In general,

$$\phi_k(d) = d_1^{\mathfrak{d}_{k1}} \cdots d_{\mathfrak{q}}^{\mathfrak{d}_{k\mathfrak{q}}}, \quad \text{for } k = 1, \dots, \mathfrak{q}_{\text{ext}}, \quad (\phi_k)$$

where

- $d_1, \dots, d_{\mathfrak{q}} \in \mathbb{R}$ are the elements of $d \in \mathbb{R}^{\mathfrak{q}}$, and
- $\mathfrak{d}_{ki} \in \mathbb{Z}_+$, is the degree of the i th element of d in the k th monomial ϕ_k .

The matrix formed from the degrees \mathfrak{d}_{ki}

$$D = [\mathfrak{d}_{ki}] \in \mathbb{R}^{\mathfrak{q}_{\text{ext}} \times \mathfrak{q}}$$

uniquely defines the vector of monomials ϕ . The degrees matrix D depends only on the number of variables \mathfrak{q} and the degree \mathfrak{d} . For example, with $\mathfrak{q} = 2$ and $\mathfrak{d} = 2$,

$$D^\top = \begin{bmatrix} 2 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 1 & 0 \end{bmatrix}.$$

The function `monomials` generates an implicit function `phi` that evaluates the 2-variate vector of monomials ϕ , with degree \mathfrak{d} .

184a `<Monomials constructor 184a> ≡` 184b>
`function [Deg, phi] = monomials(deg)`

Defines:

`monomials`, used in chunks 190a and 192e.

First an extended degrees matrix $D_{\text{ext}} \in \{0, 1, \dots, \mathfrak{d}\}^{(\mathfrak{d}+1)^2 \times 2}$, corresponding to all monomials $x^{\mathfrak{d}_x} y^{\mathfrak{d}_y}$ with degrees at most \mathfrak{d} , is generated. It can be verified that

$$D_{\text{ext}} = [\mathbf{r}_{\mathfrak{d}} \otimes \mathbf{1}_{\mathfrak{d}+1} \quad \mathbf{1}_{\mathfrak{d}+1} \otimes \mathbf{r}_{\mathfrak{d}}], \quad \text{where } \mathbf{r}_{\mathfrak{d}} := [0 \ 1 \ \cdots \ \mathfrak{d}]^\top$$

is such a matrix; moreover, the monomials are ordered in decreasing degree.

184b `<Monomials constructor 184a> ≡` <184a 185>
`Deg_ext = [kron([0:deg]', ones(deg + 1, 1)), ...`
`kron(ones(deg + 1, 1), [0:deg]')];`

Then the rows of D_{ext} are scanned and those with degree less than or equal to d are selected to form a matrix D .

```

185 (Monomials constructor 184a)+≡ <184b
    str = []; Deg = []; q = 2;
    for i = 1:size(Deg_ext, 1)
        if (sum(Deg_ext(i, :)) <= deg)
            for k = q:-1:1,
                str = sprintf('.* d(%d,:) .^ %d %s', ...
                               k, Deg_ext(i, k), str);
            end
            str = sprintf('; %s', str(4:end));
            Deg = [Deg_ext(i, :); Deg];
        end
    end
    eval(sprintf('phi = @(d) [%s];', str(2:end)))

```

Minimality of the kernel representation is equivalent to the condition that the parameter Θ is full row rank. The nonuniqueness of R_Θ corresponds to a nonuniqueness of Θ . The parameters Θ and $Q\Theta$, where Q is a nonsingular matrix, define the same model. Therefore, without loss of generality, we can assume that the representation is minimal and normalize it, so that

$$\Theta\Theta^\top = I_p.$$

Note that a $p \times q_{\text{ext}}$ full row rank matrix Θ defines via (R_Θ) a polynomial matrix R_Θ , which defines a minimal kernel representation (KER) of a model \mathcal{B}_Θ in $\mathcal{P}_{m,d}^q$. Therefore, Θ defines a function

$$\mathcal{B}_\Theta : \mathbb{R}^{p \times q_{\text{ext}}} \rightarrow \mathcal{P}_{m,d}^q.$$

Vice versa, a model \mathcal{B} in $\mathcal{P}_{m,d}^q$ corresponds to a (nonunique) $p \times q_{\text{ext}}$ full row rank matrix Θ , such that $\mathcal{B} = \mathcal{B}_\Theta$. For a given q , there are one-to-one mappings

$$q_{\text{ext}} \leftrightarrow d \quad \text{and} \quad p \leftrightarrow m,$$

defined by (q_{ext}) and $p = q - m$, respectively.

Main Results

We show a relation of the approximate modeling problems (AM) and (EM) for the model class $\mathcal{P}_{m,d}$ to low rank approximation problems.

Proposition 6.1 (Algebraic fit \iff unstructured low rank approximation) *The algebraic fitting problem for the model class of affine varieties with bounded com-*

plexity $\mathcal{P}_{m,d}$

$$\begin{aligned} & \text{minimize over } \Theta \in \mathbb{R}^{p \times q_{\text{ext}}} && \sqrt{\sum_{j=1}^N \|R_{\Theta}(d_j)\|_F^2} && (\text{AM}'_{\Theta}) \\ & \text{subject to } && \Theta \Theta^{\top} = I_p \end{aligned}$$

is equivalent to the unstructured low rank approximation problem

$$\begin{aligned} & \text{minimize over } \widehat{\Phi} \in \mathbb{R}^{q \times p} && \|\Phi_{\mathfrak{d}}(\mathcal{D}) - \widehat{\Phi}\|_F && (\text{LRA}) \\ & \text{subject to } && \text{rank}(\widehat{\Phi}) \leq q_{\text{ext}} - p. \end{aligned}$$

Proof Using the polynomial representation (R_{Θ}), the squared cost function of (AM'_{Θ}) can be rewritten as a quadratic form:

$$\begin{aligned} \sum_{j=1}^N \|R_{\Theta}(d_j)\|_F^2 &= \|\Theta \Phi_{\mathfrak{d}}(\mathcal{D})\|_F^2 \\ &= \text{trace}(\Theta \Phi_{\mathfrak{d}}(\mathcal{D}) \Phi_{\mathfrak{d}}^{\top}(\mathcal{D}) \Theta^{\top}) = \text{trace}(\Theta \Psi_{\mathfrak{d}}(\mathcal{D}) \Theta^{\top}). \end{aligned}$$

Therefore, the algebraic fitting problem is equivalent to an eigenvalue problem for $\Psi_{\mathfrak{d}}(\mathcal{D})$ or, equivalently (see the Notes section of Chap. 2), to low rank approximation problem for $\Phi_{\mathfrak{d}}(\mathcal{D})$. \square

Proposition 6.2 (Geometric fit \iff polynomial structured low rank approx) *The geometric fitting problem for the model class of affine varieties with bounded complexity $\mathcal{P}_{m,d}$*

$$\text{minimize over } \mathcal{B} \in \mathcal{P}_{m,d} \quad \text{dist}(\mathcal{D}, \mathcal{B}) \quad (\text{AM})$$

is equivalent to the polynomially structured low rank approximation problem

$$\begin{aligned} & \text{minimize over } \widehat{D} \in \mathbb{R}^{q \times N} && \|D - \widehat{D}\|_F && (\text{PSLRA}) \\ & \text{subject to } && \text{rank}(\Phi_{\mathfrak{d}}(\widehat{D})) \leq q_{\text{ext}} - p. \end{aligned}$$

Proof Problem (AM) is equivalent to

$$\begin{aligned} & \text{minimize over } \widehat{\mathcal{D}} \text{ and } \mathcal{B} && \sqrt{\sum_{j=1}^N \|d_j - \widehat{d}_j\|_2^2} && (*) \\ & \text{subject to } && \widehat{\mathcal{D}} \subset \mathcal{B} \in \mathcal{P}_{m,d}. \end{aligned}$$

Using the condition

$$\widehat{\mathcal{D}} \subset \mathcal{B} \in \mathcal{P}_{m,d} \implies \text{rank}(\Phi_{\mathfrak{d}}(\widehat{\mathcal{D}})) \leq q_{\text{ext}} - p \quad (\text{MPUM})$$

to replace the constraint of $(*)$ with a rank constraint for the structured matrix $\Phi_{\mathfrak{a}}(\widehat{\mathcal{D}}) = \Phi_{\mathfrak{a}}(\widehat{D})$, this latter problem becomes a polynomially structured low rank approximation problem (PSLRA). \square

Propositions 6.1 and 6.2 show a relation between the algebraic and geometric fitting problems.

Corollary 6.3 *The algebraic fitting problem (AM'_{Θ}) is a relaxation of the geometric fitting problem (AM) , obtained by removing the structure constraint of the approximating matrix $\Phi_{\mathfrak{a}}(\widehat{D})$.*

6.3 Algorithms

In the linear case, the misfit computation problem is a linear least norm problem. This fact is effectively used in the variable projections method. In the nonlinear case, the misfit computation problem is a nonconvex optimization problem. Thus the elimination step of the variable projections approach is not possible in the nonlinear case. This requires the data approximation $\widehat{\mathcal{D}} = \{\widehat{d}_1, \dots, \widehat{d}_N\}$ to be treated as an extra optimization variable together with the model parameter Θ . As a result, the computational complexity and sensitivity to local minima increases in the nonlinear case.

The above consideration makes critical the choice of the initial approximation. The default initial approximation is obtained from a direct method such as the algebraic fitting method. Next, we present a modification of the algebraic fitting method that is motivated by the objective of obtaining an unbiased estimate in the errors-in-variables setup.

Bias Corrected Low Rank Approximation

Assume that the data \mathcal{D} is generated according to the errors-in-variables model

$$\begin{aligned} d_j &= d_{0,j} + \widetilde{d}_j, \quad \text{where } d_{0,j} \in \mathcal{B}_0 \in \mathcal{P}_{m,q} \quad \text{and} \\ \text{vec}([\widetilde{d}_1 \ \cdots \ \widetilde{d}_N]) &\sim \mathbf{N}(0, \sigma^2 I_{qN}). \end{aligned} \tag{EIV}$$

Here \mathcal{B}_0 is the to-be-estimated true model. The estimate $\widehat{\mathcal{B}}$ obtained by the algebraic fitting method (AM'_{Θ}) is biased, i.e., $\mathbf{E}(\widehat{\mathcal{B}}) \neq \mathcal{B}_0$. In this section, we derive a bias correction procedure. The correction depends on the noise variance σ^2 , however, the noise variance can be estimated from the data \mathcal{D} together with the model parameter $\widehat{\Theta}$. The resulting bias corrected estimate $\widehat{\mathcal{B}}_c$ is invariant to rigid transformations. Simulation results show that $\widehat{\mathcal{B}}_c$ has smaller orthogonal distance to the data than alternative direct methods.

Table 6.1 Explicit expressions of the Hermite polynomials h_2, \dots, h_{10}

	$h_2(x) = x^2 - 1$
	$h_3(x) = x^3 - 3x$
	$h_4(x) = x^4 - 6x^2 + 3$
	$h_5(x) = x^5 - 10x^3 + 15x$
	$h_6(x) = x^6 - 15x^4 + 45x^2 - 15$
	$h_7(x) = x^7 - 21x^5 + 105x^3 - 105x$
	$h_8(x) = x^8 - 28x^6 + 210x^4 - 420x^2 + 105$
	$h_9(x) = x^9 - 36x^7 + 378x^5 - 1260x^3 + 945x$
	$h_{10}(x) = x^{10} - 45x^8 + 630x^6 - 3150x^4 + 4725x^2 - 945$

Define the matrices

$$\Psi := \Phi_d(\mathcal{D})\Phi_d^\top(\mathcal{D}) \quad \text{and} \quad \Psi_0 := \Phi_d(\mathcal{D}_0)\Phi_d^\top(\mathcal{D}_0).$$

The algebraic fitting method computes the rows of parameter estimate $\widehat{\Theta}$ as eigenvectors related to the p smallest eigenvalues of Ψ . We construct a “corrected” matrix Ψ_c , such that

$$\mathbf{E}(\Psi_c) = \Psi_0. \quad (*)$$

This property ensures that the corrected estimate $\widehat{\Theta}_c$, obtained from the eigenvectors related to the p smallest eigenvalues of Ψ_c , is unbiased.

188a *(Bias corrected low rank approximation 188a)* \equiv
 function [th, sh] = bclra(D, deg)
 [q, N] = size(D); qext = nchoosek(q + deg, deg);
(construct the corrected matrix Ψ_c 190a)
(estimate σ^2 and θ 190c)

Defines:

bclra, used in chunk 192e.

The key tool to achieve bias correction is the sequence of the Hermite polynomials, defined by the recursion

$$h_0(x) = 1, \quad h_1(x) = x, \quad \text{and} \\ h_k(x) = xh_{k-1}(x) - (k-2)h_{k-2}(x), \quad \text{for } k = 2, 3, \dots$$

(See Table 6.1 for explicit expressions of h_2, \dots, h_{10} .) The Hermite polynomials have the deconvolution property

$$\mathbf{E}(h_k(x_0 + \tilde{x})) = x_0^k, \quad \text{where } \tilde{x} \sim \mathbf{N}(0, 1). \quad (**)$$

The following code generates a cell array h of implicit function that evaluate the sequence of Hermite polynomials: $h\{k+1\}(d) = h_k(d)$. (The difference in the indices of the h and h is due to MATLAB convention indices to be positive integers.)

188b *(define the Hermite polynomials 188b)* \equiv (190a)
 h{1} = @(x) 1; h{2} = @(x) x;

```

for k = 3:(2 * deg + 1)
    h{k} = @(x) [x * h{k - 1}(x) zeros(1, mod(k - 2, 2))] ...
               - [0 (k - 2) * h{k - 2}(x)];
end

```

We have,

$$\Psi = \sum_{\ell=1}^N \phi(d_\ell) \phi^\top(d_\ell) = \sum_{\ell=1}^N [\phi_i(d_\ell) \phi_j(d_\ell)],$$

and, from (ϕ_k) , the (i, j) th element of Ψ is

$$\psi_{ij} = \sum_{\ell=1}^N d_{1\ell}^{\tilde{d}_{i1} + \tilde{d}_{j1}} \dots d_{q\ell}^{\tilde{d}_{iq} + \tilde{d}_{jq}} = \sum_{\ell=1}^N \prod_{k=1}^q (d_{0,k\ell} + \tilde{d}_{k\ell})^{\tilde{d}_{iq} + \tilde{d}_{jq}}.$$

By the data generating assumption (EIV), $\tilde{d}_{k\ell}$ are independent, zero mean, normally distributed. Then, using the deconvolution property (**) of the Hermite polynomials, we have

$$\psi_{c,ij} := \sum_{\ell=1}^N \prod_{k=1}^q h_{\tilde{d}_{iq} + \tilde{d}_{jq}}(d_{k\ell}) \quad (\psi_{ij})$$

has the unbiasedness property (*), i.e.,

$$\mathbf{E}(\psi_{c,ij}) = \sum_{\ell=1}^N \prod_{k=1}^q d_{0,k\ell}^{\tilde{d}_{iq} + \tilde{d}_{jq}} =: \psi_{0,ij}.$$

The elements $\psi_{c,ij}$ of the corrected matrix are even polynomials of σ of degree less than or equal to

$$d_\psi = \left\lceil \frac{qd + 1}{2} \right\rceil.$$

The following code constructs a $1 \times (d_\psi + 1)$ vector of the coefficients of $\psi_{c,ij}$ as a polynomial of σ^2 . Note that the product of Hermite polynomials in (ψ_{ij}) is a convolution of their coefficients.

```

189 (construct  $\psi_{c,ij}$  189) ≡ (190a)
    Deg_ij = Deg(i, :) + Deg(j, :);
    for l = 1:N
        psi_ijl = 1;
        for k = 1:q
            psi_ijl = conv(psi_ijl, h{Deg_ij(k) + 1}(D(k, l)));
        end
        psi_ijl = [psi_ijl zeros(1, dpsi - length(psi_ijl))];
        psi(i, j, :) = psi(i, j, :) + ...
            reshape(psi_ijl(1:dpsi), 1, 1, dpsi);
    end

```

The corrected matrix

$$\Psi_c(\sigma^2) = \Psi_{c,0} + \sigma^2 \Psi_{c,1} + \dots + \sigma^{2d_\psi} \Psi_{c,d_\psi}$$

is then obtained by computing its elements in the lower triangular part

```
190a <construct the corrected matrix  $\Psi_c$  190a>≡ (188a) 190b>
    <define the Hermite polynomials 188b>
    Deg = monomials(deg);
    dpsi = ceil((q * deg + 1) / 2);
    psi = zeros(qext, qext, dpsi);
    for i = 1:qext
        for j = 1:qext
            if i >= j
                <construct  $\psi_{c,ij}$  189>
            end
        end
    end
end
```

Uses `monomials` 184a.

and using the symmetry property to fill in the upper triangular part

```
190b <construct the corrected matrix  $\Psi_c$  190a>+≡ (188a) <190a>
    for k = 1:dpsi,
        psi(:, :, k) = psi(:, :, k) + triu(psi(:, :, k)', 1);
    end
```

The rows of the parameter $\hat{\Theta}$ form a basis for the p -dimensional (approximate) null space of $\Psi_c(\sigma^2)$

$$\Theta \Psi_c(\sigma^2) = 0.$$

Computing simultaneously σ and Θ is a *polynomial eigenvalue problem*: the noise variance estimate is the minimum eigenvalue and the parameter estimate is a corresponding eigenvector.

```
190c <estimate  $\sigma^2$  and  $\theta$  190c>≡ (188a)
    [evec, ev] = polyeig_(psi); ev(find(ev < 0)) = inf;
    [sh2, min_ind] = min(ev);
    sh = sqrt(sh2); th = evec(:, min_ind);
```

(The function `polyeig_` is a minor modification of the standard MATLAB function `polyeig`. The input to `polyeig_` is a 3-dimensional tensor while the input to `polyeig` is a sequence of matrices—the slices of the tensor in the third dimension.)

Method Based on Local Optimization

The nonlinearly structured low rank approximation problem (**PSLRA**) is solved numerically using Optimization Toolbox.

```

191a <Polynomially structured low rank approximation 191a>≡ 191b>
function [th, Dh, info] = pslra(D, phi, r, xini)
[q, N] = size(D); nt = size(phi(D), 1);

```

Defines:

pslra, used in chunk 192e.

If not specified, the initial approximation is taken as the algebraic fit and the noisy data points.

```

191b <Polynomially structured low rank approximation 191a>+≡ <191a 191c>
if (nargin < 4) | isempty(xini)
tini = lra(phi(D), r); xini = [D(:); tini(:)];
end

```

Uses lra 64.

Anonymous functions that extract the data approximation $\widehat{\mathcal{D}}$ and the model parameter θ from the optimization parameter \mathbf{x} are defined next.

```

191c <Polynomially structured low rank approximation 191a>+≡ <191b 191d>
Dh = @(x) reshape(x(1:(q * N)), q, N);
th = @(x) reshape(x((q * N + 1):end), nt - r, nt)';

```

The optimization problem is set and solved, using the Optimization Toolbox:

```

191d <Polynomially structured low rank approximation 191a>+≡ <191c>
(set optimization solver and options 85a)
prob.objective = @(x) norm(D - Dh(x), 'fro');
prob.nonlcon = @(x) deal([], ...
    [th(x)' * phi(Dh(x)),
    th(x)' * th(x) - eye(nt - r)]);
prob.x0 = xini;
(call optimization solver 85b) Dh = Dh(x); th = th(x);

```

6.4 Examples

In this section, we apply the algebraic and geometric fitting methods on a range of algebraic curve fitting problems. In all examples, except for the last one, the data \mathcal{D} are simulated in the errors-in-variables setup, see (EIV) on p. 187. The perturbations \tilde{d}_j , $j = 1, \dots, N$ are independent, zero mean, normally distributed 2×1 vectors with covariance matrix $\sigma^2 I_2$. The true model $\mathcal{B}_0 = \ker(r_0)$, the number of data points N , and the perturbation standard deviation σ are simulation parameters. The true model is plotted by a solid line, the data points by circles, the algebraic fit by a dotted line, the bias corrected fit by **dashed dotted line**, and the geometric fit by a **dashed line**.

Test Function

The test script `test_curve_fitting` assumes that the simulation parameters:

- polynomial r in x and y , defined as a symbolic object;
- degree d of r ;
- number of data points N ;
- noise standard deviation σ ; and
- coordinates ax of a rectangle for plotting the results

are already defined.

192a *(Test curve fitting 192a)*≡
(initialize the random number generator 89e)
(default parameters 192b)
(generate data 192c)
(fit data 192e)
(plot results 192f)

Defines:

`test_curve_fitting`, used in chunks 193–95.

If not specified, $q = 2$, $m = 1$.

192b *(default parameters 192b)*≡ (192a)
`if ~exist('q'), q = 2; end`
`if ~exist('m'), m = 1; end`
`if ~exist('xini'), xini = []; end`

The true (D0) and noisy (D) data points are generated by plotting the true model

192c *(generate data 192c)*≡ (192a) 192d>
`figure,`
`H = plot_model(r, ax, 'LineStyle', '-', 'color', 'k');`

Uses `plot_model 193a`.

and sampling N equidistant points on the curve

192d *(generate data 192c)*+≡ (192a) <192c
`D0 = [];`
`for h = H',`
`D0 = [D0 [get(h, 'XData'); get(h, 'YData')]];`
`end`
`D0 = D0(:, round(linspace(1, size(D0, 2), N)));`
`D = D0 + s * randn(size(D0));`

The data are fitted by the algebraic (`lra`), bias corrected (`bclra`), and geometric (`pslra`) fitting methods:

192e *(fit data 192e)*≡ (192a 196)
`qext = nchoosek(q + deg, deg); p = q - m;`
`[Deg, phi] = monomials(deg);`
`th_exc = lra(phi(D), qext - p)';`
`th_ini = bclra(D, deg);`
`[th, Dh] = pslra(D, phi, qext - p, xini);`

Uses `bclra 188a`, `lra 64`, `monomials 184a`, and `pslra 191a`.

The noisy data and the fitted models are plotted on top of the true model:

192f *(plot results 192f)*≡ (192a 196)
`hold on; plot(D(1,:), D(2,:), 'o', 'markersize', 7);`

```

plot_model(th2poly(th_exc, phi), ax, ...
           'LineStyle', ':', 'color', 'k');
plot_model(th2poly(th_ini, phi), ax, ...
           'LineStyle', '-.', 'color', 'r');
plot_model(th2poly(th, phi), ax, ...
           'LineStyle', '-', 'color', 'b');
axis(ax); print_fig(sprintf('%s-est', name))

```

Uses `plot_model` 193a, `print_fig` 25a, and `th2poly` 193b.

Plotting the algebraic curve

$$\mathcal{B} = \{d \mid \phi(d)\theta = 0\}$$

in a region, defined by the vector `rect`, is done with the function `plot_model`.

```

193a (Plot the model 193a)≡
function H = plot_model(r, rect, varargin)
H = ezplot(r, rect);
if nargin > 2, for h = H', set(h, varargin{:}); end, end

```

Defines:
`plot_model`, used in chunk 192.

The function `th2poly` converts a vector of polynomial coefficients to a function that evaluates that polynomial.

```

193b ( $\theta \mapsto R_\theta$  193b)≡
function r = th2poly(th, phi),
r = @(x, y) th' * phi([x y]');

```

Defines:
`th2poly`, used in chunk 192f.

Fitting Algebraic Curves in \mathbb{R}^2

Simulation 1: Parabola

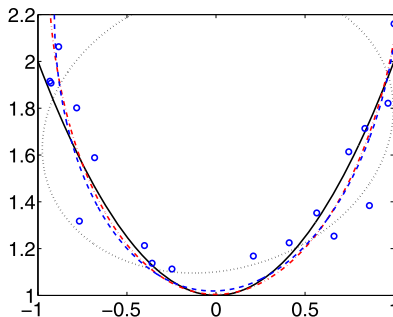
$$\mathcal{B} = \{\text{col}(x, y) \mid y = x^2 + 1\}$$

```

193c (Curve fitting examples 193c)≡
clear all
name = 'parabola';
N = 20; s = 0.1;
deg = 2; syms x y;
r = x^2 - y + 1;
ax = [-1 1 1 2.2];
test_curve_fitting

```

Uses `test_curve_fitting` 192a.



194a▷

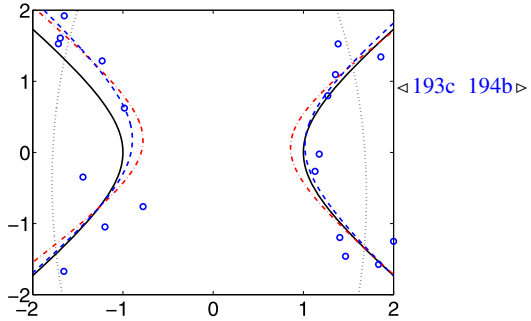
Simulation 2: Hyperbola

$$\mathcal{B} = \{\text{col}(x, y) \mid x^2 - y^2 - 1 = 0\}$$

194a

```
(Curve fitting examples 193c)+≡
name = 'hyperbola';
N = 20; s = 0.3;
deg = 2; syms x y;
r = x^2 - y^2 - 1;
ax = [-2 2 -2 2];
test_curve_fitting
```

Uses test_curve_fitting 192a.

**Simulation 3: Cissoid**

$$\mathcal{B} = \{\text{col}(x, y) \mid y^2(1+x) = (1-x)^3\}$$

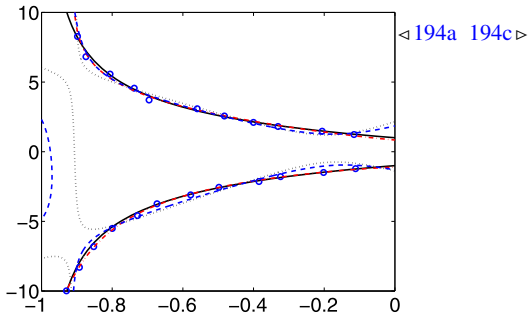
194b

```
(Curve fitting examples 193c)+≡
name = 'cissoid';
N = 25; s = 0.02;
deg = 3; syms x y;
r = y^2 * (1 + x) ...
    - (1 - x)^3;
ax = [-1 0 -10 10];
test_curve_fitting
```

Defines:

```
examples_curve_fitting,
never used.
```

Uses test_curve_fitting 192a.

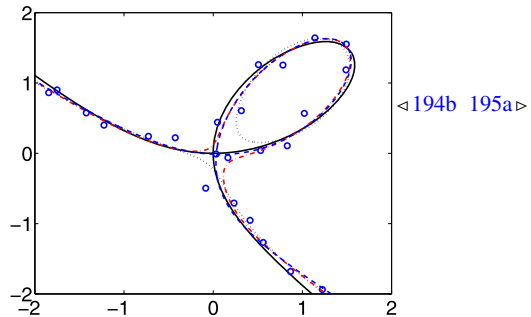
**Simulation 4: Folium of Descartes**

$$\mathcal{B} = \{\text{col}(x, y) \mid x^3 + y^3 - 3xy = 0\}$$

194c

```
(Curve fitting examples 193c)+≡
name = 'folium';
N = 25; s = 0.1;
deg = 3; syms x y;
r = x^3 + y^3
    - 3 * x * y;
ax = [-2 2 -2 2];
test_curve_fitting
```

Uses test_curve_fitting 192a.



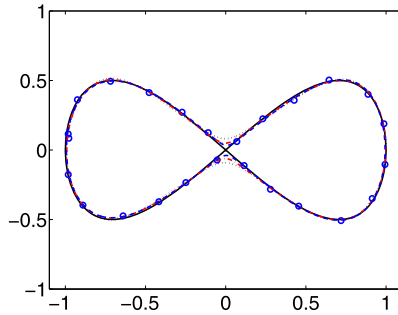
Simulation 5: Eight curve

$$\mathcal{B} = \{ \text{col}(x, y) \mid y^2 - x^2 + x^4 = 0 \}$$

195a

```
(Curve fitting examples 193c)+≡
name = 'eight';
N = 25; s = 0.01;
deg = 4; syms x y;
r = y^2 - x^2 + x^4;
ax = [-1.1 1.1 -1 1];
test_curve_fitting
```

Uses test_curve_fitting 192a.



◀194c 195b▶

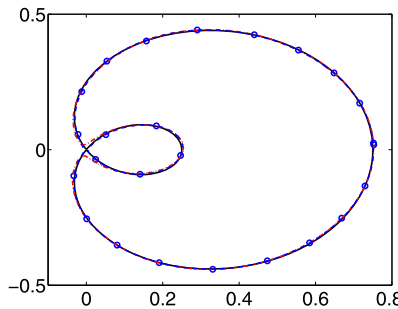
Simulation 6: Limacon of Pascal

$$\mathcal{B} = \{ \text{col}(x, y) \mid y^2 + x^2 - (4x^2 - 2x + 4y^2)^2 = 0 \}$$

195b

```
(Curve fitting examples 193c)+≡
name = 'limacon';
N = 25; s = 0.002;
deg = 4; syms x y;
r = y^2 + x^2
    - (4 * x^2 ...
      - 2 * x
      + 4 * y^2)^2;
ax = [-.1 .8 -0.5 .5];
test_curve_fitting
```

Uses test_curve_fitting 192a.



◀195a 195c▶

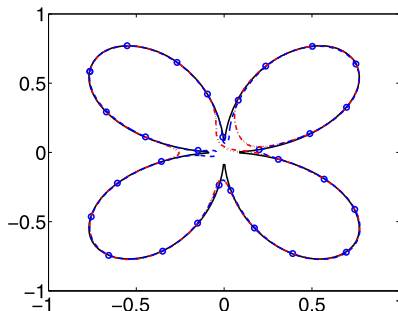
Simulation 7: Four-leaved rose

$$\mathcal{B} = \{ (x, y) \mid (x^2 + y^2)^3 - 4x^2y^2 = 0 \}$$

195c

```
(Curve fitting examples 193c)+≡
name = 'rose';
N = 30; s = 0.002;
deg = 6; syms x y;
r = (x^2 + y^2)^3 ...
    - 4 * x^2 * y^2;
ax = [-1 1 -1 1];
test_curve_fitting
```

Uses test_curve_fitting 192a.

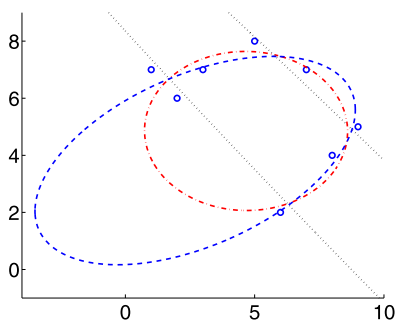


◀195b 196▶

Simulation 8: “Special data” example from Gander et al. (1994)

196

```
(Curve fitting examples 193c)+≡
name = 'special-data';
D = [1 2 5 7 9 3 6 8 ;
     7 6 8 7 5 7 2 4 ];
D0 = D; deg = 2;
xini = [D(:)' 1 0 0
        1 0 -1]';
figure,
ax = [-4 10 -1 9];
<fit data 192e> <plot results 192f>
```



< 195c

6.5 Notes

Fitting curves to data is a basic problem in coordinate metrology, see Van Huffel (1997, Part IV). In the computer vision literature, there is a large body of work on ellipsoid fitting (see, e.g., Bookstein 1979; Gander et al. 1994; Kanatani 1994; Fitzgibbon et al. 1999; Markovsky et al. 2004), which is a special case of the considered in this chapter data fitting problem when the degree of the polynomial is equal to two.

In the systems and control literature, the geometric distance is called misfit and the algebraic distance is called latency, see Lemmerling and De Moor (2001). Identification of a linear time-invariant dynamical systems, using the latency criterion leads to the autoregressive moving average exogenous setting, see Ljung (1999) and Söderström and Stoica (1989). Identification of a linear time-invariant dynamical systems, using the misfit criterion leads to the errors-in-variables setting, see Söderström (2007).

State-of-the art image segmentation methods are based on the *level set* approach (Sethian 1999). Level set methods use implicit equations to represent a contour in the same way we use kernel representations to represent a model in this chapter. The methods used for parameter estimation in the level set literature, however, are based on solution of partial differential equations while we use classical parameter estimation/optimization methods.

Relaxation of the nonlinearly structured low rank approximation problem, based on ignoring the nonlinear structure and thus solving the problem as unstructured low rank approximation (i.e., the algebraic fitting method) is known in the machine learning literature as *kernel principal component analysis* (Schölkopf et al. 1999). The *principal curves*, introduced in Hastie and Stuetzle (1989), lead to a problem of minimizing the sum of squares of the distances from data points to a curve. This is a polynomially structured low rank approximation problem. More generally, dimensionality reduction by manifold learning, see, e.g., Zhang and Zha (2005) is

related to the problem of fitting an affine variety to data, which is also polynomially structured low rank approximation.

Nonlinear (Vandermonde) structured total least squares problems are discussed in Lemmerling et al. (2002) and Rosen et al. (1998) and are applied to fitting a sum of damped exponentials model to data. Fitting a sum of damped exponentials to data, however, can be solved as a linear (Hankel) structured approximation problem. In contrast, the geometric data fitting problem considered in this chapter in general cannot be reduced to a linearly structured problem and is therefore a genuine application of nonlinearly structured low rank approximation.

The problem of passing from image to kernel representation of the model is known as the *implicitization problem* (Cox et al. 2004, p. 96) in computer algebra. The reverse transformation—passing from a kernel to an image representation of the model, is a problem of solving a system of multivariable polynomial equations.

References

- Bookstein FL (1979) Fitting conic sections to scattered data. *Comput Graph Image Process* 9:59–71
- Cox D, Little J, O’Shea D (2004) *Ideals, varieties, and algorithms*. Springer, Berlin
- Fitzgibbon A, Pilu M, Fisher R (1999) Direct least-squares fitting of ellipses. *IEEE Trans Pattern Anal Mach Intell* 21(5):476–480
- Gander W, Golub G, Strebel R (1994) Fitting of circles and ellipses: least squares solution. *BIT Numer Math* 34:558–578
- Hastie T, Stuetzle W (1989) Principal curves. *J Am Stat Assoc* 84:502–516
- Kanatani K (1994) Statistical bias of conic fitting and renormalization. *IEEE Trans Pattern Anal Mach Intell* 16(3):320–326
- Lemmerling P, De Moor B (2001) Misfit versus latency. *Automatica* 37:2057–2067
- Lemmerling P, Van Huffel S, De Moor B (2002) The structured total least squares approach for nonlinearly structured matrices. *Numer Linear Algebra Appl* 9(1–4):321–332
- Ljung L (1999) *System identification: theory for the user*. Prentice-Hall, Upper Saddle River
- Markovsky I, Kukush A, Van Huffel S (2004) Consistent least squares fitting of ellipsoids. *Numer Math* 98(1):177–194
- Rosen J, Park H, Glick J (1998) Structured total least norm for nonlinear problems. *SIAM J Matrix Anal Appl* 20(1):14–30
- Schölkopf B, Smola A, Müller K (1999) *Kernel principal component analysis*. MIT Press, Cambridge, pp 327–352
- Sethian J (1999) *Level set methods and fast marching methods*. Cambridge University Press, Cambridge
- Söderström T (2007) Errors-in-variables methods in system identification. *Automatica* 43:939–958
- Söderström T, Stoica P (1989) *System identification*. Prentice Hall, New York
- Van Huffel S (ed) (1997) *Recent advances in total least squares techniques and errors-in-variables modeling*. SIAM, Philadelphia
- Zhang Z, Zha H (2005) Principal manifolds and nonlinear dimension reduction via local tangent space alignment. *SIAM J Sci Comput* 26:313–338

Chapter 7

Fast Measurements of Slow Processes

7.1 Introduction

The core idea developed in this chapter is expressed in the following problem from Luenberger (1979, p. 53):

Problem 7.1 A thermometer reading 21°C , which has been inside a house for a long time, is taken outside. After one minute the thermometer reads 15°C ; after two minutes it reads 11°C . What is the outside temperature? (According to Newton's law of cooling, an object of higher temperature than its environment cools at a rate that is proportional to the difference in temperature.)

The solution of the problem shows that measurement of a signal from a "slow" processes can be speeded up by data processing. The solution applies to the special case of exact data from a first order single input single output linear time-invariant system. Our purpose is to generalize Problem 7.1 and its solution to multi input multi output processes with higher order linear time-invariant dynamics and to make the solution a practical tool for improvement of the speed-accuracy characteristics of measurement devices by signal processing.

A method for speeding up of measurement devices is of generic interest. Specific applications can be found in process industry where the process, the measurement device, or both have slow dynamics, e.g., processes in biotechnology that involve chemical reactions and convection. Of course, the notion of "slow process" is relative. There are applications, e.g., weight measurement, where the dynamics may be fast according to the human perception but slow according to the state-of-the-art technological requirements.

The dynamics of the process is assumed to be linear time-invariant but otherwise it need not be known. Knowledge of the process dynamics simplifies considerably the problem. In some applications, however, such knowledge is not available a priori. For example, in Problem 7.1 the heat exchange coefficient (cooling time constant) depends on unknown environmental factors, such as pressure and humidity. As another example, consider the dynamic weighing problem, where the unknown

mass of the measured object affects the dynamics of the weighting process. The linearity assumption is justifiable on the basis that nonlinear dynamical processes can be approximated by linear models, and existence of an approximate linear model is often sufficient for solving the problem to a desirable accuracy. The time-invariance assumption can be relaxed in the recursive estimation algorithms proposed by using windowing of the processed signal and forgetting factor in the recursive update formula.

Although only the steady-state measurement value is of interest, the solution of Problem 7.1 identifies explicitly the process dynamics as a byproduct. The proposed methods for dynamic weighing (see the notes and references section) are also model-based and involve estimation of model parameters. Similarly, the generalized problem considered reduces to a system identification question—find a system from step response data. Off-line and recursive methods for solving this latter problem, under different noise assumptions, exist in the literature, so that these methods can be readily used for input estimation.

In this chapter, a method for estimation of the parameter of interest—the input step value—without identifying a model of the measurement process is described. The key idea that makes model-free solution possible comes from work on data-driven estimation and control. The method proposed requires only a solution of a system of linear equations and is computationally more efficient and less expensive to implement in practice than the methods based on system identification. Modifications of the model-free algorithm compute approximate solutions that are optimal for different noise assumptions. The modifications are obtained by replacing exact solution of a system of linear equations with approximate solution by the least squares method or one of its variations. On-line versions of the data-driven method, using ordinary or weighted least squares approximation criterion, are obtained by using a standard recursive least squares algorithms. Windowing of the processed signal and forgetting factor in the recursive update allow us to make the algorithm adaptive to time-variation of the process dynamics.

The possibility of using theoretical results, algorithms, and software for different standard identification methods in the model based approach and approximation methods in the data-driven approach lead to a range of new methods for speedup of measurement devices. These new methods inherit desirable properties (such as consistency, statistical and computational efficiency) from the identification or approximation methods being used.

The generalization of Problem 7.1 considered is defined in this section. In Sect. 7.2, the simpler version of the problem when the measurement process dynamics is known is solved by reducing the problem to an equivalent state estimation problem for an augmented autonomous system. Section 7.3 reduces the general problem without known model to standard identification problems—identification from step response data as well as identification in a model class of autonomous system (sum-of-damped exponentials modeling). Then the data-driven method is derived as a modification of the method based on autonomous system identification. Examples and numerical results showing the performance of the methods are presented in Sect. 7.4. The proofs of the statements are almost identical for the con-

tinuous and discrete-time cases, so that only the proof for the discrete-time case is given.

Problem Formulation

Problem 7.1 can be restated more abstractly as follows.

Problem 7.2 A step input $u = \bar{u}s$, where $\bar{u} \in \mathbb{R}$, is applied to a first order stable linear time-invariant system with unit dc gain.¹ Find the input step level \bar{u} from the output values $y(0) = 21$, $y(1) = 15$, and $y(2) = 11$.

The heat exchange dynamics in Problem 7.1 is indeed a first order stable linear time-invariant system with unit dc gain and step input. The input step level \bar{u} is therefore equal to the steady-state value of the output and is the quantity of interest. Stating Problem 7.1 in system theoretic terms opens a way to generalizations and makes clear the link of the problem to system identification. The general problem considered is defined as follows.

Problem 7.3 Given output observations

$$y = (y(t_1), \dots, y(t_T)), \quad \text{where } y(t) \in \mathbb{R}^p$$

at given moments of time $0 \leq t_1 < \dots < t_T$, of a stable linear time-invariant system $\mathcal{B} \in \mathcal{L}_{m,n}^{p+m}$ with known dc gain $G \in \mathbb{R}^{p \times m}$, generated by a step input $u = \bar{u}s$ (but not necessarily zero initial conditions), find the input step value $\bar{u} \in \mathbb{R}^m$.

The known dc gain assumption imposed in Problem 7.3 corresponds to calibration of the measurement device in real-life metrological applications. Existence of a dc gain, however, requires stability hence the process is assumed to be a stable dynamical system. By this assumption, in a steady-state regime, the output of the device is $\bar{y} = G\bar{u}$, so that, provided G is a full column rank matrix, the value of interest is uniquely determined as $\bar{u} = G^+ \bar{y}$, where G^+ is a left inverse of G . Obviously, without prior knowledge of G and without G beginning full column rank, \bar{u} cannot be inferred from \bar{y} . Therefore, we make the following standing assumption.

Assumption 7.4 The measurement process dynamics \mathcal{B} is a stable linear time-invariant system with full column rank dc gain, i.e.,

$$\text{rank}(G) = m, \quad \text{where } G := \text{dcgain}(\mathcal{B}) \in \mathbb{R}^{p \times m}.$$

¹The dc (or steady-state) gain of a linear time-invariant system, defined by an input/output representation with transfer function H , is $G = H(0)$ in the continuous-time case and $G = H(1)$ in the discrete-time case. As the name suggest the dc gain G is the input-output amplification factor in a steady-state regime, i.e., constant input and output.

Note 7.5 (Exact vs. noisy observations) Problem 7.3, aims to determine the *exact* input value \bar{u} . Correspondingly, the given observations y of the sensor are assumed to be exact. Apart from the problem of finding the exact value of \bar{u} from exact data y , the practically relevant estimation and approximation problems are considered, where the observations are subject to measurement noise

$$y = y_0 + \tilde{y}, \quad \text{where } y_0 \in \mathcal{B} \text{ and } \tilde{y} \text{ is white process with } \tilde{y}(t) \sim N(0, V) \quad (\text{OE})$$

and/or the process dynamics is not finite dimensional linear time-invariant.

Despite the unrealistic assumption of exact observations, Problem 7.3 poses a valid theoretical question that needs to be answered before more realistic approximate and stochastic versions are considered. In addition to its theoretical (and pedagogical) value, the solution of Problem 7.3 leads to algorithms for computation of \bar{u} that with small modifications (approximate rather than exact identification and approximate solution of an overdetermined system of equations) produce approximations of \bar{u} in the case of noisy data and/or process dynamics that is not linear time-invariant.

We start in the next section by solving the simpler problem of speedup of a sensor when the measurement process dynamics \mathcal{B} is known.

Problem 7.6 Given a linear time-invariant system $\mathcal{B} \in \mathcal{L}_{m,n}^{m+p}$ and an output trajectory y of \mathcal{B} , obtained from a step input $u = \bar{u}s$, find the step level $\bar{u} \in \mathbb{R}^m$.

7.2 Estimation with Known Measurement Process Dynamics

Proposition 7.7 Define the maps between an n th order linear time-invariant system with m inputs and an $n + m$ th order autonomous system with m poles at 0 in the continuous-time case, or at 1 in the discrete-time case:

$$\mathcal{B} \mapsto \mathcal{B}_{\text{aut}} \quad \text{by} \quad \mathcal{B}_{\text{aut}} := \{y \mid \text{there is } \bar{u}, \text{ such that } (\bar{u}s, y) \in \mathcal{B}\}$$

and

$$\mathcal{B}_{\text{aut}} \mapsto \mathcal{B} \quad \text{by} \quad \mathcal{B} := \mathcal{B}_{i/s/o}(A, B, C, D), \quad \text{where } \mathcal{B}_{\text{aut}} = \mathcal{B}_{i/s/o}(A_{\text{aut}}, B_{\text{aut}}),$$

with

$$A_{\text{aut}} := \begin{bmatrix} A & B \\ 0 & I_m \end{bmatrix} \quad \text{and} \quad C_{\text{aut}} := [C \ D]. \quad (*)$$

Then

$$(\bar{u}s, y) \in \mathcal{B} \in \mathcal{L}_{m,n}^{m+p} \quad \iff \quad y \in \mathcal{B}_{\text{aut}} \in \mathcal{L}_{0,n+m}^p \quad \text{and } \mathcal{B}_{\text{aut}} \text{ has}$$

m poles at 0 in the continuous-time case, or at 1 in the discrete-time case.

The state vector x of $\mathcal{B}_{i/s/o}(A, B, C, D)$ and the state vector x_{aut} of $\mathcal{B}_{i/s/o}(A_{\text{aut}}, C_{\text{aut}})$, are related by $x_{\text{aut}} = (x, \bar{u})$.

Proof

$$\begin{aligned} (\bar{u}s, y) \in \mathcal{B} &= \mathcal{B}_{i/s/o}(A, B, C, D) \\ \iff \sigma x &= Ax + B\bar{u}s, \quad y = Cx + D\bar{u}s, \quad x(0) = x_{\text{ini}} \\ \iff \sigma x &= Ax + B\bar{u}s, \quad \sigma \bar{u} = \bar{u}, \quad y = Cx + D\bar{u}s, \quad x(0) = x_{\text{ini}} \\ \iff \sigma x_{\text{aut}} &= A_{\text{aut}}x_{\text{aut}}, \quad y = C_{\text{aut}}x_{\text{aut}}, \quad x_{\text{aut}}(0) = (x_{\text{ini}}, \bar{u}) \\ \iff y \in \mathcal{B}_{\text{aut}} &= \mathcal{B}_{i/s/o}(A_{\text{aut}}, B_{\text{aut}}). \quad \square \end{aligned}$$

Proposition 7.7 shows that Problem 7.6 can be solved as a state estimation problem for the augmented system $\mathcal{B}_{i/s/o}(A_{\text{aut}}, B_{\text{aut}})$. In the case of discrete-time exact data y , a dead-beat observer computes the exact state vector in a finite (less than $n + m$) time steps. In the case of noisy observations (OE), the maximum-likelihood state estimator is the Kalman filter.

The algorithm for input estimation, resulting from Proposition 7.7 is

203a *(Algorithm for sensor speedup in the case of known dynamics 203a)* \equiv
function uh = `stepid_kf`(y, a, b, c, d, v, x0, p0)
[p, m] = size(d); n = size(a, 1);
if nargin == 6,
x0 = zeros(n + m, 1); p0 = 1e8 * eye(n + m);
end
(model augmentation: $\mathcal{B} \mapsto \mathcal{B}_{\text{aut}}$ 203b)
(state estimation: $(y, \mathcal{B}_{\text{aut}}) \mapsto x_{\text{aut}} = (x, \bar{u})$ 203c)

Defines:

`stepid_kf`, used in chunks 213 and 222a.

The obligatory inputs to the function `stepid_kf` are a $T \times p$ matrix y of uniformly sampled outputs ($y(t) = y(t, :)'$), parameters a, b, c, d of a state space representation $\mathcal{B}_{i/s/o}(A, B, C, D)$ of the measurement process, and the output noise variance v . The output uh is a $T \times m$ matrix of the sequence of parameter \bar{u} estimates $\hat{u}(t) = uh(t, :)'$. The first step $\mathcal{B} \mapsto \mathcal{B}_{\text{aut}}$ of the algorithm is implemented as follows, see (*).

203b *(model augmentation: $\mathcal{B} \mapsto \mathcal{B}_{\text{aut}}$ 203b)* \equiv (203a)
a_aut = [a b; zeros(m, n) eye(m)]; c_aut = [c d];

Using the function `tvkf_oe`, which implements the time-varying Kalman filter for an autonomous system with output noise, the second step

$$(y, \mathcal{B}_{\text{aut}}) \mapsto x_{\text{aut}} = (x, \bar{u})$$

of the algorithm is implemented as follows.

203c *(state estimation: $(y, \mathcal{B}_{\text{aut}}) \mapsto x_{\text{aut}} = (x, \bar{u})$ 203c)* \equiv (203a)
xeh = `tvkf_oe`(y, a_aut, c_aut, v, x0, p0);
uh = xeh((n + 1):end, :);

Uses `tvkf_oe` 222c.

The optional parameters x_0 and p_0 of `stepid_kf` specify prior knowledge about mean and covariance of the initial state $x_{\text{aut}}(0)$. The default value is a highly uncertain initial condition.

Denote by n_f the order of the Kalman filter. We have $n_f = n + m$. The computational cost of processing the data by the time-varying Kalman filter is $O(n_f^2 + n_f p)$ floating point operations per discrete-time step, provided the filter gain is precomputed and stored off-line. Note, however, that in the solution of Problem 7.3, presented in Sect. 7.3, the equivalent autonomous model has state dimension $n + 1$, independent of the value of m .

Note 7.8 (Comparison of other methods with `stepid_kf`) In the case of noisy data (OE), `stepid_kf` is a statistically optimal estimator for the parameter \bar{u} . Therefore, the performance of `stepid_kf` is an upper bound for the achievable performance with the methods described in the next section.

Although `stepid_kf` is theoretically optimal estimator of \bar{u} , in practice it need not be the method of choice even when an a priori given model for the measurement process and the noise covariance are available. The reason for this, perhaps paradoxical statement, is that in practice the model of the process and the noise covariance are not known exactly and the performance of the method may degrade significantly as a result of the uncertainty in the given prior knowledge. The alternative methods that do not rely on a priori given model, may be more robust in case of large uncertainty. In specific cases, this claim can be justified theoretically on the basis of the sensitivity of the Kalman filter to the model parameters. In general cases, the claim can be observed experimentally or by numerical simulation.

7.3 Estimation with Unknown Measurement Process Dynamics

Solution by Reduction to Identification from Step Response Data

Problem 7.3 resembles a classical identification problem of finding a linear time-invariant system from step response data, except that

- the input is unknown,
- the dc gain of the system is constrained to be equal to the given matrix G , and
- the goal of the problem is to find the input rather than the unknown system dynamics.

As shown next, the first two peculiarities of Problem 7.3 are easily dealt with. The third one is addressed by the data-driven algorithm.

The following proposition shows that Problem 7.3 can be solved equivalently as a standard system identification problem from step response data.

Proposition 7.9 Let $P \in \mathbb{R}^{m \times m}$ be a nonsingular matrix and define the mappings

$$(P, \mathcal{B}) \mapsto \mathcal{B}', \quad \text{by } \mathcal{B}' := \{(Pu, y) \mid (u, y) \in \mathcal{B}\}$$

and

$$(P, \mathcal{B}') \mapsto \mathcal{B}, \quad \text{by } \mathcal{B} := \{(P^{-1}u, y) \mid (u, y) \in \mathcal{B}'\}.$$

Then, under Assumption 7.4, we have

$$\begin{aligned} (\bar{u}s, y) \in \mathcal{B} \in \mathcal{L}_{m,n}^{m+p} \quad \text{and} \quad \text{dcgain}(\mathcal{B}) = G \\ \iff (\mathbf{1}_m s, y) \in \mathcal{B}' \in \mathcal{L}_{m,n}^{m+p} \quad \text{and} \quad \text{dcgain}(\mathcal{B}') = G', \end{aligned} \quad (*)$$

where $\bar{u} = P\mathbf{1}_m$ and $GP = G'$.

Proof Obviously, $\mathcal{B} \in \mathcal{L}_{m,n}^{m+p}$ and $\text{dcgain}(\mathcal{B}) = G$ implies $\mathcal{B}' \in \mathcal{L}_{m,n}^{m+p}$ and

$$\text{dcgain}(\mathcal{B}') = GP =: G'.$$

Vice versa, if $\mathcal{B}' \in \mathcal{L}_{m,n}^{m+p}$ and $\text{dcgain}(\mathcal{B}') = G'$, then $\mathcal{B} \in \mathcal{L}_{m,n}^{m+p}$ and

$$\text{dcgain}(\mathcal{B}) = P^{-1}G' = G.$$

With $\bar{y} := \lim_{t \rightarrow \infty} y(t)$, we have

$$(\mathbf{1}_m s, y) \in \mathcal{B}' \implies G'\mathbf{1}_m = \bar{y} \quad \text{and} \quad (\bar{u}s, y) \in \mathcal{B} \implies G\bar{u} = \bar{y}.$$

Therefore, $G'\mathbf{1}_m = GP\mathbf{1}_m = G\bar{u}$. Finally, using Assumption 7.4, we obtain $P\mathbf{1}_m = \bar{u}$. \square

Note 7.10 The input value $\mathbf{1}_m$ in (*) is arbitrary. The equivalence holds for any nonzero vector \bar{u}' , in which case $\bar{u} = P\bar{u}'$.

The importance of Proposition 7.9 stems from the fact that while in the left-hand side of (*) the input \bar{u} is unknown and the gain G is known, in the right-hand side of (*), the input $\mathbf{1}_m$ is known and the gain G' is unknown. Therefore, for the case $p = m$, the standard identification problem of finding $\mathcal{B}' \in \mathcal{L}_{m,n}^{p+m}$ from the data $(\mathbf{1}_m s, y)$ is equivalent to Problem 7.3, i.e., find $\bar{u} \in \mathbb{R}^m$ and $\mathcal{B} \in \mathcal{L}_{m,n}^{p+m}$, such that $(\mathbf{1}_m s, y) \in \mathcal{B}$ and $\text{dcgain}(\mathcal{B}) = G$. (The $p = m$ condition is required in order to ensure that the system $GP = G'$ has a unique solution P for any $p \times m$ full column rank matrices G and G' .)

Next, we present an algorithm for solving Problem 7.3 using Proposition 7.9.

205 *(Algorithm for sensor speedup based on reduction to step response system identification 205)* \equiv
 function [uh, sysh] = `stepid_si`(y, g, n)
(system identification: $(\mathbf{1}_m s, y) \mapsto \mathcal{B}'$ 206a)
($\bar{u} := G^{-1}G'\mathbf{1}_m$, where $G' := \text{dcgain}(\mathcal{B}')$ 206b)

Defines:

`stepid_si`, never used.

The input to the function `stepid_si` is a $T \times p$ matrix y of uniformly sampled output values ($y(t) = y(t, :)'$), a nonsingular $p \times m$ matrix g , and the system order n . For simplicity, in `stepid_si` as well as in the following algorithms, the order n of the measurement process \mathcal{B} is assumed known. In case of unknown system order, one of the many existing order estimation methods (see, Stoica and Selén 2004) can be used.

For exact data, the map $(\mathbf{1}_{m,s}, y) \mapsto \mathcal{B}'$ is the exact identification problem of computing the most powerful unfalsified model of $(\mathbf{1}_{m,s}, y)$ in the model class of linear-time-invariant systems. For noisy data, an approximation of the “true” system \mathcal{B}' is obtained by an approximate identification method. The approximation criterion should reflect known properties of the noise, e.g., in the case of observations corrupted by measurement noise (OE), an output error identification method should be selected.

Using the function `ident_oe`, the pseudo-code of `stepid_si` is completed as follows.

206a $\langle \text{system identification: } (\mathbf{1}_{m,s}, y) \mapsto \mathcal{B}' \text{ 206a} \rangle \equiv$ (205)
`sysh = ident_oe([y ones(size(y))], n);`

Uses `ident_oe 117a`.

and

206b $\langle \bar{u} := G^{-1}G'\mathbf{1}_m, \text{ where } G' := \text{dcgain}(\mathcal{B}') \text{ 206b} \rangle \equiv$ (205)
`[p, m] = size(g); uh = sum(g \ dcgain(sysh), 2);`

Note 7.11 (Output error identification) The shown implementation of `stepid_si` is optimal (maximum likelihood) for data y generated according to (OE).

Note 7.12 (Recursive implementation) The function `ident_oe` is an off-line identification method. Replacing it with the recursive identification methods results in a recursive algorithm for solution of Problem 7.3. The easy modification of the pseudo-code for different situations, such as different noise assumptions, different model class, and batch vs. on-line implementation, is possible by the reduction of Problem 7.3 to a standard problem in system identification, for which well developed theory, algorithms, and software exist.

Solution by Identification of an Autonomous System

An alternative way of solving Problem 7.3 is to exploit its equivalence to autonomous system identification, stated in the following proposition.

Proposition 7.13 *Define the maps*

$$(\mathcal{B}, \bar{u}) \mapsto \mathcal{B}'_{\text{aut}}, \quad \text{by } \mathcal{B}'_{\text{aut}} := \{y \mid (s\bar{u}, y) \in \mathcal{B}\}$$

and

$$(\mathcal{B}'_{\text{aut}}, G) \mapsto \mathcal{B}, \quad \text{by } \mathcal{B} := \mathcal{B}_{i/s/o}(A, B, C, D),$$

$$\text{where } \mathcal{B}'_{\text{aut}} = \mathcal{B}_{i/s/o} \left(\begin{bmatrix} A & B\bar{u} \\ 0 & \lambda_1 \end{bmatrix}, [C \ D\bar{u}] \right),$$

$\lambda_1 = 0$ in the continuous-time case or $\lambda_1 = 1$ in the discrete-time case.

We have

$$(s\bar{u}, y) \in \mathcal{B} \in \mathcal{L}_{m,n}^{m+p} \quad \text{and} \quad \text{dcgain}(\mathcal{B}) = G \iff y \in \mathcal{B}'_{\text{aut}} \in \mathcal{L}_{0,n+1}^p \quad \text{and} \\ \mathcal{B}'_{\text{aut}} \text{ has a pole at } 0 \text{ in the continuous-time or } 1 \text{ in the discrete-time.} \quad (*)$$

Proof Let z_1, \dots, z_n be the poles of \mathcal{B} , i.e.,

$$\{z_1, \dots, z_n\} := \lambda(\mathcal{B}).$$

An output y of \mathcal{B} to a step input $u = \bar{u}s$ is of the form

$$y(t) = \left(\bar{u} + \sum_{i=1}^n \alpha_i p_i(t) z_i^t \right) s(t), \quad \text{for all } t,$$

where p_i is a polynomial function (of degree equal to the multiplicity of z_i minus one), and $\alpha_i \in \mathbb{R}^n$. This shows that y is a response of an autonomous linear time-invariant system with poles $1 \cup \lambda(\mathcal{B})$.

A system $\mathcal{B}'_{\text{aut}} \in \mathcal{L}_{0,n+1}^p$ with a pole at 1, admits a minimal state space representation

$$\mathcal{B}_{i/s/o} \left(\begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix}, [C \ d] \right).$$

The “ \Rightarrow ” implication in (*) follows from the definition of the map $(\mathcal{B}, \bar{u}) \mapsto \mathcal{B}'_{\text{aut}}$. The “ \Leftarrow ” implication in (*) follows from

$$\begin{aligned} y \in \mathcal{B}'_{\text{aut}} = \mathcal{B}_{i/s/o} \left(\begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix}, [C \ d] \right) \\ \implies & \text{ there exist initial conditions } x(0) \in \mathbb{R}^n \text{ and } z(0) = z_{\text{ini}} \in \mathbb{R}, \\ & \text{ such that } \sigma x = Ax + bz, \quad y = Cx + dz, \quad \sigma z = z \\ \implies & \text{ there exist initial condition } x(0) = x_{\text{ini}} \in \mathbb{R}^n \text{ and } z \in \mathbb{R}, \\ & \text{ such that } \sigma x = Ax + bz, \quad y = Cx + dz \\ \implies & (zs, y) \in \mathcal{B} = \mathcal{B}_{i/s/o}(A, b, C, d). \end{aligned}$$

Using the prior knowledge about the dc gain of the system, we have

$$\bar{y} := \lim_{t \rightarrow \infty} y(t) = G\bar{u}.$$

On the other hand, $(zs, y) \in \mathcal{B} = \mathcal{B}_{i/s/o}(A, b, C, d)$ implies that

$$\bar{y} := (C(I - A)^{-1}b + d)\bar{z}.$$

These relations give us the system of linear equations for \bar{u}

$$(C(I - A)^{-1}b + d)\bar{z} = G\bar{u}. \quad (**)$$

By Assumption 7.4, \bar{u} is uniquely determined from (**). \square

The significance of Proposition 7.13 is that Problem 7.3 can be solved equivalently as an autonomous system identification problem with a fixed pole at 0 in the continuous-time case or at 1 in the discrete-time case. The following proposition shows how a preprocessing step makes possible standard methods for autonomous system identification (or equivalently sum-of-damped exponential modeling) to be used for identification of a system with a fixed pole at 0 or 1.

Proposition 7.14

$$y \in \mathcal{B}_{i/s/o} \left(\begin{bmatrix} A & b \\ 0 & 0 \end{bmatrix}, [C \ d] \right) \iff \Delta y := \frac{d}{dt}y \in \Delta \mathcal{B} := \mathcal{B}_{i/s/o}(A, C)$$

in the continuous-time case

$$y \in \mathcal{B}_{i/s/o} \left(\begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix}, [C \ d] \right) \iff \Delta y := (1 - \sigma^{-1})y \in \Delta \mathcal{B} := \mathcal{B}_{i/s/o}(A, C)$$

in the discrete-time case

Proof Let p be the characteristic polynomial of the matrix A .

$$y \in \mathcal{B}_{i/s/o}(A_e, C_e) := \mathcal{B}_{i/s/o} \left(\begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix}, [C \ d] \right) \iff p(\sigma^{-1})(1 - \sigma^{-1})y = 0.$$

On the other hand, we have

$$\Delta y := (1 - \sigma^{-1})y \in \Delta \mathcal{B} := \mathcal{B}_{i/s/o}(A, C) \iff p(\sigma^{-1})(1 - \sigma^{-1})y = 0.$$

The initial conditions $(x_{\text{ini}}, \bar{u})$ of $\mathcal{B}_{i/s/o}(A_e, C_e)$ and Δx_{ini} of $\mathcal{B}_{i/s/o}(A, C)$ are related by

$$(I - A)x_{\text{ini}} = \Delta x_{\text{ini}}. \quad \square$$

Once the model parameters A and C are determined via autonomous system identification from Δy , the parameter of interest \bar{u} can be computed from the equation

$$y = \bar{y} + y_{\text{aut}}, \quad \text{where } \bar{y} = G\bar{u} \text{ and } y_{\text{aut}} \in \mathcal{B}_{i/s/o}(A, C) = \Delta \mathcal{B}. \quad (\text{AUT})$$

Using the fact that the columns of the extended observability matrix $\mathcal{O}_T(A, C)$ form a basis for $\Delta\mathcal{B}|_{[1, T]}$, we obtain the following system of linear equations for the estimation of \bar{u} :

$$[\mathbf{1}_T \otimes G \quad \mathcal{O}_T(A, C)] \begin{bmatrix} \bar{u} \\ x_{\text{ini}} \end{bmatrix} = \text{col}(y(t_s), \dots, y(Tt_s)). \quad (\text{SYS AUT})$$

Propositions 7.13 and 7.14, together with (SYS AUT), lead to the following algorithm for solving Problem 7.3.

209a *(Algorithm for sensor speedup based on reduction to autonomous system identification 209a) ≡*
 function [uh, sysh] = `stepid_as`(y, g, n)
(preprocessing by finite difference filter $\Delta y := (1 - \sigma^{-1})y$ 209b)
(autonomous system identification: $\Delta y \mapsto \Delta\mathcal{B}$ 209c)
(computation of \bar{u} by solving (SYS AUT) 209d)

Defines:

`stepid_as`, never used.

where

209b *(preprocessing by finite difference filter $\Delta y := (1 - \sigma^{-1})y$ 209b) ≡* (209a 210)
`dy = diff(y);`

and, using the function `ident_aut`

209c *(autonomous system identification: $\Delta y \mapsto \Delta\mathcal{B}$ 209c) ≡* (209a)
`sysh = ident_aut(dy, n);`

Uses `ident_aut` 112a.

Notes 7.11 and 7.12 apply for `stepid_as` as well. Alternatives to the prediction error method in the second step are methods for sum-of-damped exponential modeling (e.g., the Prony, Yule-Walker, or forward-backward linear prediction methods) and approximate system realization methods (e.g., Kung's method, implemented in the function `h2ss`). Theoretical results and numerical studies justify different methods as being effective for different noise assumptions. This allows us to pick the “right” identification method, to be used in the algorithm for solving Problem 7.3 under additional assumptions or prior knowledge about the measurement noise.

Finally, the third step of `stepid_as` is implemented as follows.

209d *(computation of \bar{u} by solving (SYS AUT) 209d) ≡* (209a)
`T = size(y, 1); [p, m] = size(g); yt = y'; O = sysh.c;`
`for t = 2:T`
`O = [O; O(end - p + 1:end, :) * sysh.a];`
`end`
`xe = [kron(ones(T, 1), g) O] \ yt(:); uh = xe(1:m);`

Data-Driven Solution

A signal w is persistently exciting of order 1 if the Hankel matrix $\mathcal{H}_1(w)$ has full row rank. By Lemma 4.11, a persistently exciting signals of order 1 cannot be fitted

exactly by a system in the model class $\mathcal{L}_{0,1}$. Persistency of excitation of the input is a necessary identifiability condition in exact system identification.

Assuming that Δy is persistently exciting of order n ,

$$\mathcal{B}_{\text{mpum}}(\Delta y) = \Delta \mathcal{B}.$$

Since

$$\mathcal{B}_{\text{mpum}}(\Delta y) = \text{span}(\{\sigma^\tau \Delta y \mid \tau \in \mathbb{R}\}),$$

we have

$$\mathcal{B}_{\text{mpum}}(\Delta y)|_{[1, T-n]} = \text{span}(\mathcal{H}_{T-n}(\Delta y)).$$

Then, from (AUT), we obtain the system of linear equations for \bar{u}

$$\underbrace{[\mathbf{1}_{T-n} \otimes G \quad \mathcal{H}_{T-n}(\Delta y)]}_H \begin{bmatrix} \bar{u} \\ \ell \end{bmatrix} = \text{col}(y((n+1)t_s), \dots, y(Tt_s)), \quad (\text{SYS DD})$$

which depends only on the given output data y and gain matrix G . The resulting model-free algorithm is

210 *(Data-driven algorithm for sensor speedup 210)* \equiv
 function uh = `stepid_dd`(y, g, n, ff)
 if nargin == 3, ff = 1; end
(preprocessing by finite difference filter $\Delta y := (1 - \sigma^{-1})y$ 209b)
(computation of \bar{u} by solving (SYS DD) 211)

Defines:

`stepid_dd`, used in chunks 213 and 221.

As proved next, `stepid_dd` computes the correct parameter value \bar{u} under less restrictive condition than identifiability of $\Delta \mathcal{B}$ from Δy , i.e., persistency of excitation of Δy of order n is not required.

Proposition 7.15 *Let*

$$(u, y) := \left(\underbrace{(\bar{u}, \dots, \bar{u})}_{T \text{ times}}, (y(1), \dots, y(T)) \right) \in \mathcal{B}|_{[1, T]}, \quad (*)$$

for some $\bar{u} \in \mathbb{R}^m$. Then, if the number of samples T is larger than or equal to $2n + m$, where n is the order of the data generating system \mathcal{B} , and Assumption 7.4 holds, the estimate computed by `stepid_dd` equals the true input value \bar{u} .

Proof The derivation of (SYS DD) and the exact data assumption (*) imply that there exists $\bar{\ell} \in \mathbb{R}^n$, such that $(\bar{u}, \bar{\ell})$ is a solution of (SYS DD). Our goal is to show that all solutions of (SYS DD) are of this form.

By Assumption 7.4, \mathcal{B} is a stable system, so that $1 \notin \lambda(\mathcal{B})$. It follows that for any $\bar{y} \in \mathbb{R}^p$,

$$\underbrace{(\bar{y}, \dots, \bar{y})}_{T \text{ times}} \notin \mathcal{B}_{\text{aut}}|_{[1, T]} = \Delta \mathcal{B}|_{[1, T]}.$$

Therefore,

$$\text{span}(\mathbf{1}_T \otimes G) \cap \mathcal{B}_{\text{aut}}|_{[1,T]} = \{0\}.$$

By the assumption $T \geq 2n + m$, the matrix H in (SYS DD) has at least as many rows as columns. Then using the full row rank property of G (Assumption 7.4), it follows that a solution \bar{u} of (SYS DD) is unique. \square

Note 7.16 (Nonuniqueness of a solution ℓ of (SYS DD)) Under the assumptions of Proposition 7.15, the first m elements of a solution of (SYS DD) are unique. The solution for ℓ , however, may be nonunique. This happens if and only if the order of $\mathcal{B}_{\text{mpum}}(\Delta y)$ is less than n , i.e., $\mathcal{B}_{\text{mpum}}(\Delta y) \subset \Delta \mathcal{B}$. A condition on the data for $\mathcal{B}_{\text{mpum}}(\Delta y) \subset \Delta \mathcal{B}$ is that Δy is persistently exciting of order less than n . Indeed,

$$\dim(\mathcal{B}_{\text{mpum}}(\Delta y)) = \text{rank}(\mathcal{H}_{T-n}(\Delta y)).$$

It follows from Note 7.16 that if the order n is not specified a priori but is estimated from the data by, e.g., computing the numerical rank of $\mathcal{H}_{T-n}(\Delta y)$, the system of equations (SYS DD) has a unique solution.

Note 7.17 (Relaxed assumption) The methods `stepid_si` and `stepid_as`, based on a model computed from the data y using a system identification method, require identifiability conditions. By Proposition 7.15, however, `stepid_dd` does not require identifiability. The order specification in `stepid_dd` can be relaxed to an upper bound of the true system order, in which case any solution of (SYS DD) recovers \bar{u} from its unique first m elements.

The data-driven algorithm can be implemented recursively and generalized to estimation under different noise assumptions. The following code chunk uses a standard recursive least squares algorithm, implemented in the function `rls`, for approximate solution of (SYS DD).

```

211 <computation of  $\bar{u}$  by solving (SYS DD) 211>≡ (210)
    T = size(y, 1); [p, m] = size(g);
    Tt = T - n; yt = y((n + 1):T, :)' ;
    if n == 0, dy = [0; dy]; end
    x = rls([kron(ones(Tt, 1), g), blkhank(dy, Tt)],
           yt(:), ff);
    uh = x(1:m, p:p:end)'; uh = [NaN * ones(n, m); uh];

```

Uses `blkhank` 25b and `rls` 223b.

The first estimated parameter `uh(1, :)` is computed at time

$$T_{\min} = \left\lceil \frac{n + m}{p} \right\rceil + n.$$

In `rls`, the first $\lceil (n + m)/p \rceil - 1$ samples are used for initialization, so that in order to match the index of `uh` with the actual discrete-time when the estimate is computed, `uh` is padded with n additional rows.

Using the recursive least squares algorithm `rls`, the computational cost of `stepid_dd` is $O((m+n)^2p)$. Therefore, its computational complexity compares favorably with the one of `stepid_kf` with precomputed filter gain (see Sect. 7.2). The fact that Problem 7.3 can be solved with the same order of computations with and without knowledge of the process dynamics is surprising and remarkable. We consider this fact as our main result.

In the next section, the performance of `stepid_dd` and `stepid_kf` is compared on test examples, where the data are generated according to the output error noise model (OE).

Note 7.18 (Mixed least squares Hankel structured total least squares approximation method) In case of noisy data (OE), the ordinary least squares approximate solution of (SYS DD) is not maximum likelihood. The reason for this is that the matrix in the left-hand-side of (SYS DD) depends on y , which is perturbed by the noise. A statistically better approach is to be used the mixed least squares total least squares approximation method that accounts for the fact that the block $\mathbf{1}_T \otimes G$ in H is exact but the block $\mathcal{H}_{T-n}(\Delta y)$ of H as well as the right-hand side of (SYS DD) are noisy. The least squares total least squares method, however, requires the more expensive singular value decomposition of the matrix $[H \ y]$ and is harder to implement as a recursive on-line method. In addition, although the mixed least squares total least squares approximation method improves on the standard least squares method it is also not maximum likelihood either, because it does not take into account the Hankel structure of the perturbations. A maximum-likelihood data-driven method requires an algorithm for structured total least squares.

7.4 Examples and Real-Life Testing

Simulation Setup

In the simulations, we use the output error model (OE). The exact data y_0 in the estimation problems is a uniformly sampled output trajectory $y_0 = (y_0(t_s), \dots, y_0(Tt_s))$ of a continuous-time system $\mathcal{B} = \mathcal{B}_{i/s/o}(A, B, C, D)$, obtained with input u_0 and initial condition x_{ini} .

```
212a <Test sensor speedup 212a>≡ 212b>
    <initialize the random number generator 89e>
    sys = c2d(ss(A, B, C, D), ts); G = dcgain(sys);
    [p, m] = size(G); n = size(sys, 'order');
    y0 = lsim(sys, u0, [], xini);
```

Defines:

```
test_sensor, used in chunks 215–20.
```

According to (OE), the exact trajectory y_0 is perturbed with additive noise \tilde{y} , which is modeled as a zero mean, white, stationary, Gaussian process with standard deviation Σ .

```
212b <Test sensor speedup 212a>+≡ <212a 213a>
    y = y0 + randn(T, p) * s;
```

Table 7.1 Legend for the line styles in figures showing simulation results

Line style	Corresponds to
dashed	true parameter value \bar{u}
solid	true output trajectory y_0
dotted	naive estimate $\hat{u} = G^+y$
dashed	<code>stepid_kf</code>
dashed-dotted	<code>stepid_dd</code>

After the data y is simulated, the estimation methods `stepid_kf` and `stepid_dd` are applied

```
213a <Test sensor speedup 212a> += <212b 213b>
    uh_dd = stepid_dd(y, G, n, ff);
    uh_kf = stepid_kf(y, sys.a, sys.b, sys.c, sys.d, ...
                    s^2 * eye(size(D, 1)));
```

Uses `stepid_dd` 210 and `stepid_kf` 203a.

and the corresponding estimates are plotted as functions of time, together with the “naive estimator”

$$\hat{u} := G^+y, \quad \text{where } G^+ = (G^T G)^{-1} G^T.$$

```
213b <Test sensor speedup 212a> += <213a 213c>
    figure(2 * (exm_n - 1) + 1), hold on,
    if n > 0, Tmin = 2 * n + 1; else Tmin = 2; end, t = Tmin:T;
    plot(t, y0(t, :) / G', 'k-'), plot(t, y(t, :) / G', 'k:'),
    plot(t, u0(t, :), 'k-'), plot(t, uh_dd(t, :), '-b'),
    plot(t, uh_kf(t, :), '-.r'), ax = axis;
    axis([Tmin T ax(3:4)]),
    print_fig(['example-' int2str(exm_n)])
```

Uses `print_fig` 25a.

The plotted results are in the interval $[2n+m, T]$, because $2n+m$ is the minimum number of samples needed for estimation of \bar{u} . The convention used to denote the different estimates by different line styles is summarized in Table 7.1.

Let $\hat{u}^{(i)}(t)$ be the estimate of \bar{u} , using the data $(y(1), \dots, y(t))$ in an i th Monte Carlo repetition of the estimation experiment. In addition to the results for a specific noise realization, the average estimation errors of the compared methods

$$e = \frac{1}{N} \sum_{i=1}^N \|\bar{u} - \hat{u}^{(i)}\|_1, \quad \text{where } \|x\|_1 := \sum_{i=1}^n |x_i|$$

are computed and plotted over n independent noise realizations.

```
213c <Test sensor speedup 212a> += <213b>
    N = 100; clear e_dd e_kf e_nv
    for i = 1:N
        y = y0 + randn(T, p) * s;
        e_nv(:, i) = sum(abs(u0 - y / G'), 2);
```

```

uh_dd = stepid_dd(y, G, n, ff);
e_dd(:, i) = sum(abs(u0 - uh_dd), 2);
uh_kf = stepid_kf(y, sys.a, sys.b, sys.c, sys.d, ...
                 s^2 * eye(size(D, 1)));
e_kf(:, i) = sum(abs(u0 - uh_kf), 2);
end
figure(2 * (exm_n - 1) + 2), hold on
plot(t, mean(e_dd(t, :), 2), '-b'),
plot(t, mean(e_kf(t, :), 2), '-.r'),
plot(t, mean(e_nv(t, :), 2), ':k'),
ax = axis; axis([Tmin T 0 ax(4)]),
print_fig(['example-error-' int2str(exm_n)])
exm_n = exm_n + 1;

```

Uses `print_fig` 25a, `stepid_dd` 210, and `stepid_kf` 203a.

The script file `examples_sensor_speedup.m`, listed in the rest of this section, reproduces the simulation results. The variable `exm_n` is the currently executed example.

214a *(Sensor speedup examples 214a)* ≡ 215a>
`clear all, close all, exm_n = 1;`
 Defines:
`examples_sensor_speedup`, never used.

Dynamic Cooling

The first example is the temperature measurement problem from the introduction. The heat transfer between the thermometer and its environment is governed by Newton's law of cooling, i.e., the changes in the thermometer's temperature y is proportional to the difference between the thermometer's temperature and the environment's temperature \bar{u} . We assume that the heat capacity of the environment is much larger than the heat capacity of the thermometer, so that the heat transfer between the thermometer and the environment does not change the environment's temperature. Under this assumption, the dynamics of the measurement process is given by the differential equation

$$\frac{d}{dt}y = a(\bar{u}s - y),$$

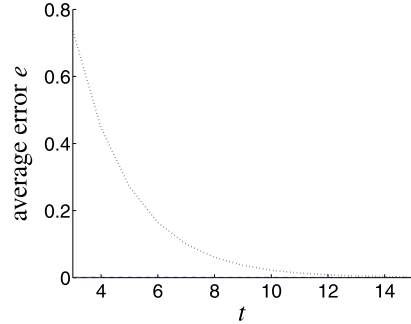
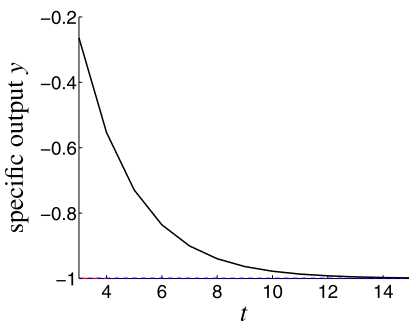
where a is a positive constant that depends on the thermometer and the environment. The differential equation defines a first order linear time-invariant dynamical system $\mathcal{B} = \mathcal{B}_{i/s/o}(-a, a, 1, 0)$ with input $u = \bar{u}s$.

214b *(cooling process 214b)* ≡ (215 220)
`A = -a; B = a; C = 1; D = 0;`

The dc gain of the system is equal to 1, so that it can be assumed known, independent of the process's parameter a . This matches the setup of Problem 7.3, where the dc gain is assumed a priori known but the process dynamics is not.

Simulation 1: Exact data

215a *(Sensor speedup examples 214a)*+≡ <214a 215b>
`a = 0.5; (cooling process 214b) T = 15; ts = 1; s = 0.0;`
`xini = 1; ff = 1; u0 = ones(T, 1) * (-1); test_sensor`
 Uses test_sensor 212a.

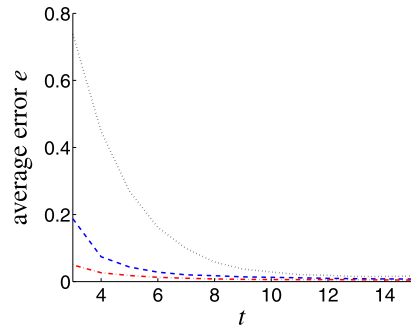
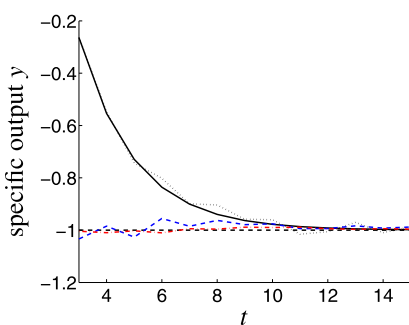


The average error for both `stepid_kf` and `stepid_dd` is zero (up to errors incurred by the numerical computation). The purpose of showing simulation results of an experiment with exact data is verification of the theoretical results stating that the methods solve Problem 7.3.

In the case of output noise (OE), `stepid_kf` is statistically optimal estimator, while `stepid_dd`, implemented with the (recursive) least squares approximation method, is not statistically optimal (see Note 7.18). In the next simulation example we show how far from optimal is `stepid_dd` in the dynamic colling example with the simulation parameters given below.

Simulation 2: Noisy data

215b *(Sensor speedup examples 214a)*+≡ <215a 216b>
`a = 0.5; (cooling process 214b) T = 15; ts = 1; s = 0.02;`
`xini = 1; ff = 1; u0 = ones(T, 1) * (-1); test_sensor`
 Uses test_sensor 212a.



Temperature-Pressure Measurement

Consider ideal gas in a closed container with a fixed volume. We measure the temperature (as described in the previous section) by a slow but accurate thermometer, and the pressure by fast but inaccurate pressure sensor. By Gay-Lussac’s law, the temperature (measured in Kelvin) is proportional to the pressure, so by proper calibration, we can measure the temperature also with the pressure sensor. Since the pressure sensor is much faster than the thermometer, we model it as a static system. The measurement process in this example is a multivariable (one input, two outputs) system $\mathcal{B}_{i/s/o}(A, B, C, D)$, where

$$A = -a, \quad B = a, \quad C = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \text{and} \quad D = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

216a *(temperature-pressure process 216a)* ≡ (216b)

$$A = -a; \quad B = a; \quad C = [1; 0]; \quad D = [0; 1];$$

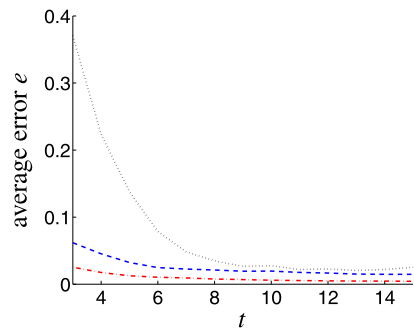
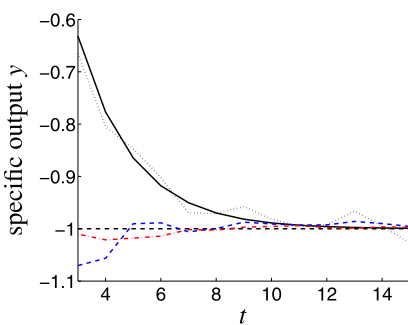
Problem 7.3 in this example can be viewed as a problem of “blending” the measurements of two sensors in such a way that a faster and more accurate measurement device is obtained. The algorithms developed can be applied directly to process the vector values data sequence y , thus solving the “data fusion” problem.

Simulation 3: Using two sensors

216b *(Sensor speedup examples 214a)* + ≡ <215b 217a>

```
a = 0.5; T = 15; ts = 1; (temperature-pressure process 216a) ff = 1;
s = diag([0.02, 0.05]); xini = 1; u0 = ones(T, 1) * (-1);
test_sensor
```

Uses test_sensor 212a.

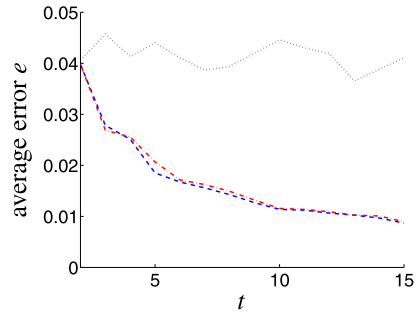
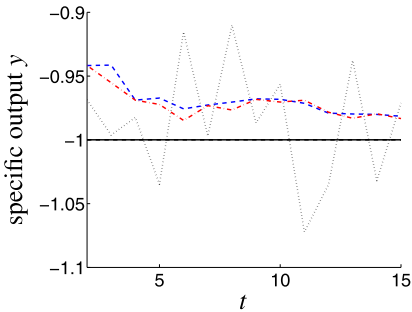


Comparing the results of Simulation 3 with the ones of Simulation 2 (experiment with the temperature sensor only), we see about two fold initial improvement of the average errors of all methods, however, in a long run the naive and `stepid_dd` methods show smaller improvement. The result is consistent with the intuition that the benefit in having fast but inaccurate second sensor is to be expected mostly in

the beginning when the estimate of the slow but accurate sensor is still far off the true value. This intuitive explanation is confirmed by the results of an experiment in which only the pressure sensor is used.

Simulation 4: Pressure sensor only

```
217a (Sensor speedup examples 214a)+≡ <216b 218b>
      A = []; B = []; C = []; D = 1; T = 15; ts = 1; s = 0.05;
      xini = []; ff = 1; u0 = ones(T, 1) * (-1); test_sensor
Uses test_sensor 212a.
```



Dynamic Weighing

The third example is the dynamic weighing problem. An object with mass M is placed on a weighing platform with mass m that is modeled as a mass, spring, damper system, see Fig. 7.1.

At the time of placing the object, the platform is in a specified (in general nonzero) initial condition. The object placement has the effect of a step input as well as a step change of the total mass of the system—platform and object. The goal is to measure the object’s mass while the platform is still in vibration.

We choose the origin of the coordinate system at the equilibrium position of the platform when there is no object placed on it with positive direction being upwards, perpendicular to the ground. With $y(t)$ being the platform’s position at time t , the measurement process \mathcal{B} is described by the differential equation

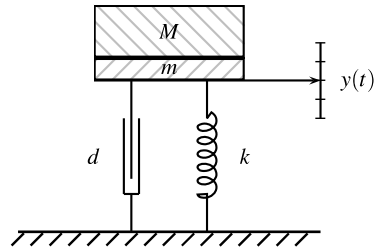
$$(M + m) \frac{d^2}{dt^2} y = -ky - d \frac{d}{dt} y - Mg,$$

where g is the gravitational constant

```
217b (define the gravitational constant 217b)≡ (218a)
      g = 9.81;
```

k is the elasticity constant of the spring, and d is the damping constant of the damper. Defining the state vector $x = (y, \frac{d}{dt}y)$ and taking as an input $u_0 = Ms$, we obtain

Fig. 7.1 Dynamic weighing setup



the following state space representation $\mathcal{B}_{\text{vis/o}}(A, b, c, 0)$ of \mathcal{B} , where

$$A = \begin{bmatrix} 0 & 1 \\ \frac{k}{M+m} & \frac{d}{M+m} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ -\frac{g}{M+m} \end{bmatrix}, \quad C = [1 \ 0], \quad \text{and} \quad D = 0.$$

218a *(weighting process 218a)* \equiv (218 219)

(define the gravitational constant 217b)

$$A = [0, 1; -k / (m + M), -d / (m + M)];$$

$$B = [0; -g / (m + M)]; \quad C = [1, 0]; \quad D = 0;$$

$$u0 = \text{ones}(T, 1) * M;$$

Note that the process parameters and, in particular, its poles depend on the unknown parameter M , however, the dc gain

$$\text{dcgain}(\mathcal{B}) = -\frac{g}{k}$$

is independent of M . Therefore, the setup in Problem 7.3—prior knowledge of the dc gain but unknown process dynamics—matches the actual setup of the dynamic weighing problem.

Next, we test the methods `stepid_kf` and `stepid_dd` on dynamic weighing problems with different object’s masses.

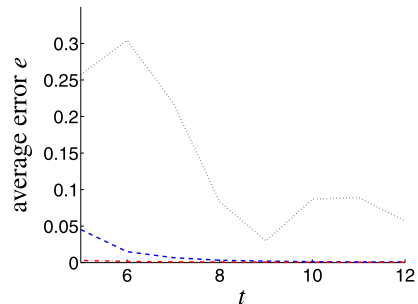
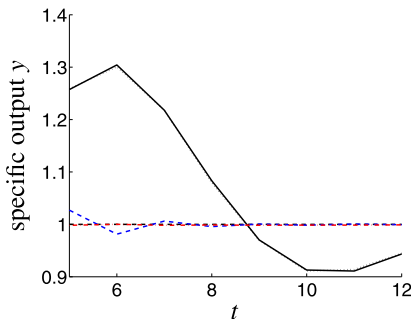
Simulation 5: $M = 1$

218b *(Sensor speedup examples 214a)* \equiv \langle 217a 219a \rangle

$$m = 1; \quad M = 1; \quad k = 1; \quad d = 1; \quad T = 12; \quad \langle \text{weighting process 218a} \rangle$$

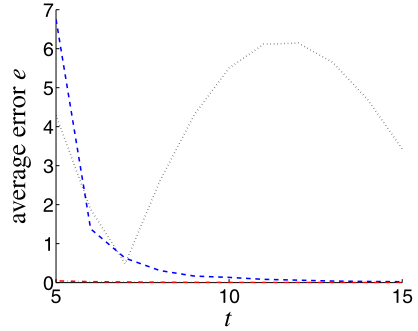
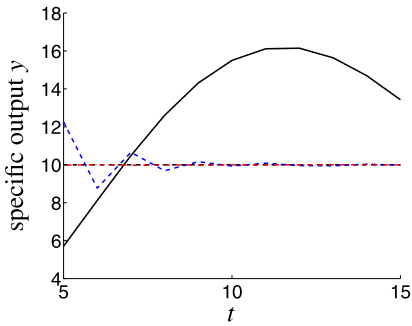
$$ts = 1; \quad s = 0.02; \quad ff = 1; \quad \text{xini} = 0.1 * [1; 1]; \quad \text{test_sensor}$$

Uses `test_sensor 212a`.



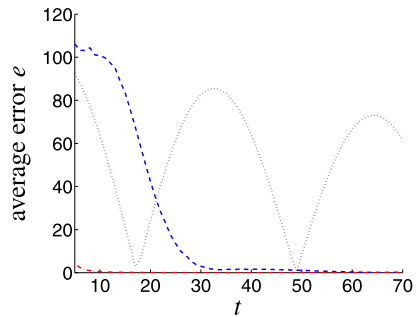
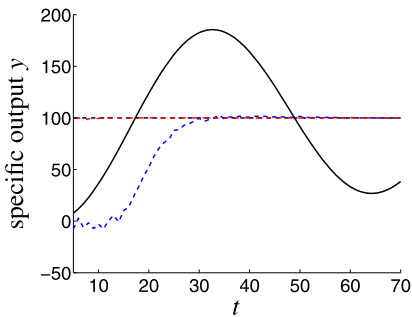
Simulation 6: $M = 10$

219a \langle Sensor speedup examples 214a $\rangle + \equiv$ \langle 218b 219b \rangle
 $m = 1; M = 10; k = 1; d = 1; T = 15;$ \langle weighting process 218a \rangle
 $ts = 1; s = 0.05; ff = 1; xini = 0.1 * [1; 1];$ test_sensor
 Uses test_sensor 212a.



Simulation 7: $M = 100$

219b \langle Sensor speedup examples 214a $\rangle + \equiv$ \langle 219a 220 \rangle
 $m = 1; M = 100; k = 1; d = 1; T = 70;$ \langle weighting process 218a \rangle
 $ts = 1; s = 0.5; ff = 1; xini = 0.1 * [1; 1];$ test_sensor
 Uses test_sensor 212a.



Time-Varying Parameter

Finally, we show an example with a time-varying measured parameter \bar{u} . The measurement setup is the cooling process with the parameter a changing from 1 to 2 at time $t = 25$. The performance of the estimates in the interval $[1, 25]$ (estimation of the initial value 1) was already observed in Simulations 1 and 2 for the exact and noisy case, respectively. The performance of the estimates in the interval $[26, 50]$

(estimation of the new value $\bar{u} = 2$) is new characteristic for the adaptive properties of the algorithms.

Simulation 8: Parameter jump

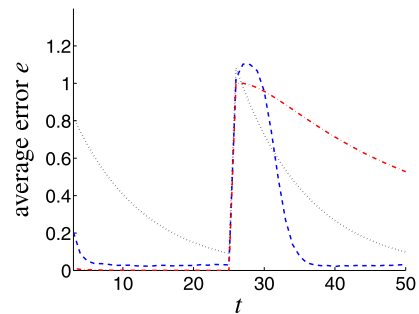
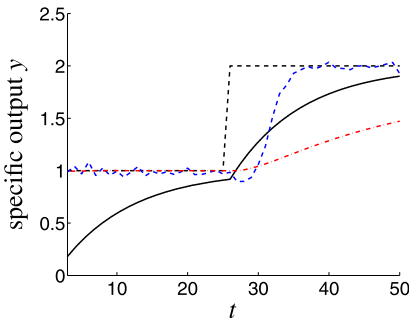
220

(Sensor speedup examples 214a)+≡

<219b

```
a = 0.1; (cooling process 214b) T = 50; ts = 1; s = 0.001; ff = 0.5;
u0 = [ones(T / 2, 1); 2 * ones(T / 2, 1)]; xini = 0;
test_sensor
```

Uses test_sensor 212a.



The result shows that by choosing “properly” the forgetting factor f ($f = 0.5$ in Simulation 8), `stepid_dd` tracks the changing parameter value. In contrast, `stepid_kf` which assumes constant parameter value is much slower in correcting the old parameter estimate $\hat{u} \approx 1$.

Currently the choice of a the forgetting factor f is based on the heuristic rule that “slowly” varying parameter requires value of f “close” to 1 and “quickly” changing parameter requires value of f close to 0. A suitable value is fine tuned by trail and error.

Another possibility for making `stepid_dd` adaptive is to include windowing of the data y by down-dating of the recursive least squares solution. In this case the tunable parameter (similar to the forgetting factor) is the window length. Again there is an obvious heuristic for choosing the window length but no systematic procedure. Windowing and exponential weighting can be combined, resulting in a method with two tunable parameter.

Real-Life Testing

The data-driven algorithms for input estimation are tested also on real-life data of the “dynamic cooling” application. The experimental setup for the data collection is based on the Lego NXT mindstorms digital signal processor and digital temperature sensor, see Fig. 7.2.



Fig. 7.2 Experimental setup: Lego NXT mindstorms brick (left) and temperature sensor (right)

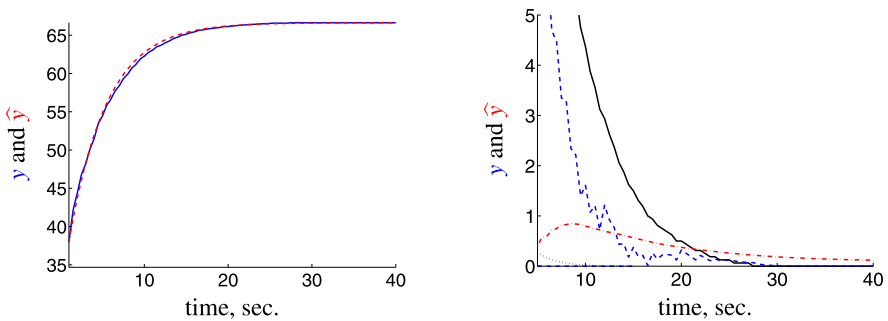


Fig. 7.3 Left: model fit to the data (solid blue—measured data, dashed red—model fit). Right: parameter estimates (solid black—naive estimator, dashed blue—`stepid_dd`, dashed dotted red—`stepid_kf`)

As a true measured value \bar{u} , we take the (approximate) steady-state temperature $\bar{y} := y(40)$. In order to apply the model based `stepid_kf` method, the measured data is fitted by a first order model and an output error standard deviation $\sigma = 0.01$ is hypothesised. The fit of the model to the data is shown in Fig. 7.3, left. The data-driven algorithm `stepid_dd` is applied with forgetting factor $f = 0.75$. The recursive estimation results are shown in Fig. 7.3, right. Although initially the model based estimation method is more accurate, after 15 sec. the data-driven method outperforms it.

First, we load the data and apply the data-driven algorithm

```

221 (Test sensor speedup methods on measured data 221)≡ 222a>
    load('y-tea.mat'); ub = y(end);
    T = length(y); G = 1; n = 1; ff = .75; s = 0.01;
    uh_dd = stepid_dd(y, G, n, ff);
Defines:
    test_lego, never used.
Uses stepid_dd 210.
    
```

Then the data are modeled by removing the steady-state value and fitting an exponential to the residual. The obtained model is a first order dynamical system, which is used for the model based input estimation method.

222a `(Test sensor speedup methods on measured data 221)+≡` <221 222b>

```
yc = y - ub; f = yc(1:end - 1) \ yc(2:end);
yh = f .^ (0:(T - 1))' * yc(1) + ub;
sys = ss(f, 1 - f, 1, 0, t(2) - t(1));
uh_kf = stepid_kf(y, sys.a, sys.b, sys.c, sys.d, s^2);
```

Uses `stepid_kf` 203a.

Finally, the estimation results for the naive, model-based, and data-driven methods are plot for comparison.

222b `(Test sensor speedup methods on measured data 221)+≡` <222a

```
figure(1), hold on, plot(t, y, 'b-', t, yh, 'r-')
axis([1 t(end) y(1) y(end)]), print_fig('lego-test-fit')
figure(2), hold on,
plot(t, abs(ub - y / G'), 'k-'),
plot(t, abs(ub - uh_dd), '-b'),
plot(t, abs(ub - uh_kf), '-.r'),
axis([t(10) t(end) 0 5]), print_fig('lego-test-est')
```

Uses `print_fig` 25a.

7.5 Auxiliary Functions

Time-Varying Kalman Filter

The function `tvkf_oe` implements the time-varying Kalman filter for the discrete-time autonomous stochastic system, described by the state space representation

$$\sigma x = Ax, \quad y = Cx + v,$$

where v is a stationary, zero mean, white, Gaussian noise with covariance V .

222c `(Time-varying Kalman filter for autonomous output error model 222c)≡` 223a>

```
function x = tvkf_oe(y, a, c, v, x0, p)
T = size(y, 1); Y = y';
```

Defines:

`tvkf_oe`, used in chunk 203c.

The optional parameters x_0 and p specify prior knowledge about the mean value of the initial state $x(0)$ and its error covariance matrix P . The default value is highly uncertain zero mean random vector.

The Kalman filter algorithm (see, Kailath et al. 2000, Theorem 9.2.1) is

$$K := APC^T(V + CPC^T)^{-1}$$

$$\begin{aligned}\sigma \widehat{x} &= A \widehat{x} + K(y - C \widehat{x}), & x(0) &= x_{\text{ini}} \\ \sigma P &= A P A^{\top} - K(V + C P C^{\top}) K^{\top}, & P(0) &= P_{\text{ini}}.\end{aligned}$$

223a *(Time-varying Kalman filter for autonomous output error model 222c)* ≡ ◀222c

```
x = zeros(size(a, 1), T); x(:, 1) = x0;
for t = 1:(T-1)
    k = (a * p * c') / (v + c * p * c');
    x(:, t + 1) = a * x(:, t) + k * (y(:, t) - c * x(:, t));
    p = a * p * a' - k * (v + c * p * c') * k';
end
```

Recursive Least Squares

The function `rls` implements an (exponentially weighted) recursive least squares algorithm (see, Kailath et al. 2000, Lemma 2.6.1 and Problem 2.6) for a system of linear equations $Ax \approx b$, where $A \in \mathbb{R}^{m \times n}$, i.e., `rls` computes the (exponentially weighted) least squares approximate solutions $x(n), \dots, x(m)$ of the sequence of problems $A_{1:i}, x^{(i)} \approx b_{1:i}$, for $i = n, \dots, m$.

223b *(Recursive least squares 223b)* ≡ 223c>

```
function x = rls(a, b, f)
[m, n] = size(a); finv = 1 / f;
Defines:
    rls, used in chunk 211.
```

The input parameter `f`, $f \in (0, 1]$, is called forgetting factor and specifies an exponential weighting $f^i r_i$ of the residual $r := Ax - b$ in the least squares approximation criterion.

Let $a(i)$ be the i th row of A and let $b(i) := b_i$. The (exponentially weighted) recursive least squares algorithm is

$$\begin{aligned}K &:= P a^{\top} \left(1 + \frac{1}{f} a P a^{\top} \right)^{-1} \\ \sigma \widehat{x} &= x + K(b - ax), & x(0) &= x_{\text{ini}} \\ \sigma P &= \frac{1}{f} (P - P a^{\top} (1 + a P a^{\top})^{-1} a P), & P(0) &= P_{\text{ini}}.\end{aligned}$$

223c *(Recursive least squares 223b)* ≡ ◀223b

```
(initialization 224)
for i = (n + 1):m
    ai = a(i, :);
    k = finv * p * ai' / (1 + finv * ai * p * ai');
    x(:, i) = x(:, i - 1) + k * (b(i) - ai * x(:, i - 1));
    p = 1 / f * (p - k * ai * p);
end
```

The algorithm is initialized with the solution of the system formed by the first n equations

$$x(0) := A_{1:n,:}^{-1} b_{1:n}, \quad P(0) := (A_{1:n,:}^\top A_{1:n,:})^{-1}.$$

224 (initialization 224) \equiv (223c)

```
ai = a(1:n, 1:n); x = zeros(n, m);
x(:, n) = ai \ b(1:n); p = inv(ai' * ai);
```

7.6 Notes

In metrology, the problem considered in this chapter is called *dynamic measurement*. The methods proposed in the literature, see the survey (Eichstädt et al. 2010) and the references there in, pose and solve the problem as a compensator design problem, i.e., the input estimation problem is solved by:

1. designing off-line a dynamical system, called compensator, such that the series connection of the measurement process with the compensator is an identity, and
2. processing on-line the measurements by the compensator.

Most authors aim at a linear time-invariant compensator and assume that a model of the measurement process is a priori given. This is done presumably due to the simplification that the linear time-invariant assumption of the compensator brings in the design stage and the reduced computational cost in the on-line implementation compared to alternative nonlinear compensators. In the case of known model, step 1 of the dynamic measurement problem reduces to the classical problem of designing an inverse system (Sain and Massey 1969). In the presence of noise, however, compensators that take into account the noise are needed. To the best of our knowledge, there is no theoretically sound solution of the dynamic measurement problem in the noisy case available in the literature although, as shown in Sect. 7.2, the problem reduces to a state estimation problem for a suitably defined autonomous linear time-invariant system. As a consequence, under standard assumptions about the measurement and process noises, the maximum-likelihood solution is given by the Kalman filter, designed for the autonomous system.

More flexible is the approach of Shu (1993), where the compensator is tuned on-line by a parameter estimation algorithm. In this case, the compensator is a nonlinear system and an a priori given model of the process is no longer required. The solutions proposed in Shu (1993) and Jafaripanah et al. (2005), however, are tailored to the dynamic weighing problem, where the measurement process dynamics is a specific second order system.

Compared with the existing results on the dynamic measurement problem in the literature the methods described in the chapter have the following advantages.

- The considered measurement process dynamics is a general linear multivariable system. This is a significant generalization of the previously considered dynamic measurement problems (single input single output, first and second order systems).

- In the case of known measurement process dynamics, the problem is shown to be equivalent to a state estimation problem for an augmented system, which implies that the standard Kalman filter, designed for the augmented system is the optimal estimator in the case of Gaussian noise. Efficient filtering algorithms for systems with m constant inputs that avoid the increase in the state dimension from the original system's order n to $n + m$ are described in Willman (1969) and Friedland (1969).
- In the case of unknown measurement process dynamics, the problem is solved as an input estimation problem. The solution leads to recursive on-line algorithms that can be interpreted as nonlinear compensators, however, we do not a priori restrict the solution to a special type of compensator, such as linear time-invariant system tuned by an adaptive filter of a specific type.
- The data-driven solution derived uses a recursive least squares algorithm, so that it can be viewed as a nonlinear compensator, similar to the one of Shu (1993) derived for the case of a second order single input single output system. The data-driven solution proposed, however, applies to higher order multivariable systems. In addition, unlike the use of recursive least squares in Shu (1993) for model parameter estimation, the data-driven algorithm estimates directly the parameter of interest. This leads to significant computational savings. The on-line computational cost of the data-driven algorithm is comparable to the one of running a full order linear time-invariant compensator in the case of known process model.

The data-driven method for estimation of the input value is similar to the data-driven simulation and control methods of Markovsky and Rapisarda (2008) and Markovsky (2010). The key link between the set of system's trajectories and the image of the Hankel matrix requiring persistency of excitation of the input and controllability of the system is proven in Willems et al. (2005), see also Markovsky et al. (2006, Sect. 8.4).

References

- Eichstädt S, Elster C, Esward T, Hessling J (2010) Deconvolution filters for the analysis of dynamic measurement processes: a tutorial. *Metrologia* 47:522–533
- Friedland B (1969) Treatment of bias in recursive filtering. *IEEE Trans Autom Control* 14(4):359–367
- Jafaripanah M, Al-Hashimi B, White N (2005) Application of analog adaptive filters for dynamic sensor compensation. *IEEE Trans Instrum Meas* 54:245–251
- Kailath T, Sayed AH, Hassibi B (2000) *Linear estimation*. Prentice Hall, New York
- Luenberger DG (1979) *Introduction to dynamical systems: theory, models and applications*. Wiley, New York
- Markovsky I (2010) Closed-loop data-driven simulation. *Int J Control* 83:2134–2139
- Markovsky I, Rapisarda P (2008) Data-driven simulation and control. *Int J Control* 81(12):1946–1959
- Markovsky I, Willems JC, Van Huffel S, De Moor B (2006) *Exact and approximate modeling of linear systems: a behavioral approach*. SIAM, Philadelphia
- Sain M, Massey J (1969) Invertibility of linear time-invariant dynamical systems. *IEEE Trans Autom Control* 14:141–149

- Shu W (1993) Dynamic weighing under nonzero initial conditions. *IEEE Trans Instrum Meas* 42(4):806–811
- Stoica P, Selén Y (2004) Model-order selection: a review of information criterion rules. *IEEE Signal Process Mag* 21:36–47
- Willems JC, Rapisarda P, Markovsky I, Moor BD (2005) A note on persistency of excitation. *Control Lett* 54(4):325–329
- Willman W (1969) On the linear smoothing problem. *IEEE Trans Autom Control* 14(1):116–117

Appendix A

Approximate Solution of an Overdetermined System of Equations

Approximate solution of an overdetermined system of linear equations $AX \approx B$ is one of the main topics in (numerical) linear algebra and is covered in any linear algebra textbook, see, e.g., Strang (1976, Sect. 3.3), Meyer (2000, Sects. 4.6 and 5.14), and Trefethen and Bau (1997, Lecture 11). The classical approach is approximate solution in the least squares sense:

$$\text{minimize over } \widehat{B} \text{ and } X \quad \|B - \widehat{B}\|_F \quad \text{subject to } AX = \widehat{B}, \quad (\text{LS})$$

where the matrix B is modified as little as possible in the sense of minimizing the correction size $\|B - \widehat{B}\|_F$, so that the modified system of equations $AX = \widehat{B}$ is compatible. The classical least squares problem has an analytic solution: assuming that the matrix A is full column rank, the unique least squares approximate solution is

$$\widehat{X}_{\text{ls}} = (A^T A)^{-1} A^T B \quad \text{and} \quad \widehat{B}_{\text{ls}} = A(A^T A)^{-1} A^T B.$$

In the case when A is rank deficient, the solution is either nonunique or does not exist. Such least squares problems are solved numerically by regularization techniques, see, e.g., Björck (1996, Sect. 2.7).

There are many variations and generalizations of the least squares method for solving approximately an overdetermined system of equations. Well known ones are methods for recursive least squares approximation (Kailath et al. 2000, Sect. 2.6), regularized least squares (Hansen 1997), linear and quadratically constrained least squares problems (Golub and Van Loan 1996, Sect. 12.1).

Next, we list generalizations related to the class of the total least squares methods because of their close connection to corresponding low rank approximation problems. Total least squares methods are low rank approximation methods using an input/output representation of the rank constraint. In all these problems the basic idea is to modify the given data as little as possible, so that the modified data define a consistent system of equations. In the different methods, however, the correction is done and its size is measured in different ways. This results in different properties of the methods in a stochastic estimation setting and motivates the use of the methods in different practical setups.

- The *data least squares* (Degroot and Dowling 1991) method is the “reverse” of the least squares method in the sense that the matrix A is modified and the matrix B is not:

$$\text{minimize over } \widehat{A} \text{ and } X \quad \|A - \widehat{A}\|_F \quad \text{subject to} \quad \widehat{A}X = B. \quad (\text{DLS})$$

As in the least squares problem, the solution of the data least squares is computable in closed form.

- The classical *total least squares* (Golub and Reinsch 1970; Golub 1973; Golub and Van Loan 1980) method modifies symmetrically the matrices A and B :

$$\begin{aligned} &\text{minimize over } \widehat{A}, \widehat{B}, \text{ and } X \quad \|[A \ B] - [\widehat{A} \ \widehat{B}]\|_F \\ &\text{subject to} \quad \widehat{A}X = \widehat{B}. \end{aligned} \quad (\text{TLS})$$

Conditions for existence and uniqueness of a total least squares approximate solution are given in terms of the singular value decomposition of the augmented data matrix $[A \ B]$. In the generic case when a unique solution exists, that solution is given in terms of the right singular vectors of $[A \ B]$ corresponding to the smallest singular values. In this case, the optimal total least squares approximation $[\widehat{A} \ \widehat{B}]$ of the data matrix $[A \ B]$ coincides with the Frobenius norm optimal low rank approximation of $[A \ B]$, i.e., in the generic case, the model obtained by the total least squares method coincides with the model obtained by the unstructured low rank approximation in the Frobenius norm.

Theorem A.1 *Let \widehat{D}^* be a solution to the low rank approximation problem*

$$\text{minimize over } \widehat{D} \quad \|D - \widehat{D}\|_F \quad \text{subject to} \quad \text{rank}(\widehat{D}) \leq m$$

and let $\widehat{\mathcal{B}}^ = \text{image}(\widehat{D}^*)$ be the corresponding optimal linear static model. The parameter \widehat{X}^* of an input/output representation $\widehat{\mathcal{B}}^* = \mathcal{B}_{i/o}(\widehat{X}^*)$ of the optimal model is a solution to the total least squares problem (TLS) with data matrices*

$$\begin{matrix} m \\ q - m \end{matrix} \left\{ \begin{matrix} [A^\top] \\ [B^\top] \end{matrix} \right\} = D \in \mathbb{R}^{q \times N}.$$

A total least squares solution \widehat{X}^ exists if and only if an input/output representation $\mathcal{B}_{i/o}(\widehat{X}^*)$ of $\widehat{\mathcal{B}}^*$ exists and is unique if and only if an optimal model $\widehat{\mathcal{B}}^*$ is unique. In the case of existence and uniqueness of a total least squares solution*

$$\widehat{D}^* = [\widehat{A}^* \ \widehat{B}^*]^\top, \quad \text{where } \widehat{A}^* \widehat{X}^* = \widehat{B}^*.$$

The theorem makes explicit the link between low rank approximation and total least squares. From a data modeling point of view,

total least squares is low rank approximation of the data matrix $D = [A \ B]^\top$, followed by input/output representation of the optimal model.

```
229a <Total least squares 229a>≡
    function [x, ah, bh] = tls(a, b)
        n = size(a, 2); [r, p, dh] = lra([a b]', n);
        <low rank approximation ↦ total least squares solution 229b>
```

Defines:
 tls, never used.
 Uses lra 64.

```
229b <low rank approximation ↦ total least squares solution 229b>≡ (229a 230)
    x = p2x(p); ah = dh(1:n, :); bh = dh((n + 1):end, :);
```

Lack of solution of the total least squares problem (TLS)—a case called non-generic total least squares problem—is caused by lack of existence of an input/output representation of the model. Nongeneric total least squares problems are considered in Van Huffel and Vandewalle (1988, 1991) and Paige and Strakos (2005).

- The *generalized total least squares* (Van Huffel and Vandewalle 1989) method measures the size of the data correction matrix

$$[\Delta A \ \Delta B] := [A \ B] - [\widehat{A} \ \widehat{B}]$$

after row and column weighting:

$$\begin{aligned} &\text{minimize over } \widehat{A}, \widehat{B}, \text{ and } X \quad \|W_1([A \ B] - [\widehat{A} \ \widehat{B}])W_r\|_F \\ &\text{subject to } \widehat{A}X = \widehat{B}. \end{aligned} \quad (\text{GTLS})$$

Here the matrices W_1 and W_r are positive semidefinite weight matrices— W_1 corresponds to weighting of the rows and W_r to weighting of the columns of the correction $[\Delta A \ \Delta B]$. Similarly to the classical total least squares problem, the existence and uniqueness of a generalized total least squares approximate solution is determined from the singular value decomposition. The data least squares and total least squares problems are special cases of the generalized total least squares problem.

- The *restricted total least squares* (Van Huffel and Zha 1991) method constrains the correction to be in the form

$$[\Delta A \ \Delta B] = P_e E L_e, \quad \text{for some } E,$$

i.e., the row and column span of the correction matrix are constrained to be within the given subspaces $\text{image}(P_e)$ and $\text{image}(L_e^\top)$, respectively. The restricted total least squares problem is:

$$\begin{aligned} &\text{minimize over } \widehat{A}, \widehat{B}, E, \text{ and } X \quad \|E\|_F \\ &\text{subject to } [A \ B] - [\widehat{A} \ \widehat{B}] = P_e E L_e \quad \text{and} \quad \widehat{A}X = \widehat{B}. \end{aligned} \quad (\text{RTLS})$$

The generalized total least squares problem is a special case of (RTLS).

- The *Procrustes problem*: given $m \times n$ real matrices A and B ,

$$\text{minimize over } X \quad \|B - AX\|_F \quad \text{subject to } X^\top X = I_n$$

is a least squares problem with a constraint that the unknown X is an orthogonal matrix. The solution is given by $X = UV^\top$, where $U\Sigma V^\top$ is the singular value decomposition of AB^\top , see Golub and Van Loan (1996, p. 601).

- The *weighted total least squares* (De Moor 1993, Sect. 4.3) method generalizes the classical total least squares problem by measuring the correction size by a weighted matrix norm $\|\cdot\|_W$

$$\begin{aligned} &\text{minimize over } \widehat{A}, \widehat{B}, \text{ and } X \quad \|[A \ B] - [\widehat{A} \ \widehat{B}]\|_W \\ &\text{subject to } \widehat{A}X = \widehat{B}. \end{aligned} \quad (\text{WTLS})$$

Special weighted total least squares problems correspond to weight matrices W with special structure, e.g., diagonal W corresponds to *element-wise weighted total least squares* (Markovsky et al. 2005). In general, the weighted total least squares problem has no analytic solution in terms of the singular value decomposition, so that contrary to the above listed generalizations, weighted total least squares problems, in general, cannot be solved globally and efficiently. Weighted low rank approximation problems, corresponding to the weighted total least squares problem are considered in Wentzell et al. (1997), Manton et al. (2003) and Markovsky and Van Huffel (2007a).

230 *(Weighted total least squares 230)*≡
 function [x, ah, bh, info] =
 wtls(a, b, s, opt) n = size(a,
 2); [p, l, info] = wlra([a b]',
 n, s, opt); dh = p * l;
(low rank approximation ↦ total least squares solution 229b)

Defines:

wtls, never used.

Uses wlra 139.

- The *regularized total least squares* (Fierro et al. 1997; Golub et al. 1999; Sima et al. 2004; Beck and Ben-Tal 2006; Sima 2006) methods is defined as

$$\begin{aligned} &\text{minimize over } \widehat{A}, \widehat{B}, \text{ and } X \quad \|[A \ B] - [\widehat{A} \ \widehat{B}]\|_F + \gamma \|DX\|_F \\ &\text{subject to } \widehat{A}X = \widehat{B}. \end{aligned} \quad (\text{RegTLS})$$

Global and efficient solution methods for solving regularized total least squares problems are derived in Beck and Ben-Tal (2006).

- The *structured total least squares* method (De Moor 1993; Abatzoglou et al. 1991) method is a total least squares method with the additional constraint that the correction should have certain specified structure

$$\begin{aligned} &\text{minimize over } \widehat{A}, \widehat{B}, \text{ and } X \quad \|[A \ B] - [\widehat{A} \ \widehat{B}]\|_F \\ &\text{subject to } \widehat{A}X = \widehat{B} \quad \text{and} \quad [\widehat{A} \ \widehat{B}] \text{ has a specified structure.} \end{aligned} \quad (\text{STLS})$$

Hankel and Toeplitz structured total least squares problems are the most often studied ones due to their application in signal processing and system theory.

- The *structured total least norm* method (Rosen et al. 1996) is the same as the structured total least squares method with a general matrix norm in the approximation criterion instead of the Frobenius norm.

For generalizations and applications of the total least squares problem in the periods 1990–1996, 1996–2001, and 2001–2006, see, respectively, the edited books (Van Huffel 1997; Van Huffel and Lemmerling 2002), and the special issues (Van Huffel et al. 2007a, 2007b). An overview of total least squares problems is given in Van Huffel and Zha (1993), Markovsky and Van Huffel (2007b) and Markovsky et al. (2010).

References

- Abatzoglou T, Mendel J, Harada G (1991) The constrained total least squares technique and its application to harmonic superresolution. *IEEE Trans Signal Process* 39:1070–1087
- Beck A, Ben-Tal A (2006) On the solution of the Tikhonov regularization of the total least squares. *SIAM J Optim* 17(1):98–118
- Björck Å (1996) Numerical methods for least squares problems. SIAM, Philadelphia
- De Moor B (1993) Structured total least squares and L_2 approximation problems. *Linear Algebra Appl* 188–189:163–207
- Degroat R, Dowling E (1991) The data least squares problem and channel equalization. *IEEE Trans Signal Process* 41:407–411
- Fierro R, Golub G, Hansen P, O’Leary D (1997) Regularization by truncated total least squares. *SIAM J Sci Comput* 18(1):1223–1241
- Golub G (1973) Some modified matrix eigenvalue problems. *SIAM Rev* 15:318–344
- Golub G, Reinsch C (1970) Singular value decomposition and least squares solutions. *Numer Math* 14:403–420
- Golub G, Van Loan C (1980) An analysis of the total least squares problem. *SIAM J Numer Anal* 17:883–893
- Golub G, Van Loan C (1996) Matrix computations, 3rd edn. Johns Hopkins University Press
- Golub G, Hansen P, O’Leary D (1999) Tikhonov regularization and total least squares. *SIAM J Matrix Anal Appl* 21(1):185–194
- Hansen PC (1997) Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion. SIAM, Philadelphia
- Kailath T, Sayed AH, Hassibi B (2000) Linear estimation. Prentice Hall, New York
- Manton J, Mahony R, Hua Y (2003) The geometry of weighted low-rank approximations. *IEEE Trans Signal Process* 51(2):500–514
- Markovsky I, Van Huffel S (2007a) Left vs right representations for solving weighted low rank approximation problems. *Linear Algebra Appl* 422:540–552
- Markovsky I, Van Huffel S (2007b) Overview of total least squares methods. *Signal Process* 87:2283–2302
- Markovsky I, Rastello ML, Premoli A, Kukush A, Van Huffel S (2005) The element-wise weighted total least squares problem. *Comput Stat Data Anal* 50(1):181–209
- Markovsky I, Sima D, Van Huffel S (2010) Total least squares methods. *Wiley Interdiscip Rev: Comput Stat* 2(2):212–217
- Meyer C (2000) Matrix analysis and applied linear algebra. SIAM, Philadelphia
- Paige C, Strakos Z (2005) Core problems in linear algebraic systems. *SIAM J Matrix Anal Appl* 27:861–875

- Rosen J, Park H, Glick J (1996) Total least norm formulation and solution of structured problems. *SIAM J Matrix Anal Appl* 17:110–126
- Sima D (2006) Regularization techniques in model fitting and parameter estimation. PhD thesis, ESAT, KU Leuven
- Sima D, Van Huffel S, Golub G (2004) Regularized total least squares based on quadratic eigenvalue problem solvers. *BIT* 44:793–812
- Strang G (1976) *Linear algebra and its applications*. Academic Press, San Diego
- Trefethen L, Bau D (1997) *Numerical linear algebra*. SIAM, Philadelphia
- Van Huffel S (ed) (1997) Recent advances in total least squares techniques and errors-in-variables modeling. SIAM, Philadelphia
- Van Huffel S, Lemmerling P (eds) (2002) *Total least squares and errors-in-variables modeling: analysis, algorithms and applications*. Kluwer, Amsterdam
- Van Huffel S, Vandewalle J (1988) Analysis and solution of the nongeneric total least squares problem. *SIAM J Matrix Anal Appl* 9:360–372
- Van Huffel S, Vandewalle J (1989) Analysis and properties of the generalized total least squares problem $AX \approx B$ when some or all columns in A are subject to error. *SIAM J Matrix Anal* 10(3):294–315
- Van Huffel S, Vandewalle J (1991) *The total least squares problem: computational aspects and analysis*. SIAM, Philadelphia
- Van Huffel S, Zha H (1991) The restricted total least squares problem: formulation, algorithm and properties. *SIAM J Matrix Anal Appl* 12(2):292–309
- Van Huffel S, Zha H (1993) The total least squares problem. In: Rao C (ed) *Handbook of statistics: comput stat, vol 9*. Elsevier, Amsterdam, pp 377–408
- Van Huffel S, Cheng CL, Mastronardi N, Paige C, Kukush A (2007a) Editorial: Total least squares and errors-in-variables modeling. *Comput Stat Data Anal* 52:1076–1079
- Van Huffel S, Markovsky I, Vaccaro RJ, Söderström T (2007b) Guest editorial: Total least squares and errors-in-variables modeling. *Signal Proc* 87:2281–2282
- Wentzell P, Andrews D, Hamilton D, Faber K, Kowalski B (1997) Maximum likelihood principal component analysis. *J Chemometr* 11:339–366

Appendix B

Proofs

... the ideas and the arguments with which the mathematician is concerned have physical, intuitive or geometrical reality long before they are recorded in the symbolism. The proof is meaningful when it answers the students doubts, when it proves what is not obvious. Intuition may fly the student to a conclusion but where doubt remains he may then be asked to call upon plodding logic to show the overland route to the same goal.

Kline (1974)

Proof of Proposition 2.23

The proof is given in Vanluyten et al. (2006). Let \widehat{D}^* be a solution to

$$\begin{aligned} \widehat{D}^* &:= \arg \min_{\widehat{D}} \|D - \widehat{D}\| \\ &\text{subject to } \text{rank}(\widehat{D}) \leq m. \end{aligned} \tag{LRA}$$

and let

$$\widehat{D}^* := U^* \Sigma^* (V^*)^\top$$

be a singular value decomposition of \widehat{D}^* . By the unitary invariance of the Frobenius norm, we have

$$\begin{aligned} \|D - \widehat{D}^*\|_F &= \|(U^*)^\top (D - \widehat{D}^*) V^*\|_F \\ &= \|\underbrace{(U^*)^\top D V^*}_{\widehat{D}} - \Sigma^*\|_F, \end{aligned}$$

which shows that Σ^* is an optimal approximation of \widehat{D} . Partition

$$\widehat{D} =: \begin{bmatrix} \widehat{D}_{11} & \widehat{D}_{12} \\ \widehat{D}_{21} & \widehat{D}_{22} \end{bmatrix}$$

conformably with $\Sigma^* =: \begin{bmatrix} \Sigma_1^* & 0 \\ 0 & 0 \end{bmatrix}$ and observe that

$$\begin{aligned} \text{rank} \left(\begin{bmatrix} \Sigma_1^* & \widehat{D}_{12} \\ 0 & 0 \end{bmatrix} \right) &\leq m \quad \text{and} \quad \widehat{D}_{12} \neq 0 \\ \implies \left\| \widehat{D} - \begin{bmatrix} \Sigma_1^* & \widehat{D}_{12} \\ 0 & 0 \end{bmatrix} \right\|_{\text{F}} &< \left\| \widehat{D} - \begin{bmatrix} \Sigma_1^* & 0 \\ 0 & 0 \end{bmatrix} \right\|_{\text{F}}, \end{aligned}$$

so that $\widehat{D}_{12} = 0$. Similarly $\widehat{D}_{21} = 0$. Observe also that

$$\begin{aligned} \text{rank} \left(\begin{bmatrix} \widehat{D}_{11} & 0 \\ 0 & 0 \end{bmatrix} \right) &\leq m \quad \text{and} \quad \widehat{D}_{11} \neq \Sigma_1^* \\ \implies \left\| \widehat{D} - \begin{bmatrix} \widehat{D}_{11} & 0 \\ 0 & 0 \end{bmatrix} \right\|_{\text{F}} &< \left\| \widehat{D} - \begin{bmatrix} \Sigma_1^* & 0 \\ 0 & 0 \end{bmatrix} \right\|_{\text{F}}, \end{aligned}$$

so that $\widehat{D}_{11} = \Sigma_1^*$. Therefore,

$$\widehat{D} = \begin{bmatrix} \Sigma_1^* & 0 \\ 0 & \widehat{D}_{22} \end{bmatrix}.$$

Let

$$\widehat{D}_{22} = U_{22} \Sigma_{22} V_{22}^\top$$

be the singular value decomposition of \widehat{D}_{22} . Then the matrix

$$\begin{bmatrix} I & 0 \\ 0 & U_{22}^\top \end{bmatrix} \widehat{D} \begin{bmatrix} I & 0 \\ 0 & V_{22} \end{bmatrix} = \begin{bmatrix} \Sigma_1^* & 0 \\ 0 & \Sigma_{22} \end{bmatrix}$$

has optimal rank- m approximation $\Sigma^* =: \begin{bmatrix} \Sigma_1^* & 0 \\ 0 & 0 \end{bmatrix}$, so that

$$\min(\text{diag}(\Sigma_1^*)) > \max(\text{diag}(\Sigma_{22}))$$

Therefore,

$$D = U^* \begin{bmatrix} I & 0 \\ 0 & U_{22} \end{bmatrix} \begin{bmatrix} \Sigma_1^* & 0 \\ 0 & \Sigma_{22} \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & V_{22}^\top \end{bmatrix} (V^*)^\top$$

is a singular value decomposition of D .

Then, if $\sigma_m > \sigma_{m+1}$, the rank- m truncated singular value decomposition

$$\widehat{D}^* = U^* \begin{bmatrix} \Sigma_1^* & 0 \\ 0 & 0 \end{bmatrix} (V^*)^\top = U^* \begin{bmatrix} I & 0 \\ 0 & U_{22} \end{bmatrix} \begin{bmatrix} \Sigma_1^* & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & V_{22}^\top \end{bmatrix} (V^*)^\top$$

is unique and \widehat{D}^* is the unique solution of (LRA). Moreover, \widehat{D}^* is simultaneously optimal in any unitarily invariant norm.

Proof of Proposition 2.32

The probability density function of the observation vector $\text{vec}(D)$ is

$$p_{\widehat{\mathcal{B}}, \widehat{D}}(\text{vec}(D)) = \begin{cases} \text{const} \cdot \exp\left(-\frac{1}{2\sigma^2} \|\text{vec}(D) - \text{vec}(\widehat{D})\|_{V^{-1}}^2\right), & \text{if } \text{image}(\widehat{D}) \subset \widehat{\mathcal{B}} \text{ and } \dim(\widehat{\mathcal{B}}) \leq m \\ 0, & \text{otherwise,} \end{cases}$$

where “const” is a term that does not depend on \widehat{D} and $\widehat{\mathcal{B}}$. The log-likelihood function is

$$\ell(\widehat{\mathcal{B}}, \widehat{D}) = \begin{cases} -\text{const} \cdot \frac{1}{2\sigma^2} \|\text{vec}(D) - \text{vec}(\widehat{D})\|_{V^{-1}}^2, & \text{if } \text{image}(\widehat{D}) \subset \widehat{\mathcal{B}} \text{ and } \dim(\widehat{\mathcal{B}}) \leq m \\ -\infty, & \text{otherwise,} \end{cases}$$

and the maximum likelihood estimation problem is

$$\begin{aligned} &\text{minimize} \quad \text{over } \widehat{\mathcal{B}} \text{ and } \widehat{D} \quad \frac{1}{2\sigma^2} \|\text{vec}(D) - \text{vec}(\widehat{D})\|_{V^{-1}}^2 \\ &\text{subject to} \quad \text{image}(\widehat{D}) \subset \widehat{\mathcal{B}} \quad \text{and} \quad \dim(\widehat{\mathcal{B}}) \leq m, \end{aligned}$$

which is an equivalent problem to Problem 2.31 with $\|\cdot\| = \|\cdot\|_{V^{-1}}$.

Note B.1 (Weight matrix in the norm specification) The weight matrix W in the norm specification is the inverse of the measurement noise covariance matrix V . In case of singular covariance matrix (e.g., missing data) the method needs modification.

Proof of Theorem 3.16

The polynomial equations (GCD) are equivalent to the following systems of algebraic equations

$$\begin{bmatrix} \widehat{p}_0 \\ \widehat{p}_1 \\ \vdots \\ \widehat{p}_n \end{bmatrix} = \mathcal{T}_{d+1}^\top(u) \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_d \end{bmatrix}, \quad \begin{bmatrix} \widehat{q}_0 \\ \widehat{q}_1 \\ \vdots \\ \widehat{q}_n \end{bmatrix} = \mathcal{T}_{d+1}^\top(v) \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_d \end{bmatrix},$$

where the Toeplitz matrix constructor \mathcal{T} is defined in (\mathcal{T}) on p. 86. Rewriting and combining the above equations, we find that a polynomial c is a common factor of

\widehat{p} and \widehat{q} with $\text{degree}(c) \leq d$ if and only if the system of equations

$$\begin{bmatrix} \widehat{p}_0 & \widehat{q}_0 \\ \widehat{p}_1 & \widehat{q}_1 \\ \vdots & \vdots \\ \widehat{p}_n & \widehat{q}_n \end{bmatrix} = \mathcal{F}_{n-d+1}^\top(c) \begin{bmatrix} u_0 & v_0 \\ u_1 & v_1 \\ \vdots & \vdots \\ u_{n-d} & v_{n-d} \end{bmatrix}$$

has a solution.

The condition $\text{degree}(c) = d$ implies that the highest power coefficient c_d of c is different from 0. Since c is determined up to a scaling factor, we can impose the normalization $c_d = 1$. Conversely, imposing the constraint $c_d = 1$ in the optimization problem to be solved ensures that $\text{degree}(c) = d$. Therefore, Problem 3.13 is equivalent to

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{p}, \widehat{q} \in \mathbb{R}^{n+1}, u, v \in \mathbb{R}^{n-d+1}, \text{ and } c_0, \dots, c_{d-1} \in \mathbb{R} \\ & && \text{trace}(\left([p \ q] - [\widehat{p} \ \widehat{q}]\right)^\top \left([p \ q] - [\widehat{p} \ \widehat{q}]\right)) \\ & \text{subject to} && [\widehat{p} \ \widehat{q}] = \mathcal{F}_{n-d+1}^\top(c) [u \ v]. \end{aligned}$$

Substituting $[\widehat{p} \ \widehat{q}]$ in the cost function and minimizing with respect to $[u \ v]$ by solving a least squares problem gives the equivalent problem (AGCD').

Proof of Theorem 5.17

First, we show that the sequence $\widehat{D}^{(1)}, \widehat{D}^{(1)}, \dots, \widehat{D}^{(k)}, \dots$ converges monotonically in the Σ -weighted norm $\|\cdot\|_\Sigma$. On each iteration, Algorithm 6 solves two optimization problems (steps 1 and 2), which cost functions and constraints coincide with the ones of problem (C₀–C₅). Therefore, the cost function $\|D - \widehat{D}^{(k)}\|_\Sigma^2$ is monotonically nonincreasing. The cost function is bounded from below, so that the sequence

$$\|D - \widehat{D}^{(1)}\|_\Sigma^2, \quad \|D - \widehat{D}^{(2)}\|_\Sigma^2, \quad \dots$$

is convergent. This proves $(f(k) \rightarrow f^*)$.

Although, $\widehat{D}^{(k)}$ converges in norm, it may not converge element-wise. A sufficient condition for element-wise convergence is that the underlying optimization problem has a solution and this solution is unique, see Kiers (2002, Theorem 5). The element-wise convergence of $\widehat{D}^{(k)}$ and the uniqueness (due to the normalization condition (A₁)) of the factors $P^{(k)}$ and $L^{(k)}$, implies element-wise convergence of the factor sequences $P^{(k)}$ and $L^{(k)}$ as well. This proves $(D^{(k)} \rightarrow D^*)$.

In order to show that the algorithm convergence to a minimum point of (C₀–C₅), we need to verify that the first order optimality conditions for (C₀–C₅) are satisfied at a cluster point of the algorithm. The algorithm converges to a cluster point if and

only if the union of the first order optimality conditions for the problems on steps 1 and 2 are satisfied. Then

$$P^{(k-1)} = P^{(k)} =: P^* \quad \text{and} \quad L^{(k-1)} = L^{(k)} =: L^* .$$

From the above conditions for a stationary point and the Lagrangians of the problems of steps 1 and 2 and (C_0-C_5) , it is easy to see that the union of the first order optimality conditions for the problems on steps 1 and 2 coincides with the first order optimality conditions of (C_0-C_5) .

References

- Kiers H (2002) Setting up alternating least squares and iterative majorization algorithms for solving various matrix optimization problems. *Comput Stat Data Anal* 41:157–170
- Kline M (1974) *Why Johnny can't add: the failure of the new math*. Random House Inc
- Vanluyten B, Willems JC, De Moor B (2006) Matrix factorization and stochastic state representations. In: *Proc 45th IEEE conf on dec and control*, San Diego, California, pp 4188–4193

Appendix P

Problems

Problem P.1 (Least squares data fitting) Verify that the least squares fits, shown in Fig. 1.1 on p. 4, minimize the sums of squares of horizontal and vertical distances. The data points are:

$$d_1 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}, \quad d_2 = \begin{bmatrix} -1 \\ 4 \end{bmatrix}, \quad d_3 = \begin{bmatrix} 0 \\ 6 \end{bmatrix}, \quad d_4 = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \quad d_5 = \begin{bmatrix} 2 \\ 1 \end{bmatrix},$$

$$d_6 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \quad d_7 = \begin{bmatrix} 1 \\ -4 \end{bmatrix}, \quad d_8 = \begin{bmatrix} 0 \\ -6 \end{bmatrix}, \quad d_9 = \begin{bmatrix} -1 \\ -4 \end{bmatrix}, \quad d_{10} = \begin{bmatrix} -2 \\ -1 \end{bmatrix}$$

Problem P.2 (Distance from a data point to a linear model) The 2-norm distance from a point $d \in \mathbb{R}^q$ to a linear static model $\mathcal{B} \subset \mathbb{R}^q$ is defined as

$$\text{dist}(d, \mathcal{B}) := \min_{\hat{d} \in \mathcal{B}} \|d - \hat{d}\|_2, \tag{dist}$$

i.e., $\text{dist}(d, \mathcal{B})$ is the shortest distance from d to a point \hat{d} in \mathcal{B} . A vector \hat{d}^* that achieves the minimum of (dist) is a point in \mathcal{B} that is closest to d .

Next we consider the special case when \mathcal{B} is a linear static model.

1. Let

$$\mathcal{B} = \text{image}(a) = \{\alpha a \mid \alpha \in \mathbb{R}\}.$$

Explain how to find $\text{dist}(d, \text{image}(a))$. Find

$$\text{dist}(\text{col}(1, 0), \text{image}(\text{col}(1, 1))).$$

Note that the best approximation \hat{d}^* of d in $\text{image}(a)$ is the orthogonal projection of d onto $\text{image}(a)$.

2. Let $\mathcal{B} = \text{image}(P)$, where P is a given full column rank matrix. Explain how to find $\text{dist}(d, \mathcal{B})$.
3. Let $\mathcal{B} = \text{ker}(R)$, where R is a given full row rank matrix. Explain how to find $\text{dist}(d, \mathcal{B})$.

4. Prove that in the linear static case, a solution \widehat{d}^* of (dist) is always unique?
5. Prove that in the linear static case, the approximation error $\Delta d^* := d - \widehat{d}^*$ is orthogonal to \mathcal{B} . Is the converse true, i.e., is it true that if for some \widehat{d} , $d - \widehat{d}$ is orthogonal to \mathcal{B} , then $\widehat{d} = \widehat{d}^*$?

Problem P.3 (Distance from a data point to an affine model) Consider again the distance $\text{dist}(d, \mathcal{B})$ defined in (dist). In this problem, \mathcal{B} is an affine static model, i.e.,

$$\mathcal{B} = \mathcal{B}' + a,$$

where \mathcal{B}' is a linear static model and a is a fixed vector.

1. Explain how to reduce the problem of computing the distance from a point to an affine static model to an equivalent problem of computing the distance from a point to a linear static model (Problem P.2).
2. Find

$$\text{dist} \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \ker([1 \ 1]) + \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right).$$

Problem P.4 (Geometric interpretation of the total least squares problem) Show that the total least squares problem

$$\begin{aligned} &\text{minimize over } x \in \mathbb{R}, \widehat{a} \in \mathbb{R}^N, \text{ and } \widehat{b} \in \mathbb{R}^N \quad \sum_{j=1}^N \left\| d_j - \begin{bmatrix} \widehat{a}_j \\ \widehat{b}_j \end{bmatrix} \right\|_2^2 && \text{(tls)} \\ &\text{subject to } \widehat{a}_j x = \widehat{b}_j, \quad \text{for } j = 1, \dots, N \end{aligned}$$

minimizes the sum of the squared orthogonal distances from the data points d_1, \dots, d_N to the fitting line

$$\mathcal{B} = \{ \text{col}(a, b) \mid xa = b \}$$

over all lines passing through the origin, except for the vertical line.

Problem P.5 (Unconstrained problem, equivalent to the total least squares problem)

A total least squares approximate solution x_{tls} of the linear system of equations $Ax \approx b$ is a solution to the following optimization problem

$$\text{minimize over } x, \widehat{A}, \text{ and } \widehat{b} \quad \|[A \ b] - [\widehat{A} \ \widehat{b}]\|_F^2 \quad \text{subject to } \widehat{A}x = \widehat{b}. \quad \text{(TLS)}$$

Show that (TLS) is equivalent to the unconstrained optimization problem

$$\text{minimize } f_{\text{tls}}(x), \quad \text{where } f_{\text{tls}}(x) := \frac{\|Ax - b\|_2^2}{\|x\|_2^2 + 1}. \quad \text{(TLS')}$$

Give an interpretation of the function f_{tls} .

Problem P.6 (Lack of total least squares solution) Using the formulation (TLS'), derived in Problem P.5, show that the total least squares line fitting problem (tls) has no solution for the data in Problem P.1.

Problem P.7 (Geometric interpretation of rank-1 approximation) Show that the rank-1 approximation problems

$$\begin{aligned} &\text{minimize over } R \in \mathbb{R}^{1 \times 2}, R \neq 0, \text{ and } \widehat{D} \in \mathbb{R}^{2 \times N} \quad \|D - \widehat{D}\|_F^2 \\ &\text{subject to } R\widehat{D} = 0. \end{aligned} \tag{Ira}_R$$

and

$$\begin{aligned} &\text{minimize over } P \in \mathbb{R}^{2 \times 1} \text{ and } L \in \mathbb{R}^{1 \times N} \quad \|D - \widehat{D}\|_F^2 \\ &\text{subject to } \widehat{D} = PL. \end{aligned} \tag{Ira}_P$$

minimize the sum of the squared orthogonal distances from the data points d_1, \dots, d_N to the fitting line $\mathcal{B} = \ker(P) = \text{image}(P)$ over all lines passing through the origin. Compare and contrast with the similar statement in Problem P.4.

Problem P.8 (Quadratically constrained problem, equivalent to rank-1 approximation) Show that ((Ira_P)) is equivalent to the quadratically constrained optimization problem

$$\text{minimize } f_{\text{Ira}}(P) \quad \text{subject to } P^\top P = 1, \tag{Ira}'_P$$

where

$$f_{\text{Ira}}(P) = \text{trace}(D^\top (I - PP^\top) D).$$

Explain how to find all solutions of ((Ira_P)) from a solution of ((Ira'_P)). Assuming that a solution to ((Ira'_P)) exists, is it unique?

Problem P.9 (Line fitting by rank-1 approximation) Plot the cost function $f_{\text{Ira}}(P)$ for the data in Problem P.1 over all P such that $P^\top P = 1$. Find from the graph of f_{Ira} the minimum points. Using the link between ((Ira'_P)) and ((Ira_P)), established in Problem P.7, interpret the minimum points of f_{Ira} in terms of the line fitting problem for the data in Problem P.1. Compare and contrast with the total least squares approach, used in Problem P.6.

Problem P.10 (Analytic solution of a rank-1 approximation problem) Show that for the data in Problem P.1,

$$f_{\text{Ira}}(P) = P^\top \begin{bmatrix} 140 & 0 \\ 0 & 20 \end{bmatrix} P.$$

Using geometric or analytic arguments, conclude that the minimum of f_{Ira} for a P on the unit circle is 20 and is achieved for

$$P^{*,1} = \text{col}(0, 1) \quad \text{and} \quad P^{*,2} = \text{col}(0, -1).$$

Compare the results with those obtained in Problem P.9.

Problem P.11 (Analytic solution of two-variate rank-1 approximation problem) Find an analytic solution of the Frobenius norm rank-1 approximation of a $2 \times N$ matrix.

Problem P.12 (Analytic solution of scalar total least squares) Find an analytic expression for the total least squares solution of the system $ax \approx b$, where $a, b \in \mathbb{R}^m$.

Problem P.13 (Alternating projections algorithm for low-rank approximation) In this problem, we consider a numerical method for rank- r approximation:

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{D} && \|D - \widehat{D}\|_F^2 \\ & \text{subject to} && \text{rank}(\widehat{D}) \leq m. \end{aligned} \tag{LRA}$$

The alternating projections algorithm, outlined next, is based on an image representation $\widehat{D} = PL$, where $P \in \mathbb{R}^{\alpha \times m}$ and $L \in \mathbb{R}^{m \times N}$, of the rank constraint.

1. Implement the algorithm and test it on random data matrices D of different dimensions with different rank specifications and initial approximations. Plot the approximation errors

$$e_k := \|D - \widehat{D}^{(k)}\|_F^2, \quad \text{for } k = 0, 1, \dots$$

as a function of the iteration step k and comment on the results.

- * 2. Give a proof or a counter example for the conjecture that the sequence of approximation errors $e := (e_0, e_1, \dots)$ is well defined, independent of the data and the initial approximation.
- * 3. Assuming that e is well defined. Give a proof or a counter example for the conjecture that e converges monotonically to a limit point e_∞ .
- * 4. Assuming that e_∞ exists, give proofs or counter examples for the conjectures that e_∞ is a local minimum of (LRA) and e_∞ is a global minimum of (LRA).

Algorithm 8 Alternating projections algorithm for low rank approximation

Input: A matrix $D \in \mathbb{R}^{\alpha \times N}$, with $\alpha \leq N$, an initial approximation $\widehat{D}^{(0)} = P^{(0)}L^{(0)}$, $P^{(0)} \in \mathbb{R}^{\alpha \times m}$, $L^{(0)} \in \mathbb{R}^{m \times N}$, with $m \leq \alpha$, and a convergence tolerance $\varepsilon > 0$.

1: Set $k := 0$.

2: **repeat**

3: $k := k + 1$.

4: Solve: $P^{(k+1)} := \arg \min_P \|D - PL^{(k)}\|_F^2$

5: Solve: $L^{(k+1)} := \arg \min_L \|D - P^{(k+1)}L\|_F^2$

6: $\widehat{D}^{(k+1)} := P^{(k+1)}L^{(k+1)}$

7: **until** $\|\widehat{D}^{(k)} - \widehat{D}^{(k+1)}\|_F < \varepsilon$

Output: Output the matrix $\widehat{D}^{(k+1)}$.

Problem P.14 (Two-sided weighted low rank approximation) Prove Theorem 2.29 on p. 65.

Problem P.15 (Most powerful unfalsified model for autonomous models) Given a trajectory

$$y = (y(1), y(2), \dots, y(T))$$

of an autonomous linear time-invariant system \mathcal{B} of order n , find a state space representation $\mathcal{B}_{i/s/o}(A, C)$ of \mathcal{B} . Modify your procedure, so that it does not require prior knowledge of the system order n but only an upper bound n_{\max} for it.

Problem P.16 (Algorithm for exact system identification) Develop an algorithm for exact system identification that computes a kernel representation of the model, i.e., implement the mapping

$$w_d \mapsto R(z), \quad \text{where } \widehat{\mathcal{B}} := \ker(R(z)) \text{ is the identified model.}$$

Consider separately the cases of known and unknown model order. You can assume that the system is single input single output and its order is known.

Problem P.17 (A simple method for approximate system identification) Modify the algorithm developed in Problem P.16, so that it can be used as an approximate identification method. You can assume that the system is single input single output and the order is known.

* **Problem P.18** (When is $\mathcal{B}_{\text{mpum}}(w_d)$ equal to the data generating system?) Choose a (random) linear time-invariant system \mathcal{B}_0 (the “true data generating system”) and a trajectory $w_d = (u_d, y_d)$ of \mathcal{B}_0 . The aim is to recover the data generating system \mathcal{B}_0 back from the data w_d . Conjecture that this can be done by computing the most powerful unfalsified model $\mathcal{B}_{\text{mpum}}(w_d)$. Verify whether and when in simulation $\mathcal{B}_{\text{mpum}}(w_d)$ coincides with \mathcal{B}_0 . Find counter examples when the conjecture is not true and based on this experience revise the conjecture. Find sufficient conditions for $\mathcal{B}_{\text{mpum}}(w_d) = \mathcal{B}_0$.

Problem P.19 (Algorithms for approximate system identification)

1. Download the file [[flutter.dat]] from a Database for System Identification (De Moor 1999).
2. Apply the function developed in Problem P.17 on the flutter data using model order $n = 3$.
3. Compute the misfit between the flutter data and the model obtained in step 1.
4. Compute a locally optimal model of order $n = 3$ and compare the misfit with the one obtained in step 3.
5. Repeat steps 2–4 for different partitions of the data into identification and validation parts (e.g., first 60% for identification and remaining 40% for validation). More specifically, use only the identification part of the data to find the models and compute the misfit on the unused validation part of the data.

Problem P.20 (Computing approximate common divisor with `slra`) Given polynomials p and q of degree n or less and an integer $d < n$, use `slra` to solve the Sylvester structured low rank approximation problem

$$\begin{aligned} & \text{minimize} && \text{over } \widehat{p}, \widehat{q} \in \mathbb{R}^{n+1} && \|[p \ q] - [\widehat{p} \ \widehat{q}]\|_F \\ & \text{subject to} && \text{rank}(\mathcal{R}_d(\widehat{p}, \widehat{q})) \leq 2n - 2d + 1 \end{aligned}$$

in order to compute an approximate common divisor c of p and q with degree at least d . Verify the answer with the alternative approach developed in Sect. 3.2.

Problem P.21 (Matrix centering) Prove Proposition 5.5.

Problem P.22 (Mean computation as an optimal modeling) Prove Proposition 5.6.

Problem P.23 (Nonnegative low rank approximation) Implement and test the algorithm for nonnegative low rank approximation (Algorithm 7 on p. 177).

Problem P.24 (Luenberger 1979, p. 53) A thermometer reading 21°C , which has been inside a house for a long time, is taken outside. After one minute the thermometer reads 15°C ; after two minutes it reads 11°C . What is the outside temperature? (According to Newton's law of cooling, an object of higher temperature than its environment cools at a rate that is proportional to the difference in temperature.)

Problem P.25 Solve first Problem P.24. Consider the system of equations

$$\left[\mathbf{1}_{T-n} \otimes G \ \mathcal{H}_{T-n}(\Delta y) \right] \begin{bmatrix} \vec{u} \\ \ell \end{bmatrix} = \text{col}(y((n+1)t_s), \dots, y(Tt_s)), \quad (\text{SYS DD})$$

(the data-driven algorithm for input estimation on p. 210) in the case of a first order single input single output system and three data points. Show that the solution of the system (SYS DD) coincides with the one obtained in Problem P.24.

Problem P.26 Consider the system of equations (SYS DD) in the case of a first order single input single output system and N data points. Derive an explicit formula for the least squares approximate solution of (SYS DD). Propose a recursive algorithm that updates the current solution when new data point is obtained.

Problem P.27 Solve first Problem P.26. Implement the solution obtained in Problem P.26 and validate it against the function `stepid_dd`.

References

- De Moor B (1999) DaISy: database for the identification of systems. www.esat.kuleuven.be/sista/daisy/
- Luenberger DG (1979) Introduction to dynamical systems: theory, models and applications. Wiley, New York

Notation

Symbolism can serve three purposes. It can communicate ideas effectively; it can conceal ideas; and it can conceal the absence of ideas.

M. Kline, Why Johnny Can't Add: The Failure of the New Math

Sets of numbers

- \mathbb{R} the set of real numbers
- \mathbb{Z}, \mathbb{Z}_+ the sets of integers and positive integers (natural numbers)

Norms and extreme eigen/singular values

- $\|x\| = \|x\|_2, x \in \mathbb{R}^n$ vector 2-norm
- $\|w\|, w \in (\mathbb{R}^q)^T$ signal 2-norm
- $\|A\|, A \in \mathbb{R}^{m \times n}$ matrix induced 2-norm
- $\|A\|_F, A \in \mathbb{R}^{m \times n}$ matrix Frobenius norm
- $\|A\|_W, W \geq 0$ matrix weighted norm
- $\|A\|_*$ nuclear norm
- $\lambda(A), A \in \mathbb{R}^{m \times m}$ spectrum (set of eigenvalues)
- $\lambda_{\min}(A), \lambda_{\max}(A)$ minimum, maximum eigenvalue of a symmetric matrix
- $\sigma_{\min}(A), \sigma_{\max}(A)$ minimum, maximum singular value of a matrix

Matrix operations

- A^+, A^\top pseudoinverse, transpose
- $\text{vec}(A)$ column-wise vectorization
- vec^{-1} operator reconstructing the matrix A back from $\text{vec}(A)$
- $\text{col}(a, b)$ the column vector $\begin{bmatrix} a \\ b \end{bmatrix}$
- $\text{col dim}(A)$ the number of block columns of A
- $\text{row dim}(A)$ the number of block rows of A
- $\text{image}(A)$ the span of the columns of A (the image or range of A)
- $\text{ker}(A)$ the null space of A (kernel of the function defined by A)
- $\text{diag}(v), v \in \mathbb{R}^n$ the diagonal matrix $\text{diag}(v_1, \dots, v_n)$
- \otimes Kronecker product $A \otimes B := [a_{ij}B]$
- \odot element-wise (Hadamard) product $A \odot B := [a_{ij}b_{ij}]$

Expectation, covariance, and normal distribution

\mathbf{E} , cov expectation, covariance operator
 $x \sim \mathbf{N}(m, V)$ x is normally distributed with mean m and covariance V

Fixed symbols

\mathcal{B}, \mathcal{M} model, model class
 \mathcal{S} structure specification
 $\mathcal{H}_i(w)$ Hankel matrix with i block rows, see (\mathcal{H}_i) on p. 10
 $\mathcal{T}_i(c)$ upper triangular Toeplitz matrix with i block rows, see (\mathcal{T}) on p. 86
 $\mathcal{R}(p, q)$ Sylvester matrix for the pair of polynomials p and q , see (\mathcal{R}) on p. 11
 $\mathcal{O}_i(A, C)$ extended observability matrix with i block-rows, see (\mathcal{O}) on p. 51
 $\mathcal{C}_i(A, B)$ extended controllability matrix with i block-columns, see (\mathcal{C}) on p. 51

Linear time-invariant model class

$m(\mathcal{B}), p(\mathcal{B})$ number of inputs, outputs of \mathcal{B}
 $l(\mathcal{B}), n(\mathcal{B})$ lag, order of \mathcal{B}
 $w|_{[1,T]}, \mathcal{B}|_{[1,T]}$ restriction of w, \mathcal{B} to the interval $[1, T]$, see $(\mathcal{B}|_{[1,T]})$ on p. 52

$$\mathcal{L}_{m,1}^{q,n} := \{ \mathcal{B} \subset (\mathbb{R}^q)^{\mathbb{Z}} \mid \mathcal{B} \text{ is linear time-invariant with} \\ m(\mathcal{B}) \leq m, l(\mathcal{B}) \leq 1, \text{ and } n(\mathcal{B}) \leq n \}$$

If m, l , or n are not specified, the corresponding invariants are not bounded.

Miscellaneous

$:= / =:$ left (right) hand side is defined by the right (left) hand side
 $:\iff$ left-hand side is defined by the right-hand side
 $\iff:$ right-hand side is defined by the left-hand side
 σ^τ the shift operator $(\sigma^\tau f)(t) = f(t + \tau)$
 \mathbf{i} imaginary unit
 δ Kronecker delta, $\delta_0 = 1$ and $\delta_t = 0$ for all $t \neq 0$
 $\mathbf{1}_n = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$ vector with n elements that are all ones
 $W > 0$ W is positive definite
 $\lceil a \rceil$ rounding to the nearest integer greater than or equal to a

With some abuse of notation, the discrete-time signal, vector, and polynomial

$$(w(1), \dots, w(T)) \iff \text{col}(w(1), \dots, w(T)) \iff z^1 w(1) + \dots + z^T w(T)$$

are all denoted by w . The intended meaning is understood from the context.

List of Code Chunks

- $\langle (S_0, S, \hat{p}) \mapsto \hat{D} = \mathcal{S}(\hat{p}) \text{ 83a} \rangle$
- $\langle (S_0, \mathbf{S}, \hat{p}) \mapsto \hat{D} = \mathcal{S}(\hat{p}) \text{ 82} \rangle$
- $\langle \hat{u} := G^{-1}G'\mathbf{1}_m, \text{ where } G' := \text{dcgain}(\mathcal{B}') \text{ 206b} \rangle$
- $\langle \hat{H} \mapsto \hat{\mathcal{B}} \text{ 109d} \rangle$
- $\langle \hat{p} \mapsto \hat{H} \text{ 109c} \rangle$
- $\langle \mathbf{S} \mapsto S \text{ 83d} \rangle$
- $\langle S \mapsto (m, n, n_p) \text{ 83b} \rangle$
- $\langle S \mapsto \mathbf{S} \text{ 83c} \rangle$
- $\langle \pi \mapsto \Pi \text{ 37a} \rangle$
- $\langle P \mapsto R \text{ 89c} \rangle$
- $\langle P \mapsto (TF) \text{ 88d} \rangle$
- $\langle R \mapsto (TF) \text{ 89b} \rangle$
- $\langle R \mapsto P \text{ 89a} \rangle$
- $\langle \text{2-norm optimal approximate realization 108b} \rangle$
- $\langle (R, \Pi) \mapsto X \text{ 42a} \rangle$
- $\langle (X, \Pi) \mapsto P \text{ 40a} \rangle$
- $\langle (X, \Pi) \mapsto R \text{ 39} \rangle$
- $\langle \text{dist}(\mathcal{D}, \mathcal{B}) \text{ (weighted low rank approximation) 142a} \rangle$
- $\langle \text{dist}(w_d, \mathcal{B}) \text{ 87b} \rangle$
- $\langle \Gamma, \Delta \rangle \mapsto (A, B, C) \text{ 74a} \rangle$
- $\langle \Theta \mapsto R_\Theta \text{ 193b} \rangle$
- $\langle H \mapsto \mathcal{B}_{i/s/o}(A, B, C, D) \text{ 76d} \rangle$
- $\langle P \mapsto R \text{ 40c} \rangle$
- $\langle R \mapsto \text{minimal } R \text{ 41} \rangle$
- $\langle R \mapsto P \text{ 40b} \rangle$
- $\langle R \mapsto \Pi \text{ 43a} \rangle$
- $\langle w \mapsto H \text{ 80b} \rangle$
- $\langle (TF) \mapsto P \text{ 88e} \rangle$
- $\langle (TF) \mapsto R \text{ 89d} \rangle$
- $\langle \text{Algorithm for sensor speedup based on reduction to autonomous system identification 209a} \rangle$
- $\langle \text{Algorithm for sensor speedup based on reduction to step response system identification 205} \rangle$
- $\langle \text{Algorithm for sensor speedup in the case of known dynamics 203a} \rangle$
- $\langle \text{alternating projections method 140b} \rangle$
- $\langle \text{approximate realization structure 109a} \rangle$
- $\langle \text{autonomous system identification: } \Delta y \mapsto \Delta \mathcal{B} \text{ 209c} \rangle$
- $\langle \text{Bias corrected low rank approximation 188a} \rangle$
- $\langle \text{bisection on } \gamma \text{ 99b} \rangle$
- $\langle \text{call c1s1-4 163b} \rangle$
- $\langle \text{call optimization solver 85b} \rangle$
- $\langle \text{check } n < \min(p_i - 1, m, j) \text{ 75b} \rangle$
- $\langle \text{check exit condition 141b} \rangle$
- $\langle \text{Compare h2ss and h2ss_opt 110c} \rangle$
- $\langle \text{Compare ident_asiso and ident_eiv 90b} \rangle$
- $\langle \text{Compare w2h2ss and ident_eiv 116d} \rangle$
- $\langle \text{Complex least squares, solution by (SOL1 } \hat{x}, \text{ SOL1 } \hat{\phi}) \text{ 160a} \rangle$
- $\langle \text{Complex least squares, solution by Algorithm 5 161} \rangle$

- (Complex least squares, solution by generalized eigenvalue decomposition 160b)
 (Complex least squares, solution by generalized singular value decomposition 160d)
 (computation of \bar{u} by solving (SYS AUT) 209d)
 (computation of \bar{u} by solving (SYS DD) 211)
 (Computation time for $c \perp s 1-4$ 162a)
 (compute L , given P 140c)
 (compute P , given L 141a)
 (construct $\psi_{c,ij}$ 189)
 (construct the corrected matrix Ψ_c 190a)
 (cooling process 214b)
 (Curve fitting examples 193c)
 (data compression 137)
 (data driven computation of the impulse response 80a)
 (Data-driven algorithm for sensor speedup 210)
 (default s_0 83e)
 (default initial approximation 85d)
 (default input/output partition 37b)
 (default parameters 192b)
 (default parameters opt 140a)
 (default tolerance tol 38b)
 (default weight matrix 83f)
 (define Δ and Γ 76c)
 (define \mathbf{H}_p , $\mathbf{H}_{f,u}$, and $\mathbf{H}_{f,y}$ 79)
 (define C , D , and n 160c)
 (define l_1 109b)
 (define the gravitational constant 217b)
 (define the Hermite polynomials 188b)
 (dimension of the Hankel matrix 75a)
 (Errors-in-variables finite impulse response identification 120a)
 (Errors-in-variables identification 115b)
 (errors-in-variables identification structure 116a)
 (estimate σ^2 and θ 190c)
 (exact identification: $\hat{w} \mapsto \hat{\mathcal{B}}$ 116b)
 (Example of finite impulse response identification 121c)
 (Example of harmonic retrieval 115a)
 (Example of output error identification 119a)
 (Finite impulse response identification structure 120b)
 (Finite time H_2 model reduction 111)
 (fit data 192e)
 (form $G(R)$ and $h(R)$ 84a)
 (generate data 192c)
 (Hankel matrix constructor 25b)
 (Harmonic retrieval 113)
 (harmonic retrieval structure 114a)
 (impulse response realization \mapsto autonomous system realization 112b)
 (initialization 224)
 (initialize the random number generator 89e)
 (inverse permutation 38a)
 (Low rank approximation 64)
 (low rank approximation \mapsto total least squares solution 229b)
 (Low rank approximation with missing data 136a)
 (matrix approximation 136b)
 (matrix valued trajectory w 26b)
 (misfit minimization 88b)
 (Missing data experiment 1: small sparsity, exact data 144c)
 (Missing data experiment 2: small sparsity, noisy data 145a)
 (Missing data experiment 3: bigger sparsity, noisy data 145b)
 (model augmentation: $\mathcal{B} \mapsto \mathcal{B}_{aut}$ 203b)
 (Monomials constructor 184a)
 (Most powerful unfalsified model in $\mathcal{L}_m^{\alpha,n}$ 80c)
 (nonlinear optimization over R 85c)
 (optional number of (block) columns 25c)
 (order selection 76b)
 (Output error finite impulse response identification 119b)
 (Output error identification 117a)
 (output error identification structure 117b)

- <Output only identification 112a>
 <parameters of the bisection algorithm 99c>
 <plot cls results 163c>
 <plot results 192f>
 <Plot the model 193a>
 <Polynomially structured low rank approximation 191a>
 <preprocessing by finite difference filter $\Delta y := (1 - \sigma^{-1})y$ 209b>
 <Print a figure 25a>
 <print progress information 141c>
 <Recursive least squares 223b>
 <Regularized nuclear norm minimization 97>
 <reshape H and define m, p, T 77>
 <reshape w and define α, T 26d>
 <Sensor speedup examples 214a>
 <set optimization solver and options 85a>
 <Single input single output system identification 88a>
 <singular value decomposition of $\mathcal{H}_{i,j}(\sigma H)$ 76a>
 <solve Problem SLRA 108a>
 <solve the convex relaxation (RLRA') for given γ parameter 98d>
 <solve the least-norm problem 84b>
 <state estimation: $(y, \mathcal{B}_{\text{aut}}) \mapsto x_{\text{aut}} = (x, \bar{u})$ 203c>
 <Structured low rank approximation 85e>
 <Structured low rank approximation misfit 84c>
 <Structured low rank approximation using the nuclear norm 98c>
 <suboptimal approximate single input single output system identification 88c>
 <system identification: $(\mathbf{1}_m s, y) \mapsto \mathcal{B}'$ 206a>
 <temperature-pressure process 216a>
 <Test h2ss_opt 110a>
 <Test harmonic_retrieval 114b>
 <Test ident_eiv 116c>
 <Test ident_fir 121a>
 <Test ident_oe 118>
 <Test ident_siso 89f>
 <Test r2io 43d>
 <Test slra_nn 100a>
 <Test slra_nn on Hankel structured problem 102b>
 <Test slra_nn on small problem with missing data 104e>
 <Test slra_nn on unstructured problem 102a>
 <Test curve fitting 192a>
 <Test missing data 103a>
 <Test missing data 2 142c>
 <Test model reduction 125a>
 <Test model transitions 44>
 <Test sensor speedup 212a>
 <Test sensor speedup methods on measured data 221>
 <Test structured low rank approximation methods on a model reduction problem 126b>
 <Test structured low rank approximation methods on system identification 128>
 <Test system identification 127a>
 <Time-varying Kalman filter for autonomous output error model 222c>
 <Toeplitz matrix constructor 87a>
 <Total least squares 229a>
 <trade-off curve 101b>
 <variable projections method 141d>
 <vector valued trajectory w 26e>
 <Weighted low rank approximation 139>
 <Weighted low rank approximation correction matrix 142b>
 <Weighted total least squares 230>
 <weighting process 218a>

Functions and Scripts Index

Here is a list of the defined functions and where they appear. Underlined entries indicate the place of definition. This index is generated automatically by noweb.

bclra: 188a, 192e
blkhank: 25b, 76a, 79, 88c, 109a,
114a, 116a, 117b, 120b, 125b, 127b,
211
blktoep: 87a, 87b
cls1: 160a, 163b
cls2: 160b, 163b
cls3: 160d, 163b
cls4: 161, 163b
examples_curve_fitting: 194b
examples_sensor_speedup:
214a
h2ss: 76d, 81, 100e, 101b, 109d,
110b, 113, 125c
h2ss_opt: 108b, 110b, 111, 112a
harmonic_retrieval: 113, 114b
ident_aut: 112a, 209c
ident_eiv: 90a, 115b, 116c
ident_fir_eiv: 120a, 121b
ident_fir_oe: 119b, 121b
ident_oe: 117a, 118, 206a
ident_siso: 88a, 89g, 127d
lra: 64, 85d, 88c, 101a, 101b, 191b,
192e, 229a
lra_md: 136a, 140a, 143e
minr: 41, 42b, 43c
misfit_siso: 87b, 88b, 116c, 118,
127c
misfit_slra: 84c, 85c, 85e
mod_red: 111, 125d
monomials: 184a, 190a, 192e
mwlra: 141d, 142a
mwlra2: 141d, 142b
nucnrm: 97, 98d, 104a, 104b
p2r: 40c, 45a
plot_model: 192c, 192f, 193a
print_fig: 25a, 101b, 114b, 126b,
128, 163c, 192f, 213b, 213c, 222b
pslra: 191a, 192e
r2io: 43a, 43d
r2p: 40b, 44, 45a
rio2x: 42a, 43c, 45a, 45b
rls: 211, 223b
slra: 85e, 100e, 108a
slra_nn: 98c, 100d, 101b, 125b,
127b
stepid_as: 209a
stepid_dd: 210, 213a, 213c, 221
stepid_kf: 203a, 213a, 213c, 222a
stepid_si: 205
test_cls: 162a
test_curve_fitting: 192a,
193c, 194a, 194b, 194c, 195a, 195b,
195c
test_h2ss_opt: 110a, 110c

test_harmonic_retrieval: [114b](#), [115a](#)
test_ident_eiv: [116c](#), [116d](#)
test_ident_fir: [121a](#), [121c](#)
test_ident_oe: [118](#), [119a](#)
test_ident_siso: [89f](#), [90b](#)
test_lego: [221](#)
test_missing_data: [103a](#), [104e](#)
test_missing_data2: [142c](#),
 [144c](#), [145a](#), [145b](#)
test_mod_red: [125a](#), [126b](#)
test_sensor: [212a](#), [215a](#), [215b](#),
 [216b](#), [217a](#), [218b](#), [219a](#), [219b](#), [220](#)
test_slra_nn: [100a](#), [102a](#), [102b](#)
test_sysid: [127a](#), [128](#)
th2poly: [192f](#), [193b](#)
tls: [229a](#)
tvkf_oe: [203c](#), [222c](#)
w2h: [80b](#), [81](#)
w2h2ss: [80c](#), [116b](#), [116c](#), [118](#)
wlra: [139](#), [143e](#), [230](#)
wtls: [230](#)
xio2p: [40a](#), [45a](#)
xio2r: [39](#), [45a](#)

Index

Symbols

$2^{\mathcal{U}}$, 53
 $\mathcal{B}_{\text{iso}}(A, B, C, D, \Pi)$, 47
 $\mathcal{B}_{\text{mpum}}(\mathcal{D})$, 54
 $\mathcal{B}_{\text{io}}(X, \Pi)$, 35
 \mathcal{B}^{\perp} , 36
 $\text{c}(\mathcal{B})$, 53
 $\mathcal{C}_j(A, B)$, 51
 $\mathcal{L}_{m,0}^{\alpha}$, 36
 $\mathcal{C}_i(A, C)$, 51
 $n(\mathcal{B})$, 46
 \otimes , 65
 $\mathcal{R}(p, q)$, 11
 $\text{dist}(\mathcal{A}, \mathcal{B})$, 55
 $\mathcal{H}_{i,j}(w)$, 25
 $\ker(R)$, 35
 $\lambda(A)$, 29
 $\|\cdot\|_*$, 96
 $\|\cdot\|_W$, 61
 $(\mathbb{R}^{\mathcal{A}})^{\mathbb{Z}}$, 45
 $\sigma^{\tau} w$, 46
 $\mathcal{T}_T(P)$, 86
 \wedge , 50

A

Adaptive
 beamforming, 11, 28
 filter, 225
Adjusted least squares, 187
Affine model, 147, 240
Affine variety, 181
Algebraic curve, 182
Algebraic fitting, 179
Algorithm
 bisection, 99
 Kung, 76, 77, 128

Levenberg–Marquardt, *see*
 Levenberg–Marquardt
 variable projections, *see* variable
 projections
Alternating projections, 23, 136, 137, 153,
 167, 176, 242
 convergence, 167
Analysis problem, 2, 37
Analytic solution, 62, 68, 152, 241, 242
Annihilator, 36
Antipalindromic, 113
Approximate
 common divisor, 11
 deconvolution, 121
 model, 54
 rank revealing factorization, 76
 realization, 9, 77
Array signal processing, 11
Autocorrelation, 9
Autonomous model, 47

B

Balanced
 approximation, 76
 model reduction, 73
Bias correction, 187
Bilinear constraint, 82
Biotechnology, 199
Bisection, 99

C

Calibration, 201
Causal dependence, 6
Centering, 147
Chemometrics, 13, 165, 176
Cholesky factorization, 82
Circulant matrix, 23, 68
Cissoid, 194

Classification, vi, 164
 Compensated least squares, 187
 Complex valued data, 156
 Complexity–accuracy trade-off, 59
 Computational complexity, 82, 93, 160, 204, 212
 Computer algebra, vi, 28
 Condition number, 29
 Conditioning of numerical problem, 2
 Conic section, 182
 Conic section fitting, 18
 Controllability
 gramian, 76
 matrix, 51
 Controllable system, 48
 Convex optimization, 16, 97
 Convex relaxation, 23, 73, 144
 Convolution, 48
 Coordinate metrology, 179
 Curve fitting, 57
 CVX, 97

D

Data clustering, 28
 Data fusion, 216
 Data modeling
 behavioral paradigm, 1
 classical paradigm, 1
 Data-driven methods, 78, 200
 Dead-beat observer, 203
 Deterministic identification, 10
 Dimensionality reduction, vi
 Diophantine equation, 124
 Direction of arrival, 11, 28
 Distance
 algebraic, 56
 geometric, 55
 horizontal, 3
 orthogonal, 4
 problem, 28
 to uncontrollability, 90, 122
 vertical, 3
 Dynamic
 measurement, 224
 weighing, 199, 217

E

Eckart–Young–Mirsky theorem, 23
 Epipolar constraint, 20
 Errors-in-variables, 29, 57, 115
 ESPRIT, 73
 Exact identification, 9
 Exact model, 54
 Expectation maximization, 24
 Explicit representation, 180

F

Factor analysis, 13
 Feature map, 18
 Fitting
 algebraic, 179
 criterion, 4
 geometric, 20, 179
 Folium of Descartes, 194
 Forgetting factor, 200, 220
 Forward-backward linear prediction, 209
 Fourier transform, 23, 68, 93
 Frobenius norm, 5
 Fundamental matrix, 20

G

Gauss–Markov, 58
 Generalized eigenvalue decomposition, 159
 Generator, 36
 Geometric fitting, 20, 179
 Grassman manifold, 176
 Greatest common divisor, 10

H

Hadamard product, 60
 Halmos, P., 14
 Hankel matrix, 10
 Hankel structured low rank approximation, *see*
 low rank approximation
 Harmonic retrieval, 112
 Hermite polynomials, 188
 Horizontal distance, 3

I

Identifiability, 54
 Identification, 27, 126
 autonomous system, 111
 errors-in-variables, 115
 finite impulse response, 119
 output error, 116
 output only, 111
 Ill-posed problem, 2
 Image mining, 176
 Image representation, 2
 image(P), 35
 Implicialization problem, 197
 Implicit representation, 180
 Infinite Hankel matrix, 8
 Information retrieval, vi
 Input/output partition, 1
 Intercept, 149
 Inverse system, 224

K

Kalman filter, 203, 222
 Kalman smoothing, 87

Kernel methods, 18
 Kernel principal component analysis, 196
 Kernel representation, 2
 Kronecker product, 65
 Kullback–Leibler divergence, 176
 Kung’s algorithm, 76, 77, 128

L

Lagrangian, 150
 Latency, 57, 196
 Latent semantic analysis, 14
 Least squares
 methods, 228
 recursive, 223
 regularized, 1
 robust, 1
 Lego NXT mindstorms, 220
 Level set method, 196
 Levenberg–Marquardt, 105
 Lexicographic ordering, 53, 181
 Limacon of Pascal, 195
 Line fitting, 3, 241
 Linear prediction, 111
 Literate programming, 24
 Loadings, 13
 Localization, 17
 Low rank approximation
 circulant structured, 23
 generalized, 23
 Hankel structured, 9
 nonnegative, 176
 restricted, 23
 Sylvester structured, 11
 two-sided weighted, 65
 weighted, 61

M

Machine learning, vi, 14, 28
 Manifold learning, 196
 Markov chains, 176
 Markov parameter, 50
 Matrix
 Hurwitz, 29
 observability, 209
 Schur, 29
 Maximum likelihood, 67, 70
 Measurement errors, 29
 Metrology, 224
 Microarray data analysis, 18, 28
 MINPACK, 105
 Misfit, 57, 196
 Missing data, 15, 103, 135
 Mixed least squares total least squares, 212
 Model

 approximate, 54
 autonomous, 47
 class, 53
 exact, 54
 finite dimensional, 46
 finite impulse response, 119
 invariants, 36
 linear dynamic, 45
 linear static, 35
 complexity, 45
 linear time-invariant
 complexity, 51
 most powerful unfalsified, 54
 reduction, 8, 110, 125
 representation, 21
 shift-invariant, 46
 static
 affine, 147
 stochastic, 9
 structure, 18
 sum-of-damped exponentials, 111, 197,
 208
 trajectory, 45
 Model-free, 200
 Most powerful unfalsified model, 54
 MovieLens data set, 146
 Multidimensional scaling, 17
 Multivariate calibration, 13
 MUSIC, 73

N

Norm
 Frobenius, 5
 nuclear, 16, 96
 unitarily invariant, 62
 weighted, 59, 61
 noweb, 24
 Numerical rank, 99, 102, 164

O

Observability
 gramian, 76
 matrix, 51
 Occam’s razor, 45
 Occlusions, 135
 Optimization Toolbox, 141, 190
 Order selection, 126, 206
 Orthogonal regression, 29

P

Palindromic, 113
 Pareto optimal solutions, 60
 Persistency of excitation, 10, 210
 Pole placement, 122
 Polynomial eigenvalue problem, 190

- Positive rank, 176
- Power set, 53, 180
- Pre-processing, 148
- Prediction error, 117
- Principal component analysis, vi, 28, 70
 - kernel, 28
- Principal curves, 196
- Procrustes problem, 230
- Projection, 55
- Prony's method, 209
- Proper orthogonal decomposition, 128
- Pseudo spectra, 29
- Psychometrics, 13

- R**
- Rank
 - estimation, 163
 - minimization, 16, 59, 60, 73
 - numerical, 75
 - revealing factorization, 8
- Rank one, 12, 241, 242
- Realizability, 50
- Realization
 - approximate, 9, 108
 - Ho-Kalman's algorithm, 128
 - Kung's algorithm, 128
 - theory, 49–51
- Recommender system, 15, 146
- Recursive least squares, 223, 227
- Reflection, 56
- Regression, 58, 180
- Regression model, 58
- Regularization, 1, 6, 227
- Representation
 - convolution, 48
 - explicit, 180
 - image, 2
 - minimal, 36
 - implicit, 19, 180
 - kernel, 2
 - minimal, 36
 - problem, 9
- Reproducible research, 24
- Residual, 56
- Riccati equation, 29, 57
- Riccati recursion, 87
- Rigid transformation, 17, 56, 187
- Robust least squares, 1
- Rotation, 56

- S**
- Schur algorithm, 87, 92
- Semidefinite optimization, 96
- Separable least squares, 171
- Shape from motion, 28
- Shift
 - operator, 46
 - structure, 74
- Singular problem, 135
- Singular value decompositions
 - generalized, 23
 - restricted, 23
- Singular value thresholding, 144, 176
- SLICOT library, 87, 104
- Smoothing, 57
- Stability radius, 29
- Stereo vision, 20
- Stochastic system, 9
- Stopping criteria, 175
- Structure
 - bilinear, 21
 - polynomial, 179
 - quadratic, 20
 - shift, 74
- Structured linear algebra, 29
- Structured total least norm, 231
- Subspace
 - identification, 7
 - methods, 23, 73
- Sum-of-damped exponentials, 111, 197, 200, 208
- Sum-of-exponentials modeling, 112
- Sylvester matrix, 11
- System
 - lag, 46
 - order, 46
- System identification, *see* identification
 - approximate, 10
- System realization, *see* realization
 - stochastic, 9

- T**
- Time-varying system, 219
- Total least squares, 4, 240
 - element-wise weighted, 230
 - generalized, 229
 - regularized, 230
 - restricted, 229
 - structured, 230
 - weighted, 230
 - with exact columns, 212
- Trade-off curve, 101
- Trajectory, 45
- Translation, 56

- V**
- Vandermonde matrix, 197
- Variable projections, 23, 81, 137, 154

- Y**
- Yule-Walker's method, 209