

Model Order Reduction. Volume 1: System- and Data-Driven Methods and Algorithms

Original

Model Order Reduction. Volume 1: System- and Data-Driven Methods and Algorithms / Peter, Benner; GRIVET TALOCIA, Stefano; Alfio, Quarteroni; Gianluigi, Rozza; Wil, Schilders; Luis Miguel Silveira,. - STAMPA. - 10:(2021), pp. 1-388. [10.1515/9783110498967]

Availability:

This version is available at: 11583/2935792 since: 2021-11-05T17:10:32Z

Publisher:

De Gruyter

Published

DOI:10.1515/9783110498967

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

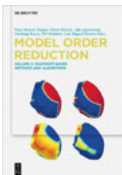
Publisher copyright

(Article begins on next page)

Peter Benner, Stefano Grivet-Talocia, Alfio Quarteroni, Gianluigi Rozza, Wil Schilders,
Luís Miguel Silveira (Eds.)

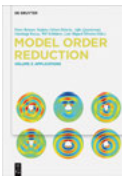
Model Order Reduction

Also of Interest



Model Order Reduction. Volume 2: Snapshot-Based Methods and Algorithms

Peter Benner, Stefano Grivet-Talocia, Alfio Quarteroni, Gianluigi Rozza, Wil Schilders, Luís Miguel Silveira (Eds.), 2020
ISBN 978-3-11-067140-7, e-ISBN (PDF) 978-3-11-067149-0,
e-ISBN (EPUB) 978-3-11-067150-6



Model Order Reduction. Volume 3: Applications

Peter Benner, Stefano Grivet-Talocia, Alfio Quarteroni, Gianluigi Rozza, Wil Schilders, Luís Miguel Silveira (Eds.), 2020
ISBN 978-3-11-050044-8, e-ISBN (PDF) 978-3-11-049900-1,
e-ISBN (EPUB) 978-3-11-049775-5



Richardson Extrapolation. Practical Aspects and Applications

Zahari Zlatev, Ivan Dimov, István Faragó, Ágnes Havasi, 2017
ISBN 978-3-11-051649-4, e-ISBN (PDF) 978-3-11-053300-2,
e-ISBN (EPUB) 978-3-11-053198-5



Multivariate Algorithms and Information-Based Complexity

Fred J. Hickernell, Peter Kritzer (Eds.), 2020
ISBN 978-3-11-063311-5, e-ISBN (PDF) 978-3-11-063546-1,
e-ISBN (EPUB) 978-3-11-063315-3



Fractional Order Crowd Dynamics. Cyber-Human System Modeling and Control

Kecai Cao, YangQuan Chen, 2018
ISBN 978-3-11-047281-3, e-ISBN (PDF) 978-3-11-047398-8,
e-ISBN (EPUB) 978-3-11-047283-7

Model Order Reduction

Volume 1: System- and Data-Driven Methods and
Algorithms

Edited by

Peter Benner, Stefano Grivet-Talocia, Alfio Quarteroni,
Gianluigi Rozza, Wil Schilders, and Luís Miguel Silveira

DE GRUYTER

Editors

Prof. Dr. Peter Benner
Max Planck Institute for Dynamics of Complex
Technical Systems
Sandtorstr. 1
39106 Magdeburg
Germany
benner@mpi-magdeburg.mpg.de

Prof. Dr. Gianluigi Rozza
Scuola Internazionale Superiore di Studi
Avanzati - SISSA
Via Bonomea 265
34136 Trieste
Italy
gianluigi.rozza@sissa.it

Prof. Dr. Stefano Grivet-Talocia
Politecnico di Torino
Dipartimento di Elettronica
Corso Duca degli Abruzzi 24
10129 Turin
Italy
stefano.grivet@polito.it

Prof. Dr. Wil Schilders
Technische Universiteit Eindhoven
Faculteit Mathematik
Postbus 513
5600 MB Eindhoven
The Netherlands
w.h.a.schilders@tue.nl

Prof. Alfio Quarteroni
Ecole Polytechnique Fédérale de Lausanne
(EPFL) and Politecnico di Milano
Dipartimento di Matematica
Piazza Leonardo da Vinci 32
20133 Milan
Italy
alfio.quarteroni@polimi.it

Prof. Dr. Luís Miguel Silveira
INESC ID Lisboa
IST Técnico Lisboa
Universidade de Lisboa
Rua Alves Redol 9
1000-029 Lisbon
Portugal
lms@inesc.pt

ISBN 978-3-11-050043-1
e-ISBN (PDF) 978-3-11-049896-7
e-ISBN (EPUB) 978-3-11-049771-7
DOI <https://doi.org/10.1515/9783110498967>



This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. For details go to <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

Library of Congress Control Number: 2021939228

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at <http://dnb.dnb.de>.

© 2021 with the authors, editing © 2021 Peter Benner, Stefano Grivet-Talocia, Alfio Quarteroni, Gianluigi Rozza, Wil Schilders, Luís Miguel Silveira, published by Walter de Gruyter GmbH, Berlin/Boston. The book is published open access at www.degruyter.com.

Cover image: Dr. Yao Yue

Typesetting: VTeX UAB, Lithuania

Printing and binding: CPI books GmbH, Leck

www.degruyter.com

Preface to the first volume of *Model Order Reduction*

This is the first of the three-volume set *Model Order Reduction* intended to be used as a handbook in partial fulfilment of the goals of the COST Action EU-MORNET. The first two volumes deal with methods and algorithms, while the third and final volume is devoted to specific applications. Before discussing the contents of Volume 1 (for the contents of Volumes 2 and 3, see the respective editorials there), we would like to explain the background of this project.

EU-MORNET: Model Order Reduction in Europe

European researchers have realized the importance of Model Order Reduction (MOR) and reduced-order modeling already since the 1990s, in the scientific computing and computational engineering communities as well as in the area of systems and control. Since then, the interest has grown steadily, with many workshops and conferences having been organized, and several MOR research groups emerging. In the early 2000s, the first workshops were organized that brought researchers from these various areas together. This includes the 2003 workshop on “Dimension Reduction of Large-Scale Systems” at the Mathematical Research Center Oberwolfach and the 2005 workshop “Model Order Reduction – Coupled Problems and Optimization” at the Lorentz Center in Leiden. Both inspired the publication of tutorial-style collections, leading to two of the first books on MOR.^{1,2} At the same time, the first research monograph fully dedicated to MOR appeared.³ In addition, comprehensive European projects like CODESTAR (“Compact modelling of on-chip passive structures at high frequencies”, 2002–2004), CHAMELEON-RF (“Comprehensive High-Accuracy Modelling of Electromagnetic Effects in Complete Nanoscale RF Blocks”, 2004–2006), and O-MOORE-NICE! (“Operational Model Order REduction for Nanoscale IC Electronics”, 2007–2010) made abundant use of MOR. The European Research Training Network COMSON (“Coupled Multiscale Simulation and Optimization in Nanoelectronics”, 2007–2009) also had a major task on MOR, and organized an autumn school on the Dutch island of Terschelling. It is still remembered by many participants, due to the nice food and luxurious accommodation, but also because many leading MOR researchers from all over the world were present. During this autumn school, there

1 Peter Benner, Volker Mehrmann, and Danny C. Sorensen (Eds.), *Dimension Reduction of Large-Scale Systems*, Lecture Notes in Computational Science and Engineering, Vol. 45, Springer-Verlag, Berlin/Heidelberg, 2005.

2 Wilhelmus H. Schilders, Henk van der Vorst, and Joost Rommes (Eds.), *Model Order Reduction: Theory, Research Aspects and Applications*, Mathematics in Industry, Vol. 13, Springer-Verlag, Berlin/Heidelberg, 2008.

3 Athanasios C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, SIAM, Philadelphia, 2005.

was a first discussion on starting a European network on MOR, but due to the lack of funding opportunities, there was no immediate follow-up.

In 2013, Peter Benner, chair of one of the MOR centers in Europe (the Max Planck Institute for Dynamics of Complex Technical Systems in Magdeburg), together with Albert Cohen (Paris), Mario Ohlberger (Münster), and Karen Willcox (then at MIT) organized a workshop in the Luminy mathematics research centre CiRM, located beautifully off the coast in the south of France, and this turned out to be the ideal setting for the preparation of a so-called COST Action on MOR. The lectures during the day and the very pleasant atmosphere in the evenings put us in the right mood for writing. The aim of the proposal was to “bring together all major groups in Europe working on a range of model reduction strategies with applications in many of the COST domains”. The proposal survived the first round, and was admitted to the second round, which meant going to Brussels for an interview with a very broad and general committee. The overall chances of success were approximately 4 %, but we succeeded and hence EU-MORNET was born. The first management committee meeting took place in Brussels in April 2014, and since then many activities have been organized and undertaken. Highlights were the MoRePaS conferences in Trieste and Nantes, the Durham workshop in August 2017, organized jointly with the London Mathematical Society, and MODRED held 2017 in Odense. The network was growing constantly, and when the funded period of EU-MORNET ended in April 2018, more than 300 researchers had joined the network. We hope to sustain this network, e. g., via its webpage eu-mor.net, as coordination of activities has turned out to be very fruitful, it has put MOR in the spotlights, and we observe that the interest in MOR is only growing: many European projects make use of it, or emphasize its importance like the recently ended ECSEL project Delphi4LED. A glimpse at some of the various applications encompassing MOR in its computational workflows is provided in Volume 3 of this handbook project. We are very grateful to the COST Organization for supporting this initiative, thereby bringing MOR in Europe to the next level. This handbook also serves as the ultimate dissemination effort of EU-MORNET and will hopefully help generations of new researchers and practitioners to get a gentle introduction into the field and to find inspiration for their own development and research work.

Introduction to Volume 1

This first volume starts with an introductory chapter to MOR in general as a very broad field of research, encompassing multiple techniques with applications in a wide variety of fields. This chapter serves two main purposes. On the one hand, it provides an introduction to the handbook project itself, helping the reader navigate through the three volumes, explaining their organization, providing pointers into the various chapters where specific methods are presented or where particular applications are further explored. Additionally, this first chapter also serves as a conduit to introduce

concepts and notation used throughout the various chapters and volumes, in an attempt to support, simplify and enrich the reader's experience when probing the information provided in the three volumes of "Model Order Reduction".

After this initial, introductory chapter, all chapters of this first volume mostly focus on the concept of MOR applied in a system-theoretical context. The common principle among methods and algorithms in this setting is the basic assumption that there is an underlying system description whose behavior can be determined from the knowledge of the dynamics of a set of state variables. Specific developments both in theory and applications, including deployment in commercial CAD tools, took place over the years in specific settings and disciplines, sometimes using different language and notation. However, all such methods share a common framework, which we attempted to capture in this book.

The second chapter in this volume, by T. Breiten and T. Stykel, is devoted to methods associated with the concept of energy of a system and with the problem of how to represent it in balanced coordinates. This enables discarding the least relevant states from an input-output perspective. The resulting truncated system has several very interesting properties, which are discussed in the context of linear and nonlinear reduction.

The third chapter by L. Feng and P. Benner delves into the realm of moment-matching methods (also known as Padé-type approximations, relating to rational interpolation) as a metric for reduction, and details methods based on projection techniques for compressing linear, nonlinear and parametric systems.

The next chapter of P. Tiso et al. is devoted to modal truncation applied to linear and nonlinear systems. This chapter discusses techniques based on analysis of the system dynamics, in particular the observation of its eigenmodes and consequent truncation leading to reduced-order models.

Enforcing specific desirable or required system properties after reduction is the target of the next chapter, by S. Grivet-Talocia and L. M. Silveira, which is devoted to post-processing techniques. In particular, the most prominent techniques for enforcing passivity of linear systems via perturbation approaches are introduced and discussed.

The following chapter serves as an interesting bridge between moment-matching methods described as rational interpolation, to data-driven interpolation techniques connecting to approaches that start from measurements of the system. This chapter, by D. Karachalios, I. V. Gosea and A. C. Antoulas, introduces the Loewner framework for system reduction and connects to moment matching, interpolation and projection.

The seventh chapter, by R. Zimmermann, continues the trend of discussing interpolation methods, but it introduces manifold interpolation as a supporting tool in the reduction framework of parameterized systems.

The final three chapters are entirely dedicated to exploring model reduction techniques fueled by data obtained from system behavior.

The eighth chapter, by P. Triverio, discusses Vector Fitting, a data-driven algorithm where samples or measurements of the system response are used to construct a reduced representation.

The ninth chapter, by G. Santin and B. Haasdonk, stays in the realm of data-driven reduction and introduces kernel methods as surrogate system models. It introduces a series of methods where the system representation is unknown or eschewed and a reduced representation is constructed or estimated from the information garnered by sampling the system or its outputs.

Last but not the least, the tenth and final chapter, by J. Kleijnen, presents Kriging techniques: a set of data-driven interpolation techniques for generating a reduced model through kernel regression assuming an underlying Gaussian distribution.

At this point, we would like to thank also all the contributing authors who brought this project to life, the numerous anonymous reviewers who ensured the quality of the 30 chapters of the three volumes of the Model Order Reduction handbook series, and last but not least Harshit Bansal, who helped with producing the index for every of the three volumes. Our gratitude also goes to the De Gruyter staff, and in particular to Nadja Schedensack, for accompanying this project constructively over more than four years, with unprecedented patience.

*Peter Benner, Stefano Grivet-Talocia, Alfio Quarteroni, Gianluigi Rozza,
Wil Schilders, Luís Miguel Silveira*

Magdeburg, Germany
Torino, Milano, Trieste, Italy
Eindhoven, The Netherlands
Lisbon, Portugal

December 2020

Contents

Preface to the first volume of *Model Order Reduction* — V

Peter Benner, Stefano Grivet-Talocia, Alfio Quarteroni, Gianluigi Rozza, Wil Schilders,
and Luís Miguel Silveira

1 Model order reduction: basic concepts and notation — 1

Tobias Breiten and Tatjana Stykel

2 Balancing-related model reduction methods — 15

Peter Benner and Lihong Feng

3 Model order reduction based on moment-matching — 57

Paolo Tiso, Morteza Karamooz Mahdiabadi, and Jacopo Marconi

4 Modal methods for reduced order modeling — 97

Stefano Grivet-Talocia and Luis Miguel Silveira

5 Post-processing methods for passivity enforcement — 139

Dimitrios S. Karachalios, Ion Victor Gosea, and Athanasios C. Antoulas

6 The Loewner framework for system identification and reduction — 181

Ralf Zimmermann

7 Manifold interpolation — 229

Piero Triverio

8 Vector fitting — 275

Gabriele Santin and Bernard Haasdonk

9 Kernel methods for surrogate modeling — 311

Jack P. C. Kleijnen

10 Kriging: methods and applications — 355

Index — 371

Peter Benner, Stefano Grivet-Talocia, Alfio Quarteroni,
Gianluigi Rozza, Wil Schilders, and Luís Miguel Silveira

1 Model order reduction: basic concepts and notation

Abstract: This is the first chapter of a three-volume series dedicated to theory and application of Model Order Reduction (MOR). We motivate and introduce the basic concepts and notation, with reference to the two main cultural approaches to MOR: the system-theoretic approach employing state-space models and transfer function concepts (Volume 1), and the numerical analysis approach as applied to partial differential operators (Volume 2), for which projection and approximation in suitable function spaces provide a rich set of tools for MOR. These two approaches are complementary but share the main objective of simplifying numerical computations while retaining accuracy. Despite the sometimes different adopted language and notation, they also share the main ideas and key concepts, which are briefly summarized in this chapter. The material is presented so that all chapters in this three-volume series are put into context, by highlighting the specific problems that they address. An overview of all MOR applications in Volume 3 is also provided.

Keywords: model order reduction, (Petrov–)Galerkin projection, snapshots, parametric operator equation, transfer function

MSC 2010: 35B30, 37M99, 41A05, 65K99, 93A15, 93C05

1.1 Overview

The ever-increasing demand for realistic simulations of complex products and processes places a heavy burden on the shoulders of mathematicians and, more generally, researchers and engineers working in the area of computational science and engineering (CSE). Realistic simulations imply that the errors of the virtual models should be small, and that different aspects of the product or process must be taken into account, resulting in complex coupled simulations.

Peter Benner, Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, Germany

Stefano Grivet-Talocia, Dipartimento di Elettronica, Politecnico di Torino, Turin, Italy

Alfio Quarteroni, Politecnico di Milano, Milan, Italy; and EPFL Lausanne, Lausanne, Switzerland

Gianluigi Rozza, SISSA, Trieste, Italy

Wil Schilders, TU Eindhoven, Eindhoven, The Netherlands

Luís Miguel Silveira, INESC ID/IST Técnico Lisboa, Universidade de Lisboa, Lisbon, Portugal

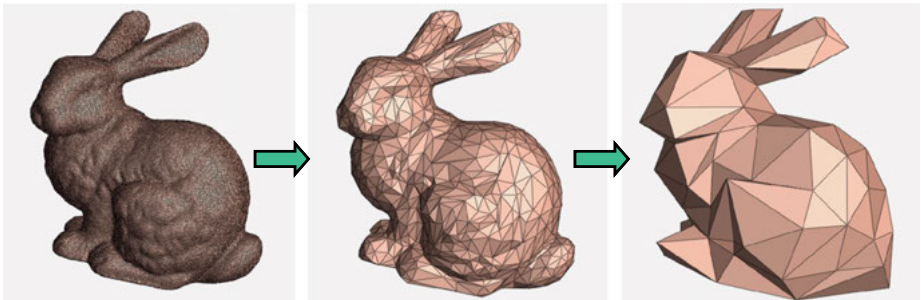
Open Access. © 2021 Peter Benner et al., published by De Gruyter.  This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

<https://doi.org/10.1515/9783110498967-001>

Often, there is a lot of superfluous detail in these simulations that is not needed to provide accurate results. With the current advances in new computer architectures, viz. the availability of many processors, one might be tempted to just use the abundant computational resources. However, this could lead to enormous energy consumption for the simulations, which should be avoided if possible. Besides, it could still lead to very lengthy and time-consuming simulations. Hence, it seems wise to use methods that can reduce the size of such huge problems, and which are able to get rid of the superfluous and unnecessary detail, still guaranteeing the accuracy of solutions.

An example is provided by the co-simulation of an electronic circuit with the interconnect structure that is mounted on top of the circuit to provide all desired connections. The metallic interconnect structure causes electromagnetic effects that may influence the behavior of the underlying circuit. There is no need, however, to solve the Maxwell equations for this complex three-dimensional structure in full detail, only the most dominant effects causing delays need to be included. This is where model order reduction (MOR) comes into play; MOR methods are able to extract the dominant behavior by reducing the size of the system to be solved.

To explain what MOR is, we often use the following picture:



If we showed the picture on the right, everyone would recognize that this is a rabbit. Hence, we do not need the detailed representation in the left picture to describe the animal. Maybe we would need a slightly more detailed description as shown in the middle picture, depending on the demands.¹ MOR works in the same way: the original problem is reduced, the representation of the solution is given with far less variables, and the hope is that this is sufficient to guarantee an accurate solution. If more accuracy is needed, clearly the problem size should be reduced to a lesser extent.

¹ Disclaimer: though our picture might indicate this, note that simply using a coarser mesh in a discretization of a continuous model is not a competitive MOR technique in general!

By now, MOR is a very active and relatively mature field of research. Much progress has been made in the past 40 years, and in many different directions. The seminal papers by Feldmann and Freund (1994/95) [9, 10], simultaneously with [12], sparked many development in the area of Krylov-type methods, strongly related to the field of numerical linear algebra. Mimetic elements were considered, leading to passivity-preserving and structure-preserving methods, necessary to retain vital properties of the original system of equations in electronics and electromagnetics. First textbooks appeared with a special focus on applications in this area, [11] even before the above papers, and after the field had matured, new textbooks [13, 22] and several collections [3, 5] were published.

Within the systems and control area, one concentrates mainly on balancing techniques, involving the solution of Lyapunov equations. The main ideas center on preserving those states in a dynamical system that can be reached with the least amount of energy, and on the other hand, those states that provide the most information when observed. In balanced coordinates, both sets of states coincide. Here, the seminal paper of [16] has to be mentioned which rendered these ideas computationally feasible. A first textbook in this area was published in 2001 by Obinata and Anderson [17].

Also, the need for MOR was already discussed in the 1960s in mechanical engineering, from which the technique of modal truncation emerged, in combination with substructuring and with further developments like component mode synthesis. These are nowadays standard techniques, found in many variants in structural analysis software, and covered by many textbooks in numerical mechanics. A comprehensive textbook focusing on this area is [18].

First textbooks and collections of tutorials that made connections between the MOR techniques developed mainly in the above-mentioned application areas started appearing in the mid-2000s, including the fundamental textbook [1] by Antoulas in 2005, and the edited volumes [6, 21].

Later, researchers started to consider parametric model order reduction, especially within the area of reduced basis methods, focusing on the fast solution of parametric partial differential equations (PDEs) [14, 19, 20]. A related, but somewhat different approach to parametric PDEs was developed in the framework of the proper generalized decomposition [8]. One can also find a collection of articles on MOR for parametric problems in [7].

Methods for nonlinear problems were also considered, important developments being the empirical interpolation methods and other so-called hyperreduction techniques. But also methods like proper orthogonal decomposition (POD), where snapshots of the solution of a nonlinear problem are used to create a basis for solutions, became popular for nonlinear problems. Basic concepts of these approaches, including also the MOR techniques already mentioned above, also with some historical perspectives, can be found in the collection of tutorials [4].

The demand for more realistic simulations led to the development of MOR methods for interconnected and coupled systems. Extensions to descriptor systems, or alternatively differential-algebraic systems, led to the creation of index-preserving methods and to the development of an interpolatory projection framework for model reduction of descriptor systems. An entirely different approach is provided by data-driven methods, in which the Loewner framework plays an important role; see the recent textbook on the interpolatory framework including the Loewner framework for more details [2].

Most recently, focus also turned to data-driven and non-intrusive MOR methods, requiring no or only incomplete knowledge of the physical model, but merely relying on tools or software to produce relevant data from which a model description can be inferred. One prominent technique in this area is dynamic mode decomposition [15], with many new methods emerging even more recently, often making connections to techniques from machine and deep learning.

The three volumes constituting this handbook of *Model Order Reduction* discuss many of the aforementioned developments and methods. This first volume contains theoretical expositions of system-theoretic, interpolatory, and data-driven methods and algorithms. The second volume treats snapshot-based methods and algorithms for parametric PDEs. The mathematical strategy behind these methods relies on Galerkin projection on finite-dimensional subspaces generated by snapshot solutions corresponding to a specific choice of parameters. The third volume contains a large variety of applications of MOR. Originally, the fields of mechanical engineering, automation and control, as well as the electronics industry were the main driving forces for the developments of MOR methods, but in recent years, MOR has been introduced in many other fields (not all covered in Volume 3, though), like chemical and biomedical engineering, life sciences, geosciences, finance, fluid mechanics and aerodynamics, to name a few. Moreover, Volume 3 also provides a chapter surveying the landscape of existing MOR software.

1.2 A quick tour

MOR is a multidisciplinary topic, which has been developed over the last decades by mathematicians, scientists and engineers from widely different communities. Although the main ideas in MOR can be classified in a relatively small set of fundamental problem statements and related reduction approaches, these ideas have been developed by different communities with different languages, notation, and scope. One of the purposes of the *Model Order Reduction* handbook project is in fact to provide a comprehensive overview of MOR approaches, hopefully forming bridges that cross different disciplines.

Several classifications can be attempted in the world of MOR. Probably the most natural high-level classification distinguishes between the two main cultural approaches of system theory on one side, and numerical analysis as applied to solving PDEs on the other. Other classifications may be considered, for instance based on the various classes of reduction approaches, which may be model-driven or data-driven, optimal or heuristic, deterministic or stochastic, or alternatively on the type of the system being addressed, which can be linear or nonlinear, uniquely defined or parameterized by some geometrical or material variable, deterministic or stochastic, finite- or infinite-dimensional. The specific methods that apply in each of these cases will be discussed in detail in the various chapters of this three-volume series. In this initial chapter we mainly distinguish the two major cultural approaches to MOR, for which reduction methods, notation and language are sometimes quite different in the existing literature.

System-theoretic approaches usually deal with a system under investigation described as a large-scale set of Ordinary Differential Equations (ODEs) or Differential-Algebraic Equations (DAEs), whose dynamics is expressed in terms of a set of state variables. The main objective is to derive some compact Reduced-Order Model (ROM) with the same structure, characterized by a significantly smaller number of states, and whose response approximates the true system response according to well-defined criteria. Very often the ROM represents a component of a larger system that is impossible or impractical to solve in its full-size formulation. In this setting, reduction is required in order to replace the original large-scale description of individual components with accurate and robust ROMs, so that a global system-level numerical simulation becomes feasible.

A second major approach to MOR addresses fast numerical solution of PDEs. In this setting, the field problem of a PDE is taken as the starting point. In the snapshot-based methods, the full-order variational form is often retained by the MOR process. This allows projection-based methods to utilize the parametric operator equations and define a reduced-order operator of the same parametric dependency. Since the starting point is the PDE form, a discretization in space and time is required, leading to a large-scale discretized model. Various methods exist to control the approximation error, often balancing rigorosity with computability.

We see that the above two approaches share their main objective of speeding up numerical computations with control over approximation errors, although the starting points are different. We should, however, consider that the two approaches practically coincide once a field problem described in terms of PDEs is discretized in its space variables in terms of suitable coefficients, which basically play the same role of the state variables in system-theoretical approaches.

We address the two approaches in Sections 1.2.1 and 1.2.2, by introducing basic notation and concepts that will be used extensively throughout Volumes 1 and 2 of this book series. Section 1.2.3 provides a glimpse at the MOR applications that are extensively discussed in Volume 3.

1.2.1 The system-theoretic approach

System-theoretic approaches consider models whose dynamics is expressed in terms of internal *state variables*, which in the finite-dimensional setting are denoted as $x(t) \in \mathcal{X} \subset \mathbb{R}^n$. These states evolve with time $t \in [t_0, T]$ according to dynamic equations which are driven by some *inputs* or *control signals* $u(t) \in \mathcal{U} \subset \mathbb{R}^m$, whereas the *quantities of interest* or *outputs* are $y(t) \in \mathcal{Y} \subset \mathbb{R}^q$, usually with $q \ll n$. Denoting the “true” system as \mathcal{S} , the MOR objective is to obtain an approximate system $\widehat{\mathcal{S}}$ with a small number $r \ll n$ of internal states $\widehat{x}(t) \in \widehat{\mathcal{X}} \subset \mathbb{R}^r$. Reduction is conducted by enforcing appropriate approximation conditions such that, given input signals $u(t)$, the ROM $\widehat{\mathcal{S}}$ provides an output $\widehat{y}(t)$ that is “close” in some sense to the corresponding output $y(t)$ of the original system \mathcal{S} .

1.2.1.1 Standard system descriptions: the LTI case

The simplest system description assumes Linearity and Time-Invariance (LTI) and is provided by a set of ODEs in *state-space* form

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t), & x(t_0) &= x_0, \\ y(t) &= Cx(t) + Du(t),\end{aligned}\tag{1.1}$$

where $\dot{x}(t)$ denotes the time derivative of $x(t)$, $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{q \times n}$, $D \in \mathbb{R}^{q \times m}$ are constant matrices, and x_0 is a prescribed initial condition. A more general formulation of LTI system dynamics can be expressed in *descriptor* form,

$$\begin{aligned}E\dot{x}(t) &= Ax(t) + Bu(t), & x(t_0) &= x_0, \\ y(t) &= Cx(t) + Du(t),\end{aligned}\tag{1.2}$$

where an additional and possibly singular matrix $E \in \mathbb{R}^{n \times n}$ enters the state equations. Casting (1.2) in the Laplace domain and assuming vanishing initial conditions, $x_0 = 0$, leads to

$$Y(s) = H(s)U(s), \quad H(s) = C(sE - A)^{-1}B + D,\tag{1.3}$$

where $H(s)$ is the transfer function of the system and $s \in \mathbb{C}$ is the Laplace variable. Well-posedness of (1.3) requires that $\det(sE - A) \neq 0$ for some s , i. e., that the pencil (A, E) is *regular*. In most cases also an (asymptotic) *stability* requirement is established, so that all finite eigenvalues of the pencil (A, E) have a (strictly) negative real part.

This system description forms the basis of most of the following chapters in this volume.

1.2.1.2 Approximation criteria

Some common approximation criteria that are appropriate for LTI systems are listed now:

- The quantities of interest of both full-scale system \mathcal{S} and reduced system $\widehat{\mathcal{S}}$ are the outputs $y(t)$ and $\widehat{y}(t)$, respectively. Therefore, it is natural to bound the output error defined as $\|\widehat{y} - y\|_{\mathcal{L}}$ within a suitable function space \mathcal{L} , with the natural choice being the Hilbert space of square integrable signals $L_2(t_0, T)$, with

$$\|y\|_{L_2(t_0, T)}^2 = \int_{t_0}^T \|y(t)\|_2^2 dt. \quad (1.4)$$

- An alternative is to control the error of the ROM transfer function $\widehat{H}(s)$ by minimizing $\|\widehat{H} - H\|_{\mathcal{H}}$, where \mathcal{H} is an appropriate function space. Common choices are the Hardy spaces \mathcal{H}_2 and \mathcal{H}_∞ , which are adequate under asymptotic stability assumptions, for which

$$\|H\|_{\mathcal{H}_2}^2 = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \|H(j\omega)\|_F^2 d\omega, \quad \|H\|_{\mathcal{H}_\infty} = \sup_{s \in \mathbb{C}_+} \|H(s)\|_2, \quad (1.5)$$

where F denotes the Frobenius norm and $j = \sqrt{-1}$. We refer to Chapter 2 in this volume for more precise definitions and for an introduction of the main system-theoretic properties that are relevant for error control in MOR.

- Data-driven approaches aim at enforcing suitable interpolation or approximation conditions starting from available samples of the original transfer function $H_k = H(s_k)$ at a set of complex frequencies s_k for $k = 1, \dots, \bar{k}$. Interpolation methods (see Chapter 6, where the Loewner framework is introduced and discussed) enforce

$$\widehat{H}(s_k) = H_k, \quad \forall k = 1, \dots, \bar{k}, \quad (1.6)$$

possibly extending this exact matching also to higher derivatives

$$\left. \frac{d^v \widehat{H}}{ds^v} \right|_{s_k} = H_k^{(v)}, \quad \forall v = 0, \dots, \bar{v}_k, \quad \forall k = 1, \dots, \bar{k}, \quad (1.7)$$

giving rise to so-called *moment-matching* methods (see Chapter 3). In some cases, the point and moment matching is performed at adaptively selected frequencies $s_k \in \mathbb{C}$; see, e. g., the IRKA algorithm in Chapter 3. Moments can also be matched implicitly through projection of the original system onto suitably-defined Krylov subspaces, also discussed in Chapter 3.

- A relaxed version of the above matching conditions involves minimization of the least squares error,

$$\sum_{k=1}^{\bar{k}} \|\widehat{H}(s_k) - H_k\|_F^2. \quad (1.8)$$

Curve fitting approaches, including the Vector Fitting (VF) methods (see Chapter 8) fall into this class. When data H_k come from measurements, only purely imaginary frequencies $s_k = j\omega_k$ are available and used.

- A fundamental class of system-theoretic approaches for MOR are based on *truncation* of state-space or descriptor systems, where those state variables that are poorly coupled to the inputs or which provide negligible contribution to the outputs are discarded. Balanced truncation methods (see Chapter 2) and modal methods (Chapter 4) belong to this class.
- Some applications require additional constraints to be enforced during reduction. A notable case is enforcements of passivity and of dissipativity, which are appropriate for systems that are unable to generate energy on their own. Dissipativity conditions for state-space systems are reviewed in Chapters 5 and 2, together with appropriate methods for their enforcement, either as a feature of the MOR scheme or as a postprocessing.

1.2.1.3 Parameterized LTI systems

An additional layer of complexity is introduced by allowing the system S to be *parameterized* by some deterministic and/or stochastic variables $\mu \in \mathcal{P} \subset \mathbb{R}^p$. Assuming that the input signals u are not parameter-dependent, we can write (1.2) in the parameterized form

$$\begin{aligned} E(\mu)\dot{x}(t, \mu) &= A(\mu)x(t, \mu) + B(\mu)u(t), & x(t_0, \mu) &= x_0(\mu), \\ y(t, \mu) &= C(\mu)x(t, \mu) + D(\mu)u(t), \end{aligned} \quad (1.9)$$

with the corresponding transfer function

$$H(s, \mu) = C(\mu)(sE(\mu) - A(\mu))^{-1}B(\mu) + D(\mu). \quad (1.10)$$

In this parameterized setting, one is usually interested in preserving a closed-form parameterization also in the ROM, so that the corresponding transfer function must match (1.10) not only throughout the frequency band of interest, but also throughout the parameter space. Chapter 3 provides an overview of moment-matching parameterized MOR (PMOR) in the case of affine dependence of $E(\mu)$ and $A(\mu)$ on the parameters. The so-called reduced basis methods discussed in Chapter 4 of Volume 2 would provide the counterpart of PMOR in the PDE reduction setting, which is extensively treated in all chapters of Volume 2.

1.2.1.4 Nonlinear systems

Generalization to nonlinear systems is also possible, although effectiveness of MOR strongly depends on the class of systems being considered. Several results are avail-

able for systems that can be cast in the form

$$\begin{aligned}\dot{x}(t) &= f(x(t)) + g(x(t))u(t), \\ y(t) &= h(x(t)),\end{aligned}\tag{1.11}$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$, $g : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$ and $h : \mathbb{R}^n \rightarrow \mathbb{R}^q$ are smooth functions. A notable particular case is the *quadratic-bilinear* form, for which the nonlinear functions can be written and/or approximated as quadratic polynomials in their variables and compactly expressed, e. g., as

$$f(x) = f(0) + A_1x + A_2(x \otimes x),\tag{1.12}$$

where \otimes is the Kronecker product and $f(0) \in \mathbb{R}^n$, $A_1 \in \mathbb{R}^{n \times n}$, $A_2 \in \mathbb{R}^{n \times n^2}$ are constant matrices. A discussion of methods applicable to MOR of such systems is available in Chapters 2 and 3.

In more general settings, supporting algorithms providing interpolation/approximation of high-dimensional nonlinear multivariate functions are indeed available. We mention the Empirical Interpolation Methods in their various formulations introduced in Chapter 1 of Volume 2 and manifold interpolation (Chapter 7 in this volume), which provides a general framework for interpolation of orthogonal bases, subspaces or positive definite system matrices. Both these approaches are recurrent tools in several modern MOR frameworks.

1.2.1.5 Surrogate modeling

Extending the framework of classical MOR, which in the system-theoretic approach is usually applied to a state-spate description, surrogate modeling approaches provide tools for processing sequences of input-output data points and constructing an approximate metamodel that explains and reproduces their relationship. The last two chapters in this volume describe two alternative approaches for surrogate modeling. Chapter 9 presents an overview of the celebrated kernel methods, an approach that is very popular in the machine learning community, both for acceleration of complex simulation models, but also for classification and signal processing. Chapter 10 discusses Kriging methods or Gaussian Processes (GPs), with emphasis on design and analysis of computer experiments. These extensions of MOR bridge the gap between control and system theory with statistics, computer science, and (big) data science, further demonstrating how pervasive the key objectives are that characterize MOR.

1.2.2 The PDE approach

The second major approach to MOR starts from a field problem defined over a continuous domain Ω . Thus, a parametric PDE is given as starting point of the MOR procedure.

Two main steps are performed: the numerical discretization in space and time and the projection of the discretized form onto a reduced-order space. The projection space is chosen such that the field variable is well approximated in a natural PDE norm or it is chosen with respect to a given output quantity of interest. These basic tools are discussed in more detail in Chapter 1 of Volume 2.

The variational or weak form of a parametric linear PDE in the continuous setting is posed over a suitable Hilbert space $V(\Omega)$ and given as

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}) \quad \forall v \in V, \quad (1.13)$$

with bilinear form $a : V \times V \times \mathcal{P} \rightarrow \mathbb{R}$ and linear form $f : V \times \mathcal{P} \rightarrow \mathbb{R}$. The parameter vector is denoted $\boldsymbol{\mu}$ and is an element of the parameter space \mathcal{P} . In many application scenarios, a particular output of interest $s : \mathcal{P} \rightarrow \mathbb{R}$ is sought, given by the linear form $l : V \times \mathcal{P} \rightarrow \mathbb{R}$ as

$$s(\boldsymbol{\mu}) = l(u(\boldsymbol{\mu}); \boldsymbol{\mu}). \quad (1.14)$$

The case of a coercive and continuous bilinear form is the setting for many introductory examples but does not cover all PDE settings. E. g., in electromagnetics, i. e., when solving Maxwell's equations, an inf-sup stable sesquilinear form is often considered. In unsteady problems, the time-dependence is often made explicit and time is treated differently from other parameters in the ROM setting; see the POD-greedy algorithm for example. Nonlinear problems require particular care and methods, which are often adapted to the particular type of nonlinearity.

A suitable discretization method is chosen to approximate the field variable u , defining a corresponding discrete space V_h . The method of weighted residuals is invoked to turn the continuous form (1.13) into a discrete variational formulation.

The weak form in the discrete setting is given as

$$a(u_h(\boldsymbol{\mu}), v_h; \boldsymbol{\mu}) = f(v_h; \boldsymbol{\mu}) \quad \forall v_h \in V_h, \quad (1.15)$$

with bilinear form $a : V_h \times V_h \times \mathcal{P} \rightarrow \mathbb{R}$ and linear form $f : V_h \times \mathcal{P} \rightarrow \mathbb{R}$. The space of all v_h is the test space, while the space of u_h is the trial space.

A discrete solution is found by invoking Galerkin orthogonality, by enforcing that the test space is orthogonal to the residual. In Ritz–Galerkin methods, the residual is tested against the same set of functions as the ansatz functions, i. e., the test space is the same as the ansatz or trial space. In a more general Petrov–Galerkin method, test space and trial space are chosen as different spaces.

Starting from the discrete high-fidelity formulation (1.15), another Galerkin projection is invoked to arrive at the reduced-order formulation. A set of solutions is computed at parameter values $S_{N_{\max}} = \{\boldsymbol{\mu}^1, \dots, \boldsymbol{\mu}^{N_{\max}}\}$, either by pre-specifying $S_{N_{\max}}$ or using an iterative algorithm such as the greedy sampling. These solutions are often called ‘snapshots’. A projection space V_N is determined by a suitable method. The dif-

ferent methods are briefly introduced below and discussed in much detail in dedicated chapters of Volume 2.

The reduced-order variational formulation is to determine $u_N(\boldsymbol{\mu}) \in V_N$, such that

$$a(u_N(\boldsymbol{\mu}), v_N; \boldsymbol{\mu}) = f(v_N; \boldsymbol{\mu}) \quad \forall v_N \in V_N. \quad (1.16)$$

With matrix \mathbb{A}_h assembling the bilinear form and the load vector \mathbf{f}_h , let $\mathbb{V} \in \mathbb{R}^{N_h \times N}$ denote the matrix of basis vectors, derived from the snapshot solutions and project (1.15) onto the reduced-order space as

$$\mathbb{V}^T \mathbb{A}_h \mathbb{V} \mathbf{u}_N = \mathbb{V}^T \mathbf{f}_h. \quad (1.17)$$

The high-order solution is then approximated as

$$\mathbf{u}_h \approx \mathbb{V} \mathbf{u}_N. \quad (1.18)$$

Typical ROM ingredients are an affine parameter dependency, an offline–online decomposition and error bounds, which are explained in Chapter 1 of Volume 2.

Pointers to subsequent chapters for accurate ROMs in the PDE setting are given in this section for Volume 2. Each chapter explains in a detailed way a different method of how to obtain the projection spaces or follows an alternate route altogether. Numerical examples can be found in the respective chapters.

- *Proper Orthogonal Decomposition*

In the Proper Orthogonal Decomposition (POD), the projection space is determined from the principal modes of the *singular value decomposition* of sampled field solutions. The sampling is uniform over the parameter domain in many cases. POD is covered in depth in Chapter 2 of Volume 2.

- *Proper Generalized Decomposition*

The Proper Generalized Decomposition (PGD) assumes a separated representation, in which all variables, i. e., space, time and parameters, can be treated in the same way; see Chapter 3 of Volume 2. Error indicators and error bounds serve to iteratively build the approximation spaces.

- *Reduced Basis Method*

Reduced Basis (RB) MOR uses residual-based error indicators and error estimators to determine the projection space by a greedy sampling; see Chapter 4 of Volume 2. It is not uncommon in the literature to consider POD as a RB method.

- *Hyperreduction*

Hyperreduction techniques are related to the *Empirical Interpolation Method* (EIM) which generally aims to approximate an affine parameter dependency for an originally non-affine problem. The EIM is introduced in Chapter 1 of Volume 2 while the chapter on hyperreduction (Chapter 5 of Volume 2) details how these techniques can be used for ROM generation.

- *Localized Model Order Reduction*

The localized model reduction aims to determine local ROMs valid over parts of the computational domain and construct a global approximation through suitable couplings of local ROMs. The localized ROMs are usually generated with POD and RB techniques; see Chapter 6 of Volume 2.

- *Dynamic Mode Decomposition*

The Dynamic Mode Decomposition (DMD) is also based on the *singular value decomposition*; see Chapter 7 of Volume 2. The starting points are measurements of the time-trajectory which aim to approximate the time-advancement operator. The DMD is thus understood as a data-driven approach, since it does not project an affinely expanded system matrix.

1.2.3 Applications

In this section, we briefly introduce the several MOR applications that are collected in the third volume of this book series. Several early developments in MOR originated in the exponentially growing field of microelectronics during the last few decades of the 20th century. The enormous growth in complexity in designing microprocessors and computing systems was requiring scalable, efficient, and especially automated design and verification methods. This necessity provided a fertile ground for research on MOR, so that many contributors from mathematics, system and control theory, and electronics engineering proposed several key ideas and algorithms that are still widely adopted in modern tools. Chapter 4 of Volume 3 reviews some of these steps and provides an overview of MOR applications in microelectronics. It is not a surprise that MOR proves very successful also in electromagnetics, since electric/electronic circuits are just a lumped approximation of the more general Maxwell's field formulations. Applications of MOR in electromagnetics are discussed in detail in Chapter 5 of Volume 3.

Not long after the initial developments, the MOR field became more and more mature, with consolidated approaches both in the system-theoretic and in the PDE communities. This enabled reaching cross- and multidisciplinary applications. Volume 3 of this book series reports on several such applications of MOR, in particular: chemical process optimization (Chapter 1 of Volume 3), mechanical engineering (Chapter 2 of Volume 3), acoustics and vibration (Chapter 3 of Volume 3), computational aerodynamics (Chapter 6 of Volume 3) and fluid dynamics (Chapter 9 of Volume 3). These chapters build on the methods discussed in the first two volumes, in some cases proposing application-driven customized versions, and testify that pervasiveness of MOR exists in practically all fields of applied engineering.

Consolidation of MOR theory made algorithms more and more reliable. Therefore, unexpected applications started to be pursued even on biological systems. One of the most striking yet successful extensions is cardiovascular modeling (Chapter 8 of Volume 3), which attempts a quantitative prediction of the behavior of the most existing

complex “system”, the human body. The same objective is shared by Chapter 7 of Volume 3 on MOR applications to the neurosciences.

MOR continues its mainstream advancement in those areas, such as mathematics and control, where methodological aspects have been introduced and are still continuously refined. Chapter 11 of Volume 3 combines classical reduction approaches with graph theory for the reduction of network systems. This contribution is quite timely nowadays, when relations between distributed systems, agents, individuals at physical or social level are often described and explained based on their networked interconnection structure. Another timely application of MOR is described in Chapter 10 of Volume 3, discussing the very important aspects of uncertainty quantification, which play a fundamental role in all applications when the description of the systems in terms of their constitutive parameters is not deterministic but subject to stochastic variations.

Chapter 12 of Volume 3 confirms the relevance of MOR in industrial production settings. The recent paradigm shift towards “Industry 4.0” augmented the requirements for sophisticated prediction methods and tools. It is now conceivable that suitably constructed abstraction layers can be devised to build so-called “digital twins”, with the objective of mimicking the behavior of actual physical systems in real time and during their lifetime. This chapter provides an overview of the state of the art in this respect, where MOR plays once again a key role.

We conclude this introduction advising the reader to check Chapter 13 of Volume 3, which provides an overview of existing MOR software. Several commercial and academic software packages are reviewed, suitably classified with respect to the type of problems being addressed. Many of the latter packages can be freely downloaded, used, and possibly extended by active MOR researchers with new features and functionalities.

Bibliography

- [1] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. Adv. Des. Control., volume 6. SIAM Publications, Philadelphia, PA, 2005.
- [2] A. C. Antoulas, C. A. Beattie, and S. Gugercin. *Interpolatory Methods for Model Reduction*. Computational Science & Engineering. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2020.
- [3] P. Benner. *System Reduction for Nanoscale IC Design*. Mathematics in Industry, volume 20. Springer, 2017.
- [4] P. Benner, A. Cohen, M. Ohlberger and K. Willcox. *Model Reduction and Approximation: Theory and Algorithms*. Computational Science & Engineering. SIAM Publications, Philadelphia, PA, 2017.
- [5] P. Benner, M. Hinze and E. J. W. ter Maten. *Model Reduction for Circuit Simulation*. Lecture Notes in Electrical Engineering, volume 74. Springer, Dordrecht, 2011.

- [6] P. Benner, V. Mehrmann and D. C. Sorensen. *Dimension Reduction of Large-Scale Systems. Lecture Notes in Computational Sciences and Engineering*, volume 45. Springer, Berlin/Heidelberg, Germany, 2005.
- [7] P. Benner, M. Ohlberger, A. T. Patera, G. Rozza and K. Urban. *Model Reduction of Parametrized Systems*. Springer, 2017.
- [8] F. Chinesta, R. Keunings, and A. Leygue. *The Proper Generalized Decomposition for Advanced Numerical Simulations. SpringerBriefs in Applied Sciences and Technology*. Springer, 2014.
- [9] P. Feldmann and R. W. Freund. Efficient linear circuit analysis by Padé approximation via the Lanczos process. In *Proc. of EURO-DAC '94 with EURO-VHDL '94*, Grenoble, France, pages 170–175. IEEE Computer Society Press, 1994.
- [10] P. Feldmann and R. W. Freund. Efficient linear circuit analysis by Padé approximation via the Lanczos process. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 14:639–649, 1995.
- [11] L. Fortuna, G. Nummari, and A. Gallo. *Model Order Reduction Techniques with Applications in Electrical Engineering*. Springer, 1992.
- [12] K. Gallivan, E. Grimme, and P. Van Dooren. Asymptotic waveform evaluation via a Lanczos method. *Appl. Math. Lett.*, 7(5):75–80, 1994.
- [13] S. Grivet-Talocia and B. Gustavsen. *Passive Macromodeling: Theory and Applications*. John Wiley and Sons, New York, 2016.
- [14] J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations. SpringerBriefs in Mathematics*. Springer, Cham, 2016.
- [15] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. Society of Industrial and Applied Mathematics, Philadelphia, USA, 2016.
- [16] B. C. Moore. Principal component analysis in linear systems: controllability, observability, and model reduction. *IEEE Trans. Autom. Control*, AC–26(1):17–32, 1981.
- [17] G. Obinata and B. D. O. Anderson. *Model Reduction for Control System Design. Comm. Control Eng.* Springer, London, UK, 2001.
- [18] Z.-Q. Qu. *Model Order Reduction Techniques: with Applications in Finite Element Analysis*. Springer, Berlin, 2004.
- [19] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations. La Matematica per il 3+2*, volume 92. Springer, 2016. ISBN 978-3-319-15430-5.
- [20] A. Quarteroni and G. Rozza, editors. *Reduced Order Methods for Modeling and Computational Reduction. MS&A – Modeling, Simulation and Applications*, volume 9. Springer, Cham, Switzerland, 2014.
- [21] W. H. A. Schilders, H. A. van der Vorst and J. Rommes. *Model Order Reduction: Theory, Research Aspects and Applications*. Springer, Berlin, Heidelberg, 2008.
- [22] S. X.-D. Tan and L. He. *Advanced Model Order Reduction Techniques in VLSI Design*. Cambridge University Press, New York, NY, USA, 2007.

2 Balancing-related model reduction methods

Abstract: This chapter provides an introduction to the concept of system balancing. An overview of the historical development is given, and application areas for balancing-based model reduction are presented. Beginning with linear systems, the idea of a balanced system is explained and illustrated by an introductory example. A detailed description of the algorithmic realization, including implementable pseudocodes, is provided and numerical challenges are pointed out. Generalizations of the classical method of balanced truncation are reviewed. In particular, more general system classes such as differential-algebraic equations as well as nonlinear systems are discussed. Two numerical examples resulting from common partial differential equations are reviewed and analyzed with respect to the applicability of balancing-related methods. Pseudocodes will allow the reader to examine the method independently.

Keywords: balanced truncation, Gramians, Lyapunov equations, Lur'e equations, differential-algebraic equations

MSC 2010: 15A24, 34A09, 65F30, 93A15, 93C05, 93D30


2.1 Introduction

Balancing-based model reduction relies on the concept of truncating a system that is given in so-called balanced coordinates. The obvious questions to be discussed are: what are balanced coordinates and how do we obtain them? Regarding the first question, we consider different Gramian matrices that represent particular energies for the underlying system. For the second question, we introduce suitable state-space transformations based on solutions of (non-)linear matrix equations that transform the system into balanced coordinates. The final reduced-order models are then obtained by discarding states from the balanced full-order model. The reasoning behind this rationale is that in balanced coordinates, we can easily find the states that contribute least to the system energy.

2.1.1 Historical development and overview

Balancing-based model reduction has its origin in the design and synthesis of digital filters; see [90]. In their work, Mullis and Roberts study optimal and equal word

Tobias Breiten, Technical University of Berlin, Berlin, Germany
Tatjana Stykel, University of Augsburg, Augsburg, Germany

Open Access. © 2021 Tobias Breiten and Tatjana Stykel, published by De Gruyter.  This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

length filters obtained by state-space transformations of discrete-time linear systems. In [89], the method was picked up from a detailed system-theoretic point of view and put into the context of *principal component analysis*. Besides relating the results from [90] to the concepts of controllability and observability, Moore already observed that preservation of stability is guaranteed for the reduced-order model. Together with the preservation of controllability and observability, this was proven in [100]. Another appealing feature of the classical balanced truncation method is the availability of an *a priori error bound* with respect to the \mathcal{H}_∞ -norm; see [40, 48]. An efficient algorithmic realization of this method was developed in [79], and its numerical robustness was enhanced in [115, 132].

Modifications of balanced truncation were proposed soon after the appearance of the work by Mullis and Roberts [90]. Let us mention *stochastic balanced truncation* introduced in [38] and further investigated in, e. g., [55, 56, 140], *positive real balanced truncation* [38, 63, 92] and *bounded real balanced truncation* [92, 94]. For systems involving slow and fast dynamics, the method of *singular perturbation approximation* was suggested and analyzed in [83]. For systems operating at a known frequency range, the method of *frequency-weighted balanced truncation* was introduced in [40] and further refined in, e. g., [139, 143], while *frequency-limited balanced truncation* was discussed in [23, 47]. In the context of designing reduced-order controllers, we mention *linear quadratic Gaussian (LQG) balanced truncation* which goes back to [137, 72, 93] and \mathcal{H}_∞ *balanced truncation* [91]. Balanced truncation for positive systems was presented in [43, 110]. Most of the balancing-related methods for standard state-space systems were extended to *differential-algebraic equations (DAEs)* [29, 87, 108, 110, 127]. Furthermore, structure-preserving *balanced truncation algorithms for second-order systems* were considered in [27, 33, 86, 106], whereas balanced realizations for *periodic discrete-time systems* were discussed in [42, 135]. Balancing transformations for *linear time-varying systems* were first introduced in [119, 138] and further investigated in [78, 116]. The concept of balanced truncation was extended to a class of linear *infinite-dimensional systems* with finite-dimensional inputs and outputs in [49] and further studied in a more general setting in [61, 105]. Based on appropriate Hamilton–Jacobi equations, Scherpen extended balancing for linear systems to *nonlinear systems* in [117].

Due to the necessity of solving a set of (non-)linear matrix equations, for a long time, balanced truncation was considered to be applicable only to small and medium size systems. With the development of so-called *low-rank methods*, see, e. g., [24, 25, 80, 97], balancing-based reduced-order models nowadays can also be computed for large-scale systems resulting from a spatial semi-discretization of multidimensional partial differential equations. For model reduction of *parametric systems*, a combination of the reduced basis method for solving parameter-dependent matrix equations and balanced truncation was presented in [118, 122]. As a further topic of recent and ongoing research, we mention the use of other algebraic Gramians for balancing of

certain classes of control systems, among them linear stochastic systems [13, 26], bilinear systems [1, 13, 16] and quadratic–bilinear systems [15].

2.1.2 Structure of this chapter

In Section 2.2, we give an introduction to the classical version of balanced truncation for linear time-invariant (LTI) control systems. We introduce the control-theoretic notation required for understanding the steps to construct a balanced reduced-order model. Based on an explicit minimal example with two states, we study the effect of a balancing transformation and its consequences with respect to internal system properties such as controllability and observability. We summarize the theoretical properties of a balanced reduced-order model in Section 2.2.4 and provide a detailed self-implementable algorithm in Section 2.2.6. This also includes a discussion on approximation methods for large-scale linear matrix equations. Section 2.3 summarizes the different classes of Gramians used in the context of positive real balancing, bounded real balancing, LQG balancing, stochastic balancing, singular perturbation approximation, and cross-Gramian-based balancing, respectively. Furthermore, Section 2.4 describes the required modifications of the method when additional algebraic constraints are present, i. e., the underlying dynamics is described by a DAE system. In Section 2.5, we give an overview of different extensions of balanced truncation that are applicable in a nonlinear setting. In Section 2.6, we briefly discuss balanced truncation of (periodic) discrete-time systems and second-order systems. Section 2.7 illustrates possibilities and limits of balancing-based model reduction by means of two test examples.

2.2 Balanced truncation

The content of this section is well known in the literature and can be found similarly in, e. g., [4, 11, 60]. For the presentation of the necessary control-theoretic concepts, we refer to any textbook on control theory, e. g., [4, 70, 123, 144].

2.2.1 Formulation of the problem

For the remainder of this section, we consider a continuous LTI system of the form

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t), & x(0) &= x_0, \\ y(t) &= Cx(t) + Du(t),\end{aligned}\tag{2.1}$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times n}$ and $D \in \mathbb{R}^{p \times m}$. For fixed time t , we call $x(t)$, $u(t)$ and $y(t)$ the *state*, *control* and *output* of the system. Unless stated otherwise, we

always assume that the system is *asymptotically stable*, i. e., the system matrix A has no eigenvalues in the closed right half-plane \mathbb{C}^+ . Given the original system (2.1) of dimension n , the goal of model reduction is to construct a reduced-order system of the same form

$$\begin{aligned}\dot{\hat{x}}(t) &= \hat{A}\hat{x}(t) + \hat{B}u(t), & \hat{x}(0) &= \hat{x}_0, \\ \hat{y}(t) &= \hat{C}\hat{x}(t) + \hat{D}u(t),\end{aligned}\tag{2.2}$$

where $\hat{A} \in \mathbb{R}^{r \times r}$, $\hat{B} \in \mathbb{R}^{r \times m}$, $\hat{C} \in \mathbb{R}^{p \times r}$ and $\hat{D} \in \mathbb{R}^{p \times m}$. Usually, for the construction of (2.2), we have the two (concurrent) goals. On the one hand, the system should actually be a reduced-order system consisting of fewer system states. Formally, we thus require $r \ll n$. On the other hand, the system should constitute an approximation of the original system and we, therefore, also demand that the reduced output approximates the original one, i. e., $y(t) \approx \hat{y}(t)$. For the construction of the reduced system matrices, we employ a *Petrov–Galerkin* projection framework: given two subspaces $\mathcal{V}, \mathcal{W} \subset \mathbb{R}^n$ of dimension r and associated basis matrices $V, W \in \mathbb{R}^{n \times r}$, we approximate $x(t)$ by $V\hat{x}(t)$ and enforce an orthogonality constraint on the residual

$$V\dot{\hat{x}}(t) - AV\hat{x}(t) - Bu(t) \perp \mathcal{W}.$$

Since the columns of W span the subspace \mathcal{W} , the latter condition can equivalently be expressed as

$$W^T(V\dot{\hat{x}}(t) - AV\hat{x}(t) - Bu(t)) = 0.\tag{2.3}$$

In case of biorthogonal matrices V, W , we have $W^T V = I$ and (2.3) yields a reduced system (2.2), where

$$\hat{A} = W^T A V, \quad \hat{B} = W^T B, \quad \hat{C} = C V.$$

Since the *feedthrough* term D is independent of the system dimension n , we may construct a reduced system with $\hat{D} = D$ and thus restrict ourselves to the case $D = 0$. Note, however, that, for other classes of systems and variants of balanced truncation, choosing $\hat{D} = D$ is not always appropriate.

2.2.2 Basics from LTI system theory

With regard to establishing a measure for the approximation quality of (2.2), recall that by means of the variation of constants formula for $x_0 = 0$ and $\hat{x}_0 = 0$, the output error is given by

$$y(t) - \hat{y}(t) = \int_0^t (C e^{A(t-\tau)} B - \hat{C} e^{\hat{A}(t-\tau)} \hat{B}) u(\tau) \, d\tau + (D - \hat{D}) u(t).$$

An application of the Laplace transformation $\mathcal{L}[\cdot]$ allows us to rewrite the difference in frequency domain as

$$\mathcal{L}[y](s) - \mathcal{L}[\hat{y}](s) = (\mathbf{G}(s) - \widehat{\mathbf{G}}(s))\mathcal{L}[u](s), \quad (2.4)$$

where

$$\mathbf{G}(s) = C(sI - A)^{-1}B + D, \quad (2.5)$$

and $\widehat{\mathbf{G}}(s) = \widehat{C}(sI - \widehat{A})^{-1}\widehat{B} + \widehat{D}$ are the transfer functions of systems (2.1) and (2.2), respectively. Assuming additionally that the reduced system matrix \widehat{A} is asymptotically stable, the transfer functions $\mathbf{G}, \widehat{\mathbf{G}}: \mathbb{C}^+ \rightarrow \mathbb{C}^{p \times m}$ are analytic in \mathbb{C}^+ and we can consider the Hardy space

$$\mathcal{H}_\infty = \{\mathbf{F}: \mathbb{C}^+ \rightarrow \mathbb{C}^{p \times m} \mid \mathbf{F} \text{ is analytic in } \mathbb{C}^+ \text{ and } \|\mathbf{F}\|_{\mathcal{H}_\infty} < \infty\},$$

where

$$\|\mathbf{F}\|_{\mathcal{H}_\infty} := \sup_{s \in \mathbb{C}^+} \|\mathbf{F}(s)\|_2$$

and $\|\cdot\|_2$ denotes the spectral matrix norm. As a consequence of (2.4), the Plancherel theorem implies

$$\|y - \hat{y}\|_{L_2(0, \infty; \mathbb{R}^p)} \leq \|\mathbf{G} - \widehat{\mathbf{G}}\|_{\mathcal{H}_\infty} \|u\|_{L_2(0, \infty; \mathbb{R}^m)}, \quad (2.6)$$

where the $L_2(0, \infty; \mathbb{R}^p)$ -norm for a time-varying function $f: (0, \infty) \rightarrow \mathbb{R}^p$ is defined as

$$\|f\|_{L^2(0, \infty; \mathbb{R}^p)} := \left(\int_0^\infty f(t)^T f(t) dt \right)^{\frac{1}{2}}.$$

Since balanced truncation yields an a priori error bound with respect to the \mathcal{H}_∞ -norm, we can relate the quality of a reduced system to the $L_2(0, \infty; \mathbb{R}^p)$ -error of the underlying output signals.

Let us further recall that the *finite-time controllability* and *observability Gramians* of system (2.1) are defined as

$$X(t) = \int_0^t e^{A\tau} B B^T e^{A^T \tau} d\tau, \quad Y(t) = \int_0^t e^{A^T \tau} C^T C e^{A\tau} d\tau.$$

The relevance of these Gramians in the context of model reduction is due to their connection to the input-output behavior of the system. In particular, given a *reachable* state $x_d \in \mathbb{R}^n$, using the *Moore–Penrose inverse* $X(t_f)^\dagger$, we can define a control

$$\tilde{u}(t) = B^T e^{A^T(t_f - t)} X(t_f)^\dagger x_d$$

that steers system (2.1) from $x(0) = 0$ to $x(t_f) = x_d$ in time t_f . Moreover, this control is optimal in the sense that, for an arbitrary control u steering the system from 0 to x_d , we find that

$$x_d^T X(t_f)^\dagger x_d = \|\tilde{u}\|_{L_2(0,t_f;\mathbb{R}^m)}^2 \leq \|u\|_{L_2(0,t_f;\mathbb{R}^m)}^2. \quad (2.7)$$

Note that, for times $t_1 \leq t_2$, we have

$$z^T X(t_2) z \geq z^T X(t_1) z \quad \text{for all } z \in \mathbb{R}^n. \quad (2.8)$$

This implies $X(t_2) \succeq X(t_1)$ or, equivalently, $X(t_2) - X(t_1) \succeq 0$ meaning that $X(t_2) - X(t_1)$ is a positive semi-definite matrix. An analogous notation $\preceq 0$ will be used for negative semi-definite matrices. The reasoning then is that the *infinite-time controllability Gramian*

$$X = \int_0^\infty e^{At} B B^T e^{A^T t} dt, \quad (2.9)$$

which exists since A is asymptotically stable, encodes the minimum input energy required to reach the target state x_d . On the infinite-time horizon, the asymptotic limit of property (2.7) is usually expressed as

$$x_d^T X^\dagger x_d = \min_{\substack{u \in L_2(-\infty, 0; \mathbb{R}^m) \\ x(-\infty) = 0, x(0) = x_d}} \int_{-\infty}^0 u^T(t) u(t) dt. \quad (2.10)$$

A similar conclusion can be drawn by noting that $Y(t_f)$ yields the output energy $\|y\|_{L^2(0,t_f;\mathbb{R}^p)}^2$ generated by the initial condition $x(0) = x_0$. This energy is given by

$$\|y\|_{L^2(0,t_f;\mathbb{R}^p)}^2 = \int_0^{t_f} (C e^{At} x_0)^T (C e^{At} x_0) d\tau = x_0^T Y(t_f) x_0. \quad (2.11)$$

Hence, the *infinite-time observability Gramian*

$$Y = \int_0^\infty e^{A^T t} C^T C e^{At} dt \quad (2.12)$$

yields a natural way of measuring the amount of energy included in given states. In particular, it can be shown that X and Y satisfy

$$AX + XA^T + BB^T = 0, \quad (2.13)$$

$$A^T Y + YA + C^T C = 0. \quad (2.14)$$

The above equations are *linear matrix equations* in the unknowns $X, Y \in \mathbb{R}^{n \times n}$ and are called *Lyapunov equations*.

2.2.3 The concept of balancing

We have seen that the Gramians X and Y contain information about the input and output energy of the system. However, it remains open how this information can be used in order to obtain a reduced-order model. In general, we cannot expect that a state that is easy to reach produces, at the same time, a large amount of output energy. To illustrate this further, let us consider two different LTI systems of the form (2.1), where the system matrices are given by

$$A = \begin{bmatrix} -2 & -200 \\ 0 & -\frac{1}{2} \end{bmatrix}, \quad B = \begin{bmatrix} 2 \\ 0.02 \end{bmatrix}, \quad C = [0.02 \quad 1], \quad D = 0, \quad (2.15)$$

and

$$\tilde{A} = \begin{bmatrix} -2 & -\frac{200}{10001} \\ -\frac{200}{10001} & -\frac{1}{2} \end{bmatrix}, \quad \tilde{B} = \begin{bmatrix} 2 \\ 0.02 \end{bmatrix}, \quad \tilde{C} = [2 \quad 0.02], \quad \tilde{D} = 0. \quad (2.16)$$

Without further knowledge, the approximability of the above two systems remains unclear. Let us thus have a look at the transfer functions \mathbf{G} and $\tilde{\mathbf{G}}$ that can (approximately) be computed as

$$\mathbf{G}(s) = \frac{0.05s}{s^2 + 2.5s + 1}, \quad \tilde{\mathbf{G}}(s) \approx \frac{4s + 1.999}{s^2 + 2.5s + 0.9996}.$$

From this point of view, one can argue that a reasonable approximation of the second system is given as

$$\tilde{\mathbf{G}}(s) \approx \frac{4s + 2}{s^2 + 2.5s + 1} = \frac{4(2s + 1)}{(2s + 1)(s + 2)} = \frac{4}{s + 2} =: \hat{\mathbf{G}}(s).$$

In other words, we expect a one-dimensional reduced-order model realized via $\hat{A} = -2$ and $\hat{B} = \hat{C} = 2$ to yield a small approximation error. On the other hand, for the first transfer function \mathbf{G} , an immediate approximation is not obvious. With the previously introduced concepts, one might ask whether there is a difference between \mathbf{G} and $\tilde{\mathbf{G}}$ from a control-theoretic point of view. In this regard, let us analyze the corresponding controllability and observability Gramians. In our example, it is possible to obtain the exact solutions as

$$X = \begin{bmatrix} 1 & 0 \\ 0 & 10^{-4} \end{bmatrix}, \quad Y = \begin{bmatrix} 10^{-4} & 0 \\ 0 & 1 \end{bmatrix},$$

$$\bar{X} = \begin{bmatrix} 1 & 0 \\ 0 & 10^{-4} \end{bmatrix}, \quad \bar{Y} = \begin{bmatrix} 1 & 0 \\ 0 & 10^{-4} \end{bmatrix}.$$

From (2.7) and (2.11), for the first system, we now conclude that states of the form $x = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}$ are easier to reach than those of the form $x = \begin{bmatrix} 0 \\ \alpha \end{bmatrix}$ with $\alpha \in \mathbb{R}$. On the other hand,

the output energy associated to states of the form $x = \begin{bmatrix} 0 \\ \alpha \end{bmatrix}$ is significantly larger than the energy of the states $x = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}$. The situation is different for the second system. Here, states of the form $x = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}$ are comparably easy to reach and, simultaneously, yield a large amount of output energy. It is thus natural to construct a reduced system by keeping the first coordinate while discarding the second one. As a surprising result, this yields the already discussed reduced transfer function $\widehat{\mathbf{G}}(s) = \frac{4}{s+2}$. Let us further emphasize that the Gramians \bar{X} and \bar{Y} of the second system are equal and diagonal, they are in *balanced form*.

The example indicates that balanced systems significantly simplify the decision process of which states to discard. Let us thus, for now, assume that system (2.1) is given in balanced form such that for the Gramians

$$X = Y = \text{diag}(\sigma_1, \dots, \sigma_n).$$

We assume without loss of generality that the diagonal entries are ordered according to $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n$. Moreover, we focus on systems that are controllable and observable, i. e., systems that satisfy $\sigma_n > 0$. For obtaining a reduced-order system, we partition the system as follows:

$$\begin{aligned} \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} &= \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u(t), \\ y(t) &= \begin{bmatrix} C_1 & C_2 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + Du(t), \end{aligned} \quad (2.17)$$

where $A_{11} \in \mathbb{R}^{r \times r}$, $A_{12} \in \mathbb{R}^{r \times (n-r)}$, $A_{21} \in \mathbb{R}^{(n-r) \times r}$, $A_{22} \in \mathbb{R}^{(n-r) \times (n-r)}$, $B_1 \in \mathbb{R}^{r \times m}$, $B_2 \in \mathbb{R}^{(n-r) \times m}$, $C_1 \in \mathbb{R}^{p \times r}$ and $C_2 \in \mathbb{R}^{p \times (n-r)}$. The crucial observation now is that the balanced form allows us to compare states partitioned according to (2.17). Indeed, given unit vectors $x_d = e_j$, $j \leq r$, and $\tilde{x}_d = e_k$, $k > r$, for the optimal controls u and \tilde{u} that steer the system (asymptotically) to x_d and \tilde{x}_d in infinite time, we find that

$$\|u\|_{L_2(-\infty, 0; \mathbb{R}^m)}^2 = e_j^T X^\dagger e_j = \frac{1}{\sigma_j} \leq \frac{1}{\sigma_k} = e_k^T X^\dagger e_k = \|\tilde{u}\|_{L_2(-\infty, 0; \mathbb{R}^m)}^2.$$

At the same time, for the associated output signals y and \tilde{y} , we have

$$\|y\|_{L_2(0, \infty; \mathbb{R}^p)}^2 = e_j^T Y e_j = \sigma_j \geq \sigma_k = e_k^T Y e_k = \|\tilde{y}\|_{L_2(0, \infty; \mathbb{R}^p)}^2.$$

It therefore seems natural to discard states of the form $x = \begin{bmatrix} 0 \\ \alpha \end{bmatrix}$ while keeping those of the form $x = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}$. As a consequence, we obtain a reduced-order model

$$\begin{aligned} \dot{x}_1(t) &= A_{11}x_1(t) + B_1u(t), \\ \hat{y}(t) &= C_1x_1(t) + Du(t), \end{aligned} \quad (2.18)$$

where \hat{y} is an approximation of y .

2.2.4 Theoretical properties

Let us now summarize the most important properties of the reduced-order system (2.18). First of all, since the original model was assumed to be balanced, from (2.13) and (2.14) we know that the Gramians satisfy

$$\begin{aligned} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} X_1 & 0 \\ 0 & X_2 \end{bmatrix} + \begin{bmatrix} X_1 & 0 \\ 0 & X_2 \end{bmatrix} \begin{bmatrix} A_{11}^T & A_{21}^T \\ A_{12}^T & A_{22}^T \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \begin{bmatrix} B_1^T \\ B_2^T \end{bmatrix} &= 0, \\ \begin{bmatrix} A_{11}^T & A_{21}^T \\ A_{12}^T & A_{22}^T \end{bmatrix} \begin{bmatrix} Y_1 & 0 \\ 0 & Y_2 \end{bmatrix} + \begin{bmatrix} Y_1 & 0 \\ 0 & Y_2 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} + \begin{bmatrix} C_1^T \\ C_2^T \end{bmatrix} \begin{bmatrix} C_1^T \\ C_2^T \end{bmatrix} &= 0, \end{aligned}$$

where $X_1 = Y_1 = \text{diag}(\sigma_1, \dots, \sigma_r)$ and $X_2 = Y_2 = \text{diag}(\sigma_{r+1}, \dots, \sigma_n)$. The eigenvalues $\sigma_1, \dots, \sigma_n$ of X and Y are called the *Hankel singular values*. In fact, they are singular values of the *Hankel operator* of the underlying system, see, e. g., [4]. Inspection of the (1, 1) blocks of these matrix equations immediately shows that the Gramians of the reduced system (2.18) are given by X_1 and Y_1 , respectively. Hence, we obtain

The reduced-order system is in balanced form.

If we additionally assume that $\sigma_r > \sigma_{r+1}$, i. e., there exists a true gap between the smallest diagonal entry of the Gramians $X_1 = Y_1$ and the largest diagonal entry of $X_2 = Y_2$, it can further be shown, see [100], that

the reduced-order system is asymptotically stable.

Since the original system was assumed to be controllable and observable, all entries of X_1 and Y_1 are nonzero, i. e.,

the reduced-order system is controllable and observable.

Moreover, an a priori error bound with respect to the \mathcal{H}_∞ -norm can be given.

For the original and reduced-order transfer functions \mathbf{G} and $\widehat{\mathbf{G}}$, we find that

$$\|\mathbf{G} - \widehat{\mathbf{G}}\|_{\mathcal{H}_\infty} \leq 2 \sum_{i=r+1}^n \sigma_i. \quad (2.19)$$

For a more detailed presentation and corresponding proofs, we refer to the original work [40, 48, 89, 90, 100] or, e. g., [4, Chapter 7].

2.2.5 Systems with inhomogeneous initial condition

In the previous discussion, system (2.1) was assumed to have a homogeneous initial condition $x(0) = 0$. If this is not the case, the Laplace transformation of the output

contains an additional term of the form $C(sI - A)^{-1}x_0$. We can thus define an augmented transfer function $\mathbf{G}(s) = C(sI - A)^{-1}[B, x_0] + [D, 0]$ and subsequently apply the classical balancing method to this system. Alternatively, we can interpret the inhomogeneous system as the homogeneous one

$$\begin{aligned}\dot{x}(t) &= Ax(t) + [B, x_0] \begin{bmatrix} u(t) \\ u_0(t) \end{bmatrix}, \quad x(0) = 0, \\ y(t) &= Cx(t) + [D, 0] \begin{bmatrix} u(t) \\ u_0(t) \end{bmatrix},\end{aligned}$$

where u_0 is considered to be a unit pulse input. This idea has been initially proposed and theoretically studied in [65]; see also [9]. A variation of this approach has recently been proposed in [8] and relies on the superposition principle for linear systems. Based on partitioning the system response into an uncontrolled part with inhomogeneous initial condition and a controlled part with homogeneous initial condition, the method reduces the associated systems individually.

2.2.6 Algorithmic realization

Up to this point, for the construction of the reduced-order model (2.18), we assumed (2.1) to be in balanced form. A natural question arises how to obtain this balanced form for a general system. In this subsection, we discuss the computation of such a balanced realization which, subsequently, can be truncated to construct a balanced reduced-order model.

Square root balanced truncation method

The following method is typically referred to as *square root balanced truncation* and goes back to [79]. Let us analyze the effect of a coordinate transform $\tilde{x} = Tx$ characterized by a regular transformation matrix $T \in \mathbb{R}^{n \times n}$. Rewriting the dynamics with respect to the coordinates \tilde{x} , we obtain the equivalent control system

$$\begin{aligned}\dot{\tilde{x}}(t) &= \tilde{A}\tilde{x}(t) + \tilde{B}u(t), \\ y(t) &= \tilde{C}\tilde{x}(t) + Du(t),\end{aligned}\tag{2.20}$$

with $\tilde{A} = TAT^{-1}$, $\tilde{B} = TB$ and $\tilde{C} = CT^{-1}$. The associated infinite-time controllability and observability Gramians are then given by

$$\begin{aligned}\tilde{X} &= \int_0^\infty e^{\tilde{A}t} \tilde{B} \tilde{B}^T e^{\tilde{A}^T t} dt = \int_0^\infty T e^{At} T^{-1} T B B^T T^T T^{-T} e^{A^T t} T^T dt = T X T^T, \\ \tilde{Y} &= \int_0^\infty e^{\tilde{A}^T t} \tilde{C}^T \tilde{C} e^{\tilde{A} t} dt = \int_0^\infty T^{-T} e^{A^T t} T^T T^{-T} C^T C T^{-1} T e^{A t} T^{-1} dt = T^{-T} Y T^{-1}.\end{aligned}$$

Hence, we seek T such that $TXT^T = T^{-T}YT^{-1} = \text{diag}(\sigma_1, \dots, \sigma_n)$. With this in mind, we assume that the symmetric positive definite Gramians X and Y are given in Cholesky form

$$X = L_X^T L_X, \quad Y = L_Y^T L_Y,$$

where $L_X, L_Y \in \mathbb{R}^{n \times n}$ are not necessarily upper triangular matrices. Let us further compute the singular value decomposition (SVD) of the product of the Cholesky factors, i. e.,

$$L_X L_Y^T = U \Sigma Z^T,$$

where $U, Z \in \mathbb{R}^{n \times n}$ are orthogonal matrices and $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n) \succ 0$ is positive definite. By algebraic manipulations, the reader may verify that a balancing coordinate transformation is given by $T = \Sigma^{-\frac{1}{2}} Z^T L_Y$. In particular, for the Gramians of the transformed system (2.20), we find that $\tilde{X} = \tilde{Y} = \Sigma$. From here, it is now possible to construct (2.18) by simply discarding the $(n - r)$ -dimensional part of the balanced dynamics. However, from a numerical point of view, it is advisable to directly compute projection matrices V and W that lead to a reduced system (2.2) which coincides with (2.18). Indeed, the balancing coordinate transformation T is ill-conditioned whenever some of the Hankel singular values are very small. Let us thus consider a partitioning

$$L_X L_Y^T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} Z_1^T \\ Z_2^T \end{bmatrix}, \quad (2.21)$$

where $U_1, Z_1 \in \mathbb{R}^{n \times r}$ and $\Sigma_1 \in \mathbb{R}^{r \times r}$. Corresponding to the Petrov–Galerkin framework, we now set

$$V = L_X^T U_1 \Sigma_1^{-\frac{1}{2}}, \quad W = L_Y^T Z_1 \Sigma_1^{-\frac{1}{2}}.$$

Again, by algebraic manipulations it can be verified that V and W satisfy $W^T V = I$ and that $(A_{11}, B_1, C_1, D) = (W^T A V, W^T B, C V, D)$.

Linear matrix equations

As we have seen, the square root balancing method relies on the computation of the Cholesky factors L_X and L_Y , which, in turn, depend on the Gramians X and Y . In order to perform the transformation step, we thus have to compute X and Y either via their integral representations or as the solutions to the Lyapunov equations (2.13) and (2.14). Here, we focus on the latter approach. For approximation techniques based on quadrature, we refer to [113]. As a representative for both X and Y , consider a linear matrix equation for the unknown X of the form

$$AX + XA^T = G \quad (2.22)$$

where $A, G \in \mathbb{R}^{n \times n}$. From an abstract linear algebra point of view, we can replace this matrix equation with an ordinary linear system of dimension n^2 . For this purpose, we recall the vectorization operator as well as the Kronecker product defined by

$$\begin{aligned} \text{vec}: \mathbb{R}^{n \times m} &\rightarrow \mathbb{R}^{nm}, \quad \text{vec}(A) \mapsto [a_{11}, \dots, a_{n1}, a_{12}, \dots, a_{1n}, \dots, a_{nm}]^T, \\ \otimes: \mathbb{R}^{n \times m} \times \mathbb{R}^{p \times q} &\rightarrow \mathbb{R}^{np \times mq}, \quad \otimes(A, B) = A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1m}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \dots & a_{nm}B \end{bmatrix}. \end{aligned}$$

For matrices A, B and C of compatible dimensions, these operators are related via the formula

$$\text{vec}(ABC) = (C^T \otimes A) \text{vec}(B).$$

We can now apply the vectorization operator to both sides in (2.22) to obtain the equivalent linear system

$$(I \otimes A + A \otimes I) \text{vec}(X) = \text{vec}(G), \quad (2.23)$$

where I denotes the identity matrix of dimension n . While this yields the possibility to compute the solution by standard solvers for linear systems, the complexity now scales with the dimension n^2 , e. g., a standard LU decomposition will have complexity $\mathcal{O}(n^6)$. For small to medium scale systems, the Bartels–Stewart algorithm [6] can be considered to be the method of choice. The main idea of this algorithm is first to transform the matrix A into a real Schur form $Q^T A Q$ by means of an orthogonal transformation $Q \in \mathbb{R}^{n \times n}$. The quasi-upper triangular form of the resulting matrix $Q^T A Q$ can then be exploited to obtain a solution of (2.22). A detailed description of the method can be found in [6]. Note that a corresponding MATLAB implementation `lyap` is provided by the Systems and Control Toolbox. As a variant of the Bartels–Stewart algorithm, let us mention Hammarling’s method [62] which directly computes a Cholesky factorization $X = L_X^T L_X$, $L_X \in \mathbb{R}^{n \times n}$, of the solution of (2.22) with $G = -BB^T$. Again, a MATLAB implementation `lyapchol` is accessible in the Systems and Control Toolbox.

For systems resulting from a spatial semi-discretization of partial differential equations (PDEs), the previous methods are often not computationally feasible. As a remedy, in recent years, there has been an increasing interest in finding efficient approximation techniques for linear matrix equations of the form (2.22). Most of these methods can be categorized into Krylov subspaces methods, alternating directions implicit (ADI) based methods as well as iterative solvers that exploit the specific Kronecker structure. For a recent survey of low-rank methods for large-scale matrix equations, we refer to [28, 120]. A more detailed discussion with application to model reduction by balanced truncation can be found in [11]. Common for all these methods is that they rely on a *low-rank* approximation of the true solution which is known to exist for a large class of systems including parabolic second-order PDEs; see [5, 51, 57, 95, 98].

Pseudocode

A pseudocode summarizing the classical balanced truncation method is shown in Algorithm 2.1. For large-scale systems, steps 1 and 2 should be replaced by an approximation technique as described previously. Let us emphasize that the remaining steps remain unchanged when

$$X = L_X^T L_X \approx \hat{L}_X^T \hat{L}_X, \quad Y = L_Y^T L_Y \approx \hat{L}_Y^T \hat{L}_Y$$

are approximated by low-rank matrices with factors $\hat{L}_X \in \mathbb{R}^{k_1 \times n}$ and $\hat{L}_Y \in \mathbb{R}^{k_2 \times n}$, where $k_1, k_2 \ll n$.

Algorithm 2.1: Balanced truncation for LTI control systems in MATLAB.

Require: Original system A, B, C, D ; error tolerance tol .

Ensure: Reduced system $\hat{A}, \hat{B}, \hat{C}, \hat{D}$ with $\|\mathbf{G} - \hat{\mathbf{G}}\|_{\mathcal{H}_\infty} \leq \text{tol}$.

- 1: Compute the solution $X = L_X^T L_X$ of (2.13). ▷ $LX = \text{lyapchol}(A, B)$
 - 2: Compute the solution $Y = L_Y^T L_Y$ of (2.14). ▷ $LY = \text{lyapchol}(A', C')$
 - 3: Compute the SVD (2.21) of $L_X L_Y^T$, ▷ $[U, \Sigma, Z] = \text{svd}(LX * LY')$
 where r is chosen such that $2 \sum_{i=r+1}^n \sigma_i \leq \text{tol}$.
 - 4: Define $V = L_X^T U_1 \Sigma_1^{-\frac{1}{2}}$ and $W = L_Y^T Z_1 \Sigma_1^{-\frac{1}{2}}$.
 - 5: Define $\hat{A} = W^T A V, \hat{B} = W^T B, \hat{C} = C V$ and $\hat{D} = D$.
-

2.3 Variants of classical balancing

In this section, we present a brief survey on other balancing-related model reduction techniques which have been developed for various classes of control systems with different control-theoretic properties. A key idea of all these methods is to define a pair of Gramians which characterize the inherent properties of the particular class of systems. Then the reduced-order model is obtained by transforming the system into a balanced form such that the Gramians of the transformed system are equal and diagonal and truncating the state components corresponding to the small diagonal elements of the Gramians.

We start our consideration by introducing a concept of dissipativity which was extensively studied by Willems [141, 142]. System (2.1) is called *dissipative* with respect to a supply rate $w(u(t), y(t))$ if there exists a non-negative function $\mathcal{S}: \mathbb{R}^n \rightarrow \mathbb{R}$ satisfying the dissipation inequality

$$\mathcal{S}(x(t_0)) + \int_{t_0}^{t_1} w(u(\tau), y(\tau)) \, d\tau \geq \mathcal{S}(x(t_1)) \quad (2.24)$$

for all $t_0, t_1 \in \mathbb{R}$ with $t_1 \geq t_0$ and all $u \in L_2(t_0, t_1; \mathbb{R}^m)$. The function S is called the *storage function*. The dissipation inequality implies that the increase in the internal energy described by the storage function S on the time interval (t_0, t_1) does not exceed the energy supplied to the system. By choosing the quadratic supply rate

$$w(u(t), y(t)) = y^T(t)Qy(t) + 2y^T(t)Su(t) + u^T(t)Ru(t) \quad (2.25)$$

with $Q = Q^T \in \mathbb{R}^{p \times p}$, $S \in \mathbb{R}^{p \times m}$ and $R = R^T \in \mathbb{R}^{m \times m}$, the dissipativity of (2.1) can be characterized in terms of the Kalman–Yakubovich–Popov lemma [3, 32]. If system (2.1) is minimal, i. e., it is controllable and observable, then the following statements are equivalent:

1. System (2.1) is dissipative with respect to the supply rate w as in (2.25).
2. There exists a symmetric, positive definite matrix $Y \in \mathbb{R}^{n \times n}$ such that the linear matrix inequality (LMI)

$$\begin{bmatrix} A^T Y + YA - C^T Q C & YB - C^T (QD + S) \\ B^T Y - (QD + S)^T C & -R - D^T Q D - D^T S - S^T D \end{bmatrix} \preceq 0 \quad (2.26)$$

is fulfilled.

3. There exists a matrix triple (Y, K, J) with $K \in \mathbb{R}^{k \times n}$, $J \in \mathbb{R}^{k \times m}$ and symmetric, positive definite $Y \in \mathbb{R}^{n \times n}$ satisfying the *Lur'e matrix equation*

$$\begin{bmatrix} A^T Y + YA - C^T Q C & YB - C^T (QD + S) \\ B^T Y - (QD + S)^T C & -R - D^T Q D - D^T S - S^T D \end{bmatrix} = - \begin{bmatrix} K^T \\ J^T \end{bmatrix} \begin{bmatrix} K^T \\ J^T \end{bmatrix}^T. \quad (2.27)$$

Note that, if $R_Y = R + D^T Q D + D^T S + S^T D$ is positive definite, then J is nonsingular and, hence, the Lur'e equation (2.27) can equivalently be written as the *Riccati equation*

$$A^T Y + YA - C^T Q C + (YB - C^T (QD + S))R_Y^{-1}(YB - C^T (QD + S))^T = 0.$$

In general, the solution of (2.26) is not unique. There exist, however, unique solutions Y_{\min} and Y_{\max} such that $0 < Y_{\min} \leq Y \leq Y_{\max}$ for all symmetric solutions Y of (2.26). These extremal solutions can be used to characterize the *required supply* and *available storage* of system (2.1) defined as

$$\begin{aligned} \mathcal{S}_r(x_0) &= \inf_{\substack{u \in L_2(-\infty, 0; \mathbb{R}^m) \\ x(-\infty)=0, x(0)=x_0}} \int_{-\infty}^0 w(u(\tau), y(\tau)) \, d\tau, \\ \mathcal{S}_a(x_0) &= \sup_{\substack{u \in L_2(0, \infty; \mathbb{R}^m) \\ x(0)=x_0, x(\infty)=0}} - \int_0^{\infty} w(u(\tau), y(\tau)) \, d\tau, \end{aligned}$$

respectively. The required supply $\mathcal{S}_r(x_0)$ describes the minimum amount of energy that has to be supplied to the system to steer it from the zero state into the state $x(0) = x_0$,

whereas the available storage $\mathcal{S}_a(x_0)$ determines the maximum amount of energy that can be extracted from the system starting at the initial state $x(0) = x_0$ and reaching the zero state. For these quadratic functionals, we find that

$$\mathcal{S}_r(x_0) = x_0^T Y_{\max} x_0, \quad \mathcal{S}_a(x_0) = x_0^T Y_{\min} x_0,$$

see [142]. These energy representations suggest, just as in the classical balancing approach, to take Y_{\max}^{-1} and Y_{\min} as a pair of Gramians and use them for balancing and truncation. Since the numerical solution of LMIs is, in general, much more expensive than that of Lur'e or Riccati equations, we restrict ourselves to the definition of the Gramians as solutions to matrix equations. Let us further emphasize that the dual Lur'e equation

$$\begin{bmatrix} AX + XA^T - BQB^T & XC^T - B(DQ + S)^T \\ CX - (DQ + S)B^T & -Q - DRD^T - DS^T - SD^T \end{bmatrix} = - \begin{bmatrix} L \\ F \end{bmatrix} \begin{bmatrix} L \\ F \end{bmatrix}^T \quad (2.28)$$

for a matrix triple (X, L, F) with $X \in \mathbb{R}^{n \times n}$, $L \in \mathbb{R}^{n \times l}$ and $F \in \mathbb{R}^{m \times l}$ is also of great interest. One can show, e. g., [56, 92] that the extremal solutions of (2.27) and (2.28) satisfy the relations

$$Y_{\min} = X_{\max}^{-1}, \quad Y_{\max} = X_{\min}^{-1}.$$

Thus, to avoid the explicit inversion of Y_{\max} , we can balance the minimal solutions Y_{\min} and X_{\min} of the Lur'e equations (2.27) and (2.28), respectively. In the following, we show that several particular choices of Q , S and R in (2.25) correspond to various physical properties of the control system (2.1) and lead to different balanced truncation methods that preserve these properties in reduced-order models.

2.3.1 Positive real balancing

First, we consider system (2.1) with $m = p$ and set $Q = 0$, $S = \frac{1}{2}I_m$ and $R = 0$. Then the supply rate takes the form

$$w(u(t), y(t)) = y^T(t)u(t).$$

In this case, system (2.1) satisfying the dissipation inequality (2.24) is called *passive*. Such systems play an important role in circuit theory and network analysis [3]; see also Chapter 5 of this volume. Passivity-preserving balanced truncation for such systems was considered in [63, 101]. If system (2.1) is controllable, then the dissipation inequality (2.24) is equivalent to the condition

$$\int_0^t y^T(\tau)u(\tau) \, d\tau \geq 0$$

which holds for all $t > 0$ and all $u \in L_2(0, \infty; \mathbb{R}^m)$; see [142]. This condition is often used as a definition of passivity. It is equivalent to the *positive realness* of the transfer function $\mathbf{G}(s) = C(sI - A)^{-1}B + D$ meaning that \mathbf{G} is analytic in \mathbb{C}^+ and $\mathbf{G}(s) + \mathbf{G}^T(\bar{s}) \geq 0$ for all $s \in \mathbb{C}^+$; see [3]. This property is, further, equivalent to the solvability of the positive real Lur'e equations

$$\begin{bmatrix} AX + XA^T & XC^T - B \\ CX - B^T & -D - D^T \end{bmatrix} = - \begin{bmatrix} L \\ F \end{bmatrix} \begin{bmatrix} L \\ F \end{bmatrix}^T \quad (2.29)$$

and

$$\begin{bmatrix} A^T Y + YA & YB - C^T \\ B^T Y - C & -D^T - D \end{bmatrix} = - \begin{bmatrix} K^T \\ J^T \end{bmatrix} \begin{bmatrix} K^T \\ J^T \end{bmatrix}^T. \quad (2.30)$$

The *positive real Gramians* of system (2.1) are then defined as the minimal solutions

$$X^{\text{PR}} = X_{\min}, \quad Y^{\text{PR}} = Y_{\min}$$

of these equations. In positive real balancing, system (2.1) is transformed into the coordinates such that $X^{\text{PR}} = Y^{\text{PR}} = \text{diag}(\sigma_1^{\text{PR}}, \dots, \sigma_n^{\text{PR}})$ with the *positive real characteristic values* σ_i^{PR} ordered decreasingly. Applying the square root balanced truncation method as in Algorithm 2.1 with X and Y replaced by X^{PR} and Y^{PR} , respectively, we get the reduced-order system (2.2) which is passive and satisfies the error bound

$$\|\mathbf{G} - \widehat{\mathbf{G}}\|_{\mathcal{H}_\infty} \leq 2\|(D + D^T)^{-1}\|_2 \|\mathbf{G} + D^T\|_{\mathcal{H}_\infty} \|\widehat{\mathbf{G}} + D^T\|_{\mathcal{H}_\infty} \sum_{i=r+1}^n \sigma_i^{\text{PR}}$$

provided $D + D^T$ is nonsingular; see [60, 92].

2.3.2 Bounded real balancing

An important property of network systems in the scattering form [3] is *contractivity*. This property corresponds to dissipativity with respect to the supply rate

$$w(u(t), y(t)) = -y^T(t)y(t) + u^T(t)u(t)$$

obtained from (2.25) by taking $Q = -I_p$, $S = 0$ and $R = I_m$. The controllable system (2.1) is contractive if and only if its transfer function \mathbf{G} is *bounded real*, i. e., \mathbf{G} is analytic in \mathbb{C}^+ and $I - \mathbf{G}^T(\bar{s})\mathbf{G}(s) \geq 0$ for all $s \in \mathbb{C}^+$; see [3]. Such systems are useful, for example, in L_2 -gain constraint controller design [50]. To verify contractivity, we can also use the bounded real Lur'e equations

$$\begin{bmatrix} AX + XA^T + BB^T & XC^T + BD^T \\ CX + DB^T & DD^T - I \end{bmatrix} = - \begin{bmatrix} L \\ F \end{bmatrix} \begin{bmatrix} L \\ F \end{bmatrix}^T$$

and

$$\begin{bmatrix} A^T Y + YA + C^T C & YB + C^T D \\ B^T Y + D^T C & D^T D - I \end{bmatrix} = - \begin{bmatrix} K^T \\ J^T \end{bmatrix} \begin{bmatrix} K^T \\ J^T \end{bmatrix}^T.$$

Their minimal solutions define the *bounded real Gramians*

$$X^{\text{BR}} = X_{\min}, \quad Y^{\text{BR}} = Y_{\min}.$$

Transforming the LTI system (2.1) into a bounded real balanced form such that $X^{\text{BR}} = Y^{\text{BR}} = \text{diag}(\sigma_1^{\text{BR}}, \dots, \sigma_n^{\text{BR}})$ and truncating the states corresponding to small *bounded real characteristic values* σ_i^{BR} results in a contractive reduced-order model (2.2) satisfying the error bound

$$\|\mathbf{G} - \widehat{\mathbf{G}}\|_{\mathcal{H}_\infty} \leq 2 \sum_{i=r+1}^n \sigma_i^{\text{BR}}.$$

These properties of the bounded real balanced truncation method were proved in [94].

It is well known that the square transfer function $\mathbf{G}(s)$ is bounded real if and only if its Moebius transform given by $\mathbf{G}_M(s) = (\mathbf{G}(s) - I)(\mathbf{G}(s) + I)^{-1}$ is positive real [39]. Note that this transform coincides with its inverse, i. e.,

$$\mathbf{G}(s) = (\mathbf{G}_M(s) - I)(\mathbf{G}_M(s) + I)^{-1}.$$

This relation leads to another model reduction method which preserves contractivity (resp. passivity). It consists in computing a reduced-order system

$$\widetilde{\mathbf{G}}(s) = (\widetilde{\mathbf{G}}_M(s) - I)(\widetilde{\mathbf{G}}_M(s) + I)^{-1},$$

where the approximation $\widetilde{\mathbf{G}}_M(s)$ is obtained by the positive real (resp. bounded real) balanced truncation applied to the Moebius-transformed system $\mathbf{G}_M(s)$; see [108] for details.

2.3.3 Linear-quadratic Gaussian balancing

Although Lyapunov-based balanced truncation is a well-established model reduction method, it suffers from some limitations when applied to controller reduction. For such a problem, the LQG balanced truncation approach was developed in [72, 93, 137] which can also be applied to unstable systems and guarantees closed-loop stability with the reduced-order controller. This approach relies on the supply rate

$$w(u(t), y(t)) = y^T(t)y(t) + u^T(t)u(t)$$

obtained from (2.25) by setting $Q = I_p$, $S = 0$ and $R = I_m$. For the linear quadratic optimal regulator problem

$$J(x_0) = \min_{\substack{u \in L_2(0, \infty; \mathbb{R}^m) \\ x(0) = x_0 \\ \lim_{t \rightarrow \infty} x(t) = 0}} \int_0^\infty w(u(\tau), y(\tau)) \, d\tau,$$

the optimal cost is given by $J(x_0) = x_0^T Y_+ x_0$, where (Y_+, K_+, L_+) is the stabilizing solution of the Lur'e equation

$$\begin{bmatrix} A^T Y + Y A + C^T C & Y B + C^T D \\ B^T Y + D^T C & I + D^T D \end{bmatrix} = \begin{bmatrix} K^T \\ J^T \end{bmatrix} \begin{bmatrix} K^T \\ J^T \end{bmatrix}^T.$$

Furthermore, the stabilizing solution (X_+, L_+, F_+) of the dual Lur'e equation

$$\begin{bmatrix} A X + X A^T + B B^T & X C^T + B D^T \\ C X + D B^T & I + D D^T \end{bmatrix} = \begin{bmatrix} L \\ F \end{bmatrix} \begin{bmatrix} L \\ F \end{bmatrix}^T$$

can be used to describe the optimal filter cost. Note that the stabilizing solutions are characterized by the property that all finite eigenvalues of the pencils

$$\begin{bmatrix} \lambda I - A & -B \\ -K_+ & -J_+ \end{bmatrix}, \quad \begin{bmatrix} \lambda I - A & -L_+ \\ -C & -F_+ \end{bmatrix}$$

have negative real part. The *LQG Gramians* can now be defined as

$$X^{\text{LQG}} = X_+, \quad Y^{\text{LQG}} = Y_+.$$

In LQG balanced coordinates, it holds $X^{\text{LQG}} = Y^{\text{LQG}} = \text{diag}(\sigma_1^{\text{LQG}}, \dots, \sigma_n^{\text{LQG}})$, where σ_i^{LQG} are called *LQG characteristic values*. The reduced-order model (2.2) is then determined by projection onto the subspace corresponding to r dominant LQG characteristic values. Since (2.1) and (2.2) are not necessarily asymptotically stable, the \mathcal{H}_∞ -norm of the error $\mathbf{G} - \widehat{\mathbf{G}}$ is generally not defined. In [91], an error bound in the gap metric was presented. It is based on the normalized left coprime factorization $\mathbf{G}(s) = \mathbf{M}^{-1}(s)\mathbf{N}(s)$, where

$$[\mathbf{N}(s), \mathbf{M}(s)] = [0, -I] \begin{bmatrix} sI - A & -L_+ \\ -C & -F_+ \end{bmatrix}^{-1} \begin{bmatrix} B & 0 \\ D & I \end{bmatrix}$$

is the stable rational function. The LQG balanced truncation can then be interpreted as the classical balanced truncation applied to the system $[\mathbf{N}, \mathbf{M}] \in \mathcal{H}_\infty$. The resulting reduced-order system $[\widehat{\mathbf{N}}, \widehat{\mathbf{M}}]$ is stable and provides the normalized left coprime factorization of $\widehat{\mathbf{G}}$, i. e., $\widehat{\mathbf{G}}(s) = \widehat{\mathbf{M}}^{-1}(s)\widehat{\mathbf{N}}(s)$. Then the gap metric error bound is given by

$$\|[\mathbf{N}, \mathbf{M}] - [\widehat{\mathbf{N}}, \widehat{\mathbf{M}}]\|_{\mathcal{H}_\infty} \leq 2 \sum_{i=r+1}^n \frac{\sigma_i^{\text{LQG}}}{\sqrt{1 + (\sigma_i^{\text{LQG}})^2}};$$

see [91] for the proof.

2.3.4 Stochastic balancing

Stochastic balanced truncation was first introduced in [38] for stochastic processes and further studied in [55, 56, 63, 136]. This approach belongs to a class of relative error methods which attempt to minimize the relative error $\|\mathbf{G}^{-1}(\mathbf{G} - \widehat{\mathbf{G}})\|_{\mathcal{H}_\infty}$. It does not rely on a special form of the supply rate any more but rather on spectral factors as defined below.

Let $\Phi(s) = \mathbf{Z}(s) + \mathbf{Z}^T(-s)$ be the *power spectrum* of the positive real transfer function $\mathbf{Z}(s) = \mathbf{C}_Z(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}_Z + \mathbf{D}_Z$ with a minimal realization $(\mathbf{A}, \mathbf{B}_Z, \mathbf{C}_Z, \mathbf{D}_Z)$. Furthermore, we consider the *left spectral factor* $\mathbf{V}(s)$ and the *right spectral factor* $\mathbf{W}(s)$ of $\Phi(s)$ satisfying

$$\Phi(s) = \mathbf{V}(s)\mathbf{V}^T(-s) = \mathbf{W}^T(-s)\mathbf{W}(s).$$

Solving the positive real Lur'e equations

$$\begin{bmatrix} \mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{A}^T & \mathbf{X}\mathbf{C}_Z^T - \mathbf{B}_Z \\ \mathbf{C}_Z\mathbf{X} - \mathbf{B}_Z^T & -\mathbf{D}_Z - \mathbf{D}_Z^T \end{bmatrix} = - \begin{bmatrix} \mathbf{L}_Z \\ \mathbf{F}_Z \end{bmatrix} \begin{bmatrix} \mathbf{L}_Z \\ \mathbf{F}_Z \end{bmatrix}^T \quad (2.31)$$

and

$$\begin{bmatrix} \mathbf{A}^T\mathbf{Y} + \mathbf{Y}\mathbf{A} & \mathbf{Y}\mathbf{B}_Z - \mathbf{C}_Z^T \\ \mathbf{B}_Z^T\mathbf{Y} - \mathbf{C}_Z & -\mathbf{D}_Z^T - \mathbf{D}_Z \end{bmatrix} = - \begin{bmatrix} \mathbf{K}_Z^T \\ \mathbf{J}_Z^T \end{bmatrix} \begin{bmatrix} \mathbf{K}_Z^T \\ \mathbf{J}_Z^T \end{bmatrix}^T \quad (2.32)$$

for $(\mathbf{X}, \mathbf{L}_Z, \mathbf{F}_Z)$ and $(\mathbf{Y}, \mathbf{K}_Z, \mathbf{J}_Z)$, respectively, the spectral factors can be realized as

$$\mathbf{V}(s) = \mathbf{C}_Z(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{L}_Z + \mathbf{F}_Z, \quad (2.33)$$

$$\mathbf{W}(s) = \mathbf{K}_Z(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}_Z + \mathbf{J}_Z; \quad (2.34)$$

see [56] for details. Then the balanced stochastic realization is obtained by performing a state-space transformation of the realizations of $\mathbf{Z}(s)$, $\mathbf{V}(s)$ and $\mathbf{W}(s)$ such that the controllability Gramian of $\mathbf{V}(s)$ is equal to the observability Gramian of $\mathbf{W}(s)$. These Gramians solve the Lyapunov equations

$$\mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{A}^T = -\mathbf{L}_Z\mathbf{L}_Z^T, \quad (2.35)$$

$$\mathbf{A}^T\mathbf{Y} + \mathbf{Y}\mathbf{A} = -\mathbf{K}_Z^T\mathbf{K}_Z, \quad (2.36)$$

respectively.

Depending on whether the system $\mathbf{V}(s)$ or $\mathbf{W}(s)$ has to be reduced, one obtains two different model reduction methods: *left spectral factor balanced truncation* and *right spectral factor balanced truncation*. In the first method, given a square nonsingular transfer function $\mathbf{G}(s) = \mathbf{V}(s)$ as in (2.33), where all eigenvalues of \mathbf{A} have negative real part, we first calculate its controllability Gramian \mathbf{X} by solving (2.35). Then using (2.31), we find

$$\mathbf{B}_Z = \mathbf{X}\mathbf{C}_Z^T + \mathbf{L}_Z\mathbf{F}_Z^T, \quad \mathbf{D}_Z + \mathbf{D}_Z^T = \mathbf{F}_Z\mathbf{F}_Z^T.$$

Inserting these matrices into the Lur'e equation (2.32), we determine, finally, the minimal solution Y_{\min} of (2.32). Note that this matrix is also the observability Gramian of \mathbf{W} since it satisfies the Lyapunov equation (2.36). We use now the *stochastic Gramians*

$$X^{\text{ST}} = X, \quad Y^{\text{ST}} = Y_{\min}$$

to define the stochastic balanced realization of $\mathbf{G}(s) = \mathbf{V}(s)$ such that X^{ST} and Y^{ST} are equal and diagonal, i. e., $X^{\text{ST}} = Y^{\text{ST}} = \text{diag}(\sigma_1^{\text{ST}}, \dots, \sigma_n^{\text{ST}})$. Reducing this realization by truncating the states corresponding to small *stochastic characteristic values* σ_i^{ST} , we obtain an approximation $\widehat{\mathbf{G}}(s)$ which satisfies the relative error bound

$$\|\mathbf{G}^{-1}(\mathbf{G} - \widehat{\mathbf{G}})\|_{\mathcal{H}_{\infty}} \leq \prod_{i=r+1}^n \frac{1 + \sigma_i^{\text{ST}}}{1 - \sigma_i^{\text{ST}}} - 1,$$

derived in [56]. Note that the stochastic characteristic values satisfy $0 \leq \sigma_i^{\text{ST}} \leq 1$. Moreover, if r exceeds the number of σ_i^{ST} with $\sigma_i^{\text{ST}} = 1$, then $\widehat{\mathbf{G}}(s)$ preserves zeros of $\mathbf{G}(s)$ in \mathbb{C}^+ ; see [55]. This property immediately implies that, if $\mathbf{G}(s)$ is *minimum phase*, i. e., it has no zeros in \mathbb{C}^+ , then $\widehat{\mathbf{G}}(s)$ is also minimum phase.

For given $\mathbf{G}(s) = \mathbf{W}(s)$ as in (2.34), the stochastic balanced realization can be determined by first solving the Lyapunov equation (2.36) for Y and then computing the minimal solution X_{\min} of the Lur'e equation (2.31) with

$$C_Z = B_Z^T Y + J_Z^T K_Z, \quad D_Z^T + D_Z = J_Z^T J_Z$$

obtained from (2.32). Balancing the Gramian pair (X_{\min}, Y) and performing model reduction, we obtain an approximation $\widehat{\mathbf{G}}(s)$ with similar properties as in the previous method.

The balancing-related model reduction methods considered above require solving Lur'e or Riccati matrix equations. Numerical methods for large-scale Lur'e equations presented in [102, 103] are based on deflating subspaces of an associated even matrix pencil. Furthermore, an extension of the ADI method to Lur'e equations was proposed in [85]. For Riccati equations, many different numerical methods have been developed over the last 30 years; see [30]. Let us just mention some of them relying on a low-rank approximation: Krylov subspace methods [67, 121], Newton's method [19, 24, 28], and ADI-type methods [12, 81, 84].

2.3.5 Singular perturbation approximation

Another variant of classical balancing relies on partitioning the balanced system (2.17) into slow and fast dynamics. Instead of setting $x_2 = 0$ and obtaining the reduced-order model via (2.18), consider the dynamics of x_2 to reach a steady state such that $\dot{x}_2(t) = 0$.

As a consequence, the partitioning (2.17) allows to explicitly solve the second equation for the variable x_2 . In particular, we find that

$$x_2(t) = -A_{22}^{-1}A_{21}x_1(t) - A_{22}^{-1}B_2u(t).$$

Inserting this expression into the first equation leads to

$$\begin{aligned}\dot{x}_1(t) &= (A_{11} - A_{12}A_{22}^{-1}A_{21})x_1(t) + (B_1 - A_{12}A_{22}^{-1}B_2)u(t), \\ \hat{y}(t) &= (C_1 - C_2A_{22}^{-1}A_{21})x_1(t) - C_2A_{22}^{-1}B_2u(t) + Du(t)\end{aligned}\tag{2.37}$$

from which we obtain another reduced-order model of the form

$$\begin{aligned}\hat{A} &= (A_{11} - A_{12}A_{22}^{-1}A_{21}), & \hat{B} &= B_1 - A_{12}A_{22}^{-1}B_2, \\ \hat{C} &= C_1 - C_2A_{22}^{-1}A_{21}, & \hat{D} &= -C_2A_{22}^{-1}B_2 + D.\end{aligned}$$

Let us emphasize that, for a controllable and observable linear system (2.17), the matrix A_{22} is guaranteed to be regular if Σ_1 and Σ_2 have no common diagonal entries; see [83]. It has also been shown in [83] that the reduced-order model (2.37) satisfies the a priori error bound (2.19). Let us also mention the interesting fact that singular perturbation approximation can be interpreted as classical balanced truncation applied to the *reciprocal system*

$$\bar{A} = A^{-1}, \quad \bar{B} = A^{-1}B, \quad \bar{C} = CA^{-1}.$$

Moreover, for the transfer function of the original system, we find that

$$\begin{aligned}G(0) &= -[C_1 \quad C_2] \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1} \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} + D \\ &= -[C_1 \quad C_2] \begin{bmatrix} \hat{A}^{-1} & -\hat{A}^{-1}A_{12}A_{22}^{-1} \\ -A_{22}^{-1}A_{21}\hat{A}^{-1} & A_{22}^{-1}A_{21}\hat{A}^{-1}A_{12}A_{22}^{-1} + A_{22}^{-1} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} + D \\ &= -[C_1 \quad C_2] \begin{bmatrix} \hat{A}^{-1}\hat{B} \\ -A_{22}^{-1}A_{21}\hat{A}^{-1}\hat{B} + A_{22}^{-1}B_2 \end{bmatrix} + D \\ &= -\hat{C}\hat{A}^{-1}\hat{B} + \hat{D} = \hat{G}(0).\end{aligned}$$

The above relation additionally implies that singular perturbation approximation yields a reduced-order model that is exact at the frequency $s = 0$. We refer to the original reference [83], where more details of the method can be found.

2.3.6 Cross-Gramian balanced truncation

With the intention of studying controllability and observability concepts at the same time, in [44] the authors introduced (for the SISO case) the so-called *cross-Gramian*

$$X_c = \int_0^\infty e^{At} B C e^{At} dt.$$

Generalizations to symmetric MIMO systems were subsequently considered in [45]. Similar to the controllability and observability Gramians, the matrix X_c can be shown to satisfy the Sylvester equation,

$$AX_c + X_cA + BC = 0.$$

Moreover, the eigenvalues of the cross-Gramian are invariant under state space transformations and coincide with the Hankel singular values of system (2.1). As a consequence, it is possible to perform the balancing step with respect to X_c instead of X and Y . The main advantage is that only one linear matrix equation has to be solved which allows for a computationally more tractable method; see, e.g., [7, 124]. Furthermore, the cross-Gramian and some empirical variants thereof can be computed by means of simulated trajectories of the system; see [68] for a more detailed overview and comparison.

2.4 Balancing for DAEs

Control problems governed by DAEs arise in a variety of practical applications including circuit simulation, computational electromagnetics, fluid dynamics, mechanical and chemical engineering; see [18, Chapters 2, 4 and 5] for practical examples. Unlike ordinary differential equations (ODEs), DAEs contain (hidden) algebraic constraints restricting the solution to a manifold. These constraints usually result from physical laws as, for example, conservation laws in incompressible (Navier–)Stokes equations and Maxwell’s equations, and Kirchhoff’s laws in network problems, or from geometric and kinematic constraints in mechanical systems. For solvability of DAE control systems, it is required that the initial values satisfy certain consistency conditions imposed by algebraic constraints and that the input function or some of its components are sufficiently smooth. In [31, 76, 77], different frameworks have been presented for structural analysis and numerical treatment of linear and nonlinear DAEs. Also, different index concepts have been introduced there to characterize various structural properties of DAEs. In general, a high index characterizes the difficulty of analyzing a DAE theoretically and numerically.

We consider a linear DAE control system,

$$\begin{aligned} E\dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{2.38}$$

where $E, A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times n}$ and $D \in \mathbb{R}^{p \times m}$. If E is nonsingular, this system can be transformed into the standard state-space form (2.1) by inversion of E . Control systems of the form (2.38) with a singular matrix E are also known as *descriptor systems*, *generalized state-space systems* or *singular systems*. Model reduction of such

systems by balanced truncation was first considered in [99] and further improved in [82, 127]. We also refer to [29] for a comprehensive survey on model reduction of DAEs.

Assuming that a pencil $\lambda E - A$ is *regular*, i. e., $\det(\lambda E - A) \neq 0$ for some $\lambda \in \mathbb{C}$, we can introduce a transfer function for (2.38) given by $\mathbf{G}(s) = C(sE - A)^{-1}B + D$. Comparing it with the transfer function (2.5) of the standard-state space system (2.1), one might think that all control-theoretic concepts and related model reduction methods for ODEs can be extended to DAEs just by replacing the identity matrix with E . However, in practice, it does not work, since DAEs, especially higher index DAEs, exhibit much more complex behavior than ODEs.

A useful tool for investigating the structural properties of linear DAEs is a *Weierstrass canonical form*. For a regular pencil $\lambda E - A$, it is given by

$$E = T_l \begin{bmatrix} I_{n_f} & 0 \\ 0 & N \end{bmatrix} T_r, \quad A = T_l \begin{bmatrix} J & 0 \\ 0 & I_{n_{\infty}} \end{bmatrix} T_r, \quad (2.39)$$

where $T_l, T_r \in \mathbb{R}^{n \times n}$ nonsingular matrices and $N \in \mathbb{R}^{n_{\infty} \times n_{\infty}}$ is nilpotent with nilpotency index ν defined as a smallest integer such that $N^{\nu} = 0$. This quantity also defines the (differentiation) index of the DAE system (2.38). The eigenvalues of $J \in \mathbb{R}^{n_f \times n_f}$ are the finite eigenvalues of the pencil $\lambda E - A$, and N corresponds to the eigenvalue at infinity. Using the Weierstrass canonical form (2.39), we can decompose (2.38) into the differential part (also called the slow subsystem) and the algebraic part (also called the fast subsystem). The differential part is in the ODE form and, therefore, it can be reduced in a standard way by balancing and truncation. On the contrary, the algebraic part determines a solution manifold which has to be preserved in the reduced-order model. This can be achieved if we only remove redundant equations and those state components which do not contribute to the input-output energy transfer. As a result, we get a minimal realization of the algebraic part.

For the differential part, we introduce the *proper controllability* and *observability Gramians* X_{pr} and Y_{pr} as symmetric, positive semi-definite solutions of the projected continuous-time Lyapunov equations

$$\begin{aligned} AX_{pr}E^T + EX_{pr}A^T &= -P_l BB^T P_l^T, & X_{pr} &= P_r X_{pr} P_r^T, \\ A^T Y_{pr}E + E^T Y_{pr}A &= -P_r^T C^T C P_r, & Y_{pr} &= P_l^T Y_{pr} P_l, \end{aligned} \quad (2.40)$$

where

$$P_l = T_l \begin{bmatrix} I_{n_f} & 0 \\ 0 & 0 \end{bmatrix} T_l^{-1}, \quad P_r = T_r^{-1} \begin{bmatrix} I_{n_f} & 0 \\ 0 & 0 \end{bmatrix} T_r$$

are the spectral projectors onto the left and right deflating subspaces of the pencil $\lambda E - A$ corresponding to the finite eigenvalues. Such Gramians exist and are unique if the DAE system (2.38) is asymptotically stable, i. e., all finite eigenvalues of $\lambda E - A$ have

negative real part. The proper Gramians have a similar energy interpretation as the Gramians in the standard state-space case [127]. For the algebraic part, we define the *improper controllability* and *observability Gramians* X_{im} and Y_{im} as symmetric, positive semi-definite solutions of the projected discrete-time Lyapunov equations

$$\begin{aligned} AX_{im}A^T - EX_{im}E^T &= -Q_l BB^T Q_l^T, & X_{im} &= Q_r X_{im} Q_r^T, \\ A^T Y_{im} A - E^T Y_{im} E &= -Q_r^T C^T C Q_r, & Y_{im} &= Q_l^T Y_{im} Q_l, \end{aligned} \quad (2.41)$$

where $Q_l = I - P_l$ and $Q_r = I - P_r$ are the spectral projectors onto the left and right deflating subspaces of $\lambda E - A$ corresponding to the eigenvalue at infinity. These two pairs of the Gramians provide two sets of Hankel singular values of the DAE system (2.38). Let

$$\begin{aligned} X_{pr} &= L_{X,pr}^T L_{X,pr}, & Y_{pr} &= L_{Y,pr}^T L_{Y,pr}, \\ X_{im} &= L_{X,im}^T L_{X,im}, & Y_{im} &= L_{Y,im}^T L_{Y,im} \end{aligned}$$

be the Cholesky factorizations of the Gramians. Then the *proper Hankel singular values*, denoted by σ_j , are defined as the largest n_f singular values of the matrix $L_{X,pr} E^T L_{Y,pr}^T$, and the *improper Hankel singular values*, denoted by θ_j , are defined as the largest n_{∞} singular values of the matrix $L_{X,im} A^T L_{Y,im}^T$. The SVDs of these matrices can be used to transform (2.38) into a balanced form such that the Gramians of the transformed system satisfy

$$\tilde{X}_{pr} + \tilde{X}_{im} = \tilde{Y}_{pr} + \tilde{Y}_{im} = \text{diag}(\sigma_1, \dots, \sigma_{n_f}, \theta_1, \dots, \theta_{n_{\infty}}).$$

Then a reduced-order model can be computed by truncation of the state components of the balanced system corresponding to small proper Hankel singular values and zero improper Hankel singular values. Note that the truncation of the states corresponding to non-zero improper Hankel singular values may lead to an inaccurate and physically meaningless approximation; see [82, 130] for some examples. An extension of the square root balanced truncation method to DAE control systems is presented in Algorithm 2.2.

One can show that the resulting reduced-order system is balanced, asymptotically stable and has an index which does not exceed the index of the original system [127]. Moreover, we have the error estimate

$$\|\mathbf{G} - \widehat{\mathbf{G}}\|_{\mathcal{H}_{\infty}} \leq 2 \sum_{i=r_f+1}^{n_f} \sigma_i.$$

For solving the projected continuous-time Lyapunov equations (2.40), one can use the low-rank generalized ADI method [129] or (rational) Krylov subspace methods [131]. The projected discrete-time Lyapunov equations (2.41) can be solved using the generalized Smith method [129]. The main difficulty in all these methods is the determi-

Algorithm 2.2: Balanced truncation for DAE control systems.**Require:** Original system E, A, B, C, D ; error tolerance tol .**Ensure:** Reduced system $\hat{E}, \hat{A}, \hat{B}, \hat{C}, \hat{D}$ with $\|\mathbf{G} - \hat{\mathbf{G}}\|_{\mathcal{H}_\infty} \leq \text{tol}$.

- 1: Compute the solutions $X_{pr} = L_{X,pr}^T L_{X,pr}$ and $Y_{pr} = L_{Y,pr}^T L_{Y,pr}$ of (2.40).
- 2: Compute the solutions $X_{im} = L_{X,im}^T L_{X,im}$ and $Y_{im} = L_{Y,im}^T L_{Y,im}$ of (2.41).
- 3: Compute the SVDs

$$L_{X,pr} E^T L_{Y,pr}^T = [U_{pr,1} \quad U_{pr,2}] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} Z_{pr,1}^T \\ Z_{pr,2}^T \end{bmatrix},$$

$$L_{X,im} A^T L_{Y,im}^T = [U_{im,1} \quad U_{im,2}] \begin{bmatrix} \Theta_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Z_{im,1}^T \\ Z_{im,2}^T \end{bmatrix},$$

where $\Sigma_1 = \text{diag}(\sigma_1, \dots, \sigma_{r_f})$ and $\Sigma_2 = \text{diag}(\sigma_{r_f+1}, \dots, \sigma_{n_f})$ with r_f chosen such that

$$2 \sum_{i=r_f+1}^{n_f} \sigma_i \leq \text{tol}, \text{ and } \Theta_1 = \text{diag}(\theta_1, \dots, \theta_{r_\infty}) \text{ is positive definite.}$$

- 4: Define

$$V = [L_{X,pr}^T U_{pr,1} \Sigma_1^{-\frac{1}{2}}, L_{X,im}^T U_{im,1} \Theta_1^{-\frac{1}{2}}],$$

$$W = [L_{Y,pr}^T Z_{pr,1} \Sigma_1^{-\frac{1}{2}}, L_{Y,im}^T Z_{im,1} \Theta_1^{-\frac{1}{2}}].$$

- 5: Define $\hat{E} = W^T E V$, $\hat{A} = W^T A V$, $\hat{B} = W^T B$, $\hat{C} = C V$ and $\hat{D} = D$.

nation of the spectral projectors P_r and P_∞ . For some special classes of linear DAEs such as semi-explicit DAEs of index 1 [46, 111], magneto-quasistatic systems of index 1 [73], Stokes-like DAEs of index 2 [66, 128], and constrained mechanical systems of index 2 and 3 [29, 114], it is possible to avoid the explicit construction of the projectors. By making use of a special block structure of the system matrices, such systems can be transformed into the ODE form allowing the application of standard model reduction methods. It should, however, be emphasized that the resulting ODE systems do not preserve the sparsity in the matrix coefficients. Therefore, the ODE systems will never be computed explicitly. Instead, again exploiting the system structure, all computations can be performed in terms of the original data. For other structured DAEs such as circuit equations of index 1 and 2, the required projectors have been derived in [107, 109]. In numerical implementations, however, only projector-vector products will be computed, avoiding explicitly forming the (possibly full) projector matrices and significantly reducing computational costs.

We conclude this section by referring to [29, 87, 108] for an extension of other types of balancing-based model reduction methods to DAE control systems.

2.5 Balancing for nonlinear systems

A generalization of the concept of balancing for the nonlinear case has been introduced in [117]. For summarizing the main idea, consider a nonlinear system of the form

$$\begin{aligned}\dot{x}(t) &= f(x(t)) + g(x(t))u(t), \\ y(t) &= h(x(t)),\end{aligned}\tag{2.42}$$

where $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$, $g: \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$ and $h: \mathbb{R}^n \rightarrow \mathbb{R}^p$ are smooth functions. Additionally, zero is assumed to be an equilibrium, i. e., $f(0) = 0$ and $h(0) = 0$. In analogy to the linear case, Scherpen defined the energy functionals

$$\begin{aligned}L_c(x_d) &= \min_{\substack{u \in L_2(-\infty, 0; \mathbb{R}^m) \\ x(-\infty) = 0, x(0) = x_d}} \frac{1}{2} \int_{-\infty}^0 u^T(t)u(t) dt, \\ L_o(x_0) &= \frac{1}{2} \int_0^\infty y^T(t)y(t) dt, \quad x(0) = x_0, \quad u(t) \equiv 0, \quad 0 \leq t < \infty,\end{aligned}\tag{2.43}$$

which are defined to be infinite if x_d cannot be reached from 0 or if the system is unstable, respectively. While the Gramians X and Y of a linear system satisfy the linear matrix equations (2.13) and (2.14), under suitable solvability assumptions L_c and L_o solve the partial differential equations

$$\begin{aligned}\frac{\partial L_c}{\partial x}(x)f(x) + \frac{1}{2} \frac{\partial L_c}{\partial x}(x)g(x) \left(\frac{\partial L_c}{\partial x}(x)g(x) \right)^T &= 0, \quad L_c(0) = 0, \\ \frac{\partial L_o}{\partial x}(x)f(x) + \frac{1}{2} (h(x))^T h(x) &= 0, \quad L_o(0) = 0,\end{aligned}$$

for all x in a neighborhood of the origin; see [117, Theorem 3.2]. The idea of transforming the system coordinates into balanced form and subsequently obtaining a reduced-order system by truncation remains the same. However, in general this can only be achieved locally (around the origin) by a nonlinear coordinate transform $x = \psi(\tilde{x})$, $\psi(0) = 0$. For the transformed balanced functionals \tilde{L}_c and \tilde{L}_o , it then holds

$$\begin{aligned}\tilde{L}_c(\tilde{x}) &:= L_c(\psi(\tilde{x})) = \frac{1}{2} \tilde{x}^T \tilde{x}, \\ \tilde{L}_o(\tilde{x}) &= L_o(\psi(\tilde{x})) = \frac{1}{2} \tilde{x}^T \text{diag}(\sigma_1(\tilde{x}), \dots, \sigma_n(\tilde{x})) \tilde{x},\end{aligned}$$

where the so-called *singular value functions* $\sigma_1(\tilde{x}) \geq \dots \geq \sigma_n(\tilde{x})$ extend the notion of the Hankel singular values to the nonlinear setting. In particular, it can be shown that a linearization of this approach yields the classical balancing technique applied to the linearization (around 0) of the nonlinear system (2.42). For more details on theoretical

properties such as local asymptotic stability or preservation of balanced coordinates, we refer to the original reference [117].

Since computing L_c and L_o as the solutions (or approximations thereof) of (2.43) suffers from *the curse of dimensionality*, recent work has focused on replacing L_c and L_o by algebraic approximations. Particularly for the class of *bilinear control systems*

$$\begin{aligned}\dot{x}(t) &= Ax(t) + \sum_{i=1}^m N_i x(t) u_i(t) + Bu(t), \\ y(t) &= Cx(t),\end{aligned}\tag{2.44}$$

with $A, N_i \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$ and $C \in \mathbb{R}^{p \times n}$, below we summarize an alternative way of generalizing model reduction by balanced truncation. Let us emphasize that bilinear systems not only arise in several practical applications such as nuclear fission, biology [88], the Fokker–Planck equation [64] and heat transfer processes [41], but can also be used to approximate more general nonlinear systems by a *Carleman linearization*; see [112]. Controllability and observability concepts for bilinear control systems have already been studied in [35, 71]. Their use in context of model reduction has been first discussed in [1, 2] and is based on a generalization of the integral representations (2.9) and (2.12). In particular, consider the following recursive series of time-dependent matrices:

$$\begin{aligned}X_1(t_1) &= e^{At_1} B, \\ X_k(t_1, \dots, t_k) &= [e^{At_k} N_1 X_{k-1} \quad \dots \quad e^{At_k} N_m X_{k-1}], \quad k = 2, 3, \dots,\end{aligned}$$

as well as

$$\begin{aligned}Y_1(t_1) &= C e^{At_1}, \\ Y_k(t_1, \dots, t_k) &= [(Y_{k-1} N_1 e^{At_1})^T \quad \dots \quad (Y_{k-1} N_m e^{At_1})^T]^T, \quad k = 2, 3, \dots,\end{aligned}$$

and define

$$\begin{aligned}X &= \sum_{k=1}^{\infty} \int_0^{\infty} \dots \int_0^{\infty} X_k(t_1, \dots, t_k) X_k(t_1, \dots, t_k)^T dt_1 \dots dt_k, \\ Y &= \sum_{k=1}^{\infty} \int_0^{\infty} \dots \int_0^{\infty} Y_k^T(t_1, \dots, t_k) Y_k(t_1, \dots, t_k) dt_1 \dots dt_k.\end{aligned}\tag{2.45}$$

If X and Y exist, it can be shown, see, e. g., [1, Theorem 1], that they satisfy the generalized Lyapunov equations

$$\begin{aligned}AX + XA^T + \sum_{i=1}^m N_i X N_i^T + BB^T &= 0, \\ A^T Y + YA + \sum_{i=1}^m N_i^T Y N_i + C^T C &= 0.\end{aligned}\tag{2.46}$$

The main advantage of using these algebraic Gramians X and Y is that a global static coordinate transform $\tilde{x} = Tx$ can be used to transform (2.44) into a balanced form. The construction is completely analogous to the linear case, i. e., the transformation matrix $T \in \mathbb{R}^{n \times n}$ can be obtained via $T = \Sigma^{-\frac{1}{2}} Z^T L_Y$, where

$$X = L_X^T L_X, \quad Y = L_Y^T L_Y, \quad L_X L_Y^T = U \Sigma Z^T.$$

We summarize the main steps of balanced truncation for bilinear control systems in Algorithm 2.3.

Algorithm 2.3: Balanced truncation for bilinear control systems.

Require: Original system A, B, C and N_1, \dots, N_m .

Ensure: Reduced system $\hat{A}, \hat{B}, \hat{C}$ and $\hat{N}_1, \dots, \hat{N}_m$.

- 1: Compute the solutions $X = L_X^T L_X$ and $Y = L_Y^T L_Y$ of (2.46).
 - 2: Compute the SVD of $L_X L_Y^T$ as in (2.21).
 - 3: Define $V = L_X^T U_1 \Sigma_1^{-\frac{1}{2}}$ and $W = L_Y^T Z_1 \Sigma_1^{-\frac{1}{2}}$.
 - 4: Define $\hat{A} = W^T A V$, $\hat{B} = W^T B$, $\hat{C} = C V$ and $\hat{N}_i = W^T N_i V$, $i = 1, \dots, m$.
-

A few remarks on the properties of the method are in order. The quadratic forms $x^T X^{-1} x$, $x^T Y x$ and some related variants have been compared to the energy functionals L_c and L_o defined in (2.43) in several publications, e. g., [13, 52, 53, 54]. In [13, Propositions 3.5 and 3.8], it is shown that the input energy L_c (resp. the output energy L_o) can locally (around the origin) be bounded from below (resp. from above) by means of $x^T X^{-1} x$ (resp. $x^T Y x$). Similar to the linear case, preservation of system stability is guaranteed for the reduced-order model; see [14]. Let us mention that the notion of stability used in the latter work is based on the eigenvalues of the generalized Lyapunov operator and is stronger than asymptotic stability of the system matrix A ; see [36]. Recently, a further pair of generalized algebraic Gramians has been suggested in [104]. These Gramians are inspired by similar quantities introduced in the context of model reduction of linear stochastic systems in [37]. In contrast to (2.46), here nonlinear matrix inequalities have to be solved. The benefit of these Gramians is that an \mathcal{H}_∞ -type error bound can be shown; see [104, Theorem 4.1]. However, from a computational point of view, even for large-scale systems, it is easier to solve generalized matrix equations of the form (2.46). For references on (low-rank) approximation techniques, we refer to [10, 11, 36, 75].

As a further field of current research, we mention balancing-based model reduction for *quadratic–bilinear* control systems of the form

$$\begin{aligned} \dot{x}(t) &= Ax(t) + H(x(t) \otimes x(t)) + \sum_{i=1}^m N_i x(t) u_i(t) + Bu(t), \\ y(t) &= Cx(t), \end{aligned}$$

where A, N_i, B and C are as before and, moreover, $H \in \mathbb{R}^{n \times n^2}$. The interest in such systems goes back to the results obtained in [58, 59], where the author shows that a certain class of smooth, nonlinear, control affine systems can equivalently be expressed in such a form. In [15], the authors follow an approach that is similar to the bilinear version described above. In particular, a pair of nonlinear coupled generalized Lyapunov equations

$$\begin{aligned} AX + XA^T + H(X \otimes X)H^T + \sum_{i=1}^m N_i X N_i^T + BB^T &= 0, \\ A^T Y + YA + \mathcal{H}^{(2)}(X \otimes Y)(\mathcal{H}^{(2)})^T + \sum_{i=1}^m N_i^T Y N_i + C^T C &= 0 \end{aligned}$$

is derived and also compared to the energy functionals L_c and L_o . Here, $\mathcal{H}^{(2)}$ denotes the mode-2 matricization of the three-dimensional tensor $\mathcal{H} \in \mathbb{R}^{n \times n \times n}$ associated to the matrix $H \in \mathbb{R}^{n \times n^2}$. Moreover, an approximation procedure based on a series of generalized linear matrix equations is discussed in [15] and numerically investigated for several nonlinear PDEs.

2.6 Other balancing issues

2.6.1 Balanced truncation for discrete-time systems

The balanced truncation model reduction method can also be formulated for discrete-time control systems of the form

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k, \\ y_k &= Cx_k + Du_k, \end{aligned} \tag{2.47}$$

with the state x_k , control u_k and output y_k . For such systems, instead of the continuous-time Lyapunov equations (2.13) and (2.14), one has to solve the *discrete-time Lyapunov equations*

$$AXA^T - X = -BB^T, \quad A^T YA - Y = -C^T C \tag{2.48}$$

for the controllability and observability Gramians X and Y provided all eigenvalues of A lie inside the unit disc. Then the reduced-order model is computed analogously to the continuous-time case, by balancing X and Y and truncating the states corresponding to small eigenvalues of the Gramians. The preservation of stability and error bound similar to (2.19) were proved in [69, 100]. Other balancing-related techniques for discrete-time systems were considered in [38, 92].

The Lyapunov-based balanced truncation approach was also extended to periodic discrete-time systems in [42, 133, 134, 135, 136]. The Gramians for such systems can be determined as solutions to the periodic discrete-time Lyapunov equations. Using a lifted representation [125] for the periodic system, these equations can be written in the form (2.48) with block structured matrices A , B and C . Efficient numerical methods for such structured equations were presented in [74]. They exploit the block sparsity in the lifted system matrices and rely on low-rank techniques. Model reduction by balanced truncation for (periodic) discrete-time descriptor systems was considered in [20, 21, 34, 126].

2.6.2 Balanced truncation for second-order systems

Model reduction of second-order control systems has received a lot of attention because of their importance in structural mechanics, acoustics and vibration problems; see [18, Chapters 2 and 3]. Balanced truncation for such systems was first considered in [86] and then further investigated in [27, 33, 106]. We consider the second-order system

$$\begin{aligned} M\ddot{q}(t) + D\dot{q}(t) + Kq(t) &= B_2u(t), \\ C_1\dot{q}(t) + C_0q(t) &= y(t), \end{aligned} \quad (2.49)$$

where $M, D, K \in \mathbb{R}^{n \times n}$ are the mass, damping and stiffness matrices. This system can be written as a first-order system

$$\begin{aligned} E\dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t), \end{aligned} \quad (2.50)$$

with $x = [q^T \quad \dot{q}^T]^T$ and

$$E = \begin{bmatrix} N & 0 \\ 0 & M \end{bmatrix}, \quad A = \begin{bmatrix} 0 & N \\ -K & -D \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ B_2 \end{bmatrix}, \quad C = [C_0 \quad C_1], \quad (2.51)$$

where N is an arbitrary nonsingular matrix. The controllability and observability Gramians of this system solve the generalized Lyapunov equations

$$EXA^T + AXE^T = -BB^T, \quad E^T YA + A^T YE = -C^T C. \quad (2.52)$$

Applying the balanced truncation method to (2.50) as described in Section 2.2, we obtain a reduced first-order model which, in general, cannot be turned into the second-order form. Ensuring the second-order structure in the reduced model often guarantees the preservation of the physical properties and allows the use of software tools specially developed for second-order systems.

Thus, we partition the Gramians X and Y into $n \times n$ blocks as

$$X = \begin{bmatrix} X_p & X_{12} \\ X_{12}^T & X_v \end{bmatrix}, \quad Y = \begin{bmatrix} Y_p & Y_{12} \\ Y_{12}^T & Y_v \end{bmatrix}.$$

Then X_p and Y_p define the *position controllability* and *observability Gramians*, and X_v and Y_v define the *velocity controllability* and *observability Gramians* of the second-order system (2.49). We refer to [33, 86] for the energy interpretation of these Gramians. By balancing one of the pairs (X_p, Y_p) , (X_p, Y_v) , (X_v, Y_p) or (X_v, Y_v) , we get the position–position, position–velocity, velocity–position or velocity–velocity balanced realizations, respectively. Considering the Cholesky factorizations

$$X_p = L_{X,p}^T L_{X,p}, \quad X_v = L_{X,v}^T L_{X,v}, \quad Y_p = L_{Y,p}^T L_{Y,p}, \quad Y_v = L_{Y,v}^T L_{Y,v},$$

the corresponding balancing transformation matrices can be determined from the SVD of the matrices $L_{X,p} L_{Y,p}^T$, $L_{X,p} M^T L_{Y,v}^T$, $L_{X,v} L_{Y,p}^T$ and $L_{X,v} M^T L_{Y,v}^T$; see [106]. This leads to four different second-order balanced truncation approaches which provide the reduced model in the second-order form,

$$\begin{aligned} \widehat{M} \ddot{\hat{q}}(t) + \widehat{D} \dot{\hat{q}}(t) + \widehat{K} \hat{q}(t) &= \widehat{B}_2 u(t), \\ \widehat{C}_1 \dot{\hat{q}}(t) + \widehat{C}_0 \hat{q}(t) &= \widehat{y}(t), \end{aligned}$$

where $\widehat{M} = W^T M V$, $\widehat{D} = W^T D V$, $\widehat{K} = W^T K V$, $\widehat{B}_2 = W^T B_2$, $\widehat{C}_0 = W^T C_0$, and $\widehat{C}_1 = W^T C_1$ with appropriate projection matrices W and V . Unlike the first-order balanced truncation, stability is not necessarily preserved in the reduced model, see [106] for some examples, and there exists no error bound. However, for symmetric second-order systems with $M = M^T > 0$, $D = D^T > 0$, $K = K^T > 0$, $C_0 = 0$ and $C_1 = B_2^T$, it was shown in [22] that choosing $N = -K$ in (2.51) one can guarantee the preservation of stability and symmetry for the position–position and velocity–velocity balanced truncation methods. Moreover, for $N = I$, the position–velocity balanced truncation also preserves stability and symmetry in the reduced-order model [106]. Finally note that the low-rank Cholesky factors of the position and velocity Gramians can be computed using the ADI method applied to (2.52) without explicit forming the double sized matrices E , A , B and C as in (2.51); see [22, 27] for details.

2.7 Numerical examples

In this section, we present two numerical examples which illustrate the applicability and the limitations of classical balanced truncation.

2.7.1 Heat equation

As a first example, let us focus a one-dimensional linear heat equation with boundary control. In particular, we consider

$$\begin{aligned} x_t &= x_{\xi\xi}, & (\xi, t) &\in (0, 1) \times (0, T), \\ x(0, t) &= 0, & t &\in (0, T), \\ x_\xi(1, t) &= u(t), & t &\in (0, T), \\ x(\xi, 0) &= 0, & \xi &\in (0, 1), \end{aligned}$$

where $x(\xi, t)$ describes the evolution of a temperature distribution on the interval $[0, 1]$. We assume that the average temperature can be measured such that the output variable $y(t)$ is given by

$$y(t) = \int_0^1 x(\xi, t) d\xi \quad \text{for } t > 0.$$

A finite difference discretization on a grid with n interior grid points and grid size $h = \frac{1}{n+1}$ then yields a finite-dimensional control system (2.1) with system matrices

$$A = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -1 \end{bmatrix}, \quad B = \frac{1}{h} \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \quad C^T = h \begin{bmatrix} 1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 1 \end{bmatrix}.$$

In our results, we have used a discretization with $n = 5000$. According to the pseudocode from Algorithm 2.1, we employed the MATLAB routine `lyapchol` to compute the solutions X and Y of the controllability and observability Lyapunov equations (2.13) and (2.14). The numerical rank of the matrix $L_X L_Y^T$ in this case was only 11. Hence, a numerically (up to machine precision) exact copy of the original system can be realized by a system of dimension $r = 11$. This is confirmed by the results from Figure 2.1. The so-called *Bode plot* of the reduced transfer function $\widehat{\mathbf{G}}(s) = \widehat{C}(sI - \widehat{A})^{-1}\widehat{B}$ cannot be distinguished from the original transfer function $\mathbf{G}(s) = C(sI - A)^{-1}B$. A similar conclusion can be drawn for the impulse response functions $h(t) = Ce^{At}B$ and $\widehat{h}(t) = \widehat{C}e^{\widehat{A}t}\widehat{B}$ presented in Figure 2.2. Figure 2.3 compares the \mathcal{H}_∞ -error corresponding to balanced reduced-order systems of dimension $r = 1, \dots, 11$ with the guaranteed error bound determined by the neglected Hankel singular values. We emphasize that the heat equation and, more general, parabolic PDEs with finite-dimensional input or output space are well-suited for model reduction by balanced truncation. For more details on a functional analytic discussion, we refer to [96].

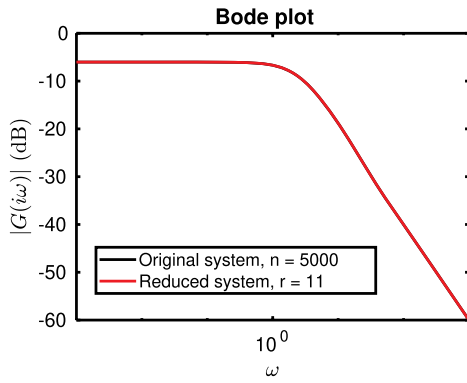


Figure 2.1: Bode plots for the 1D heat equation.

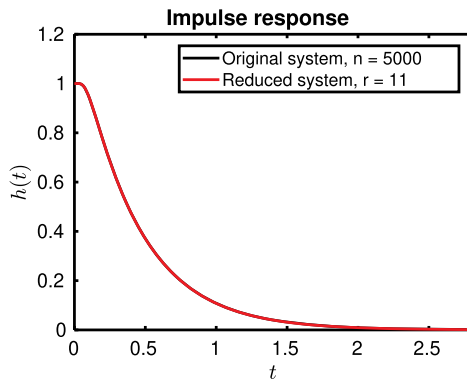


Figure 2.2: Impulse responses for the 1D heat equation.

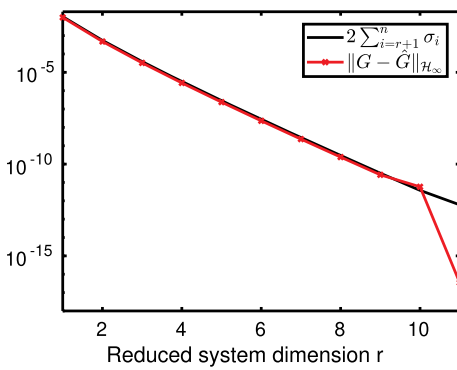


Figure 2.3: \mathcal{H}_{∞} -error for varying reduced system dimensions r for the 1D heat equation.

2.7.2 Transport equation

The second example is a one-dimensional boundary controlled transport equation and has been discussed in more detail in [96]. Let us consider

$$\begin{aligned} x_t &= -x_\xi, & (\xi, t) &\in (0, 1) \times (0, T), \\ x(0, t) &= u(t), & t &\in (0, T), \\ x(\xi, 0) &= 0, & \xi &\in (0, 1). \end{aligned}$$

We assume that the right boundary can be observed. Hence, we obtain the output variable $y(t) = x(1, t)$ for $t > 0$.

For the spatial discretization, we employ an upwind scheme and thus use a backward finite difference for the approximation of the transport term. We consider a grid with n grid points equally distributed over the interval $[h, 1]$ where $h = \frac{1}{n}$. Note that in contrast to the first example, we only eliminate the left boundary value which is prescribed by the dynamics. As a result, we obtain the following system of matrices:

$$A = \frac{1}{h} \begin{bmatrix} -1 & & & & \\ 1 & -1 & & & \\ & \ddots & \ddots & & \\ & & & 1 & -1 \end{bmatrix}, \quad B = \frac{1}{h} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad C^T = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

As mentioned in [96], the transfer function of the above transport equation is $\mathbf{G}_{\text{true}}(s) = e^{-s}$. Note that this irrational transfer function is approximated by rational transfer functions of the form $\mathbf{G}(s) = C(sI - A)^{-1}B$. In Figure 2.4, the Bode plots of the original and reduced transfer functions are shown. Note that the spatial discretization and its realization (A, B, C) already introduces a visible error. In contrast to the analytic expression $\mathbf{G}_{\text{true}}(s) = e^{-s}$ which has constant modulus 1 along the imaginary axis, the magnitude of $\mathbf{G}(s) = C(sI - A)^{-1}B$ decreases for increasing frequencies. Let us emphasize that transport equations as the one from above are generally less suitable for model reduction by balanced truncation. This is also apparent from the relatively large reduced system dimension ($r = 180$) of the numerically (up to machine precision) exact copy of the original semi-discretized system.

Since the original transfer function is given by $\mathbf{G}_{\text{true}}(s) = e^{-s}$, its underlying impulse response is $h(t) = \delta(t-1)$, i. e., a unit pulse mass at 1. Again, the results presented in Figure 2.5 underlines that the spatially discretized system only approximates the original dynamics. Nevertheless, the impulse responses $h(t) = Ce^{At}B$ and $\hat{h}(t) = \hat{C}e^{\hat{A}t}\hat{B}$ are almost identical. Finally, Figure 2.6 shows the decrease of the \mathcal{H}_∞ -error for increasing reduced system dimension r , together with the available a priori error bound. We observe that the approximability is worse than in the case of the heat equation. A theoretical explanation can be given in terms of the decay rate of the Hankel singular values of the system; see [96].

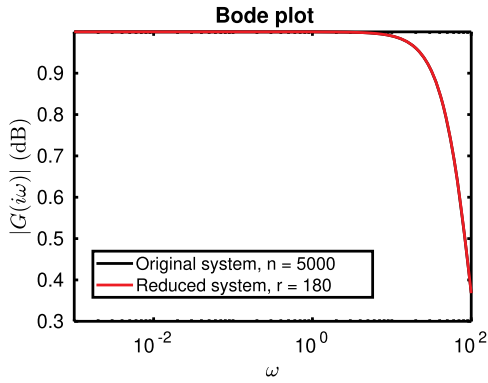


Figure 2.4: Bode plots for the 1D transport equation.

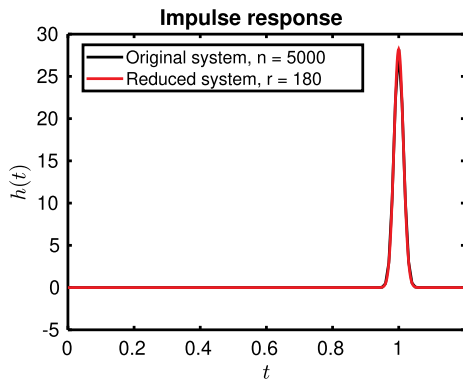


Figure 2.5: Impulse responses for the 1D transport equation.

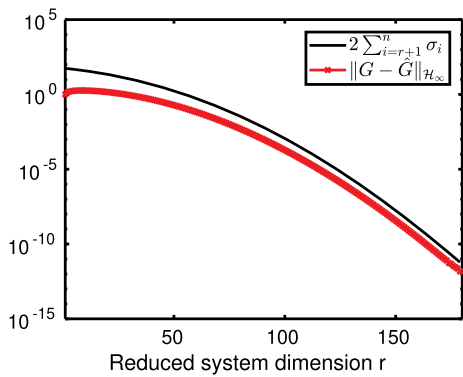


Figure 2.6: \mathcal{H}_∞ -error for varying reduced system dimensions r for the 1D transport equation.

2.8 Conclusions

This chapter provided an introduction to the general concept of system balancing. The focus was on classical balanced truncation and some of its most common variants. Based on an illustrative example, the main idea was derived and also shown to be applicable to spatially semi-discretized partial differential equations. While the methods were mainly discussed by means of finite-dimensional continuous-time control systems, several generalizations to other system classes exist and can be found in the literature. We refer to [18, Chapter 13] for a review of existing software tools implementing balanced truncation model reduction algorithms. With increasing complexity and available data, the need for efficient implementations is still an active research topic. In particular, so-called data-driven methods as presented in [17, Chapter 7] seem to become an indispensable part for future methods.

Bibliography

- [1] S. Al-Baiyat and M. Bettayeb. A new model reduction scheme for k-power bilinear systems. In *Proceedings of the 32nd Conference on Decision and Control*, pages 22–27, 1993.
- [2] S. Al-Baiyat, M. Bettayeb, and U. Al-Saggaf. New model reduction scheme for bilinear systems. *Int. J. Syst. Sci.*, 25:1631–1642, 1994.
- [3] B. Anderson and S. Vongpanitlerd. *Network Analysis and Synthesis: A Modern Systems Theory Approach*. Prentice Hall, Englewood Cliffs, NJ, 1973.
- [4] A. Antoulas. *Approximation of Large-Scale Dynamical Systems*. Society for Industrial and Applied Mathematics, 2005.
- [5] A. Antoulas, D. Sorensen, and Y. Zhou. On the decay rate of Hankel singular values and related issues. *Syst. Control Lett.*, 46:323–342, 2002.
- [6] R. Bartels and G. Stewart. Solution of the matrix equation $AX + XB = C$. *Commun. ACM*, 15:820–826, 1972.
- [7] U. Baur and P. Benner. Gramian-based model reduction for data-sparse systems. *SIAM J. Sci. Comput.*, 31:776–798, 2008.
- [8] C. Beattie, S. Gugercin, and V. Mehrmann. Model reduction for systems with inhomogeneous initial conditions. *Syst. Control Lett.*, 99:99–106, 2017.
- [9] P. Benner, U. Baur, and L. Feng. Model order reduction for linear and nonlinear systems: a system-theoretic perspective. *Arch. Comput. Methods Eng.*, 21:331–358, 2014.
- [10] P. Benner and T. Breiten. Low rank methods for a class of generalized Lyapunov equations and related issues. *Numer. Math.*, 124:441–470, 2013.
- [11] P. Benner and T. Breiten. Model order reduction based on system balancing. In P. Benner, M. Ohlberger, A. Cohen and K. Willcox, editors, *Model Reduction and Approximation: Theory and Algorithms, Computational Science & Engineering*, pages 261–295. SIAM, Philadelphia, PA, 2017.
- [12] P. Benner, Z. Bujanović, P. Kürschner, and J. Saak. RADl: A low-rank ADI-type algorithm for large scale algebraic Riccati equations. *Numer. Math.*, 138:301–330, 2018.
- [13] P. Benner and T. Damm. Lyapunov equations, energy functionals, and model order reduction of bilinear and stochastic systems. *SIAM J. Control Optim.*, 49:686–711, 2011.

- [14] P. Benner, T. Damm, M. Redmann, and Y. R. Cruz. Positive operators and stable truncation. *Linear Algebra Appl.*, 498:74–87, 2016.
- [15] P. Benner and P. Goyal. Balanced truncation model order reduction for quadratic–bilinear control systems, 2017. arXiv:1705.00160.
- [16] P. Benner, P. Goyal, and M. Redmann. Truncated Gramians for bilinear systems and their advantages in model order reduction. In P. Benner, M. Ohlberger, A. Patera, G. Rozza and K. Urban, editors, *Model Reduction of Parametrized Systems*, pages 285–300. Springer, Cham, 2017.
- [17] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders and L. Silveira, editors. In *Model Order Reduction. Volume 2: Snapshot-Based Methods and Algorithms*. De Gruyter, Berlin, 2020.
- [18] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders and L. Silveira, editors. In *Model Order Reduction. Volume 3: Applications*. De Gruyter, Berlin, 2020.
- [19] P. Benner, M. Heinkenschloss, J. Saak, and W. H. K. Inexact low-rank Newton-ADI method for large-scale algebraic Riccati equations. *Appl. Numer. Math.*, 108:125–142, 2016.
- [20] P. Benner and M.-S. Hossain. Structure preserving iterative methods for periodic projected Lyapunov equations and their application in model reduction of periodic descriptor systems. *Numer. Algorithms*, 76:881–904, 2017.
- [21] P. Benner, M.-S. Hossain, and T. Stykel. Low-rank iterative methods for periodic projected Lyapunov equations and their application in model reduction of periodic descriptor systems. *Numer. Algorithms*, 67:669–690, 2014.
- [22] P. Benner, P. Kürschner, and J. Saak. An improved numerical method for balanced truncation for symmetric second-order systems. *Math. Comput. Model. Dyn. Syst.*, 12:593–615, 2013.
- [23] P. Benner, P. Kürschner, and J. Saak. Frequency-limited balanced truncation with low-rank approximations. *SIAM J. Sci. Comput.*, 38:A471–A499, 2016.
- [24] P. Benner, J.-R. Li, and T. Penzl. Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems. *Numer. Linear Algebra Appl.*, 15:755–777, 2008.
- [25] P. Benner and E. Quintana-Orti. Solving stable generalized Lyapunov equations with the matrix sign function. *Numer. Algorithms*, 20:75–100, 1999.
- [26] P. Benner and M. Redmann. Model reduction for stochastic systems. *Stoch. Partial Differ. Equ., Anal. Computat.*, 3:291–338, 2015.
- [27] P. Benner and J. Saak. Efficient balancing-based MOR for large-scale second-order systems. *Math. Comput. Model. Dyn. Syst.*, 17:123–143, 2011.
- [28] P. Benner and J. Saak. Numerical solution of large and sparse continuous time algebraic matrix Riccati and Lyapunov equations: a state of the art survey. *GAMM-Mitt.*, 36:32–52, 2013.
- [29] P. Benner and T. Stykel. Model order reduction of differential-algebraic equations: a survey. In A. Ilchmann and T. Reis, editors, *Surveys in Differential-Algebraic Equations IV, Differential-Algebraic Equations Forum*, pages 107–160. Springer, Switzerland, 2017.
- [30] D. Bini, B. Iannazzo, and B. Meini. *Numerical Solution of Algebraic Riccati Equations*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2012.
- [31] K. Brenan, S. Campbell, and L. Petzold. *The Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations. Classics in Applied Mathematics*, volume 14. SIAM, Philadelphia, PA, 1996.
- [32] B. Brogliato, B. Maschke, R. Lozano, and O. Egeland. Kalman–Yakubovich–Popov lemma. In *Dissipative Systems Analysis and Control, Communications and Control Engineering*, pages 69–176. Springer, London, 2007.
- [33] Y. Chahlaoui, D. Lemonnier, A. Vandendorpe, and P. Van. Dooren, Second-order balanced truncation. *Linear Algebra Appl.*, 415:373–384, 2006.

- [34] E.-W. Chu, H.-Y. Fan, and W.-W. Lin. Projected generalized discrete-time periodic Lyapunov equations and balanced realization of periodic descriptor systems. *SIAM J. Matrix Anal. Appl.*, 29:982–1006, 2007.
- [35] P. D'Alessandro, A. Isidori, and A. Ruberti. Realization and structure theory of bilinear dynamical systems. *SIAM J. Control*, 12:517–535, 1974.
- [36] T. Damm. Direct methods and ADI-preconditioned Krylov subspace methods for generalized Lyapunov equations. *Numer. Linear Algebra Appl.*, 15:853–871, 2008.
- [37] T. Damm and P. Benner. Balanced truncation for stochastic linear systems with guaranteed error bound. In *Proceedings of The 21st International Symposium on Mathematical Theory of Networks and Systems (MTNS 2014)*, pages 1492–1497, 2014.
- [38] U. Desai and D. Pal. A transformation approach to stochastic model reduction. *IEEE Trans. Autom. Control*, 29:1097–1100, 1984.
- [39] C. Desoer and M. Vidyasagar. *Feedback Systems: Input-Output Properties. Classics in Applied Mathematics*, volume 55. SIAM, Philadelphia, PA, 1975.
- [40] D. Enns. Model reduction with balanced realization: an error bound and a frequency weighted generalization. In *Proceedings of the 23rd IEEE Conference on Decision and Control (Las Vegas, 1984)*, pages 127–132. IEEE, New York, 1984.
- [41] K. Eppler and F. Tröltzsch. Fast optimization methods in the selective cooling of steel. In M. Grötschel, S. Krumke and J. Rambau, editors, *Online Optimization of Large Scale Systems*, pages 185–204. Springer, Berlin Heidelberg, 2001.
- [42] M. Farhood, C. Beck, and G. Dullerud. Model reduction of periodic systems: a lifting approach. *Automatica*, 41:1085–1090, 2005.
- [43] J. Feng, J. Lam, Z. Shu, and Q. Wang. Internal positivity preserved model reduction. *Int. J. Control*, 83:575–584, 2010.
- [44] K. Fernando and H. Nicholson. On the structure of balanced and other principal representations of SISO system. *IEEE Trans. Autom. Control*, 28:228–231, 1983.
- [45] K. Fernando and H. Nicholson. On the cross-Gramian for symmetric MIMO systems. *IEEE Trans. Circuits Syst.*, 32:487–489, 1985.
- [46] F. Freitas, J. Rommes, and N. Martins. Gramian-based reduction method applied to large sparse power system descriptor models. *IEEE Trans. Power Syst.*, 23:1258–1270, 2008.
- [47] W. Gawronski and J. Juang. Model reduction in limited time and frequency intervals. *Int. J. Syst. Sci.*, 21:349–376, 1990.
- [48] K. Glover. All optimal Hankel-norm approximations of linear multivariable systems and their L_∞ -error bounds. *Int. J. Control*, 39:1115–1193, 1984.
- [49] K. Glover, R. Curtain, and J. Partington. Realisation and approximation of linear infinite-dimensional systems with error bounds. *SIAM J. Control Optim.*, 26:863–898, 1988.
- [50] K. Glover and J. Doyle. State-space formulae for all stabilizing controllers that satisfy an H_∞ -norm bound and relations to risk sensitivity. *Syst. Control Lett.*, 11:167–172, 1988.
- [51] L. Grasedyck. Existence and computation of low Kronecker-rank approximations for large linear systems of tensor product structure. *Computing*, 72:247–265, 2004.
- [52] W. Gray and J. Mesko. Energy functions and algebraic Gramians for bilinear systems. In *Preprints of the 4th IFAC Nonlinear Control Systems Design Symposium*, pages 103–108, 1998.
- [53] W. Gray and J. Scherpen. On the nonuniqueness of singular value functions and balanced nonlinear realizations. *Syst. Control Lett.*, 44:219–232, 2001.
- [54] W. Gray and E. Verriest. Algebraically defined Gramians for nonlinear systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*, volume 1, pages 3730–3735, 2006.
- [55] M. Green. Balanced stochastic realizations. *Linear Algebra Appl.*, 98:211–247, 1988.
- [56] M. Green. A relative error bound for balanced stochastic truncation. *IEEE Trans. Autom. Control*, 33:961–965, 1988.

- [57] L. Grubišić and D. Kressner. On the eigenvalue decay of solutions to operator Lyapunov equations. *Syst. Control Lett.*, 73:42–47, 2014.
- [58] C. Gu. Model Order Reduction of Nonlinear Dynamical Systems. PhD thesis, University of California, Berkeley 2011.
- [59] C. Gu. QLMOR: A projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 30:1307–1320, 2011.
- [60] S. Gugercin and A. Antoulas. A survey of model reduction by balanced truncation and some new results. *Int. J. Control*, 77:748–766, 2004.
- [61] C. Guiver and M. Opmeer. Model reduction by balanced truncation for systems with nuclear Hankel operators. *SIAM J. Control Optim.*, 52:1366–1401, 2014.
- [62] S. Hammarling. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA J. Numer. Anal.*, 2:303–323, 1982.
- [63] P. Harshavardhana, E. A. Jonckheere, and L. M. Silverman. Stochastic balancing and approximation - stability and minimality. In *Proceedings of the 22nd IEEE Conference on Decision and Control*, pages 1260–1265, 1983.
- [64] C. Hartmann, B. Schäfer-Bung, and A. Thöns-Zueva. Balanced averaging of bilinear systems with applications to stochastic control. *SIAM J. Control Optim.*, 51:2356–2378, 2013.
- [65] M. Heinkenschloss, T. Reis, and A. Antoulas. Balanced truncation model reduction for systems with inhomogeneous initial conditions. *Automatica*, 47:559–564, 2011.
- [66] M. Heinkenschloss, D. Sorensen, and K. Sun. Balanced truncation model reduction for a class of descriptor systems with application to the Oseen equations. *SIAM J. Sci. Comput.*, 30:1038–1063, 2008.
- [67] M. Heyouni and K. Jbilou. An extended block Arnoldi algorithm for large-scale solutions of the continuous-time algebraic Riccati equation. *Electron. Trans. Numer. Anal.*, 33:53–62, 2009.
- [68] C. Himpe and M. Ohlberger. Cross-gramian-based model reduction: A comparison. In P. Benner, M. Ohlberger, A. Patera, G. Rozza and K. Urban, editors, *Model Reduction of Parametrized Systems*, pages 271–283. Springer, Cham, 2017.
- [69] D. Hinrichsen and A. Pritchard. An improved error estimate for reduced-order models of discrete-time systems. *IEEE Trans. Autom. Control*, 35:317–320, 1990.
- [70] D. Hinrichsen and A. Pritchard. In *Mathematical Systems Theory I: Modelling, State Space Analysis, Stability and Robustness*, volume 48. Springer, Berlin Heidelberg, 2005.
- [71] A. Isidori. Direct construction of minimal bilinear realizations from nonlinear input-output maps. *IEEE Trans. Autom. Control*, 18:626–631, 1973.
- [72] E. A. Jonckheere and L. M. Silverman. A new set of invariants for linear systems—application to reduced order compensator design. *IEEE Trans. Autom. Control*, 28:953–964, 1983.
- [73] J. Kerler-Back and T. Stykel. Model order reduction for linear and nonlinear magneto-quasistatic equations. *Int. J. Numer. Methods Eng.*, 111:1274–1299, 2017.
- [74] D. Kressner. Large periodic Lyapunov equations: algorithms and applications. In *Proceedings of the European Control Conference, (ECC03, Cambridge, UK, 2003)*, pages 951–956, 2003.
- [75] D. Kressner and C. Tobler. Preconditioned low-rank methods for high-dimensional elliptic PDE eigenvalue problems. *Comput. Methods Appl. Math.*, 11:363–381, 2011.
- [76] P. Kunkel and V. Mehrmann. *Differential-Algebraic Equations. Analysis and Numerical Solution*. EMS Publishing House, Zürich, Switzerland, 2006.
- [77] R. Lamour, R. März, and C. Tischendorf. *Differential-Algebraic Equations: A Projector Based Analysis, Differential-Algebraic Equations Forum*. Springer, Berlin Heidelberg, 2013.
- [78] N. Lang, J. Saak, and T. Stykel. Balanced truncation model reduction for linear time-varying systems. *Math. Comput. Model. Dyn. Syst.*, 22:267–281, 2016.
- [79] A. Laub, M. Heath, C. Paige, and R. Ward. Computation of system balancing transformations and other applications of simultaneous diagonalization algorithms. *IEEE Trans. Autom. Control*, 32:115–122, 1987.

- [80] J.-R. Li and J. White. Low rank solution of Lyapunov equations. *SIAM J. Matrix Anal. Appl.*, 24:260–280, 2002.
- [81] Y. Lin and V. Simoncini. A new subspace iteration method for the algebraic Riccati equation. *Numer. Linear Algebra Appl.*, 22:26–47, 2015.
- [82] W. Liu and V. Sreeram. Model reduction of singular systems. *Int. J. Syst. Sci.*, 32:1205–1215, 2001.
- [83] Y. Liu and B. Anderson. Singular perturbation approximation of balanced systems. *Int. J. Control*, 50:1379–1405, 1989.
- [84] A. Massoudi, M. R. Opmeer, and T. Reis. Analysis of an iteration method for the algebraic Riccati equation. *SIAM J. Matrix Anal. Appl.*, 37:624–648, 2016.
- [85] A. Massoudi, M. R. Opmeer, and T. Reis. The ADI method for bounded real and positive real Lur’e equations. *Numer. Math.*, 135:431–458, 2017.
- [86] D. G. Meyer and S. Srinivasan. Balancing and model reduction for second-order form linear systems. *IEEE Trans. Autom. Control*, 41:1632–1644, 1996.
- [87] L. Möckel, T. reis, and T. Stykel. Linear-quadratic Gaussian balancing for model reduction of differential-algebraic equations. *Int. J. Control*, 84:1627–1643, 2011.
- [88] R. Mohler. Natural bilinear control processes. *IEEE Trans. Syst. Sci. Cybern.*, 6:192–197, 1970.
- [89] B. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Trans. Autom. Control*, 26:17–32, 1981.
- [90] C. Mullis and R. Roberts. Synthesis of minimum roundoff noise fixed point digital filters. *IEEE Trans. Circuits Syst.*, 23:551–562, 1976.
- [91] D. Mustafa and K. Glover. Controller reduction by \mathcal{H}_∞ -balanced truncation. *IEEE Trans. Autom. Control*, 36:668–682, 1991.
- [92] R. Ober. Balanced parametrization of classes of linear systems. *SIAM J. Control Optim.*, 29:1251–1287, 1991.
- [93] P. Opdenacker and E. A. Jonckheere. LQG balancing and reduced LQG compensation of symmetric passive systems. *Int. J. Control*, 41:73–109, 1985.
- [94] P. C. Opdenacker and E. A. Jonckheere. A contraction mapping preserving balanced reduction scheme and its infinity norm error bounds. *IEEE Trans. Circuits Syst.*, 35:184–189, 1988.
- [95] M. Opmeer. Decay of Hankel singular values of analytic control systems. *Syst. Control Lett.*, 59:635–638, 2010.
- [96] M. Opmeer. Model reduction for distributed parameter systems: a functional analytic view. In *Proceedings of the American Control Conference 2012*, pages 1418–1423, 2012.
- [97] T. Penzl. A cyclic low-rank Smith method for large sparse Lyapunov equations. *SIAM J. Sci. Comput.*, 21:1401–1418, 1999.
- [98] T. Penzl. Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case. *Syst. Control Lett.*, 40:139–144, 2000.
- [99] K. Perev and B. Shafai. Balanced realization and model reduction of singular systems. *Int. J. Syst. Sci.*, 25:1039–1052, 1994.
- [100] L. Pernebo and L. Silverman. Model reduction via balanced state space representations. *IEEE Trans. Autom. Control*, 27:382–387, 1982.
- [101] J. Phillips, L. Daniel, and M. Silveira. Guaranteed passive balancing transformations for model order reduction. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 22:1027–1041, 2003.
- [102] F. Poloni and T. Reis. A deflation approach for large-scale Lur’e equations. *SIAM J. Matrix Anal. Appl.*, 33:1339–1368, 2012.
- [103] F. Poloni and T. Reis. A structured doubling algorithm for Lur’e equations. *Numer. Linear Algebra Appl.*, 23:169–186, 2016.
- [104] M. Redmann. Type II balanced truncation for deterministic bilinear control systems. *SIAM J. Control Optim.*, 56:2593–2612, 2018.

- [105] T. Reis and T. Selig. Balancing transformations for infinite-dimensional systems with nuclear Hankel operator. *Integral Equ. Oper. Theory*, 79:67–105, 2014.
- [106] T. Reis and T. Stykel. Balanced truncation model reduction of second-order systems. *Math. Comput. Model. Dyn. Syst.*, 14:391–406, 2008.
- [107] T. Reis and T. Stykel. PABTEC: Passivity-preserving balanced truncation for electrical circuits. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 29:1354–1367, 2010.
- [108] T. Reis and T. Stykel. Positive real and bounded real balancing for model reduction of descriptor systems. *Int. J. Control*, 83:74–88, 2010.
- [109] T. Reis and T. Stykel. Lyapunov balancing for passivity-preserving model reduction of RC circuits. *SIAM J. Appl. Dyn. Syst.*, 10:1–34, 2011.
- [110] T. Reis and E. Virnik. Positivity preserving balanced truncation for descriptor systems. *SIAM J. Control Optim.*, 48:2600–2619, 2009.
- [111] J. Rommes and N. Martins. Exploiting structure in large-scale electrical circuit and power system problems. *Linear Algebra Appl.*, 431:318–333, 2009.
- [112] W. Rugh. *Nonlinear System Theory*, volume 1981. The Johns Hopkins University Press, 1981.
- [113] Y. Saad. Numerical solution of large Lyapunov equations. In M. Kaashoek, J. van Schuppen and A. Ran, editors, *Signal Processing, Scattering and Operator Theory, and Numerical Methods, Proceedings of the International Symposium MTNS-89*, volume III, pages 503–511. Birkhäuser, 1990.
- [114] J. Saak and M. Voigt. Model reduction of constrained mechanical systems in M-M.E.S.S. *IFAC-PapersOnLine*, 51:661–666, 2018.
- [115] M. Safonov and R. Chiang. A Schur method for balanced truncation model reduction. *IEEE Trans. Autom. Control*, 34:729–733, 1989.
- [116] H. Sandberg and A. Rantzer. Balanced truncation of linear time-varying systems. *IEEE Trans. Autom. Control*, 49:217–229, 2004.
- [117] J. Scherpen. Balancing for nonlinear systems. *Syst. Control Lett.*, 21:143–153, 1993.
- [118] A. Schmidt and B. Haasdonk. Reduced basis approximation of large scale parametric algebraic Riccati equations. *ESAIM Control Optim. Calc. Var.*, 24:129–151, 2018.
- [119] S. Shokoohi, L. Silverman, and P. Van. Dooren, Linear time-variable systems: balancing and model reduction. *IEEE Trans. Autom. Control*, 28:810–822, 1983.
- [120] V. Simoncini. Computational methods for linear matrix equations. *SIAM Rev.*, 58:377–441, 2016.
- [121] V. Simoncini, D. Szyld, and M. Monsalve. On two numerical methods for the solution of large-scale algebraic Riccati equations. *IMA J. Numer. Anal.*, 34:904–920, 2014.
- [122] N. Son and T. Stykel. Solving parameter-dependent Lyapunov equations using the reduced basis method with application to parametric model order reduction. *SIAM J. Matrix Anal. Appl.*, 38:478–504, 2017.
- [123] E. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, volume 6. Springer Verlag, 1998.
- [124] D. Sorensen and A. Antoulas. The Sylvester equation and approximate balanced reduction. *Linear Algebra Appl.*, 351–352:671–700, 2002.
- [125] J. Sreedhar, P. Van Dooren, and P. Misra. Minimal order time invariant representation of periodic descriptor systems. In *Proceedings of American Control Conference (San Diego, California, June 1999)*, volume 2, pages 1309–1313, 1999.
- [126] T. Stykel. Analysis and Numerical Solution of Generalized Lyapunov Equations. PhD thesis, Technische Universität Berlin 2002.
- [127] T. Stykel. Gramian based model reduction for descriptor systems. *Math. Control Signals Syst.*, 16:297–319, 2004.
- [128] T. Stykel. Balanced truncation model reduction for semidiscretized Stokes equation. *Linear Algebra Appl.*, 415:262–289, 2006.

- [129] T. Stykel. Low-rank iterative methods for projected generalized Lyapunov equations. *Electron. Trans. Numer. Anal.*, 30:187–202, 2008.
- [130] T. Stykel. Balancing-related model reduction of circuit equations using topological structure. In P. Benner, M. Hinze and E. ter Maten, editors, *Model Reduction for Circuit Simulation. Lecture Notes in Electrical Engineering*, volume 74, pages 53–80. Springer-Verlag, Berlin, Heidelberg, 2011.
- [131] T. Stykel and V. Simoncini. Krylov subspace methods for projected Lyapunov equations. *Appl. Numer. Math.*, 62:35–50, 2012.
- [132] A. Varga. Efficient minimal realization procedure based on balancing. In *Proceedings of the IMAGS Symposium on Modelling and Control Technological Systems (Lille, France, May 1991)*, volume 2, pages 42–47, 1991.
- [133] A. Varga. Periodic Lyapunov equations: some applications and new algorithms. *Int. J. Control*, 67:69–87, 1997.
- [134] A. Varga. Balancing related methods for minimal realization of periodic systems. *Syst. Control Lett.*, 36:339–349, 1999.
- [135] A. Varga. Balanced truncation model reduction of periodic systems. In *Proceedings of the 39-th IEEE Conference on Decision and Control (Sydney, Australia, December 2000)*, pages 2379–2384, 2000.
- [136] A. Varga. On stochastic balancing related model reduction. In *Proceedings of the 39-th IEEE Conference on Decision and Control (Sydney, Australia, December 2000)*, pages 2385–2390, 2000.
- [137] E. I. Verriest. Low sensitivity design and optimal order reduction for the LQG-problem. In *Proceedings of the 24th Midwest Symposium on Circuits and Systems*, Albuquerque, NM, pages 365–369, 1981.
- [138] E. I. Verriest and T. Kailath. On generalized balanced realizations. *IEEE Trans. Autom. Control*, 28:833–844, 1983.
- [139] G. Wang, V. Sreeram, and W. Liu. A new frequency-weighted balanced truncation method and an error bound. *IEEE Trans. Autom. Control*, 44:1734–1737, 1999.
- [140] W. Wang and M. Safonov. A tighter relative-error bound for balanced stochastic truncation. *Syst. Control Lett.*, 14:307–317, 1990.
- [141] J. Willems. Dissipative dynamical systems Part I: General theory. *Arch. Ration. Mech. Anal.*, 45:321–351, 1972.
- [142] J. Willems. Dissipative dynamical systems Part II: Linear systems with quadratic supply rates. *Arch. Ration. Mech. Anal.*, 45:352–393, 1972.
- [143] K. Zhou. Frequency-weighted \mathcal{L}_∞ -norm and optimal Hankel norm model reduction. *IEEE Trans. Autom. Control*, 40:1687–1699, 1995.
- [144] K. Zhou, J. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice Hall, Upper Saddle River, NJ, 1996.

3 Model order reduction based on moment-matching

Abstract: This is a survey of model order reduction (MOR) methods based on moment-matching. Moment-matching methods for linear non-parametric and parametric systems are reviewed in detail. Extensions of moment-matching methods to nonlinear systems are also discussed. Efficient algorithms for computing the reduced-order models (ROMs) are presented.

Keywords: moment-matching, linear time-invariant systems, parametric systems, error estimation, greedy-type algorithm

MSC 2010: 37M05, 65P99, 65L80

3.1 Introduction

In this chapter, we focus on linear time-invariant (LTI) systems

$$\begin{aligned} E(\mu) \frac{d\mathbf{x}(t, \mu)}{dt} &= A(\mu)\mathbf{x}(t, \mu) + B(\mu)\mathbf{u}(t), \\ \mathbf{y}(t, \mu) &= C(\mu)\mathbf{x}(t, \mu) + D(\mu)\mathbf{u}(t), \end{aligned} \quad (3.1)$$

and mildly nonlinear systems

$$\begin{aligned} E(\mu) \frac{d\mathbf{x}(t, \mu)}{dt} &= \mathbf{f}(\mathbf{x}(t, \mu), \mu) + B(\mu)\mathbf{u}(t), \\ \mathbf{y}(t, \mu) &= C(\mu)\mathbf{x}(t, \mu) + D(\mu)\mathbf{u}(t), \end{aligned}$$

with and without parameters. Here $\mathbf{x}(t, \mu) \in \mathbb{R}^n$ is the state vector, and its entries are called state variables. n is often referred to as the *order* of the system. The vector $\mu \in \mathbb{R}^m$ includes all of the geometrical and physical parameters. The system matrices $E(\mu), A(\mu) \in \mathbb{R}^{n \times n}$, and $B(\mu) \in \mathbb{R}^{n \times n_{\mathbf{u}}}$, $C(\mu) \in \mathbb{R}^{n_{\mathbf{y}} \times n}$, $D(\mu) \in \mathbb{R}^{n_{\mathbf{y}} \times n_{\mathbf{u}}}$ may depend on the parameters. The vector $\mathbf{f}(\mathbf{x}, \mu) \in \mathbb{R}^n$ is a nonlinear function. The system in (3.1) is called the *state-space* representation of the system. It may result from the spatial discretization of partial differential equations (PDEs) describing certain processes like fluid dynamics, temperature distribution in devices, electric circuits, etc.

For most MOR methods, the term $D(\mu)\mathbf{u}(t)$ remains unchanged during the process of MOR. For simplicity, we therefore assume that $D(\mu) = 0$, a zero matrix. There are

Peter Benner, Lihong Feng, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstr. 1, 39106 Magdeburg, Germany, e-mails: benner@mpi-magdeburg.mpg.de, feng@mpi-magdeburg.mpg.de

n_I input terminals and n_O output terminals. When $n_I = n_O = 1$, the system is called a *single-input single-output (SISO) system*. Otherwise, if $n_I, n_O > 1$, it is called a *multi-input multi-output (MIMO) system*.

The basic idea of (P)MOR methods is as follows. Find a low-dimensional trial subspace S_1 which well approximates the manifold where the state vector $\mathbf{x}(t, \mu)$ resides. $\mathbf{x}(t, \mu)$ is approximated by a vector $\hat{\mathbf{x}}(t, \mu)$ in S_1 , which causes a residual of the state equation. The reduced-order model (ROM) is obtained by a (Petrov)–Galerkin projection of the residual onto a test subspace S_2 . In particular, one computes an orthonormal matrix $V = (v_1, v_2, \dots, v_r)$ whose columns span S_1 . The ROM is derived by the following two steps.

1. By replacing $\mathbf{x}(t, \mu)$ in (3.1) with $V\mathbf{z}(t, \mu)$, we obtain

$$\begin{aligned} E(\mu) \frac{dV\mathbf{z}(t, \mu)}{dt} &\approx A(\mu)V\mathbf{z}(t, \mu) + B(\mu)\mathbf{u}(t), \\ \mathbf{y}(t) &\approx C(\mu)V\mathbf{z}(t, \mu). \end{aligned} \quad (3.2)$$

2. Notice that the equations in (3.1) do not hold any longer. Therefore, we can only use “ \approx ” in (3.2). Denote the residual as $\mathbf{e}(t, \mu) = AV\mathbf{z}(t, \mu) + B(\mu)\mathbf{u}(t) - E(\mu) \frac{dV\mathbf{z}(t, \mu)}{dt}$, which in general is nonzero over the whole vector space \mathbb{R}^n . However, it is possible to force $\mathbf{e} = \mathbf{0}$ in a properly chosen subspace S_2 of \mathbb{R}^n . If we have computed a matrix $W \in \mathbb{R}^{n \times r}$, whose columns span S_2 , then $\mathbf{e} = \mathbf{0}$ in S_2 means \mathbf{e} is orthogonal to each column in W , i. e. $W^T \mathbf{e} = \mathbf{0} \iff W^T E \frac{dV\mathbf{z}(t, \mu)}{dt} = W^T AV\mathbf{z}(t, \mu) + W^T B\mathbf{u}(t)$. Finally, we obtain the ROM

$$\begin{aligned} \hat{E}(\mu) \frac{d\mathbf{z}(t, \mu)}{dt} &= \hat{A}(\mu)\mathbf{z}(t, \mu) + \hat{B}(\mu)\mathbf{u}(t), \\ \hat{\mathbf{y}}(t, \mu) &= \hat{C}\mathbf{z}(t, \mu), \end{aligned} \quad (3.3)$$

where $\hat{E}(\mu) = W^T E(\mu)V \in \mathbb{R}^{r \times r}$, $\hat{A}(\mu) = W^T A(\mu)V \in \mathbb{R}^{r \times r}$, $\hat{B}(\mu) = W^T B(\mu) \in \mathbb{R}^{r \times n_I}$, $\hat{C}(\mu) = C(\mu)V \in \mathbb{R}^{n_O \times r}$. $\mathbf{z}(\mu) \in \mathbb{R}^r$ is a vector of length $r \ll n$. Then $\mathbf{x}(t, \mu)$ can be approximated by $\mathbf{x}(t, \mu) \approx V\mathbf{z}(t, \mu)$. The system in (3.3) is referred to as the *reduced-order model* (ROM), since it is of much smaller order than the original system in (3.1), i. e. $r \ll n$. The ROM can then replace the original system for fast simulation.

MOR methods differ in computing the two matrices W and V . One common goal of all methods is that the input-output behavior of the ROM should be sufficiently “close” to that of the original model. The error between the transfer functions (see (3.6)) is also used to measure the accuracy of the ROM.

Moment-matching relates to a class of methods which construct the ROM by building the projection matrices W, V from the system information in the frequency domain. The early moment-matching methods are only applicable to linear non-parametric systems. Later on, these methods were extended to linear parametric systems. Based on nonlinear system theory [52], multi-moment-matching methods based on variational analysis were proposed and are successful in reducing weakly

nonlinear systems. In contrast to the snapshot based time-domain methods, e.g., proper orthogonal decomposition or the reduced basis method, moment-matching methods can be considered as frequency-domain methods, and are independent of the inputs. Therefore, these methods are robust for systems with varying inputs. The rest of this chapter is organized as follows. In Section 3.2, we introduce moment-matching methods for linear non-parametric systems, where methods based on rational interpolation are particularly discussed. The extension of those methods to linear parametric systems is introduced in Section 3.3. Methods based on moment-matching and multi-moment-matching for nonlinear systems are reviewed in Section 3.4, and their extension to parametric nonlinear systems is discussed in Section 3.5. Conclusions are drawn in the end.

3.2 Moment-matching for linear non-parametric systems

This section reviews moment-matching methods for linear non-parametric systems, so that the vector of parameters μ can be dropped from the system (3.1). Among the early works of moment-matching MOR, the method of Asymptotic Waveform Evaluation (AWE) in [48] was shown to be able to reduce large-scale interconnected electrical circuit models, which stimulated broad interests in this kind of methods. The AWE method tries to find a Padé approximation of the transfer function $H(s)$, which can be computed much more quickly than computing $H(s)$ itself.

Transfer function

For all the methods introduced in this chapter, the transfer function of the system is used to either derive the ROM, or to perform the error estimation. The transfer function of the system in (3.1) is the input/output relation of the system in the frequency domain. By applying the Laplace transform to both sides of the equations in (3.1), we obtain

$$sEX(s) - E\mathbf{x}(0) = AX(s) + BU(s), \quad (3.4)$$

$$Y(s) = CX(s). \quad (3.5)$$

Here, $X(s)$ is the Laplace transform of $\mathbf{x}(t)$, and $\mathbf{x}(0)$ is the initial state of the system. Assuming that $\mathbf{x}(0) = \mathbf{0}$, we obtain the expression for the transfer function

$$H(s) = Y(s)/U(s) = C(sE - A)^{-1}B, \quad (3.6)$$

where the right division “/” has to be understood in a formal way for MIMO systems. For a SISO system, the transfer function $H(s)$ is a scalar function. The Padé approximation of a scalar function can be defined as follows.

Padé approximation

The Padé approximation of a function $H(s)$ is a rational function $H_{p,q}(s)$ whose Taylor series at $s = 0$ agrees with that of $H(s)$ in at least the first $p + q + 1$ terms [21].

For a MIMO system, the transfer function $H(s)$ is a matrix function and each entry of it can be approximated by the above Padé approximation. For clarity of explanation, we use a SISO system as an example to briefly describe the method.

From the definition of the Padé approximation, we know that, if $H(s) = H_{p,q}(s_0 + \sigma) = P_p(\sigma)/Q_q(\sigma)$ is a Padé approximation of the transfer function $H(s_0 + \sigma)$, then we have

$$H_{p,q}(s_0 + \sigma) = H(s_0 + \sigma) + O(\sigma^{p+q+1}). \quad (3.7)$$

The derivatives of $H(s_0 + \sigma)$ at $\sigma = 0$ are actually the derivatives of $H(s)$ at $s = s_0$, they are also the moments $m_i(s_0)$, $i = 0, 1, \dots$, (see the definition in Section 3.2.1.1) of the transfer function. By definition, the Padé approximation $H_{p,q}(s_0 + \sigma)$ matches the first $p + q + 1$ moments of the transfer function.

If the coefficients of the two polynomials $P_p(\sigma)$ and $Q_q(\sigma)$ in $H_{p,q}(s_0 + \sigma)$ are computed, then $H_{p,q}(s_0 + \sigma)$ is obtained. The coefficients can be obtained by solving two groups of equations which are derived from equating the coefficients of the Taylor series expansion (at $\sigma = 0$) on both sides of (3.7).

Since the moments are the coefficients of the Taylor series expansion of $H(s_0 + \sigma)$ at $\sigma = 0$, they are involved in solving the equations to obtain the coefficients of $P_p(\sigma)$ and $Q_q(\sigma)$. However, in the AWE method, the moments are computed explicitly, which can cause serious numerical instability.

In order to overcome the numerical instability of AWE, a more robust method “Padé via Lanczos” (PVL) [21] (see also [32]) was proposed. PVL also computes the Padé approximation of $H(s) = H(s_0 + \sigma)$, however, the moments of $H(s)$ do not have to be computed explicitly. Instead, an orthonormal basis of the subspace spanned by the moment vectors is computed, which constitutes the projection matrix V , and the projection matrix W is also computed simultaneously. Both of them are computed by the nonsymmetric Lanczos process. It is proved in [21] that the transfer function of the ROM produced by W and V is the Padé approximation of the original transfer function $H(s)$. The PVL method avoids explicit computation of the moments, and therefore avoids the possible numerical instability.

Unfortunately, PVL does not necessarily preserve passivity of the original system, which is a problem in some engineering applications, especially in Integrated Circuit (IC) design. For this target, the method “Passive and Reduced-order Interconnect Macromodeling Algorithm” (PRIMA) [45] was proposed. The resulting ROM preserves the passivity of the original system, under certain assumptions on the system matrices. The trade-off is that only half the number of moments can be matched by PRIMA as compared to PVL, if the matrix V in both methods expands the same subspace. Such an approximation of the transfer function is called a *Padé-type approximation*.

The rational interpolation method proposed in [35] extended the Padé-approximation method and the Padé-type-approximation method based on a single expansion point to the case of multiple expansion points. All these methods can be called moment-matching methods, because all the ROMs match the moments of the original system to different extents. For survey papers on moment-matching model reduction; see [4, 31] and [3, 35].

Moment-matching MOR methods try to derive a ROM whose transfer function matches the moments of the transfer function of the original system. Generally speaking, the more moments matched, the more accurate the ROM will be. In the following, we first introduce the definition of the moments and moment vectors, then we show how to compute the matrices W and V based on moment-matching.

3.2.1 Basic idea

3.2.1.1 Moments and moment vectors

If we expand the transfer function $H(s)$ into its Taylor series about an expansion point s_0 so that $s_0E - A$ is a nonsingular matrix,

$$\begin{aligned} H(s) &= C^T [(s - s_0 + s_0)E - A]^{-1} B \\ &= C^T [(s - s_0)E + (s_0E - A)]^{-1} B \\ &= C^T [I + (s_0E - A)^{-1}E(s - s_0)]^{-1} (s_0E - A)^{-1} B \\ &= \sum_{i=0}^{\infty} \underbrace{C^T [-(s_0E - A)^{-1}E]^i}_{=: m_i(s_0)} (s_0E - A)^{-1} B (s - s_0)^i, \end{aligned} \quad (3.8)$$

and if the system is a SISO system, then $m_i(s_0)$, $i = 0, 1, 2, \dots$, are called the moments of the transfer function $H(s)$. If the system is a MIMO system, then $m_i(s_0)$, $i = 0, 1, 2, \dots$, are matrices and they are called block moments [45]. In the field of circuit design, the entry in the j th row, k th column of $m_i(s_0)$ is called the i th moment of the current that flows into port j when the voltage source at port k is the only nonzero source [45]. Analogies exist for other application areas, such as mechanics. In this chapter, we only consider $m_i(s_0)$ as a whole, and do not consider its entries individually. This means, when we talk about moments of the transfer function, we mean $m_i(s_0)$, $i = 0, 1, \dots$, which refers either to the moments of a SISO system or to the block moments of a MIMO system.

From (3.4) and (3.8), it is straightforward to obtain the corresponding Taylor series expansion of $X(s)$,

$$X(s) = \sum_{i=0}^{\infty} [-(s_0E - A)^{-1}E]^i (s_0E - A)^{-1} B U(s) (s - s_0)^i. \quad (3.9)$$

Here, we call the vectors $[-(s_0E - A)^{-1}E]^i (s_0E - A)^{-1} B$, $i = 0, 1, \dots$, moment vectors which are to be used to compute the projection matrices W , V . Notice that when the system

in (3.1) has multiple inputs, i. e. $n_I > 1$, then $[-(s_0 E - A)^{-1} E]^i (s_0 E - A)^{-1} B$, $i = 0, 1, \dots$, are matrices rather than vectors. For simplicity, we still call them moment vectors.

3.2.1.2 Computation of the projection matrices W and V

Computation of V

Approximating $X(s)$ by the truncated series in (3.9) means that $X(s) \approx VZ(s)$. The columns of $V \in \mathbb{R}^{n \times r}$ constitute an orthonormal basis of S_1 , which is a subspace spanned by the moment vectors in the truncated series. After inverse Laplace transform, we obtain the corresponding approximation of $\mathbf{x}(t)$ in S_1 , i. e. $\mathbf{x}(t) \approx V\mathbf{z}(t)$, where $\mathbf{z}(t)$ is the inverse Laplace transform of $Z(s)$. This means that $\mathbf{x}(t)$ in the time domain can be approximated by $V\mathbf{z}(t)$. Usually, the more moment vectors included in S_1 , the more accurate the approximation $V\mathbf{z}(t)$ will be. However, in order to keep the ROM small, we usually choose a small number of moment vectors starting from $i = 0$, i. e. the columns of the orthonormal matrix V span the subspace

$$\text{range}\{V\} = \text{span}\{\tilde{B}(s_0), \tilde{A}(s_0)\tilde{B}(s_0), \dots, \tilde{A}^{q-1}(s_0)\tilde{B}(s_0)\}, \quad (3.10)$$

where $\tilde{A}(s_0) = (s_0 E - A)^{-1} E$, $\tilde{B}(s_0) = (s_0 E - A)^{-1} B$ and $q \ll n$.

Computation of W

To obtain the ROM, we also need to compute the (Petrov-)Galerkin projection matrix W . The columns of the matrix W span the subspace below, i. e.

$$\text{range}\{W\} = \text{span}\{\tilde{C}(s_0), \tilde{A}_c(s_0)\tilde{C}(s_0), \dots, \tilde{A}_c^{q-1}(s_0)\tilde{C}(s_0)\}, \quad (3.11)$$

where $\tilde{A}_c(s_0) = (s_0 E - A)^{-T} E^T$, $\tilde{C}(s_0) = (s_0 E - A)^{-T} C^T$. Note that the two subspaces in (3.10) and (3.11) are actually two Krylov subspaces $K_q(\tilde{A}(s_0), \tilde{B}(s_0))$ and $K_q(\tilde{A}_c(s_0), \tilde{C}(s_0))$, respectively. Moment-matching methods based on computing W, V from Krylov subspaces are often called *Krylov-based methods*. If the above two matrices W and V are used to obtain the ROM (3.3), the transfer function of the ROM matches the first $2q$ moments of the transfer function of the original model [35]. We summarize this in the following theorem.

Theorem 3.1. *If V and W span the subspaces in (3.10) and (3.11), respectively, then the transfer function $\hat{H}(s) = \hat{C}(s\hat{E} + \hat{A})^{-1}\hat{B}$ of the ROM (3.3) matches the first $2q$ moments of the transfer function of the original system, i. e.*

$$m_i(s_0) = \hat{m}_i(s_0), \quad i = 0, 1, \dots, 2q - 1,$$

where $\hat{m}_i(s_0) = \hat{C}[-(s_0\hat{E} - \hat{A})\hat{E}]^{-i}(s_0\hat{E} - \hat{A})^{-1}\hat{B}$, $i = 0, 1, \dots, 2q - 1$, are the i th-order moments of \hat{H} .

Note that in order to ensure the projector property of VW^T , one also needs to enforce the bi-orthogonality condition $W^T V = I$ (assuming here that the subspace basis ma-

trices V, W are formed over \mathbb{R}). The moment-matching MOR method PRIMA [45] uses $W = V$. In this case, only q moments of the transfer function are matched.

The orthonormal matrices W and V in (3.10), (3.11) can be computed by rational Krylov subspace algorithms (rational Lanczos algorithm, rational Arnoldi algorithm) [35]. Only (sparse) matrix factorizations, (sparse) forward/backward solves, and matrix-vector multiplications are used in these algorithms, such that the complexity of the moment-matching MOR is in $O(nq^2)$ for sparse matrices E, A .

3.2.2 Stability

In general, the moment-matching methods do not preserve stability of the original system. Only for systems with special structures, there exist several approaches based on Galerkin projection where the ROM are guaranteed to be stable and passive; see, e. g., [45]. For details on passivity of LTI systems, we refer to Chapter 5 of this volume. The passivity preservation of the moment-matching method for RLC circuits can be mathematically described as follows [45].

Theorem 3.2. *If the system matrices E and A satisfy $E^T + E \geq 0$ and $A^T + A \leq 0$, respectively, and if $C = B$, then the ROM obtained by Galerkin projection, i. e., $W = V$ preserves the passivity of the original system in (3.1).*

Stability is naturally guaranteed by passivity, therefore the ROM obtained by moment-matching with Galerkin projection preserves stability as well. Benefiting from the preservation of passivity and low computational complexity, the moment-matching method is very popular in circuit simulation and in micro-electro-mechanical systems (MEMS) simulation as well.

3.2.3 Multiple expansion points

The accuracy of the moment-matching methods depends not only on the number of moments matched, but also on the expansion points. Since the Taylor expansion in (3.8) is only accurate within a certain radius around the expansion point s_0 , the ROM becomes inaccurate beyond this radius.

To increase the accuracy of a single-point expansion, one may use more than one expansion point. Moment-matching by multi-point expansion is also known as rational interpolation [35]. For example, if using a set of q distinct expansion points $\{s_1, \dots, s_q\}$, the ROM obtained by, e. g.,

$$\begin{aligned}\text{range}\{V\} &= \text{span}\{\tilde{B}(s_1), \dots, \tilde{B}(s_l)\}, \\ \text{range}\{W\} &= \text{span}\{\tilde{C}(s_1), \dots, \tilde{C}(s_l)\},\end{aligned}$$

matches the first two moments $m_0(s_i), m_1(s_i)$ at each $s_i, i = 1, \dots, l$ [35]. Here, $\tilde{B}(s_i) = (s_i E - A)^{-1} B$, $\tilde{C}(s_i) = (s_i E - A)^{-T} C^T, i = 1, \dots, l$.

More generally, if we use

$$\text{range}\{V\} = \text{span}\{\tilde{B}(s_1), \dots, \tilde{A}^{q_1-1}(s_1)\tilde{B}(s_1), \dots, \tilde{B}(s_l), \dots, \tilde{A}^{q_l-1}(s_l)\tilde{B}(s_l)\}, \quad (3.12)$$

$$\text{range}\{W\} = \text{span}\{\tilde{C}(s_1), \dots, \tilde{A}_c^{q_1-1}(s_1)\tilde{C}(s_1), \dots, \tilde{C}(s_l), \dots, \tilde{A}_c^{q_l-1}(s_l)\tilde{C}(s_l)\}, \quad (3.13)$$

where $\tilde{A}(s_k) = (s_k E - A)^{-1}E$, $\tilde{A}_c(s_k) = (s_k E - A)^{-T}E^T$, $k = 1, \dots, l$, then we have the following moment-matching property.

Theorem 3.3 ([35]). *If*

$$\text{range}\{V\} \supseteq \text{span}\{\tilde{B}(s_1), \dots, \tilde{A}^{q_1-1}(s_1)\tilde{B}(s_1), \dots, \tilde{B}(s_l), \dots, \tilde{A}^{q_l-1}(s_l)\tilde{B}(s_l)\},$$

and

$$\text{range}\{W\} \supseteq \text{span}\{\tilde{C}(s_1), \dots, \tilde{A}_c^{q_1-1}(s_1)\tilde{C}(s_1), \dots, \tilde{C}(s_l), \dots, \tilde{A}_c^{q_l-1}(s_l)\tilde{C}(s_l)\},$$

then the transfer function $\hat{H}(s) = \hat{C}(s\hat{E} + \hat{A})^{-1}\hat{B}$ of the ROM (3.3) matches the first $2q_k$ moments of the transfer function of the original system at each expansion point s_k , i. e.

$$m_i(s_k) = \hat{m}_i(s_k), \quad i = 0, 1, \dots, 2q_k - 1, \quad k = 1, \dots, l,$$

where $\hat{m}_i(s_k) = \hat{C}[-(s_k\hat{E} - \hat{A})\hat{E}]^{-i}(s_k\hat{E} - \hat{A})^{-1}\hat{B}$, $i = 0, 1, \dots, 2q_k - 1$, are the i th-order moments of \hat{H} at s_k .

Given expansion points s_1, \dots, s_l , Algorithm 3.1 presents a procedure for computing the projection matrix V in (3.12).

The matrix W can also be computed using Algorithm 3.1, only by replacing $\tilde{B}(s_k)$ with $\tilde{C}(s_k)$, and $\tilde{A}(s_k)$ with $\tilde{A}_c(s_k)$. Algorithm 3.1 is also applicable to SISO systems, as we can see from Step 7. However, for SISO systems, the algorithm can be further simplified, and the two matrices W , V can be easily computed in parallel. Algorithm 3.2 is a version for SISO systems. In fact, the computed V , $W \in \mathbb{R}^{n \times r}$ from either Algorithm 3.1, or Algorithm 3.2, are not bi-orthogonal, which is not required by the moment-matching Theorem 3.3. However, for systems with $E = I$, the identity matrix, it is preferred that the reduced matrix $\hat{E} = I_r$, the identity matrix of dimension of r . Then we can use the transform $W \leftarrow W(V^T W)^{-1}$ to obtain a new W , so that $\hat{E} = W^T E V = W^T V = I_r$. In the final steps of both algorithms, we need to orthogonalize the columns of the intermediate matrices using the modified Gram–Schmidt process, which is an algorithm for orthogonalizing any given group of vectors. The details of the algorithm are given in Algorithm 3.3. The finally obtained orthogonal vectors are actually orthonormal, i. e., their norms are all 1. The number of orthogonal vectors are $\tilde{l} \leq l$, because once deflation ($\|a_k\|_2 \leq \varepsilon$) in Step 7 occurs, \tilde{l} will not be increased.

Remark 3.1. Let $\text{size}(M, 2)$ be the MATLAB notation for the number of columns in a matrix M . Then it could happen that $\text{size}(V, 2) \neq \text{size}(W, 2)$. In this situation, more computations should be done as follows. Denote $r_V = \text{size}(V, 2)$, $r_W = \text{size}(W, 2)$, if

Algorithm 3.1: Compute V in (3.12) for a non-parametric MIMO system (3.1).

Input: System matrices E, A, B, C , expansion points s_1, \dots, s_l .

Output: Projection matrix V .

```

1: Initialize  $a_1 = 0, a_2 = 0, sum = 0, col = 0$ .
2: for  $k = 1, \dots, l$  do
3:   if (multiple input) then
4:     Orthogonalize the columns in  $\tilde{B}(s_k)$  using the modified Gram–Schmidt process:  $[v_1, v_2, \dots, v_{m_k}] = \text{orth}\{\tilde{B}(s_k)\}$ ,
5:      $sum = m_k$ . ( $m_k$  is the number of remaining columns after deflation.)
6:   else
7:     Compute the first column in  $V$ :  $v_1 = \tilde{B}(s_k)/\|\tilde{B}(s_k)\|_2$ ,
8:      $sum = 1$ .
9:   end if
10:  Orthogonalize the columns in  $\tilde{A}(s_k)\tilde{B}(s_k), \dots, \tilde{A}(s_k)^{q_k-1}\tilde{B}(s_k)$  iteratively as follows:
11:  for  $i = 1, 2, \dots, q_k - 1$  do
12:     $a_2 = sum$ .
13:    if  $a_1 = a_2$  then
14:      break; go to Step 2
15:    else
16:      for  $j = a_1 + 1, \dots, a_2$  do
17:         $w = \tilde{A}(s_k)v_j, col = sum + 1$ .
18:        for  $d = 1, 2, \dots, col - 1$  do
19:           $h = v_d^T w, w = w - hv_d$ .
20:        end for
21:        if  $\|w\|_2 > \varepsilon$  ( $\varepsilon > 0$  is a small value indicating deflation, e. g.,  $\varepsilon = 10^{-7}$ ) then
22:           $v_{col} = \frac{w}{\|w\|_2}, sum = col$ .
23:        end if
24:      end for ( $j$ )
25:    end if
26:     $a_1 = a_2$ .
27:  end for ( $i$ )
28:   $V_k = [v_1, \dots, v_{sum}]$ ,
29: end for ( $k$ )
30: Orthogonalize the columns in  $[V_1, \dots, V_l]$  by the modified Gram–Schmidt process to obtain  $V$ , i. e.  $V := \text{orth}\{V_1, \dots, V_l\}$ .

```

$r_V < r_W$, then add $r_W - r_V$ random orthogonal columns to V , and vice versa. This way, the moment-matching property of the ROM remains unchanged due to the definitions of V, W in Theorem 3.3.

Algorithm 3.2: Compute V in (3.12) and W in (3.13) for a non-parametric SISO system (3.1).

Input: System matrices E, A, B, C , expansion points s_1, \dots, s_l .

Output: Projection matrices V, W .

```

1: Initialize  $col = 0, col_v = 0, col_w = 0$ .
2: for  $k = 1, \dots, l$  do
3:   Compute the first column of  $V_k$ :  $v_1 = \tilde{B}(s_k) / \|\tilde{B}(s_k)\|_2$ .
4:   Compute the first column of  $W_k$ :  $w_1 = \tilde{C}(s_k) / \|\tilde{C}(s_k)\|_2$ .
5:    $col = col + 1$ .
6:   Orthogonalize the vectors  $\tilde{A}(s_k)\tilde{B}(s_k), \dots, \tilde{A}(s_k)^{q_k-1}\tilde{B}(s_k)$  against  $v_1$  iteratively, orthogonalize the vectors  $\tilde{A}_c(s_k)\tilde{C}(s_k), \dots, \tilde{A}_c(s_k)^{q_k-1}\tilde{C}(s_k)$  against  $w_1$  iteratively, as follows:
7:   for  $i = 1, 2, \dots, q_k - 1$  do
8:      $v = \tilde{A}(s_k)v_i$ ,
9:      $w = \tilde{A}_c(s_k)w_i$ 
10:    for  $j = 1, 2, \dots, col$  do
11:       $h = v_j^T v$ ,  $v = v - hv_j$ .
12:       $h = w_j^T w$ ,  $w = w - hw_j$ .
13:    end for
14:     $col = col + 1$ .
15:    if  $\|v\|_2 > \varepsilon$  ( $\varepsilon > 0$  is a small value indicating deflation, e. g.,  $\varepsilon = 10^{-7}$ ) then
16:       $v_{col} = \frac{v}{\|v\|_2}$ ,
17:    else
18:       $col_v = col - 1$ , stop updating  $V_k$ .
19:    end if
20:    if  $\|w\|_2 > \varepsilon$  then
21:       $w_{col} = \frac{w}{\|w\|_2}$ ,
22:    else
23:       $col_w = col - 1$ , stop updating  $W_k$ .
24:    end if
25:  end for
26:   $V_k = [v_1, \dots, v_{col_v}]$ ,  $W_k = [w_1, \dots, w_{col_w}]$ .
27: end for
28: Orthogonalize the columns in  $[V_1, \dots, V_l]$  by the modified Gram–Schmidt process to obtain  $V$ , i. e.  $V := \text{orth}\{V_1, \dots, V_l\}$ .
29: Orthogonalize the columns in  $[W_1, \dots, W_l]$  by the modified Gram–Schmidt process to obtain  $W$ , i. e.  $W := \text{orth}\{W_1, \dots, W_l\}$ .

```

The issue is then how to (adaptively) choose the multiple expansion points. Many adaptive techniques have been proposed during the last years [8, 14, 25, 40, 39, 38, 30, 28, 56], where some are more or less heuristic [8, 14, 25, 40]. Based on system theory, an

Algorithm 3.3: Modified Gram–Schmidt process.

Input: A group of nonzero vectors a_1, \dots, a_l , a deflation tolerance $\varepsilon > 0$.

Output: A group of orthogonalized vectors $v_1, \dots, v_{\tilde{l}}$, $\tilde{l} \leq l$.

```

1:  $v_1 = a_1 / \|a_1\|_2$ ,  $\tilde{l} = 1$ .
2: for  $k = 2, \dots, l$  do
3:   for  $i = 1, \dots, \tilde{l}$  do
4:      $h = v_i^T a_k$ ,
5:      $a_k = a_k - h v_i$ .
6:   end for
7:   if  $\|a_k\|_2 > \varepsilon$  then
8:      $\tilde{l} = \tilde{l} + 1$ ,
9:      $v_{\tilde{l}} = \frac{a_k}{\|a_k\|_2}$ ,
10:  end if
11: end for
```

error bound is derived in [56], but it faces high computational complexity. The residual of the state vector is simply used in [39] as the error estimator of the ROM. In the next subsection, we introduce several typical techniques of adaptivity [38, 25, 30, 28].

3.2.4 Selection of expansion points

3.2.4.1 \mathcal{H}_2 -optimal iterative rational Krylov algorithm

The iterative rational Krylov algorithm (IRKA) is proposed in [38]. Given a group of initial expansion points, IRKA adaptively updates the expansion points, and upon convergence, IRKA produces a ROM satisfying \mathcal{H}_2 -optimal necessary conditions. (See Equation (5) in Chapter 1 of this volume for the definition of the \mathcal{H}_2 -norm.) The expansion points are selected as the mirror images of the poles of the updated ROM at each iteration. The algorithm is presented as Algorithm 3.4.

Moment-matching property

For single-input single-output (SISO) systems, IRKA leads to the following interpolation property upon convergence:

$$\begin{aligned} \hat{H}(-\hat{\lambda}_i) &= H(-\hat{\lambda}_i), \\ \frac{\partial \hat{H}(-\hat{\lambda}_i)}{\partial s} &= \frac{\partial H(-\hat{\lambda}_i)}{\partial s}. \end{aligned} \tag{3.14}$$

Here $\hat{\lambda}_i, i = 1, \dots, r$, are the eigenvalues of the ROM defined in Step 3(b) of Algorithm 3.4. They are also the poles of the reduced transfer function $\hat{H}(s)$.

Algorithm 3.4: Iterative Rational Krylov Algorithm (IRKA).

- 1: Make an initial selection of the expansion points closed under conjugation, i. e. $s_1, \dots, s_i, \bar{s}_i, \dots, s_r$, if s_i is a complex variable. Fix a tolerance ϵ for the accuracy of the ROM. Choose initial directions $\tilde{b}_1, \dots, \tilde{b}_r, \tilde{c}_1, \dots, \tilde{c}_r$.
- 2: Choose V_r, W_r so that

$$\begin{aligned}\text{range}(V_r) &= \text{span}\{\tilde{B}(s_1)\tilde{b}_1, \dots, \tilde{B}(s_i)\tilde{b}_i, \tilde{B}(\bar{s}_i)\tilde{b}_{i+1}, \dots, \tilde{B}(s_r)\tilde{b}_r\}, \\ \text{range}(W_r) &= \text{span}\{\tilde{C}(s_1)\tilde{c}_1, \dots, \tilde{C}(s_i)\tilde{c}_i, \tilde{C}(\bar{s}_i)\tilde{c}_{i+1}, \dots, \tilde{C}(s_r)\tilde{c}_r\},\end{aligned}$$

and $W_r = W_r(V_r^T W_r)^{-1}$. Here $\tilde{B}(s_i) = (s_i E - A)^{-1} B$, $\tilde{C}(s_i) = (s_i E - A)^{-T} C^T$, $i = 1, \dots, r$.

- 3: While $(\max_{j=1, \dots, r} \{\frac{s_j - \bar{s}_j^{\text{old}}}{s_j}\} > \epsilon)$
 - (a) $\hat{E} = W_r^T E V_r$, $\hat{A} = W_r^T A V_r$, $\hat{B} = W_r^T B$, $\hat{C} = C V_r$.
 - (b) Compute eigenvalues, -vectors of $\lambda E - A$ so that $(\lambda_i \hat{E} - \hat{A})y_i = \lambda_i y_i$, $i = 1, \dots, r$.
 - (c) Assign $s_i \leftarrow -\lambda_i$ for $i = 1, \dots, r$, $Y = (y_1, \dots, y_r)$.
 - (d) $\tilde{B} = \hat{B}^T Y^{-T}$, $\tilde{C} = \hat{C} Y$, $(\tilde{b}_1, \dots, \tilde{b}_r) \leftarrow \tilde{B}$, $(\tilde{c}_1, \dots, \tilde{c}_r) \leftarrow \tilde{C}$.
 - (e) Update V_r and W_r :

$$\text{range}(V_r) = \text{span}\{\tilde{B}(s_1)\tilde{b}_1, \dots, \tilde{B}(s_r)\tilde{b}_r\},$$

$$\text{range}(W_r) = \text{span}\{\tilde{C}(s_1)\tilde{c}_1, \dots, \tilde{C}(s_r)\tilde{c}_r\},$$

and $W_r = W_r(V_r^T W_r)^{-1}$.

- 4: $\hat{E} = W_r^T E V_r$, $\hat{A} = W_r^T A V_r$, $\hat{B} = W_r^T B$, $\hat{C} = C V_r$.

It is easy to see that the images of the poles of the ROM are selected as the expansion points, and are updated every time the ROM is updated. In IRKA, \bar{s}_i is the conjugate of s_i . From the definition of the moments of the transfer function, we know that the first-order derivative of the transfer function at s_i is the first-order moment $m_1(s_i)$. The value of the transfer function at s_i is the zeroth-order moment $m_0(s_i)$. Therefore, IRKA generates ROMs matching the first two moments of the transfer function at each expansion point s_i , $i = 1, \dots, r$.

Optimality property [38]

The ROM computed by IRKA satisfies the following necessary conditions of optimality.

Theorem 3.4. *Let $H(s)$ be the transfer function of a stable SISO system, and \hat{H} be a local minimizer of dimension r for the optimal \mathcal{H}_2 -model reduction problem*

$$\|H - \hat{H}\|_{\mathcal{H}_2} = \min_{\dim(\tilde{H})=r, \tilde{H}:\text{stable}} \|H - \tilde{H}\|_{\mathcal{H}_2},$$

and suppose that $\hat{H}(s)$ has simple poles at $\hat{\lambda}_i, i = 1, \dots, r$, then $\hat{H}(s)$ interpolates $H(s)$ and its first derivative at $\hat{\lambda}_i, i = 1, \dots, r$:

$$\hat{H}(-\hat{\lambda}_i) = H(-\hat{\lambda}_i), \quad \frac{\partial \hat{H}(-\hat{\lambda}_i)}{\partial s} = \frac{\partial H(-\hat{\lambda}_i)}{\partial s}.$$

Comparing Theorem 3.4 with the moment-matching property in (3.14), we see that IRKA constructs a ROM that satisfies the necessary condition of the local optimal property in Theorem 3.4.

3.2.4.2 A heuristic technique

In [25], an adaptive scheme for both choosing expansion points and deciding the number of moments is delineated. Generally speaking, the expansion points are chosen based on a binary principle. The number of moments matched at each expansion point is determined by a tested point which is known to cause the largest error in the interval of each pair of neighboring expansion points. Using this technique, the projection matrix V in (3.12) is adaptively computed, and the ROM is obtained by Galerkin projection using $W = V$. The only inputs of the algorithm are an acceptable dimension of the ROM, say r_{\max} , as well as the acceptable accuracy of the ROM, tol . r_{\max} will be adjusted to a proper number during the adaptive scheme if it was selected too small. The details of the algorithm can be found in [25]. From the numerical examples in [25], the method shows its success in automatically obtaining ROMs for several circuit examples. It is nevertheless clarified in [25] that the proposed method has difficulty in dealing with multi-input and multi-output (MIMO) models with many resonances in the output responses. The proposed method may obtain good results for a single-input and single-output (SISO) system with many resonances in the output, but it will fail when the system is MIMO and possesses multiple resonances in all the output responses of all the I-O ports. For such systems, an efficient error estimation may help to construct more robust and reliable ROMs. In the next subsection, we introduce a greedy-type algorithm which adaptively selects the expansion points using a recently developed *a posteriori* error bound.

3.2.4.3 Scheme based on a posteriori error estimation

In [28], an *a posteriori* error bound $\Delta(\tilde{\mu})$ for the transfer function of the ROM is proposed. This will be discussed in Section 3.3.4, where the error bound is defined for linear parametric systems, and can straightforwardly treat linear non-parametric systems as a special case. For linear non-parametric systems, the error bound $\Delta(\tilde{\mu})$ actually depends only on s , i. e. $\tilde{\mu} = s$. $\Delta(s)$ can be computed following (3.25) and (3.26), except that $\tilde{\mu}$ is replaced by s .

Similar to the reduced basis method (see Chapter 4 of Volume 2), the next expansion point s_i is iteratively selected as the point at which the error bound is maximized. Using the error bound, the ROM can be automatically generated by Algorithm 3.5. The projection matrices W and V for constructing the ROM are extended iteratively by the matrices $W_{\hat{s}}$ and $V_{\hat{s}}$ generated at the selected expansion point \hat{s} , until the error bound is below the error tolerance ϵ_{tol} . The so-called training set Ξ_{train} is a set of samples of s , which is given by the user, and which should cover the interesting range of the frequency axis. The expansion points are selected from Ξ_{train} . The matrices W^{du} and V^{du} are used to compute the error bound; see (3.25).

Algorithm 3.5: Automatic generation of a reduced model by adaptively selecting expansion points \hat{s} for non-parametrized LTI systems.

Input: System matrices $E, A, B, C, \epsilon_{\text{tol}} > 0, \Xi_{\text{train}}$: a large set of samples of s , taken over the interesting range of the frequency.

Output: The projection matrices W, V .

```

1:  $W = [], V = []$ , set  $\epsilon = \epsilon_{\text{tol}} + 1$ .
2: Initial expansion point:  $\hat{s} \in \Xi_{\text{train}}$ .
3: while  $\epsilon > \epsilon_{\text{tol}}$  do
4:    $\text{range}(V_{\hat{s}}) = \text{span}\{\tilde{B}(\hat{s}), \tilde{A}(\hat{s})\tilde{B}(\hat{s}), \dots, \tilde{A}^{q-1}(\hat{s})\tilde{B}(\hat{s})\}$ ,
5:    $\text{range}(W_{\hat{s}}) = \text{span}\{\tilde{C}(\hat{s}), \tilde{A}_c(\hat{s})\tilde{C}(\hat{s}), \dots, \tilde{A}_c^{q-1}(\hat{s})\tilde{C}(\hat{s})\}$ .
6:    $V = \text{orth}\{V, V_{\hat{s}}\}$ ,  $W^{du} = V$ .
7:    $W = \text{orth}\{W, W_{\hat{s}}\}$ ,  $V^{du} = W$ .
8:    $\hat{s} = \arg \max_{s \in \Xi_{\text{train}}} \Delta(s)$ .
9:    $\epsilon = \Delta(\hat{s})$ .
10: end while
```

Either $V_{\hat{s}}$ in Step 6 or $W_{\hat{s}}$ in Step 7 in Algorithm 3.5 can be computed by Step 1, Steps 3–29 plus Step 31 in Algorithm 3.1. Step 6 or Step 7 of Algorithm 3.5 implements the modified Gram–Schmidt process, Algorithm 3.3.

3.2.4.4 Complex expansion points

Note that the projection matrices V, W computed by the moment-matching method, as well as the multi-moment-matching method in the next section, could be complex, if the expansion point for the variable s is taken as a complex number. The ROM then has complex system matrices, even if the original system matrices are real.

In order to obtain real reduced system matrices, each complex matrix should be separated into its real part and its imaginary part, which should then be combined to obtain a real projection matrix for MOR, i. e. we need to do the following extra

step:

$$\begin{aligned} V &\leftarrow \text{orth}\{\text{Re}(V), \text{Im}(V)\}. \\ W &\leftarrow \text{orth}\{\text{Re}(W), \text{Im}(W)\}. \end{aligned}$$

Here and below, $\text{Re}(\cdot)$ and $\text{Im}(\cdot)$ is the real and imaginary part of a complex variable, respectively. Since, in \mathbb{C}^n ,

$$\text{range}\{V\} = \text{colspan}\{\text{Re}(V), \text{Im}(V)\}, \quad \text{range}\{W\} = \text{colspan}\{\text{Re}(W), \text{Im}(W)\},$$

over \mathbb{C} , the moment-matching property in Theorem 3.3 remains unchanged.

The algorithm IRKA in Section 3.2.4.1 can also introduce complex interpolation points s_i and \bar{s}_i , where \bar{s}_i is the conjugate of s_i . If s_i is complex, then in Step 2 of IRKA, $(s_i E - A)B\tilde{B}_i$ and $(\bar{s}_i E - A)B\tilde{B}_{i+1}$ are both complex vectors, which may produce complex matrices V_r, W_r . It is nevertheless not difficult to verify that the conjugate of $(s_i E - A)B\tilde{B}_i$ is $(\bar{s}_i E - A)B\tilde{B}_{i+1}$, so that they have the same real and imaginary (up to the sign) parts. Therefore, in Step 2, we can replace $(s_i E - A)B\tilde{B}_i$ and $(\bar{s}_i E - A)B\tilde{B}_{i+1}$ with $\text{Re}[(s_i E - A)B\tilde{B}_i]$ and $\text{Im}[(s_i E - A)B\tilde{B}_i]$ for any complex s_i , without changing the subspace.

3.3 Multi-moment-matching for linear parametric systems

Some parametric model order reduction (PMOR) methods are basically extensions of MOR methods for non-parametric systems. PMOR methods can be used to compute the ROM of the parametric system in (3.1), where the vector of parameters μ should be symbolically preserved in the ROM as follows:

$$\begin{aligned} \hat{E}(\mu) \frac{d\mathbf{z}(t, \mu)}{dt} &= \hat{A}(\mu)\mathbf{z}(t, \mu) + \hat{B}(\mu)\mathbf{u}(t), \\ \hat{\mathbf{y}}(t, \mu) &= \hat{C}(\mu)\mathbf{z}. \end{aligned}$$

Here $\hat{E}(\mu) = W^T E(\mu) V$, $\hat{A}(\mu) = W^T A(\mu) V$, $\hat{B}(\mu) = W^T B(\mu)$ and $\hat{C}(\mu) = C(\mu) V$. A survey of PMOR methods can be found in [13].

3.3.1 A robust algorithm

Multi-moment matching PMOR methods are reported in [17, 24], which are generalizations of the moment-matching method [35]. In this section, the robust PMOR algorithm proposed in [24] is reviewed. For ease of notation, we call this method PMOR-MM. Both methods in [17, 24] are based on Galerkin projection, i. e. $W = V$. Note that the method

in [24] is already extended to Petrov–Galerkin in [2]. For clarity and simplicity, we use Galerkin projection to explain the idea and the algorithm. Assume that $E(\mu)$, $A(\mu)$ are either in the affine form defined as

$$\begin{aligned} E(\mu) &= E_0 + E_1\mu_1 + \cdots + E_m\mu_m, \\ A(\mu) &= A_0 + A_1\mu_1 + \cdots + A_m\mu_m, \end{aligned}$$

or can be approximated in the affine form above.

To compute the matrix V , a series expansion of the state \mathbf{x} in the frequency domain is used. After Laplace transform (with zero initial condition), the original parametric system in (3.1) can be written as

$$\begin{aligned} G(\tilde{\mu})X(\tilde{\mu}) &= B(\tilde{\mu})U(\tilde{\mu}), \\ Y(\tilde{\mu}) &= C(\tilde{\mu})X(\tilde{\mu}), \end{aligned} \tag{3.15}$$

where the entries in $\tilde{\mu} = (\tilde{\mu}_1, \dots, \tilde{\mu}_p)$ are sufficiently smooth functions of the parameters μ_1, \dots, μ_m and the Laplace variable s . $U(\tilde{\mu})$ is the Laplace transform of $u(t)$. Due to the affine form of $E(\mu)$ and $A(\mu)$, $G(\tilde{\mu})$ can also be written in affine form as

$$G(\tilde{\mu}) = G_0 + G_1\tilde{\mu}_1 + \cdots + G_p\tilde{\mu}_p.$$

As a result, the state vector in the frequency domain can be written as

$$\begin{aligned} X(\tilde{\mu}) &= [G(\tilde{\mu})]^{-1}B(\tilde{\mu})U(\tilde{\mu}) \\ &= [G_0 + G_1\tilde{\mu}_1 + \cdots + G_p\tilde{\mu}_p]^{-1}B(\tilde{\mu})U(\tilde{\mu}). \end{aligned} \tag{3.16}$$

Given an expansion point $\tilde{\mu}^0 = [\tilde{\mu}_1^0, \dots, \tilde{\mu}_p^0]$, $X(\tilde{\mu})$ in (3.16) can be expanded as

$$\begin{aligned} X(\tilde{\mu}) &= [I - (\sigma_1 M_1 + \cdots + \sigma_p M_p)]^{-1} \tilde{B}_M U(\tilde{\mu}) \\ &= \sum_{i=0}^{\infty} (\sigma_1 M_1 + \cdots + \sigma_p M_p)^i \tilde{B}_M U(\tilde{\mu}), \end{aligned} \tag{3.17}$$

where $\tilde{B}_M = [G(\tilde{\mu}^0)]^{-1}B(\tilde{\mu})$, $M_i = -[G(\tilde{\mu}^0)]^{-1}G_i$, $i = 1, 2, \dots, p$, and $\sigma_i = \tilde{\mu}_i - \tilde{\mu}_i^0$, $i = 1, 2, \dots, p$. We call the coefficients in the above series expansion the *moment vectors (matrices)* of the parametrized system. The corresponding multi-moments of the transfer function are those moment vectors multiplied by C from the left.

To obtain the projection matrix V , instead of directly computing the moment vectors [17], a numerically robust method is proposed in [29], and a detailed algorithm is presented in [24]. The method combines the recursions in (3.18), with a repeated modified Gram–Schmidt process so that the moment vectors are computed implicitly. We

have

$$\begin{aligned}
 R_0 &= B_M, \\
 R_1 &= [M_1 R_0, \dots, M_p R_0], \\
 R_2 &= [M_1 R_1, \dots, M_p R_1], \\
 &\vdots \\
 R_q &= [M_1 R_{q-1}, \dots, M_p R_{q-1}], \\
 &\vdots
 \end{aligned} \tag{3.18}$$

Here, $B_M = \tilde{B}_M$, if $B(\tilde{\mu})$ does not depend on μ , i.e. $B(\tilde{\mu}) = B$. Otherwise, $B_M = [\tilde{B}_{M_1}, \dots, \tilde{B}_{M_p}]$, $\tilde{B}_{M_i} = [G(\tilde{\mu}^0)]^{-1} B_i$, $i = 1, \dots, p$, if $B(\tilde{\mu})$ can be approximated in affine form, e.g., $B(\tilde{\mu}) \approx B_1 \tilde{\mu}_1 + \dots + B_p \tilde{\mu}_p$.

Then $V := V_{\tilde{\mu}^0}$ is computed as

$$\text{range}(V_{\tilde{\mu}^0}) = \text{span}\{R_0, R_1, \dots, R_q\}_{\tilde{\mu}^0}, \tag{3.19}$$

with the sub-index denoting the dependance on the expansion point $\tilde{\mu}^0$. It is proved in [24] that the leading multi-moments of the original system match those of the ROM. The accuracy of the ROM can be improved by increasing the number of terms in (3.19), whereby more multi-moments can be matched. To be self-contained, we present the method in Algorithm 3.6.

It is noticed that the dimensions of $R_j, j = 1, \dots, q$, increase exponentially. If the number p of the parameters is larger than 2, it is advantageous to use multiple point expansion, such that only the low order moment matrices, e.g., $R_j, j \leq 1$, have to be computed for each expansion point. As a result, the order of the ROM can be kept small. Given a group of expansion points $\tilde{\mu}^i, i = 0, \dots, \ell$, a matrix $V_{\tilde{\mu}^i}$ can be computed from (3.19) for each $\tilde{\mu}^i$ as

$$\text{range}(V_{\tilde{\mu}^i}) = \text{span}\{R_0, R_1, \dots, R_q\}_{\tilde{\mu}^i}, \tag{3.20}$$

where $R_j, j = 1, \dots, q$, are defined as in (3.18), with $R_0 = B_M, B_M = [G(\tilde{\mu}^i)]^{-1} B$, or $B_M = [\tilde{B}_{M_1}, \dots, \tilde{B}_{M_p}]$, $\tilde{B}_{M_j} = [G(\tilde{\mu}^i)]^{-1} B_j, M_j = -[G(\tilde{\mu}^i)]^{-1} G_j, j = 1, 2, \dots, p$. The final projection matrix V is a combination (orthogonalization) of all the matrices $V_{\tilde{\mu}^i}$,

$$V = \text{orth}\{V_{\tilde{\mu}^0}, \dots, V_{\tilde{\mu}^\ell}\}. \tag{3.21}$$

The multi-moment-matching PMOR method is very efficient for linear parametric systems, especially for systems with affine matrices $E(\mu), A(\mu)$ [22]. The method also performs well if the matrices are not affine, but it can be well approximated in affine form [13, 30].

Algorithm 3.6: Compute $V = [v_1, v_2, \dots, v_q]$ for a parametric system (3.1), where $B(\mu)$ is generally considered as a matrix.

Input: Expansion point $\tilde{\mu}^0$, moment vectors in R_1, R_2, \dots, R_q .

Output: Projection matrix V .

```

1: Initialize  $a_1 = 0, a_2 = 0, \text{sum} = 0$ .
2: if (multiple input) then
3:   Orthogonalize the columns in  $R_0$  using the modified Gram–Schmidt
     process:  $[v_1, v_2, \dots, v_{q_1}] = \text{orth}\{R_0\}$ ,
4:    $\text{sum} = q_1$ . ( $q_1$  is the number of remaining columns after orthogonalization.)
5: else
6:   Compute the first column in  $V$ :  $v_1 = R_0 / \|R_0\|_2, \text{sum} = 1$ .
7: end if
8: Orthogonalize the columns in  $R_1, R_2, \dots, R_q$  iteratively as follows:
9: for  $i = 1, 2, \dots, q$  do
10:   $a_2 = \text{sum}$ ;
11:  for  $d = 1, 2, \dots, p$  do
12:    if  $a_1 = a_2$  then
13:      stop
14:    else
15:      for  $j = a_1 + 1, \dots, a_2$  do
16:         $w = \tilde{G}^{-1}(\tilde{\mu}^0) G_d v_j, \text{col} = \text{sum} + 1$ .
17:        for  $k = 1, 2, \dots, \text{col} - 1$  do
18:           $h = v_k^T w, w = w - h v_k$ .
19:        end for
20:        if  $\|w\|_2 > \varepsilon$  (a small value indicating deflation, e. g.,  $\varepsilon = 10^{-7}$ ) then
21:           $v_{\text{col}} = \frac{w}{\|w\|_2}, \text{sum} = \text{col}$ .
22:        end if
23:      end for ( $j$ )
24:    end if
25:  end for ( $d$ )
26:   $a_1 = a_2$ ;
27: end for ( $i$ )
28: Orthogonalize the columns in  $V$  by the modified Gram–Schmidt process.

```

3.3.2 Applicability to steady systems

The above PMOR method computes ROMs of the dynamical systems in (3.1). It is easy to see that the method can be naturally applied to steady systems:

$$\begin{aligned}
 (E_0 + E_1 \mu_1 + \dots + E_m \mu_m) \mathbf{x}(\mu) &= B(\mu) u(\mu), \\
 \mathbf{y}(\mu) &= C(\mu) \mathbf{x}(\mu).
 \end{aligned} \tag{3.22}$$

Comparing (3.22) with the Laplace transformed system (3.15), we see that they have an identical form. Consequently, the series expansion of \mathbf{x} in (3.22) can be obtained similarly to (3.17). The corresponding moment vectors can also be defined according to (3.18). Algorithm 3.6 can then be used to compute a projection matrix V . Then the ROM of (3.22) is constructed by a Galerkin projection,

$$(V^T E_0 V + V^T E_1 V \mu_1 + \cdots + V^T E_m V \mu_q) \mathbf{x}(\mu) = V^T B(\mu) u(\mu),$$

$$\hat{\mathbf{y}}(\mu) = C(\mu) V \mathbf{x}(\mu).$$

3.3.3 Structure-preserving (P)MOR for second-order systems

For the second-order systems

$$M(\mu) \frac{d^2 \mathbf{x}(t, \mu)}{dt^2} + D(\mu) \frac{d \mathbf{x}(t, \mu)}{dt} + K(\mu) \mathbf{x}(t, \mu) = B(\mu) u(t), \quad (3.23)$$

$$\mathbf{y}(t, \mu) = C(\mu) \mathbf{x}(t, \mu),$$

often arising from, e. g., mechanical engineering, it is desired that the ROM preserves the second-order structure, i. e.,

$$W^T M(\mu) V \frac{d^2 \mathbf{z}(t, \mu)}{dt^2} + W^T D(\mu) V \frac{d \mathbf{z}(t, \mu)}{dt} + W^T K(\mu) V \mathbf{z}(t, \mu) = W^T B(\mu) u(t), \quad (3.24)$$

$$\mathbf{y}(t, \mu) = C(\mu) V \mathbf{z}(t, \mu).$$

Note that PMOR-MM computes the projection matrix using the series expansion of the state vector \mathbf{x} in the frequency domain. After a Laplace transformation (assume $\mathbf{x}(t = 0, \mu) = \mathbf{0}$), the first equation in (3.23) becomes

$$s^2 M(\mu) X(s, \mu) + s D(\mu) X(s, \mu) + K(\mu) X(s, \mu) = B(\mu) U(s),$$

$$Y(s, \mu) = C(\mu) X(s, \mu).$$

Thus

$$[s^2 M(\mu) + s D(\mu) + K(\mu)] X(s, \mu) = B(\mu) U(s),$$

where $U(s)$ is the Laplace transform of the input signal $u(t)$. Defining $G(\tilde{\mu}) := s^2 M(\mu) + s D(\mu) + K(\mu)$, $\tilde{\mu} = (\tilde{\mu}_1, \tilde{\mu}_2, \tilde{\mu}_3) := (s^2, s, \mu)$, a projection matrix V can be computed following (3.15)–(3.21) in Section 3.3.1. Applying a Petrov-Galerkin projection to the second-order system, we can obtain a second-order ROM as in (3.24). Note that with a Galerkin projection, also the symmetry and definiteness properties of the coefficient matrices can be preserved in the ROM.

3.3.4 Selecting expansion points based on a posteriori error estimation

Note that the projection matrix V in (3.21) depends on the multiple expansion points $\tilde{\mu}^i, i = 1, \dots, \ell$.

In this section, we introduce an *a posteriori* error bound proposed in [28], which is an error bound for the transfer function $\hat{H}(\tilde{\mu})$ of the ROM. Given the error bound $\Delta(\tilde{\mu})$ for the ROM, the expansion points $\tilde{\mu}^i$ can be adaptively selected, and the projection matrix V can be automatically computed as shown in Algorithm 3.7.

Algorithm 3.7: Adaptively selecting expansion points $\tilde{\mu}^i$, and automatically computing V .

Input: ε_{tol} , set $\varepsilon = \varepsilon_{\text{tol}} + 1$, Ξ_{train} : a set of samples of $\tilde{\mu}$ covering the interesting domain.

Output: V .

```

1:  $V = [], V^{du} = []$ .
2: Choose an initial expansion point:  $\tilde{\mu}^0 \in \Xi, i = 0$ .
3: while  $\varepsilon > \varepsilon_{\text{tol}}$  do
4:    $\text{range}(V_{\tilde{\mu}^i}) = \text{span}\{R_0, R_1, \dots, R_q\}_{\tilde{\mu}^i}$ . (By applying Algorithm 3.6 at expansion
     point  $\tilde{\mu}^i$ .)
5:    $\text{range}(V_{\tilde{\mu}^i}^{du}) = \text{span}\{R_0^{du}, R_1^{du}, \dots, R_q^{du}\}_{\tilde{\mu}^i}$ . (By applying Algorithm 3.6 at expansion
     point  $\tilde{\mu}^i$ , and replacing  $R_0, R_1, \dots, R_q$  with  $R_0^{du}, R_1^{du}, \dots, R_q^{du}$  in (3.27).)
6:    $V = \text{orth}\{V, V_{\tilde{\mu}^i}\}, W = V$ .
7:    $V^{du} = \text{orth}\{V^{du}, V_{\tilde{\mu}^i}^{du}\}, W^{du} = V^{du}$ .
8:    $i = i + 1$ .
9:    $\tilde{\mu}^i = \arg \max_{\tilde{\mu} \in \Xi_{\text{train}}} \Delta(\tilde{\mu})$ .
10:   $\varepsilon = \Delta(\tilde{\mu}^i)$ .
11: end while.
```

For a MIMO system, the error bound $\Delta(\tilde{\mu})$ is defined as

$$\Delta(\tilde{\mu}) = \max_{ij} \Delta_{ij}(\tilde{\mu}).$$

Here $\Delta_{ij}(\tilde{\mu})$ is the error bound for the (i, j) th entry of the transfer function (it is a matrix for MIMO systems) of the ROM, i. e.,

$$|H_{ij}(\tilde{\mu}) - \hat{H}_{ij}(\tilde{\mu})| \leq \Delta_{ij}(\tilde{\mu}).$$

For a SISO system, there is no need to take the maximum. $\Delta_{ij}(\tilde{\mu})$ can be computed as

$$\Delta_{ij}(\tilde{\mu}) = \frac{\|\mathbf{r}_i^{du}(\tilde{\mu})\|_2 \|\mathbf{r}_j^{pr}(\tilde{\mu})\|_2}{\beta(\tilde{\mu})} + |(\hat{\mathbf{x}}^{du})^* \mathbf{r}_j^{pr}(\tilde{\mu})|. \quad (3.25)$$

Here and below, $(\cdot)^*$ is the conjugate transpose of a vector or a matrix. Let \mathbf{c}_i be the i th row of $C(\tilde{\mu})$ and \mathbf{b}_j be the j th column of $B(\tilde{\mu})$ in (3.15),

$$\begin{aligned} \mathbf{r}_j^{pr}(\tilde{\mu}) &= \mathbf{b}_j - G(\tilde{\mu})\hat{\mathbf{x}}_j^{pr}, \\ \hat{\mathbf{x}}_j^{pr} &= V[W^T G(\tilde{\mu})V]^{-1}W^T \mathbf{b}_j, \end{aligned} \quad (3.26)$$

$$\begin{aligned}\mathbf{r}_i^{du}(\tilde{\mu}) &= -\mathbf{c}_i^T - G^*(\tilde{\mu})\hat{\mathbf{x}}_i^{du}, \\ \hat{\mathbf{x}}_i^{du} &= -V^{du}[(W^{du})^T G^*(\tilde{\mu})V^{du}]^{-1}(W^{du})^T \mathbf{c}_i^T,\end{aligned}$$

where $\hat{\mathbf{x}}_j^{pr}$ is the approximate solution to the primal system

$$G(\tilde{\mu})\mathbf{x}_j^{pr} = \mathbf{b}_j,$$

and can be computed from the ROM of the primal system obtained with the projection matrices W, V . $\hat{\mathbf{x}}_i^{du}$ is the approximate solution to the dual system

$$G^*(\tilde{\mu})\mathbf{x}_i^{du} = -\mathbf{c}_i^T,$$

and can be computed from the ROM of the dual system obtained with the projection matrices W^{du}, V^{du} . The variable $\beta(\tilde{\mu})$ is the smallest singular value of the matrix $G(\tilde{\mu})$. The matrix V^{du} can be computed, for example, using (3.20) and (3.21), by replacing R_0, \dots, R_{qr} with $R_0^{du}, R_1^{du}, \dots, R_{qr}^{du}$, where the matrices $G(\tilde{\mu}^i)$ in R_0, \dots, R_{qr} are substituted by $G^*(\tilde{\mu}^i)$, and B is replaced with $-C^T$. More specifically,

$$\begin{aligned}R_0^{du} &= C_M^{du}, \\ R_1^{du} &= [M_1^{du} R_0^{du}, \dots, M_p^{du} R_0^{du}], \\ R_2^{du} &= [M_1^{du} R_1^{du}, \dots, M_p^{du} R_1^{du}], \\ &\vdots \\ R_q^{du} &= [M_1^{du} R_{q-1}^{du}, \dots, M_p^{du} R_{q-1}^{du}], \\ &\vdots\end{aligned}\tag{3.27}$$

$R_0^{du} = C_M^{du} = -[G^*(\tilde{\mu}^i)]^{-1}C^T$, $M_j = [G^*(\tilde{\mu}^i)]^{-1}G_j^T$, $j = 1, 2, \dots, p$. We can take $W^{du} = V^{du}$. The derivation of $\Delta(\tilde{\mu})$ is detailed in [28].

It is worth pointing out that, although the error bound is dependent on the parameter $\tilde{\mu}$, many $\tilde{\mu}$ -independent terms constituting the error bound need to be pre-computed only once, and they are repeatedly used in Algorithm 3.7 for the many samples of $\tilde{\mu}$ in Ξ_{train} . For example, when computing $\hat{\mathbf{x}}_j^{pr}$ in \mathbf{r}_j^{pr} , $W^T G_k V$, $k = 1, \dots, m$, are μ -independent, and need to be computed only once, $W^T G(\tilde{\mu})V$ is then derived by assembling $W^T G_k V$ for any value of $\tilde{\mu}$.

Algorithm 3.7 is similar to Algorithm 3.5, except that the projection matrix V is constructed for system (3.1) in the parametric case using Algorithm 3.6. At the i th iteration step, the expansion point $\tilde{\mu}^i$ is selected as the one maximizing the error bound $\Delta(\tilde{\mu})$. The projection matrix V is enriched by the matrix $V_{\tilde{\mu}^i}$ corresponding to $\tilde{\mu}^i$. The matrix V^{du} aids in computing the error bound. The most costly part of the error bound is $\beta(\tilde{\mu})$, since we need to compute the smallest singular value of the matrix $G(\tilde{\mu})$ of full dimension. The smallest singular value of the projected matrix $W^T G(\tilde{\mu})V$ could be

heuristically used to approximate $\beta(\bar{\mu})$. In [44], a method of interpolation is proposed to compute an approximation of $\beta(\bar{\mu})$, which has been shown to be accurate and cheap.

At the end of this section, we mention a method from [9], which deals with linear parametric systems with time-varying parameters. There, it is shown that these parametric systems can be equivalently considered as bilinear systems. Then suitable MOR methods for bilinear systems can be applied. MOR for bilinear systems will be discussed in the next section, where MOR based on multi-moment matching or bilinear IRKA (BIRKA) [9, 10, 12] are introduced.

3.4 Moment-matching MOR for nonlinear systems

In this section we consider mildly nonlinear systems without parameter variations,

$$\begin{aligned} E \frac{d\mathbf{x}(t)}{dt} &= \mathbf{f}(\mathbf{x}(t)) + Bu(t), \\ \mathbf{y}(t) &= C\mathbf{x}(t), \end{aligned} \tag{3.28}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ and $f(\cdot) \in \mathbb{R}^n$ is a nonlinear, vector-valued function depending on $\mathbf{x}(t)$. These nonlinear systems usually come from spatial discretizations of nonlinear PDEs. The ROM via Petrov–Galerkin projection is obtained as follows:

$$W^T E V \frac{d\mathbf{z}(t)}{dt} = W^T \mathbf{f}(V\mathbf{z}(t)) + W^T Bu(t),$$

where $W \in \mathbb{R}^{n \times r}$ and $V \in \mathbb{R}^{n \times r}$ with $W^T V = I$.

In the literature, some MOR methods for nonlinear systems are based on moment-matching. The quadratic method [16] is the simplest one. The bilinearization method [5, 46] is more accurate than the quadratic method. Methods based on variational analysis [11, 27, 37, 47, 52], in general, yield smaller errors than the previous two. A method based on a piece-wise linear approximation of the nonlinear function $f(\cdot)$ [50] could be used when dealing with strong nonlinearities. These methods are extensions of the moment-matching methods for linear systems.

3.4.1 Quadratic method

We first analyze the quadratic MOR method proposed in [16]. This method approximates the nonlinear function $\mathbf{f}(\cdot)$ by its power series expansion at, e. g., $\mathbf{x}(0) = \mathbf{0}$, which can be rewritten into a Kronecker product formulation of $\mathbf{x}(t)$ [52],

$$\begin{aligned} \mathbf{f}(\mathbf{x}(t)) &= \mathbf{f}(\mathbf{0}) + A_1 \mathbf{x}(t) + A_2 (\mathbf{x}(t) \otimes \mathbf{x}(t)) \\ &\quad + A_3 (\mathbf{x}(t) \otimes \mathbf{x}(t) \otimes \mathbf{x}(t)) + \dots, \end{aligned} \tag{3.29}$$

where $A_1 \in \mathbb{R}^{n \times n}$ is the Jacobian of \mathbf{f} and, in general, $A_j \in \mathbb{R}^{n \times n^j}$ denotes a matrix whose entries correspond to the j th-order partial derivatives of f_i w. r. t. x_1, \dots, x_n , $1 \leq i \leq n$. Here f_i, x_k are the i th and k th entry of \mathbf{f} and $\mathbf{x}(t)$, respectively. A quadratic system can be obtained by a truncation of (3.29),

$$\begin{aligned} E \frac{d\mathbf{x}(t)}{dt} &= A_1 \mathbf{x}(t) + A_2(\mathbf{x}(t) \otimes \mathbf{x}(t)) + Bu(t) + \mathbf{f}(0), \\ \mathbf{y}(t) &= C\mathbf{x}(t). \end{aligned} \quad (3.30)$$

If $\mathbf{f}(0) = 0$, the projection matrix V is computed as an orthonormal basis of the Krylov subspace $K_q(A_1^{-1}, A_1^{-1}B)$ as follows:

$$\text{range}(V) = \text{span}\{A_1^{-1}B, A_1^{-2}B, \dots, A_1^{-q}B\}.$$

Note that V is constructed only by use of the linear part of the quadratic system. A ROM is derived as

$$\begin{aligned} V^T E V \frac{d\mathbf{z}(t)}{dt} &= V^T A_1 V \mathbf{z}(t) + V^T A_2 (V \mathbf{z}(t) \otimes V \mathbf{z}(t)) + V^T Bu(t), \\ \mathbf{y}(t) &= CV \mathbf{z}(t). \end{aligned}$$

It can be seen that the idea of the quadratic method comes from the moment-matching method for linear systems. The projection matrix V is computed in the same way as in the previous moment-matching methods, but is applied to the quadratic system.

If $\mathbf{f}(0) \neq \mathbf{0}$, then the system in (3.30) can be reformulated into

$$\begin{aligned} E \frac{d\mathbf{x}(t)}{dt} &= A_1 \mathbf{x}(t) + A_2(\mathbf{x}(t) \otimes \mathbf{x}(t)) + [B, \mathbf{f}(0)][u(t), 1]^T, \\ \mathbf{y}(t) &= C\mathbf{x}(t). \end{aligned}$$

The input matrix B in (3.30) is replaced by the matrix $[B, \mathbf{f}(0)]$, which means $\mathbf{f}(0)$ can always be treated as a part of the input matrix of the system, therefore, for simplicity, we assume below that $\mathbf{f}(0) = \mathbf{0}$.

3.4.2 Bilinearization method

For a nonlinear system with $E = I$, and with single input, i. e. B is a vector \mathbf{b} , a bilinear system can be obtained by applying the Carleman linearization process to the nonlinear system (3.28) [52]. In [5, 46], the bilinear system is derived by approximating $\mathbf{f}(\mathbf{x}(t))$ with a degree-2 polynomial in the Carleman linearization process. More specifically, by use of the first three terms in (3.29), we obtain the following approximation of $\mathbf{f}(\mathbf{x}(t))$:

$$\mathbf{f}(\mathbf{x}(t)) \approx A_1 \mathbf{x}(t) + A_2(\mathbf{x}(t) \otimes \mathbf{x}(t)).$$

With the definitions

$$\begin{aligned}\mathbf{x}_\otimes &= \begin{pmatrix} \mathbf{x}(t) \\ \mathbf{x}(t) \otimes \mathbf{x}(t) \end{pmatrix}, \quad B_\otimes = \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix}, \quad C_\otimes = (C, \mathbf{0}), \\ A_\otimes &= \begin{pmatrix} A_1 & A_2 \\ 0 & A_1 \otimes I + I \otimes A_1 \end{pmatrix}, \\ N_\otimes &= \begin{pmatrix} 0 & 0 \\ \mathbf{b} \otimes I + I \otimes \mathbf{b} & 0 \end{pmatrix},\end{aligned}$$

the nonlinear system (3.28) ($E = I$) can be approximated by the following bilinear system:

$$\begin{aligned}\frac{d\mathbf{x}_\otimes}{dt} &= A_\otimes \mathbf{x}_\otimes + N_\otimes \mathbf{x}_\otimes u(t) + B_\otimes u(t), \\ \mathbf{y}(t) &= C_\otimes \mathbf{x}_\otimes.\end{aligned}\tag{3.31}$$

The derivation can be easily extended to multi-input systems with $B \in \mathbb{R}^{n \times n_I}$ being a matrix. After a few more calculations, the following bilinear system with multiple inputs can be obtained:

$$\begin{aligned}\frac{d\mathbf{x}_\otimes}{dt} &= A_\otimes \mathbf{x}_\otimes + \sum_{i=1}^{n_I} N_\otimes^{(i)} \mathbf{x}_\otimes u_i(t) + B_\otimes u(t), \\ \mathbf{y}(t) &= C_\otimes \mathbf{x}_\otimes,\end{aligned}\tag{3.32}$$

where $u(t) := (u_1(t), \dots, u_{n_I}(t))^T$, $B := (\mathbf{b}_1, \dots, \mathbf{b}_{n_I})$ and

$$B_\otimes = \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix}, \quad N_\otimes^{(i)} = \begin{pmatrix} 0 & 0 \\ \mathbf{b}_i \otimes I + I \otimes \mathbf{b}_i & 0 \end{pmatrix}.$$

Given a bilinear system with E singular, several modified MOR schemes are proposed in [1, 12, 33, 34], but will not be discussed in this chapter due to space limitations. We can see that the above bilinear system is of much larger state-space dimension than the original nonlinear system (3.28). In the following we will introduce the process of constructing the projection matrix V for MOR.

Once the nonlinear system is approximated by the bilinear system (3.31) or (3.32), there are several choices for applying MOR. Multi-moment-matching methods extend the moment-matching methods for linear systems to bilinear systems by studying the transfer function of the bilinear system. Gramian-based bilinear MOR methods construct the matrices W and V by exploring the Gramian matrices of the bilinear systems. We focus on multi-moment-matching methods in this chapter.

3.4.2.1 Constructing W and V

Multivariate transfer functions and multi-moment-matching

The bilinearization MOR methods in [5, 46, 23] construct the matrices W, V , $W = V$ by approximating the transfer function of the bilinear system. Note that only SISO systems are considered in [5, 46]. For MIMO systems, the expression of the transfer function will be different; see [43]. In [43], a method similar to that in [46] was extended to MIMO systems. In the following description, we consider only SISO bilinear systems.

Under the assumption $E = I$, the identity matrix, the output response of the bilinear system (3.31) can be expressed by a Volterra series [52],

$$y(t) = \sum_{k=1}^{\infty} y_k(t),$$

where $y_k(t)$ is described in (3.33) and (3.34). In (3.34), $h_k^{(\text{reg})}$ is called the *regular kernel of degree k* . The multivariate Laplace transform of this kernel defines the k th multivariate transfer function $H_k^{(\text{reg})}$ in (3.35). We have

$$y_k(t) = \int_0^t \int_0^{t_1} \dots \int_0^{t_{k-1}} h_k^{(\text{reg})}(t_1, \dots, t_k) u(t - t_1 - \dots - t_k) \dots u(t - t_k) dt_k \dots dt_1, \quad (3.33)$$

$$h_k^{(\text{reg})}(t_1, \dots, t_k) = C_{\otimes} e^{A_{\otimes} t_k} N_{\otimes} e^{A_{\otimes} t_{k-1}} \dots N_{\otimes} e^{A_{\otimes} t_1} B_{\otimes}, \quad (3.34)$$

$$H_k^{(\text{reg})}(s_1, \dots, s_k) = C_{\otimes} (s_k I - A_{\otimes})^{-1} N_{\otimes} (s_{k-1} I - A_{\otimes})^{-1} \dots N_{\otimes} (s_1 I - A_{\otimes})^{-1} B_{\otimes}. \quad (3.35)$$

By using the Neumann expansion,

$$(I - sA_{\otimes}^{-1})^{-1} = I + sA_{\otimes}^{-1} + s^2 A_{\otimes}^{-2} + s^3 A_{\otimes}^{-3} + \dots,$$

$H_k^{(\text{reg})}(s_1, s_2, \dots, s_k)$ can be expanded into a multivariable Maclaurin series in (3.36), with the so-called k th-order multi-moments $m(l_1, \dots, l_k)$ being defined in (3.37). We have

$$H_k^{(\text{reg})}(s_1, \dots, s_k) = \sum_{l_k=1}^{\infty} \dots \sum_{l_1=1}^{\infty} m(l_1, \dots, l_k) s_1^{l_1-1} s_2^{l_2-1} \dots s_k^{l_k-1}, \quad (3.36)$$

$$m(l_1, \dots, l_k) = (-1)^k C_{\otimes} A_{\otimes}^{-l_k} N_{\otimes} \dots A_{\otimes}^{-l_2} N_{\otimes} A_{\otimes}^{-l_1} B_{\otimes}, \quad l_1, \dots, l_k = 1, 2, \dots \quad (3.37)$$

Deriving W and V

In [23, 7], the BICOMB method constructs the projection matrix V from a series of Krylov subspaces in the following steps:

$$\text{range}(V^{(1)}) = K_{q_1}(A_{\otimes}^{-1}, A_{\otimes}^{-1} B_{\otimes}), \quad (3.38)$$

and for $j > 1$

$$\text{range}(V^{(j)}) = K_{q_j}(A_{\otimes}^{-1}, A_{\otimes}^{-1} N_{\otimes} V^{(j-1)}). \quad (3.39)$$

The final projection matrix V is

$$\text{range}(V) = \text{orth}\{V^{(1)}, \dots, V^{(J)}\}. \quad (3.40)$$

Here,

$$K_{q_j}(A^{-1}, R) := \{R, A^{-1}R, \dots, A^{-q_j+1}R\}$$

is a block Krylov subspace generated by $R = A_{\otimes}^{-1}B_{\otimes}$, $j = 1$, or $R = A_{\otimes}^{-1}N_{\otimes}V^{(j-1)}$, $j > 1$. Note that each $V^{(k)}$ actually tries to match the multi-moments of the k th transfer function $H_k^{(\text{reg})}$; the multi-moment-matching property of the ROM can be found in, e. g., [23]. Applying $x_{\otimes} \approx Vz_{\otimes}$ to (3.31), the ROM of the nonlinear system (3.28) is given by

$$\begin{aligned} \frac{dz_{\otimes}}{dt} &= \hat{A}_{\otimes}z_{\otimes} + \hat{N}_{\otimes}z_{\otimes}u(t) + \hat{B}_{\otimes}u(t), \\ \hat{y}(t) &= \hat{C}_{\otimes}z_{\otimes}, \end{aligned} \quad (3.41)$$

where $\hat{A}_{\otimes} = V^T A_{\otimes} V$, $\hat{N}_{\otimes} = V^T N_{\otimes} V$, $\hat{B}_{\otimes} = V^T B_{\otimes}$, $\hat{C}_{\otimes} = C_{\otimes} V$. For simulation results of the BICOMB method, we refer to [7]. Algorithm 3.1 can be mildly modified to Algorithm 3.8 to compute $V^{(j)}$ in (3.38)–(3.40). Algorithm 3.8 computes a matrix V from the block Krylov subspaces defined as follows:

$$\text{range}(V^{(j)}) = K_{q_j}(M^{-1}, R_j), \quad j = 1, \dots, J. \quad (3.42)$$

The final projection matrix V is computed as in (3.40). Let $M = A_{\otimes}$, $R_1 = A_{\otimes}^{-1}B_{\otimes}$ and $R_j = A_{\otimes}^{-1}N_{\otimes}V^{(j-1)}$ for $j > 1$, Algorithm 3.8 can be used to compute the matrix V in (3.40) for the ROM (3.41) of the bilinear system.

3.4.3 Variational analysis method

The third set of nonlinear MOR methods [11, 27, 37, 42, 47] originates from variational analysis of nonlinear system theory [52].

3.4.3.1 Methods using polynomial approximation

In [27, 42, 47], the original nonlinear system is first approximated by a polynomial system, then variational analysis is applied to the polynomial system to obtain a reduced polynomial system. In the following, we describe the method developed in [27]. Its main difference from the method in [47] is the construction of the projection matrices V_2 and V_3 , and will be explained later.

Algorithm 3.8: Compute V in (3.40).

Input: Matrices of the block Krylov subspace in (3.42): $M, R_j, j = 1, 2, \dots, J$.

Output: Matrix V in (3.40) with orthonormal columns.

```

1: Initialize  $a_1 = 0, a_2 = 0, \text{sum} = 0$ .
2: for  $j = 1, \dots, J$  do
3:   if  $R_j$  is a matrix then
4:     Orthogonalize the columns in  $R_j$  using a modified Gram–Schmidt
       process:  $[v_1, v_2, \dots, v_{m_j}] = \text{orth}\{R_j\}$ .
5:      $\text{sum} = m_j$ . ( $m_j$  is the number of remaining columns after deflation.)
6:   else
7:     Compute the first column in  $V_j$ :  $v_1 = R_j / \|R_j\|_2$ . ( $R_j$  is a vector.)
8:      $\text{sum} = 1$ .
9:   end if
10:  for  $i = 1, 2, \dots, q_j - 1$  do
11:     $a_2 = \text{sum}$ .
12:    if  $a_1 = a_2$  then
13:      break, go to 2.
14:    else
15:      for  $d = a_1 + 1, \dots, a_2$  do
16:         $w = Mv_d, \text{col} = \text{sum} + 1$ .
17:        for  $k = 1, 2, \dots, \text{col} - 1$  do
18:           $h = v_k^T w, w = w - hv_k$ .
19:        end for
20:        if  $\|w\|_2 > \varepsilon$  (a small value indicating deflation, e. g.,  $\varepsilon = 10^{-7}$ ) then
21:           $v_{\text{col}} = \frac{w}{\|w\|_2}, \text{sum} = \text{col}$ .
22:        end if
23:      end for ( $d$ )
24:    end if
25:     $a_1 = a_2$ .
26:  end for ( $i$ )
27:   $V_j = [v_1, \dots, v_{\text{sum}}]$ ,
28: end for ( $j$ )
29: Orthogonalize the columns in  $[V_1, \dots, V_J]$  by the modified Gram–Schmidt process
    to obtain  $V$ , i. e.  $V := \text{orth}\{V_1, \dots, V_J\}$ .

```

With the power series expansion of $\mathbf{f}(\mathbf{x}(t))$ in (3.29), the original nonlinear system (3.28) is first approximated by a degree-2 polynomial system

$$\begin{aligned}
 \frac{d\mathbf{x}(t)}{dt} &= A_1 \mathbf{x}(t) + A_2 (\mathbf{x}(t) \otimes \mathbf{x}(t)) + B u(t), \\
 \mathbf{y}(t) &= C \mathbf{x}(t),
 \end{aligned} \tag{3.43}$$

or by a degree-3 polynomial system

$$\begin{aligned}\frac{d\mathbf{x}(t)}{dt} &= A_1\mathbf{x}(t) + A_2(\mathbf{x}(t) \otimes \mathbf{x}(t)) \\ &\quad + A_3(\mathbf{x}(t) \otimes \mathbf{x}(t) \otimes \mathbf{x}(t)) + Bu(t), \\ \mathbf{y}(t) &= C\mathbf{x}(t).\end{aligned}\tag{3.44}$$

Consider the response of (3.28) to a variation of the input $au(t)$,

$$\begin{aligned}\frac{d\mathbf{x}(t)}{dt} &= f(\mathbf{x}(t)) + B(au(t)), \\ \mathbf{y}(t) &= C\mathbf{x}(t),\end{aligned}\tag{3.45}$$

where α is an arbitrarily small-valued variable. Assuming that the response to $u(t) = 0$ is $\mathbf{x}(t) = \mathbf{0}$ (in [52], it is called a forced response), then $\mathbf{x}(t)$, as a function of α , can be expanded into a power series of α around $\alpha_0 = 0$,

$$\mathbf{x}(t) = \alpha\mathbf{x}_1(t) + \alpha^2\mathbf{x}_2(t) + \alpha^3\mathbf{x}_3(t) + \cdots,\tag{3.46}$$

where the first term of the series is $\mathbf{x}_0(t) = \mathbf{x}(t, \alpha_0 = 0) = \mathbf{0}$, since when $\alpha_0 = 0$, $\alpha_0 u(t) = 0$. The corresponding response $\mathbf{x}(t, \alpha_0 = 0) = \mathbf{0}$ is then removed from the above expansion. Substituting both (3.46) and (3.29) into the right hand side and (3.46) into the left hand side of (3.45), we get

$$\begin{aligned}\alpha\frac{d\mathbf{x}_1(t)}{dt} + \alpha^2\frac{d\mathbf{x}_2(t)}{dt} + \alpha^3\frac{d\mathbf{x}_3(t)}{dt} + \cdots &= \alpha A_1\mathbf{x}_1(t) \\ &\quad + \alpha^2[A_1\mathbf{x}_2(t) + A_2(\mathbf{x}_1(t) \otimes \mathbf{x}_1(t))] + \cdots + B(au(t)).\end{aligned}$$

Since this equation holds for all α , coefficients of powers of α can be equated. This gives the variational equations:

$$\frac{d\mathbf{x}_1(t)}{dt} = A_1\mathbf{x}_1(t) + Bu(t),\tag{3.47}$$

$$\frac{d\mathbf{x}_2(t)}{dt} = A_1\mathbf{x}_2(t) + A_2(\mathbf{x}_1(t) \otimes \mathbf{x}_1(t)),\tag{3.48}$$

$$\begin{aligned}\frac{d\mathbf{x}_3(t)}{dt} &= A_1\mathbf{x}_3(t) + A_2(\mathbf{x}_1(t) \otimes \mathbf{x}_2(t) + \mathbf{x}_2(t) \otimes \mathbf{x}_1(t)) \\ &\quad + A_3(\mathbf{x}_1(t) \otimes \mathbf{x}_1(t) \otimes \mathbf{x}_1(t)),\end{aligned}\tag{3.49}$$

\vdots

It is worth pointing out that the assumptions on the forced response can be relaxed, and similar variational equations of $\mathbf{x}_\delta = \mathbf{x}(t) - \hat{\mathbf{x}}(t)$ can be derived. Here, $\hat{\mathbf{x}}(t)$ is the response to a certain input $\hat{u}(t)$ for a fixed initial state $\mathbf{x}(t = 0) = \mathbf{x}_0^*$. For a detailed discussion; see Section 3.4 in [52].

Deriving W and V

We notice that all of these variational equations are linear systems of order n for the vectors of the unknowns $\mathbf{x}_1(t)$, $\mathbf{x}_2(t)$, \dots , respectively. Since $\mathbf{x}(t)$ is a linear combination of $\mathbf{x}_1(t)$, $\mathbf{x}_2(t)$, \dots (see (3.46)), it is in the subspace spanned by $\mathbf{x}_1(t)$, $\mathbf{x}_2(t)$, \dots . The projection matrix V can be computed from the subspace containing $\mathbf{x}_1(t)$, $\mathbf{x}_2(t)$, \dots .

Building upon this observation, the method in [27] constructs V based on the linear variational equations (3.47)–(3.49) rather than from the nonlinear system. From the moment-matching MOR for linear systems, a projection matrix V_1 for $\mathbf{x}_1(t)$ of the first linear system (3.47) is constructed as

$$\text{range}(V_1) = \text{span}\{A_1^{-1}B, A_1^{-2}B, \dots, A_1^{-q_1}B\}. \quad (3.50)$$

Then $\mathbf{x}_1(t)$ can be approximated by $\mathbf{x}_1(t) \approx V_1 \mathbf{z}_1(t)$. A projection matrix V_2 for $\mathbf{x}_2(t)$ of the second linear system (3.48) is similarly constructed by

$$\text{range}(V_2) = \text{span}\{A_1^{-1}A_2, A_1^{-2}A_2, \dots, A_1^{-q_2}A_2\}, \quad (3.51)$$

such that $\mathbf{x}_2(t) \approx V_2 \mathbf{z}_2(t)$. A projection matrix V_3 for $\mathbf{x}_3(t)$ in (3.49) can be derived in a similar way. From (3.46), we have

$$\mathbf{x}(t) \approx \alpha V_1 \mathbf{z}_1(t) + \alpha^2 V_2 \mathbf{z}_2(t),$$

or

$$\mathbf{x}(t) \approx \alpha V_1 \mathbf{z}_1(t) + \alpha^2 V_2 \mathbf{z}_2(t) + \alpha^3 V_3 \mathbf{z}_3(t),$$

which indicates that the solution $\mathbf{x}(t)$ to (3.43) or (3.44) can be approximated by a linear combination of the columns of V_1 , V_2 or V_1 , V_2 , V_3 . Therefore the final projection matrix V can be computed as

$$\text{range}(V) = \text{orth}\{V_1, \dots, V_J\}, \quad J = 2 \text{ or } 3. \quad (3.52)$$

The ROM is thus derived from the polynomial system (3.43) or (3.44) as follows:

$$\begin{aligned} \frac{d\mathbf{z}(t)}{dt} &= V^T A_1 V \mathbf{z}(t) + V^T A_2 (V \mathbf{z}(t) \otimes V \mathbf{z}(t)) + V^T B u(t), \\ y(t) &= C V \mathbf{z}(t), \end{aligned} \quad (3.53)$$

or

$$\begin{aligned} \frac{d\mathbf{z}(t)}{dt} &= V^T A_1 V \mathbf{z}(t) + V^T A_2 (V \mathbf{z}(t) \otimes V \mathbf{z}(t)) \\ &\quad + V^T A_3 (V \mathbf{z}(t) \otimes V \mathbf{z}(t) \otimes V \mathbf{z}(t)) + V^T B u(t), \\ y(t) &= C V \mathbf{z}(t). \end{aligned} \quad (3.54)$$

The advantage of this method is that it has the flexibility of using a more accurate polynomial system (3.44) to approximate the original nonlinear system. It is possible that for the quadratic method, the system (3.30) can also be replaced by a more accurate polynomial system. However, the projection matrix V computed by the quadratic method might be less accurate than the matrix V in (3.52), because it is computed using only the linear part of the nonlinear system. For an approximation of the original nonlinear system (3.28), the bilinear system is less accurate than the polynomial system (3.44). Moreover, since the bilinear system is derived by approximating the nonlinear function $\mathbf{f}(\mathbf{x})$ by its power series expansion up to second order, the projection matrix V also only uses the information of the series expansion of $\mathbf{f}(\mathbf{x})$ at most to the second order, which is less accurate than the matrix V computed by the variational analysis method.

Again, let $M = A_1$, $R_1 = A_1^{-1}B$, $R_2 = A_1^{-1}A_2$ in (3.50), (3.51), then Algorithm 3.8 can be used to compute V in (3.52). Since there are many columns in A_2 , it is not possible to use all the columns. Instead, one may take only the first several columns, e. g., $R_2 = A_1^{-1}A_2(:, 1:q)$, $q \ll n$, where MATLAB notation is used.

In [47], the second projection matrix \tilde{V}_2 is constructed from the approximate system by replacing \mathbf{x}_1 with $V_1\mathbf{z}_1$ in (3.48) to get

$$\frac{d\mathbf{x}_2(t)}{dt} = A_1\mathbf{x}_2(t) + A_2(V_1\mathbf{z}_1(t) \otimes V_1\mathbf{z}_1(t)).$$

Then

$$\text{range}(\tilde{V}_2) = \text{span}\{A_1^{-1}A_2(V_1 \otimes V_1), A_1^{-2}A_2(V_1 \otimes V_1), \dots, A_1^{-q_2}A_2(V_1 \otimes V_1)\}. \quad (3.55)$$

The advantage of this approach is that there are much fewer columns in $A_2(V_1 \otimes V_1)$ than in A_2 in (3.51). Thus, \tilde{V}_2 matches more moments than V_2 in (3.51) if the matrices have the same number of columns. However, \tilde{V}_2 only matches approximate moments because the input matrix A_2 in (3.48) is approximated by $A_2(V_1 \otimes V_1)$ in (3.55). Therefore, although \tilde{V}_2 matches more moments, its accuracy is impaired by the approximate moments. The accuracy of the two methods is compared in [7].

At the end of this subsection, we would like to mention another method [42] which is based on both the Volterra series expansion of the output response and variational analysis. In [42], the original system (3.28) is approximated by the polynomial system in (3.44). Then the Volterra series representation of the output response of the polynomial system is employed to introduce the nonlinear transfer functions of (3.44). The k th-order nonlinear transfer function is similar to the k th transfer function $H_k^{(\text{reg})}(s_1, s_2, \dots, s_k)$ for the bilinear system. The projection matrix V is constructed based on the moments of the nonlinear transfer functions. Instead of performing the Laplace transform of the Volterra kernels as in (3.35), the nonlinear transfer functions are computed from the variational linear systems (3.47)–(3.49), whose transfer functions are equivalent with the first-order, second-order and third-order nonlinear

transfer functions, respectively. The basic idea of [42] is quite similar to the methods in [5] and [46]. The main difference is that [5] and [46] are based on a bilinear approximation of the original nonlinear system, whereas [42] is based on the more accurate approximation (3.44).

3.4.3.2 Methods based on quadratic–bilinearization

All those previous nonlinear model reduction methods first approximate the nonlinear function $\mathbf{f}(\mathbf{x}(t))$ by a polynomial, then reduce the approximate polynomial system to a small dimension. When the function $\mathbf{f}(\mathbf{x}(t))$ is weakly nonlinear, it is usually sufficient to approximate it by a degree-2 polynomial or degree-3 polynomial. Meanwhile, when $\mathbf{f}(\mathbf{x}(t))$ is strongly nonlinear, the low-degree polynomial approximation is not accurate. It is possible to employ higher order polynomials to improve the accuracy, but with much more complexity. Furthermore, the storage requirement for higher order polynomials is prohibitive if the matrix dimension is very large. Therefore, these methods are more suitable for weakly nonlinear systems.

The methods based on quadratic–bilinearization provide a solution to the above issues of polynomial approximation. Instead of approximating the nonlinear part $\mathbf{f}(\mathbf{x})$ by a polynomial function, equivalent transformations are applied to the nonlinear system in (3.28). The nonlinear system is first “lifted” to a polynomial system by adding polynomial algebraic equations or by taking Lie derivatives and adding more differential equations. The polynomial system is then transformed into a quadratic–bilinear system by either adding quadratic algebraic equations or taking Lie derivatives again. No accuracy is lost during the transformations. The detailed explanation can be found in [36, 37].

The equivalent quadratic–bilinear system is

$$G_0 \tilde{\mathbf{x}} = G_1 \tilde{\mathbf{x}} + G_2(\tilde{\mathbf{x}} \otimes \tilde{\mathbf{x}}) + D_1 \tilde{\mathbf{x}}u + D_2(\tilde{\mathbf{x}} \otimes \tilde{\mathbf{x}})u + \tilde{B}u(t), \quad (3.56)$$

where $\tilde{\mathbf{x}}$ is the lifted state vector after more state variables are added to the state vector \mathbf{x} . Notice that in [36, 37], the system (3.56) is called quadratic-linear differential algebraic equation (QLDAE). However, the system above obviously includes the bilinear term $D_1 \tilde{\mathbf{x}}u$ and the quadratic–bilinear term $D_2(\tilde{\mathbf{x}} \otimes \tilde{\mathbf{x}})u$. Therefore, the notion quadratic–bilinear differential algebraic equations (QBDAEs) introduced in [11] is used in this paper.

Once the QBDAEs are derived after several steps of transformations, the variational analysis (3.45)–(3.49) in the previous subsection can be applied to the QBDAEs. The projection matrix V can also be computed likewise. Then a Galerkin projection can be applied to (3.56) to get the reduced QBDAEs, which is considered as the ROM for the original nonlinear system in (3.28).

Recall that, from the second variational equation (3.48), the input matrix has many vectors, which makes the computation of the projection matrix V_2 tricky.

In [36, 37], a different way of computing the projection matrix V is proposed based on the transfer functions of the QBDAEs (3.56). The expression of the transfer functions of the QBDAEs can be originally found in [52]. For example, assuming for simplicity $G_0 = I$, the first two transfer functions are

$$\begin{aligned} H_1(s) &= L^T (sI - G_1)^{-1} B, \\ H_2(s_1, s_2) &= \frac{1}{2!} L^T [(s_1 + s_2)I - G_1]^{-1} \\ &\quad \times \{G_2[H_1(s_1) \otimes H_1(s_2) + H_1(s_2) \otimes H_1(s_1)] + D_1[H_1(s_1) + H_2(s_2)]\}. \end{aligned} \quad (3.57)$$

Using Taylor series expansions of the transfer functions, the matrix V can be recursively computed from the coefficients of the series expansions. The series expansions of H_1 and H_2 about zero (adaptation to nonzero expansion points is straightforward) are given as

$$\begin{aligned} H_1(s) &= L^T \sum_{k=0}^{\infty} A^k \tilde{B} s^k, \\ H_2(s_1, s_2) &= \frac{1}{2!} L^T \sum_{i=0}^k A^{k+1} (s_1 + s_2)^k \left\{ G_2 \left[\left(\sum_{k=0}^{\infty} A^k \tilde{B} s_1^k \right) \otimes \left(\sum_{k=0}^{\infty} A^k \tilde{B} s_2^k \right) \right. \right. \\ &\quad \left. \left. + \left(\sum_{k=0}^{\infty} A^k \tilde{B} s_2^k \right) \otimes \left(\sum_{k=0}^{\infty} A^k \tilde{B} s_1^k \right) \right] \right. \\ &\quad \left. + D_1 \left[\sum_{k=0}^{\infty} A^k \tilde{B} s_1^k + \sum_{k=0}^{\infty} A^k \tilde{B} s_2^k \right] \right\}, \end{aligned}$$

where $A = G_1^{-1}$, $\tilde{B} = -G_1^{-1}B$. In [36, 37], the projection matrix V is constructed as

$$\begin{aligned} \text{range}(V_1) &= \text{span}\{A^i \tilde{B}, i \leq q\}, \\ \text{range}(V_2) &= \text{span}\{A^{i+1} D_1 A^j \tilde{B}, i + j \leq q\}, \\ \text{range}(V_3) &= \text{span}\{A^{i+1} G_2 (A^j \tilde{B}) \otimes (A^k \tilde{B}), i + j + k \leq q, k \leq j\}, \\ \text{range}(V) &= \text{span}\{V_1, V_2, V_3\}. \end{aligned} \quad (3.58)$$

It can be seen that, if the system matrix B is a vector, the Kronecker product $(A^j \tilde{B}) \otimes (A^k \tilde{B})$ is also a vector so that the construction of V_3 is easy. In general, if B has m columns, $(A^j \tilde{B}) \otimes (A^k \tilde{B})$ has m^2 columns. The number of columns in $(A^j \tilde{B}) \otimes (A^k \tilde{B})$ is still moderate if m is small. This is an advantage over the way of computing V through variational analysis.

Algorithm 3.8 can also be used to compute V in (3.58), where we need to let $M = G_1$, $R_1 = \tilde{B}$, $R_2 = D_1 V_1$, $R_3 = G_2 A^j V_2 \otimes V_2$. Note that in order to compute V_2 , we use V_1 instead of $A^j \tilde{B}$ in R_2 , since V_1 is already the basis of the subspace spanned by $A^j \tilde{B}$, $j \leq q$. Similarly, we use $V_2 \otimes V_2$ rather than $A^j \tilde{B} \otimes A^k \tilde{B}$. This way, we avoid the issues of how to choose proper values of j, k . When $A^j \tilde{B}$ is replaced by V_1 in R_2 , V_1 is a matrix instead of

a vector. However, usually there are a few columns in V_1 , which still keeps the column dimensions of R_2, R_3 moderate.

In [11], the method is extended to two-sided projection based on the transfer functions (3.57) of the QBDAEs. It is proved that by using two-sided projection, the reduced transfer function matches almost twice as many moments of the original transfer functions as with the one-sided projection used in [36, 37]. Simulation results also show better accuracy than the one-sided projection. However, the two-sided projection sometimes causes numerical instability, which may produce unstable reduced models [11].

Note that the subspace dimension in (3.58) will grow exponentially if the coefficients of the series expansion of the higher order transfer functions, e. g. $H_3(s_1, s_2, s_3)$, are also included to compute the projection matrix V . This easily leads to a ROM with no reduced number of equations. In [58], the higher order multivariate transfer functions $H_2(s_1, s_2)$, $H_3(s_1, s_2, s_3)$, \dots , are transformed to single- s transfer functions $H_2(s)$, $H_3(s)$, \dots by association of variables without losing accuracy. The series expansion of $H_2(s)$ or $H_3(s)$ only depends on the single variable s , such that the exponential growth of the subspace dimension can be avoided. Compared with the method in [37], a more compact ROM with the same accuracy can be obtained. The theory on association of variables can be found in [52].

Recall that, if the original nonlinear system is a system of ODEs, the QBDAEs usually constitute a system of differential-algebraic equations after quadratic–bilinearization, i. e., G_0 could be singular. In general, it is still unclear how to determine the index of the QBDAEs which may cause problems when the ROM is solved.

3.4.3.3 Other variants

Algorithm IRKA has been extended to the *bilinear iterative rational Krylov algorithm* (BIRKA) in [10] to compute the ROMs of bilinear systems, which iteratively updates a set of interpolation points such that the ROM satisfies the necessary conditions of \mathcal{H}_2 -optimality. Upon convergence, the BIRKA method produces a ROM whose Volterra series interpolates that of the original bilinear system at the mirror images of the poles of the ROM. However, the computational cost of BIRKA for large-scale systems is high and it is also not possible to match higher-order derivatives. Regarding computational cost of BIRKA, efforts have been done in [12] to reduce the computational cost for some special systems.

3.4.4 Trajectory piece-wise linear method

The trajectory piece-wise linear method in [49, 50] is proposed to deal with strongly nonlinear systems. An error bound for this method is proposed in [51], where the

stability and passivity of the ROM are also discussed. The trajectory piece-wise linear method first linearizes the nonlinear function $\mathbf{f}(\mathbf{x}(t))$ at a number of linearization points \mathbf{x}_i , $i = 0, 1, 2, \dots, k$, then approximates $\mathbf{f}(\mathbf{x}(t))$ by the weighted sum of these linearizations, $\mathbf{f}(\mathbf{x}_i) + \mathbf{A}_i(\mathbf{x} - \mathbf{x}_i)$. Finally, the original nonlinear system is approximated by the following weighted sum of linear systems:

$$\begin{aligned} \frac{d\mathbf{x}(t)}{dt} &= \sum_{i=0}^{s-1} \tilde{w}_i(\mathbf{x}) \mathbf{f}(\mathbf{x}_i) + \sum_{i=0}^{s-1} \tilde{w}_i(\mathbf{x}) \mathbf{A}_i(\mathbf{x} - \mathbf{x}_i) + \mathbf{B}u(t), \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t). \end{aligned}$$

Once a projection matrix V is obtained, the ROM can be obtained using a Galerkin projection. In [49, 50], V is obtained by applying the moment-matching method to each linearized system.

The linearization points are chosen by selecting a training input and simulating the original nonlinear system. The procedure is simply as follows: (1) A linearized model around state \mathbf{x}_i (initially $i = 0$) is generated. (2) The original nonlinear system is simulated while $\min_{0 \leq j \leq i} \|\mathbf{x} - \mathbf{x}_j\| < \delta$, i. e. while the current state \mathbf{x} is close enough to any of the previous linearization points. (3) A new linearization point \mathbf{x}_{i+1} is taken as the first state violating $\|\mathbf{x} - \mathbf{x}_i\| < \delta$, then return to step (1). Note that in order to get the linearization points, the original full system has to be simulated. Instead of simulating the full system, a fast algorithm for computing an approximate trajectory is also proposed in the paper.

The weak point of this method is that training inputs have to be chosen. In general, it is unclear how to choose the optimal training inputs so that the trajectory represents the behavior of the state vector $\mathbf{x}(t)$. If the training inputs are chosen far away from the actual inputs, then the computed trajectory of the unknown vector will depart from the real behavior of the state vector $\mathbf{x}(t)$ and the ROM will lose accuracy. Computation of the weight functions \tilde{w}_i in the above linear system is also more or less heuristic. Some related papers based on piece-wise linear ideas are [15, 18, 19, 20, 53, 54, 55].

3.5 Extension to parametric nonlinear systems

Some of the above nonlinear MOR methods could be extended to deal with parametric nonlinear systems, though little relevant work has been done. Often the nonlinear system also involves parameter variations, i. e.,

$$\begin{aligned} E(\mu) \frac{d\mathbf{x}(t, \mu)}{dt} &= \mathbf{f}(\mathbf{x}(t, \mu), \mu) + \mathbf{B}(\mu)u(t), \\ \mathbf{y}(t, \mu) &= \mathbf{C}(\mu)\mathbf{x}(t, \mu), \end{aligned} \tag{3.59}$$

where $\mu \in \mathbb{R}^p$ is the vector of geometrical or physical parameters. When $\mathbf{f}(\mathbf{x}(t, \mu), \mu)$ is mildly nonlinear, many of the above introduced methods can be extended to solving (3.59).

The nonlinear system in (3.59) can be transformed to a parametric bilinear system using the same technique as introduced in Section 3.4.2, so the resulting system could be considered as a linear parametric system, where the input $u(t)$ could be taken as an extra parameter. The PMOR-MM method can then be applied to obtain the ROM. Extension of both the quadratic method and the variational analysis approach introduced in Section 3.4.1 and Section 3.4.3 is straightforward. In the following, we discuss these extensions in more detail.

3.5.1 Quadratic PMOR

The quadratic method in Section 3.4.1 depends on the power series expansion of the nonlinear function $\mathbf{f}(\mathbf{x}(t))$. For parameter dependent $\mathbf{f}(\mathbf{x}(t, \mu), \mu)$, the corresponding power series expansion may be written as

$$\begin{aligned} \mathbf{f}(\mathbf{x}(t, \mu), \mu) &= \mathbf{f}(\mathbf{0}) + A_1(\mu)\mathbf{x}(t, \mu) + A_2(\mu)(\mathbf{x}(t, \mu) \otimes \mathbf{x}(t, \mu)) \\ &\quad + A_3(\mu)(\mathbf{x}(t, \mu) \otimes \mathbf{x}(t, \mu) \otimes \mathbf{x}(t, \mu)) + \dots \end{aligned} \quad (3.60)$$

The approximated quadratic system is

$$\begin{aligned} E(\mu) \frac{d\mathbf{x}(t, \mu)}{dt} &= A_1(\mu)\mathbf{x}(t, \mu) + A_2(\mu)(\mathbf{x}(t, \mu) \otimes \mathbf{x}(t, \mu)) + \tilde{B}(\mu)\tilde{u}(t), \\ \mathbf{y}(t, \mu) &= C(\mu)\mathbf{x}(t, \mu), \end{aligned} \quad (3.61)$$

where $\tilde{B}(\mu) = [B(\mu), \mathbf{f}(\mathbf{0})]$, $\tilde{u}(t) = (u(t)^T, 1)^T$. We seek a projection matrix V , which is used to reduce the linear parametric system

$$\begin{aligned} E(\mu) \frac{d\mathbf{x}(t, \mu)}{dt} &= A_1(\mu)\mathbf{x}(t, \mu) + \tilde{B}(\mu)\tilde{u}(t), \\ \mathbf{y}(t, \mu) &= C(\mu)\mathbf{x}(t, \mu). \end{aligned} \quad (3.62)$$

Once V is computed from the linear system in (3.62), the ROM is then obtained by applying a Galerkin projection with V to the *quadratic* system in (3.63), i. e. the ROM of (3.59) is

$$\begin{aligned} V^T E(\mu) V \frac{d\mathbf{z}(t, \mu)}{dt} &= V^T A_1(\mu) V \mathbf{z}(t, \mu) + V^T A_2(\mu) V (\mathbf{z}(t, \mu) \otimes \mathbf{z}(t, \mu)) \\ &\quad + V^T \tilde{B}(\mu) \tilde{u}(t), \\ \mathbf{y}(t, \mu) &= C(\mu) V \mathbf{z}(t, \mu). \end{aligned} \quad (3.63)$$

Since V is computed from (3.62), the PMOR-MM method can be directly used to compute V . PMOR-MM is shown to be accurate for MOR of quadratic parametric systems [6, 26, 57], when the magnitude of the input signal is relatively small. The extension to two-sided (Petrov-Galerkin) projection is straightforward.

3.5.2 PMOR after bilinearization

Following the bilinearization method introduced in Section 3.4.2, the parametric nonlinear system in (3.59) can be approximated by a parametric bilinear system as

$$\begin{aligned}\frac{dx_{\otimes}}{dt} &= A_{\otimes}(\mu)x_{\otimes} + N_{\otimes}(\mu)x_{\otimes}u(t) + B_{\otimes}(\mu)u(t), \\ \mathbf{y}(t, \mu) &= C_{\otimes}(\mu)x_{\otimes}.\end{aligned}$$

By considering $u(t)$ associated with the bilinear term $N_{\otimes}(\mu)x_{\otimes}u(t)$ to be an extra parameter, say $u(t) = \mu_{m+1}(t)$, the above system can be viewed as a linear parametric system,

$$\begin{aligned}E_{\otimes}(\mu)\frac{dx_{\otimes}}{dt} &= A_{\otimes}(\mu)x_{\otimes} + N_{\otimes}(\mu)x_{\otimes}\mu_{m+1}(t) + B_{\otimes}(\mu)u(t), \\ \mathbf{y}(t, \mu) &= C_{\otimes}(\mu)x_{\otimes}.\end{aligned}\tag{3.64}$$

Note that the parameter $\mu_{m+1}(t)$ is time-varying. Strictly speaking, the PMOR-MM method in Section 3.3 cannot be directly used, since the Laplace transform of (3.64) cannot be applied as in (3.15). However, it is found that directly applying PMOR-MM to some systems with time-varying parameters [24, 41] or to the bilinear system [2] may also produce accurate results.

3.5.3 PMOR based on variational analysis

The variational analysis method in Section 3.4.3 can easily be extended to deal with parametric nonlinear systems. It can be seen that from the power series expansion of $\mathbf{f}(\mathbf{x}(t, \mu), \mu)$ in (3.60), one can obtain the parametric variational equations

$$\frac{d\mathbf{x}_1(t, \mu)}{dt} = A_1(\mu)\mathbf{x}_1(t, \mu) + B(\mu)u(t),\tag{3.65}$$

$$\frac{d\mathbf{x}_2(t, \mu)}{dt} = A_1(\mu)\mathbf{x}_2(t, \mu) + A_2(\mu)(\mathbf{x}_1(t, \mu) \otimes \mathbf{x}_1(t, \mu)),\tag{3.66}$$

$$\begin{aligned}\frac{d\mathbf{x}_3(t, \mu)}{dt} &= A_1(\mu)\mathbf{x}_3(t, \mu) + A_2(\mu)(\mathbf{x}_1(t, \mu) \otimes \mathbf{x}_2(t, \mu) + \mathbf{x}_2(t, \mu) \otimes \mathbf{x}_1(t, \mu)) \\ &\quad + A_3(\mu)(\mathbf{x}_1(t, \mu) \otimes \mathbf{x}_1(t, \mu) \otimes \mathbf{x}_1(t, \mu)),\end{aligned}\tag{3.67}$$

\vdots

For each linear parametric system in (3.65)–(3.67), the PMOR-MM method can be used to compute the projection matrices V_1, V_2, V_3 corresponding to (3.65)–(3.67), respectively. The final projection matrix V for the parametric nonlinear system is then the combination of V_1, V_2, V_3 , and can be computed following (3.52). The ROM is of a form

similar to (3.54):

$$\begin{aligned}\frac{d\mathbf{z}(t, \mu)}{dt} &= V^T A_1(\mu) V \mathbf{z}(t, \mu) + V^T A_2(\mu) (V \mathbf{z}(t, \mu) \otimes V \mathbf{z}(t, \mu)) \\ &\quad + V^T A_3(\mu) (V \mathbf{z}(t, \mu) \otimes V \mathbf{z}(t, \mu) \otimes V \mathbf{z}(t, \mu)) + V^T B(\mu) u(t), \\ \mathbf{y}(t, \mu) &= C(\mu) V \mathbf{z}(t, \mu).\end{aligned}$$

3.6 Conclusions

This chapter reviews moment-matching methods for MOR of a wide range of systems, including standard linear time-invariant (LTI) systems, parametric LTI systems, nonlinear non-parametric and nonlinear parametric systems. Sufficient algorithms are provided to enable most of the methods to be realizable and the results in the literature to be reproducible. Some algorithms, e. g., Algorithms 3.1–3.2 and Algorithm 3.8 have not appeared elsewhere. The discussions in some sections, e. g. Sections 3.3.2, 3.3.3, and 3.5 are also new. It has been demonstrated in numerous publications that moment-matching methods are powerful MOR tools for many systems and are useful in many application areas.

Bibliography

- [1] M. I. Ahmad, P. Benner, and P. Goyal. Krylov subspace-based model reduction for a class of bilinear descriptor systems. *J. Comput. Appl. Math.*, 315:303–318, 2017.
- [2] M. I. Ahmad, P. Benner, and L. Feng. Interpolatory model reduction for quadratic–bilinear systems using error estimators. *Eng. Comput.*, 36(1):25–44, 2018.
- [3] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. SIAM Publications, Philadelphia, PA, 2005.
- [4] Z. Bai. Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems. *Appl. Numer. Math.*, 43(1–2):9–44, 2002.
- [5] Z. Bai and D. Skoogh. A projection method for model reduction of bilinear dynamical systems. *Linear Algebra Appl.*, 415(2–3):406–425, 2006.
- [6] N. Banagaaya, P. Benner, and L. Feng. Parametric model order reduction for electro-thermal coupled problems with many inputs. In *Proceedings of European Consortium for Mathematics in Industry ECMI 2016: Progress in Industrial Mathematics at ECMI 2016*, volume 26, pages 263–270. Springer, 2018.
- [7] U. Baur, P. Benner, and L. Feng. Model order reduction for linear and nonlinear systems: a system-theoretic perspective. *Arch. Comput. Methods Eng.*, 21:331–358, 2014.
- [8] T. Bechtold, E. B. Rudnyi, and J. G. Korvink. Error indicators for fully automatic extraction of heat-transfer macromodels for MEMS. *J. Micromech. Microeng.*, 15(3):430–440, 2005.
- [9] P. Benner and T. Breiten. On H_2 -model reduction of linear parameter-varying systems. *Proc. Appl. Math. Mech.*, 11(1):805–806, 2011.
- [10] P. Benner and T. Breiten. Interpolation-based \mathcal{H}_2 -model reduction of bilinear control systems. *SIAM J. Matrix Anal. Appl.*, 33(3):859–885, 2012.

- [11] P. Benner and T. Breiten. Two-sided projection methods for nonlinear model order reduction. *SIAM J. Sci. Comput.*, 37(2):B239–B260, 2015.
- [12] P. Benner and P. Goyal. Multipoint interpolation of Volterra series and \mathcal{H}_2 -model reduction for a family of bilinear descriptor systems. *Syst. Control Lett.*, 97:1–11, 2016.
- [13] P. Benner, S. Gugercin, and K. Willcox. A survey of model reduction methods for parametric systems. *SIAM Rev.*, 57(4):483–531, 2015.
- [14] A. Bodendiek and M. Bollhöfer. Adaptive-order rational Arnoldi-type methods in computational electromagnetism. *BIT Numer. Math.*, 54:357–380, 2014.
- [15] B. Bond and L. Daniel. Parameterized model order reduction of nonlinear dynamical systems. In *Proc. IEEE/ACM Conference on Computer-Aided Design, ICCAD-2005, November*, pages 487–494, 2005.
- [16] C.-T. Chen. *Linear System Theory and Design*. Oxford University Press, New York/Oxford, 1999.
- [17] L. Daniel, O. C. Siong, L. S. Chay, K. H. Lee, and J. White. A multiparameter moment-matching model-reduction approach for generating geometrically parameterized interconnect performance models. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 23(5):678–693, 2004.
- [18] N. Dong and J. Roychowdhury. Piecewise polynomial nonlinear model reduction. In *Proc. Design Automation Conference*, pages 484–489, 2003.
- [19] N. Dong and J. Roychowdhury. Automated extraction of broadly applicable nonlinear analog macromodels from SPICE-level descriptions. In *Proc. IEEE Custom Integrated Circuits Conference, 2004*, pages 117–120, 2004.
- [20] N. Dong and J. Roychowdhury. Automated nonlinear macromodelling of output buffers for high-speed digital applications. In *Proc. Design Automation Conference*, pages 51–56, 2005.
- [21] P. Feldmann and R. W. Freund. Efficient linear circuit analysis by Padé approximation via the Lanczos process. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 14(5):639–649, 1995.
- [22] L. Feng, A. C. Antoulas, and P. Benner. Automatic generation of reduced order models for linear parametric systems. In *European Conference on Mathematics for Industry (ECMI). The European Consortium for Mathematics in Industry*, volume 22, 2016.
- [23] L. Feng and P. Benner. A note on projection techniques for model order reduction of bilinear systems. In *Numerical Analysis and Applied Mathematics: International Conference of Numerical Analysis and Applied Mathematics*, pages 208–211, 2007.
- [24] L. Feng and P. Benner. A robust algorithm for parametric model order reduction based on implicit moment matching. In *Reduced Order Methods for Modeling and Computational Reduction. MS&A Series*, volume 9, pages 159–186. Springer, Berlin, Heidelberg, New York, 2014.
- [25] L. Feng, J. G. Korvink, and P. Benner. A fully adaptive scheme for model order reduction based on moment-matching. *IEEE Trans. Compon. Packag. Manuf. Technol.*, 5(12):1872–1884, 2015.
- [26] L. Feng, Y. Yue, N. Banagaaya, P. Meuris, W. Schoenmaker, and P. Benner. Parametric modeling and model order reduction for (electro-)thermal analysis of nanoelectronic structures. *J. Math. Ind.*, 6(1):1–10, 2016.
- [27] L. Feng, X. Zeng, C. Chiang, D. Zhou, and Q. Fang. Direct nonlinear order reduction with variational analysis. In *Proc. Design Automation and Test in Europe*, pages 1316–1321, 2004.
- [28] L. Feng, A. C. Antoulas, and P. Benner. Some a posteriori error bounds for reduced order modelling of (non-)parametrized linear systems. *ESAIM: M2AN*, 51(6):2157–2158, 2017.
- [29] L. Feng and P. Benner. A robust algorithm for parametric model order reduction. In *Proceedings in Applied Mathematics and Mechanics (ICIAM)*, volume 7(1), pages 10215.01–10215.02, 2007.
- [30] L. Feng, P. Benner, P. Meuris, and W. Schoenmaker. Parametric and reduced-order modeling for the thermal analysis of nanoelectronic structures. In *Proceedings of SCEE-2014, Scientific Computing in Electrical Engineering*, pages 115–163. Springer, 2015.

- [31] R. W. Freund. Model reduction methods based on Krylov subspaces. *Acta Numer.*, 12:267–319, 2003.
- [32] K. Gallivan, E. Grimme, and P. Van Dooren. Asymptotic waveform evaluation via a Lanczos method. *Appl. Math. Lett.*, 7(5):75–80, 1994.
- [33] P. Goyal, M. I. Ahmad, and P. Benner. Model reduction of quadratic–bilinear descriptor systems via Carleman bilinearization. In *Proc. European Control Conf. 2015, Linz*, pages 1177–1182. IEEE, 2015.
- [34] P. Goyal and P. Benner. An iterative model order reduction scheme for a special class of bilinear descriptor systems appearing in constraint circuit simulation. In *ECCOMAS Congress 2016, VII European Congress on Computational Methods in Applied Sciences and Engineering*, volume 2, pages 4196–4212, 2016.
- [35] E. J. Grimme. Krylov projection methods for model reduction. PhD thesis, Univ. Illinois, Urbana-Champaign 1997.
- [36] C. Gu. QLMOR: A new projection-based approach for nonlinear model order reduction. In *Proc. International Conference on Computer-Aided Design*, November, pages 389–396, 2009.
- [37] C. Gu. QLMOR: A projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 30(9):1307–1320, 2011.
- [38] S. Gugercin, A. C. Antoulas, and C. Beattie. \mathcal{H}_2 model reduction for large-scale linear dynamical systems. *SIAM J. Matrix Anal. Appl.*, 30(2):609–638, 2008.
- [39] U. Hetmaniuk, R. Tezaur, and C. Farhat. An adaptive scheme for a class of interpolatory model reduction methods for frequency response problems. *Int. J. Numer. Methods Eng.*, 93:1109–1124, 2013.
- [40] H.-J. Lee, C.-Ch. Chu, and W.-Sh. Feng. An adaptive-order rational Arnoldi method for model-order reductions of linear time-invariant systems. *Linear Algebra Appl.*, 235–261, 2006.
- [41] L. Feng, D. Koziol, E. B. Rudnyi, and J. G. Korvink. Parametric model reduction for fast simulation of cyclic voltammograms. *Sens. Lett.*, 4:165–173, 2006.
- [42] P. Li and L. T. Pileggi. NORM: Compact model order reduction of weakly nonlinear systems. In *Proc. Design Automation Conference*, pages 472–477, 2003.
- [43] Y. Lin, L. Bao, and Y. Wei. A model-order reduction method based on Krylov subspace for MIMO bilinear dynamical systems. *J. Appl. Math. Comput.*, 25(1–2):293–304, 2007.
- [44] A. Manzoni and F. Negri. Heuristic strategies for the approximation of stability factors in quadratically nonlinear parametrized PDEs. *Adv. Comput. Math.*, 41(5):1255–1288, 2015.
- [45] A. Odabasioglu, M. Celik, and L. T. Pileggi. PRIMA: passive reduced-order interconnect macromodeling algorithm. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 17(8):645–654, 1998.
- [46] J. R. Phillips. Projection frameworks for model reduction of weakly nonlinear systems. In *Proc. Design Automation Conference*, pages 184–189, 2000.
- [47] J. R. Phillips. Projection-based approaches for model reduction of weakly nonlinear time-varying systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 22(2):171–187, 2003.
- [48] L. T. Pillage and R. A. Rohrer. Asymptotic waveform evaluation for timing analysis. *IEEE Trans. Comput.-Aided Des.*, 9(4):352–366, 1990.
- [49] M. J. Rewieński. A Trajectory Piecewise-Linear Approach to Model Order Reduction of Nonlinear Dynamical Systems. Ph.D. Thesis, Massachusetts Institute of Technology 2003.
- [50] M. J. Rewieński and J. White. A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 22(2):155–170, 2003.
- [51] M. J. Rewieński and J. White. Model order reduction for nonlinear dynamical systems based on trajectory piecewise-linear approximations. *Linear Algebra Appl.*, 415(2–3):426–454, 2006.

- [52] W. J. Rugh. *Nonlinear System Theory*. The Johns Hopkins University Press, Baltimore, 1981.
- [53] S. Tiwary and R. Rutenbar. Scalable trajectory methods for on-demand analog macromodel extraction. In *Proc. Design Automation Conference*, pages 403–408, 2005.
- [54] D. Vasilyev, M. Rewienski, and J. White. A TBR-based trajectory piecewise-linear algorithm for generating accurate low-order models for nonlinear analog circuits and MEMS. In *Proc. Design Automation Conference*, pages 490–495, 2003.
- [55] T. Voß, R. Pulch, E. J. W. Maten, and A. El Guennouni. Trajectory piecewise linear approach for nonlinear differential-algebraic equations in circuit simulation. In *Scientific Computing in Electrical Engineering SCEE 2006, Mathematics in Industry*, volume 11, pages 167–174, Springer, Berlin, Heidelberg, 2007.
- [56] T. Wolf, H. Panzer, and B. Lohmann. Gramian-based error bound in model reduction by Krylov subspace methods. In *Proceedings of IFAC World Congress*, pages 3587–3591, 2011.
- [57] Y. Yue, L. Feng, P. Meuris, W. Schoenmaker, and P. Benner. Application of Krylov-type parametric model order reduction in efficient uncertainty quantification of electro-thermal circuit models. In *Progress In Electromagnetics Research Symposium (PIERS 2015)*, pages 379–384, 2015.
- [58] Y. Zhang, H. Liu, Q. Wang, N. Fong, and N. Wong. Fast nonlinear model order reduction via associated transforms of high-order Volterra transfer functions. In *Proc. Design Automation Conference*, pages 289–294, 2012.

4 Modal methods for reduced order modeling

Abstract: In this chapter, we present an overview of the so-called *modal methods* for reduced order modeling. The naming is loosely referring to techniques that aim at constructing the reduced order basis for reduction without resorting to data, typically obtained by full order simulations. We focus primarily on linear and nonlinear mechanical systems stemming from a finite element discretization of the underlying strong form equations. The nonlinearity is of a geometric nature, i. e. due to redirection of internal stresses due to large displacements. Intrusive vs non-intrusive techniques (i. e. requiring or not access to the finite element formulation to construct the reduced order model) are discussed, and an overview of the most popular methods is presented.

Keywords: Galerkin projection, geometric nonlinearities, modal derivatives, non-intrusive reduction, nonlinear manifolds

MSC 2010: 74H15, 74H45

4.1 Introduction

Structural dynamics often relies on Finite Element (FE) models leading to large system of equations, which need to be integrated in time to predict time histories of displacements, strain and stresses. Even in the case of linear models, this task is often prohibitive in a design context, as the large size of the systems and the large number of simulations required result in excessive time and storage requirements. This situation is exacerbated by the presence of nonlinearities, which essentially implies the evaluation and factorization of configuration-dependent residuals and Jacobians, and an iterative process for convergence at each time step. Clearly, model order reduction is a must in these cases.

Historically, linear structural dynamics have been resorting to modal decomposition since its dawn. Essentially, the displacement field is approximated as a superposition of few eigenmodes of the system to achieve reduction.

When nonlinearities are present, a plethora of phenomena might occur, namely a richer harmonic spectrum in the response than that of the applied forcing, internal resonances, bifurcation, etc. The challenge for a nonlinear reduced order model (NL-ROM) is to efficiently capture all these phenomena. As the type of nonlinearity strongly affects the response, it would make sense to use all the knowledge of the FE model,

Paolo Tiso, Morteza Karamooz Mahdiabadi, ETH Zürich, Institute for Mechanical Systems, Zurich, Switzerland

Jacopo Marconi, Department of Mechanical Engineering, Politecnico Di Milano, Milan, Italy

Open Access. © 2021 Paolo Tiso et al., published by De Gruyter.  This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

usually referred to as the High Fidelity Model (HFM) to devise methods to construct the NLROM. This strategy, which we name here *model-driven* or *system-driven*, is in some sense opposed to a *data-driven* approach, which constructs the NLROM essentially from data coming from simulations of the HFM. This chapter attempts to give an overview to methods belonging to the former category, and focuses on structural systems featuring geometric nonlinearities.

Essentially, a NLROM technique tackling such systems faces two challenges. First, the derivation of a compact reduction basis on which to project the HFM. Second, an efficient calculation of the projected nonlinear terms. As we will discuss, the nonlinear terms of a FE based HFM are never available in an explicit form at system level, and therefore their reduction and projection face some computational bottlenecks that need to be addressed.

An often employed categorization of NLROMs distinguishes between *intrusive* and *non-intrusive* methods. Respectively, the construction of the NLROM requires, or not, the access to element level formulation, and FE assembly. Clearly, a non-intrusive technique is preferable when one has only the availability of a commercial FE program, which typically only release forces and, in some cases, Jacobians, at the assembled level. On the other hand, intrusive methods are typically more systematic and theoretically sound, at the price of the mentioned need to access codes at the element level.

In this contribution, we discuss the main methods and point out recent developments and trends. We first recap the main concepts applicable to linear systems, and then tackle the relevant cases of geometric nonlinearities.

4.2 Linear systems

Let us first consider the linear system resulting from a FE discretization of second-order ordinary differential equations describing an elastic continuum. This can be written

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{p}(t), \quad (4.1)$$

where $\mathbf{u} \in \mathbb{R}^n$ is the vector of nodal generalized displacements, $\mathbf{M} \in \mathbb{R}^{n \times n}$ is the mass matrix, $\mathbf{C} \in \mathbb{R}^{n \times n}$ is the damping matrix, \mathbf{K} is the stiffness matrix and $\mathbf{p}(t) \in \mathbb{R}^n$ is a forcing vector. The system (4.1) implies a linearization about an equilibrium point. For many structural dynamics applications, n is typically very large due to a required fine discretization.

A well established reduction technique in linear structural dynamics is based on the so-called *vibration modes*, i. e. eigenvectors associated to the free undamped vibration problem, obtained by setting $\mathbf{C} = \mathbf{0}$ and $\mathbf{p}(t) = \mathbf{0}$ in (4.1),

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{0}. \quad (4.2)$$

The solution of (4.2) reads

$$\mathbf{u}(t) = \boldsymbol{\phi} e^{i\omega t}. \quad (4.3)$$

When (4.3) is inserted in (4.2), we obtain the eigenvalue problem

$$(\mathbf{K} - \omega^2 \mathbf{M})\boldsymbol{\phi} = \mathbf{0}, \quad (4.4)$$

which admits eigenpairs ω_j , $\boldsymbol{\phi}_j$, $j = 1, \dots, n$ as solutions. Physically, each mode $\boldsymbol{\phi}_j$ represents the shape at which the system freely vibrates at a frequency ω_j . It can be easily shown that the vibration modes are orthogonal with respect to the mass and the stiffness matrices. Compactly, we can write

$$\boldsymbol{\phi}_j^T \mathbf{M} \boldsymbol{\phi}_k = \delta_{jk}, \quad \boldsymbol{\phi}_j^T \mathbf{K} \boldsymbol{\phi}_k = \omega_j^2 \delta_{jk}, \quad (4.5)$$

where $\delta_{jk} = 1$ for $j = k$, and zero otherwise. In the above, we also use a so called *mass normalization* for the vibration mode, i.e. a scaling $\boldsymbol{\phi}_j$ that guarantees that $\boldsymbol{\phi}_j^T \mathbf{M} \boldsymbol{\phi}_j = 1$, $j = 1, \dots, n$. Since the eigenmodes of the system constitute an orthonormal basis spanning \mathbb{R}^n , we can operate the following coordinate transformation, known as *Mode Superposition*:

$$\mathbf{u}(t) = \sum_{k=1}^n \boldsymbol{\phi}_k q_k(t) = \boldsymbol{\Phi} \mathbf{q}(t), \quad (4.6)$$

where $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_n]$ is a matrix containing all the eigenmodes and $\mathbf{q}(t) = [q_1(t), \dots, q_n(t)]^T$ is the vector of modal coordinates. By substituting (4.6) in (4.1) and projecting onto $\boldsymbol{\Phi}$, we obtain

$$\boldsymbol{\Phi}^T \mathbf{M} \boldsymbol{\Phi} \ddot{\mathbf{q}} + \boldsymbol{\Phi}^T \mathbf{C} \boldsymbol{\Phi} \dot{\mathbf{q}} + \boldsymbol{\Phi}^T \mathbf{K} \boldsymbol{\Phi} \mathbf{q} = \boldsymbol{\Phi}^T \mathbf{p}. \quad (4.7)$$

It is often the case, in structural dynamics applications, that the damping matrix \mathbf{C} is given as linear combination of \mathbf{K} and \mathbf{M} ,

$$\mathbf{C} = \alpha \mathbf{K} + \beta \mathbf{M}, \quad (4.8)$$

where α and β are user defined coefficients. This is done for two reasons. First, the actual damping is due to several mechanisms (joint friction, internal heating, interaction with surrounding fluid, etc.) which cannot be physically described by a simple viscous term as in (4.1), but rather by much more complex, nonlinear laws. The term $\mathbf{C}\dot{\mathbf{u}}(t)$ is then designed to provide a velocity dependent linear term in (4.1) by globally representing the dissipation occurring during motion. Second, the form (4.8), also known as *Rayleigh Damping*, allows one to make use of the orthogonality property (4.5), and yields a decoupled set of equations. By virtue of mode orthogonality and mass normalization of modes, (4.7) becomes

$$\ddot{\mathbf{q}} + \mathbf{D} \dot{\mathbf{q}} + \boldsymbol{\Omega}^2 \mathbf{q} = \boldsymbol{\Theta}, \quad (4.9)$$

where $\Theta = [\phi_1^T \mathbf{p}, \dots, \phi_n^T \mathbf{p}]^T \in \mathbb{R}^n$ is the *load participation vector*,

$$\Omega^2 = \begin{bmatrix} \omega_1^2 & & \\ & \ddots & \\ & & \omega_n^2 \end{bmatrix} \quad (4.10)$$

is a diagonal matrix containing the eigenvalues ω_j^2 , $j = 1, \dots, n$ along its diagonal, and

$$\mathbf{D} = \Phi^T \mathbf{C} \Phi = \Phi^T (\alpha \mathbf{K} + \beta \mathbf{M}) \Phi = \begin{bmatrix} 2\xi_1 \omega_1 & & \\ & \ddots & \\ & & 2\xi_n \omega_n \end{bmatrix}, \quad (4.11)$$

with the damping ratios ξ_j found as

$$\xi_j = \frac{1}{2} \left(\alpha \omega_j + \frac{\beta}{\omega_j} \right). \quad (4.12)$$

Equation (4.9) indicates that the response of a damped linear system to a given excitation can be computed from a set of n uncoupled single DOF equations. Each of these uncoupled equations can be separately integrated using for instance the Laplace transform and convolution products (see [18]). However, the computation of all eigenmodes for a large dynamical system with several DOFs is computationally expensive in practice. It is anyway possible to approximate the response of the system using only few modes in (4.6), as explained in the next section.

Modal displacement method

To reduce the computational costs for obtaining the response of a system using mode superposition, one could only include the modes in (4.6) which effectively participate in the response. This technique is known as *Modal Displacement Method* (MDM). If this strategy holds, not only just a few eigenmodes of the system have to be calculated, but also the number of uncoupled modal equations that need to be solved is significantly reduced with respect to the size n of the original problem (4.1).

Let us first decompose the applied external load to a spatial distribution vector and a time-varying function as

$$\mathbf{p} = \mathbf{P}_0 \mathbf{g}(t), \quad (4.13)$$

where $\mathbf{P}_0 \in \mathbb{R}^{n \times p}$ is a matrix containing p spatial distributions of the applied load and $\mathbf{g}(t) \in \mathbb{R}^p$ represents their time-varying functions. According to the MDM, the motion of the system (4.1) can be approximated as a superposition of truncated number of modes by

$$\mathbf{u}(t) \approx \sum_{k=1}^m \phi_k \tilde{q}_k(t) = \tilde{\Phi} \tilde{\mathbf{q}}(t), \quad (4.14)$$

where $\tilde{\Phi} \in \mathbb{R}^{n \times m}$ is a matrix containing the truncated set of $m \ll n$ kept eigenmodes of the system, and $\tilde{\mathbf{q}}(t)$ are the corresponding modal amplitudes. By substituting (4.14) in (4.1) and projecting onto $\tilde{\Phi}$, the reduced set of modal equations reads

$$\ddot{\tilde{\mathbf{q}}} + \tilde{\mathbf{D}}\dot{\tilde{\mathbf{q}}} + \tilde{\mathbf{\Omega}}^2\tilde{\mathbf{q}} = \tilde{\Phi}^T \mathbf{P}_0 \mathbf{g}(t) = \tilde{\mathbf{P}}_0 \mathbf{g}(t), \quad (4.15)$$

where $\tilde{\mathbf{D}} = \tilde{\Phi}^T \mathbf{D} \tilde{\Phi}$ and $\tilde{\mathbf{\Omega}}^2 = \tilde{\Phi}^T \mathbf{K} \tilde{\Phi}$. Once the reduced system in (4.15) is integrated, the approximate displacement $\tilde{\mathbf{u}}$ is recovered as

$$\tilde{\mathbf{u}}(t) = \sum_{k=1}^m \boldsymbol{\phi}_k \tilde{q}_k(t). \quad (4.16)$$

The MDM can significantly reduce the computational cost of time integration. Typically, the eigenmodes that are used for the reduction are those whose frequencies spans the frequency content of $\mathbf{g}(t)$. While representing the spectral content of $\mathbf{g}(t)$, one should also strive for an accurate projection of the spatial distribution of the load $\tilde{\mathbf{P}}_0$. If this is not guaranteed, a significant error can arise. To quantify this error, let us define the residual spatial distribution vector, \mathbf{P}_r , as the portion of the load distribution vector \mathbf{P}_0 , which is neglected after projecting it on the truncation modes basis. This residual can be obtained:

$$\mathbf{P}_r = \mathbf{P}_0 - \mathbf{P}_t, \quad (4.17)$$

where

$$\mathbf{P}_t = \mathbf{M} \tilde{\Phi} \tilde{\Phi}^T \mathbf{P}_0; \quad (4.18)$$

see [14] for details of the derivation. Note that, as m increases, $\tilde{\Phi} \tilde{\Phi}^T \rightarrow \Phi \Phi^T = \mathbf{M}^{-1}$ and therefore $\mathbf{P}_r \rightarrow \mathbf{0}$, but the required computational effort increases as well. In order to keep m low and compensate for the effect of modal truncation onto the load, two main methods are outlined here.

Mode acceleration correction

The MDM is based on the assumption that the reduction basis spans the whole frequency content of the applied load. Therefore, the modes that are left out would contribute to the solution in a quasi-static manner, i. e. without changing the velocity and the acceleration of the full response appreciably. The Mode Acceleration Correction (MAC) method computes the quasi-static effect of truncated modes and augments it to the displacement response obtained from reduced system using the Modal Displacement method. This is done as follows. From (4.1) we can write

$$\mathbf{K}\mathbf{u} = \mathbf{p} - \mathbf{M}\ddot{\mathbf{u}} - \mathbf{C}\dot{\mathbf{u}}. \quad (4.19)$$

Since the acceleration of the system is assumed not to be affected by modal reduction, we substitute (4.16) into (4.19) and solve it for the displacement, giving

$$\mathbf{u} \approx \bar{\mathbf{u}} = \mathbf{K}^{-1} \mathbf{p} - \sum_{k=1}^m \frac{\boldsymbol{\phi}_k \ddot{q}_k(t)}{\omega_k^2} - \sum_{k=1}^m \frac{2\xi_k \boldsymbol{\phi}_k \dot{q}_k(t)}{\omega_k}. \quad (4.20)$$

The acceleration of the k th generalized coordinate can be calculated from (4.9) as

$$\ddot{q}_k = \boldsymbol{\phi}_k^T \mathbf{p} - \omega_k^2 q_k - 2\xi_k \omega_k \dot{q}_k. \quad (4.21)$$

Equation (4.21) is then introduced to (4.20), and rearranged to introduce the final form of the displacement:

$$\bar{\mathbf{u}} = \sum_{k=1}^m \boldsymbol{\phi}_k q_k + \left(\mathbf{K}^{-1} - \sum_{k=1}^m \frac{\boldsymbol{\phi}_k \boldsymbol{\phi}_k^T}{\omega_k^2} \right) \mathbf{p}. \quad (4.22)$$

The last term in (4.22) represents a quasi-static correction of the displacement using MAC. In other words, MAC represents a statically complete alternative for the MDM by augmenting the static response of the deleted modes to the system. Note that the missing modes in the truncation are never computed; simply, their static contribution is added a posteriori.

Modal truncation augmentation

The inaccuracies of the applied load's spatial distribution, caused by mode displacement projection, can be also improved by adding few correction vectors to the reduction basis. This technique goes under the name of *Modal Truncation Augmentation* (MTA) [15]. Let us first compute the static response of the system due to the residual spatial forces, which is simply

$$\mathbf{K}\mathbf{X} = \mathbf{P}_r, \quad (4.23)$$

where \mathbf{X} is a matrix collecting Ritz vectors columnwise. This matrix is then employed to reduce the mass and stiffness matrices of the system as

$$\hat{\mathbf{M}} = \mathbf{X}^T \mathbf{M} \mathbf{X}, \quad \hat{\mathbf{K}} = \mathbf{X}^T \mathbf{K} \mathbf{X}. \quad (4.24)$$

Here, $\hat{\mathbf{M}}$ and $\hat{\mathbf{K}}$ are the reduced mass and stiffness matrices projected onto the space spanned by the columns of \mathbf{X} . In the next step, an eigenvalue problem of the form

$$(\hat{\mathbf{K}} - \omega_k^2 \hat{\mathbf{M}}) \mathbf{w}_k = \mathbf{0} \quad (4.25)$$

is solved to obtain the eigenvectors \mathbf{w}_k . Finally, the modal truncation augmentation vectors are found by projecting the Ritz vectors onto the space spanned by the eigenvectors calculated from (4.25). This is simply

$$\boldsymbol{\Phi}_{mt} = \mathbf{X} \mathbf{W}, \quad (4.26)$$

where $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_p]$ is a matrix containing the eigenvectors obtained from (4.25) in its columns. The final step is to append the modal truncation augmentation vectors to the reduction basis of truncated eigenmodes as

$$\mathbf{\Psi} = [\tilde{\mathbf{\Phi}} \quad \mathbf{\Phi}_{mt}], \quad (4.27)$$

where $\mathbf{\Psi}$ is the enrichment to the basis of mode displacement method. From here, the MTA procedure follows the MDM, namely, the matrix $\mathbf{\Psi}$ can be used instead of $\tilde{\mathbf{\Phi}}$ in (4.14) to reduce the size of the system to also take into account the quasi-static contribution of the truncated mode in the system.

Once the response of the generalized coordinates \mathbf{q} is obtained the physical displacement can be retrieved as

$$\hat{\mathbf{u}} \approx \mathbf{\Psi} \hat{\mathbf{q}}. \quad (4.28)$$

Likewise, the physical acceleration is

$$\ddot{\mathbf{u}} \approx \mathbf{\Psi} \ddot{\mathbf{q}}. \quad (4.29)$$

As can be seen from (4.29), the effect of truncated modes is also reflected on the acceleration of the system through the basis $\mathbf{\Psi}$. This is different from the MAC method, which attempts only to correct for the displacement of the mode displacement reduced system in a strictly static manner.

4.3 Substructuring and component mode synthesis

Before discussing nonlinear problems, it is worth to briefly mention here another practical strategy for model reduction, namely *substructuring*. A thorough discussion of this topic is beyond the scope of this chapter, and the interested reader should refer to [1] for a comprehensive overview. The reduction methods discussed in this contribution tackle the whole high fidelity model monolithically, meaning that only one reduction basis is constructed to reduce the full solution. It is possible, however, to decompose the system into *substructures*, which communicate with each other through common interfaces. Then each substructure could be reduced independently from the others, while preserving the interface DOFs to allow for assembly. This approach was originated by the seminal work of Hurty in [25] and then later developed by Craig and Bampton in [12].

The division of the system into substructures is often dictated by the practical reason of dealing with the detailed design and the analysis of single components independently. The assembled model will then feature common interface DOFs and few, reduced generalized coordinates relative to each substructure, thus achieving model order reduction. All techniques aiming at reducing the substructures are based on

what discussed in the previous section. However, significant differences will arise depending on how the interface compatibility across substructures is enforced, namely whether a *primal* or *dual* assembly is adopted. In the former, common interface degrees of freedom are eliminated, and therefore the interface forces are not present in the assembled system. In the latter case, interface forces appear as Lagrange's multipliers, and thus compatibility is not a priori satisfied. For primal assembly, the Hurty–Graig–Bampton method [12, 25] achieves reduction for each substructure with a basis formed by static modes obtained by applying unit displacements at the interface, and vibration modes relative to the substructure clamped at the interface. In the case of dual assembly, it is natural to resort to the method of McNeal [42] and Rubin [56], which construct a reduction basis using vibration modes of the *free* substructure complemented with so-called attachment modes obtained by applying unit forces at the interface DOFs.

In case of several substructures featuring interfaces with many DOFs, the resulting assembled reduced order model could still be of large size. Then a second reduction could be performed on the interface DOFs, which could be applied before assembly to each component independently, or on the global interface after assembly. A recent review of the available methods can be found in [39].

4.4 Geometrically nonlinear structural dynamics

When the displacements about the equilibrium point cannot be considered as small, nonlinear geometrical effects might arise and (4.1) loses its validity. In particular, internal stresses due to deformations are rotated with respect to the reference, undeformed configuration. Typical engineering applications prone to geometric nonlinearities are thin-walled structures, where low-frequency and bending-dominated modes cause in-plane stretching when the displacements are in the order of the thickness of the structure. Then the system (4.1) is modified as

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{f}(\mathbf{u}(t)) = \mathbf{p}(t), \quad (4.30)$$

where $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is the function of internal nodal forces. The nonlinearity here included can take various forms, depending on the kinematic and material model adopted. For instance, for linear elastic constitutive law and Green–Lagrange strain tensor, the resulting internal forces are linear, quadratic and cubic in \mathbf{u} [43]. The same holds for the approximate kinematic model due to von Karman, which captures the behavior of beams and plates undergoing out-of-plane deflections in the order of one thickness of the structure [54]. Other kinematic descriptions, i. e. obtained by means of the co-rotational formulation, lead to more complex forms of $\mathbf{f}(\mathbf{u})$ [10, 13].

4.4.1 Galerkin projection

In order to achieve a low order version of (4.30), the generalized displacement vector $\mathbf{u}(t)$ can be approximated by a linear combination of vectors, contained in the columns of the matrix \mathbf{V} , as

$$\mathbf{u}(t) \approx \mathbf{V}\mathbf{q}(t) \quad (4.31)$$

where $\mathbf{q}(t)$ are the generalized modal amplitudes, unknowns of the Nonlinear Reduced Order Model (NLROM). By substituting (4.31) in equation (4.30), we get

$$\mathbf{M}\mathbf{V}\ddot{\mathbf{q}}(t) + \mathbf{C}\mathbf{V}\dot{\mathbf{q}}(t) + \mathbf{f}(\mathbf{V}\mathbf{q}(t)) - \mathbf{p}(t) = \mathbf{r}(t), \quad (4.32)$$

where $\mathbf{r}(t)$ is the residual resulting from the approximation. We can enforce this residual to be orthogonal to the reduction subspace, by requiring that

$$\mathbf{V}^T \mathbf{r} = \mathbf{0}. \quad (4.33)$$

This implies

$$\tilde{\mathbf{M}}\ddot{\mathbf{q}}(t) + \tilde{\mathbf{C}}\dot{\mathbf{q}}(t) + \tilde{\mathbf{f}}(\mathbf{V}\mathbf{q}(t)) = \tilde{\mathbf{p}}(t), \quad (4.34)$$

where

$$\tilde{\mathbf{M}} = \mathbf{V}^T \mathbf{M} \mathbf{V}, \quad \tilde{\mathbf{C}} = \mathbf{V}^T \mathbf{C} \mathbf{V}, \quad \tilde{\mathbf{p}}(t) = \mathbf{V}^T \mathbf{p}(t) \quad (4.35)$$

and

$$\tilde{\mathbf{f}}(t) = \mathbf{V}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t)). \quad (4.36)$$

The above technique is known as Galerkin projection, according to which the subspaces on which the solution is sought and on which the residual is projected coincide. This preserves the symmetry—and thus stability—properties of the structural dynamics equations. In other cases, when symmetry is not featured in the full model, the right and left subspaces do not have to be equal. This latter procedure goes under the name of Petrov–Galerkin projection. Note that the reduced operators $\tilde{\mathbf{M}}$, $\tilde{\mathbf{C}}$ and $\tilde{\mathbf{p}}(t)$ can be pre-computed *offline* before time-integrating the ROM. The nonlinear reduced force $\tilde{\mathbf{f}}$, however, still requires evaluation over the whole mesh, and hinders any significant speed-up that may be achieved from the ROM. In short, an accurate and efficient reduction relies on

1. a good choice of \mathbf{V} , and
2. an efficient computation of $\tilde{\mathbf{f}}$.

There is a variety of methods that aim at constructing a proper basis \mathbf{V} . In general, \mathbf{V} should fulfill the following, in some sense related, features:

1. *Span the HFM solution subspace.* The choice of the reduction basis vectors is usually based on the expected spectrum of the response with respect to the spectrum of the excitation. While in the linear case, because of the superposition principle, this is an easily fulfilled requirement, nonlinear systems pose several challenges related to the raise of sub and super-harmonics in the response.
2. *Can be cheaply computed, in terms of computational time.*
3. *Can handle different cases.* It is for instance desirable to construct a NLRM able to provide accurate responses for different forcing amplitudes and frequencies, or different material and geometric parameters.

4.4.1.1 Evaluation of nonlinear terms

For FE applications, the nonlinear forces are evaluated as follows:

$$\tilde{\mathbf{f}}(\mathbf{q}(t)) = \sum_{e=1}^{n_e} \mathbf{V}_e^T \mathbf{f}_e(\mathbf{V}_e \mathbf{q}(t)) \quad (4.37)$$

where $\mathbf{f}_e(\mathbf{u}_e) \in \mathbb{R}^{N_e}$ is the contribution of the element e for the vector $\mathbf{f}(\mathbf{u})$ (N_e being the number of DOFs for the element e), \mathbf{V}_e is the restriction of \mathbf{V} to the rows indexed by the DOFs corresponding to element e , and n_e is the total number of elements of the mesh. Since the reduced nonlinear term $\tilde{\mathbf{f}}(\mathbf{q}(t))$ is evaluated in the space of full variables, the computational cost associated to its evaluation does not scale with m alone. Indeed, (4.37) shows that this cost scales linearly with the number of elements in the structure, and can hence be high for large systems. Thus, despite the reduction in dimensionality achieved in (4.34), the evaluation of $\tilde{\mathbf{f}}(\mathbf{q}(t))$ hinders any fast prediction of system response using the NLRM. Different strategies are available to overcome this problem.

4.4.1.2 Exact reduced forces

In the case of polynomial nonlinearities, as the one arising from von Karman kinematic model of linear elastic bodies, or solid continuum FE described in a total Lagrangian framework, the i th component of the HFM nonlinear force $\mathbf{f}(\mathbf{u})$ can be written as

$$f_i = K_{ij}u_j + B_{ijk}u_ju_k + C_{ijkl}u_ju_ku_l, \quad i, j, k, l = 1, \dots, n, \quad (4.38)$$

where K_{ij} , B_{ijk} and C_{ijkl} are constant tensors of second, third and fourth order, respectively. Note that the form (4.38) is never available as written, i. e. the tensors B_{ijk} and

C_{ijkl} are typically never assembled within a FE model, while K_{ij} is the linear stiffness matrix. When a reduction $\mathbf{u} \approx \mathbf{V}\mathbf{q}$ is inserted in (4.38) and a projection on \mathbf{V} is performed, then \tilde{f}_i can be expressed *exactly* by

$$\tilde{f}_i = \tilde{K}_{ij}u_j + \tilde{B}_{ijk}q_jq_k + \tilde{C}_{ijkl}q_jq_kq_l, \quad i, j, k, l = 1, \dots, m. \quad (4.39)$$

Once (4.39) is available, no connection to the HFM is needed any longer. For details of the computation of the reduced tensors \tilde{K}_{ij} , \tilde{B}_{ijk} and \tilde{C}_{ijkl} see for instance [66].

4.4.1.3 Approximate reduced forces

When the nonlinearity is not polynomial, for instance because of a nonlinear material law, or geometric nonlinearities described by a co-rotational formulation [13], one is facing essentially two possibilities:

1. assuming a certain form of $\tilde{\mathbf{f}}(\mathbf{q})$ and identifying its coefficients, or
 2. trying to compute $\tilde{\mathbf{f}}(\mathbf{q})$ affordably, but still querying the underlying HFM mesh.
- This strategy is addressed by the so-called *hyper-reduction* methods, which are discussed, at least to some extent, in Section 4.4.1.4.

For option 1., in the case of NLROMs addressing geometrically nonlinear problems, a typical choice is

$$\tilde{f}_i = \tilde{K}_{ij}^{(1)}q_j + \tilde{K}_{ijk}^{(2)}q_jq_k + \tilde{K}_{ijkl}^{(3)}q_jq_kq_l, \quad i, j, k, l = 1, 2, \dots, m, \quad (4.40)$$

where $\tilde{K}_{ij}^{(1)}$, $\tilde{K}_{ijk}^{(2)}$ and $\tilde{K}_{ijkl}^{(3)}$ are two, three and four dimensional matrices, respectively. Note the formal equivalence between (4.40) and (4.39). However, (4.40) is simply a pre-defined, convenient model of something which is essentially unknown. We will discuss in the following that this form is typically used in non-intrusive methods, in which the access to element information (which would enable (4.39)) is not possible. The problem then shifts to designing an efficient identification technique for the coefficients $\tilde{K}_{ijk}^{(2)}$ and $\tilde{K}_{ijkl}^{(3)}$, as the reduced stiffness matrix $\tilde{K}_{ij}^{(1)}$ is typically available.

4.4.1.4 Hyper-reduction

In the cases described before, the nonlinear term is written directly with respect to the modal reduced coordinates, in order to avoid any element level function evaluations during the numerical time integration of the NLROM. Another option is possible, which aims at scaling the computation of the nonlinear reduced terms with the size of the reduced coordinate vector m , rather than with n (size of the HFM). This is achievable only if the nonlinear terms are evaluated in a sparse manner, and if the missing

contributions are in some sense compensated for. Methods pursuing this strategy go under the name of *hyper-reduction*, meaning with this that they introduce a further reduction upon the one arising from projection. Among the various proposals, two methods (and numerous variants) are gaining popularity, namely the *Discrete Empirical Interpolation Method* (DEIM) [11] and the *Energy Conserving Sampling and Weighting* (ECSW) [16]. These two methods are briefly outlined now.

ECSW

In ECSW, the nonlinear projected terms are approximated as

$$\tilde{\mathbf{f}}(t) = \sum_{e=1}^{n_e} \mathbf{V}_e^T \mathbf{f}_e(\mathbf{V}_e \mathbf{q}(t)) \approx \sum_{e \in E} \xi_e \mathbf{V}_e^T \mathbf{f}_e(\mathbf{V}_e \mathbf{q}(t)), \quad (4.41)$$

where $\xi_e \in \mathbb{R}^+$ are positive weights, and $|E| < n_e$. In other words, (4.41) weights the element forces of the element set E in order to match the work done by the internal forces onto the displacements induced by the reduction basis \mathbf{V} . This procedure shows striking similarities with Gauss quadrature, where a defined integral is approximated by the evaluation of the integrand function at specific points, and weighted with pre-defined coefficients. In ECSW, the weights are determined by matching the work done by the nonlinear forces onto the reduction basis for a set of n_t sampled training forces. This translates into the minimization problem

$$\boldsymbol{\xi} : \arg \min_{\boldsymbol{\xi} \in \mathbb{R}^{n_e}, \boldsymbol{\xi} \geq 0} \|\mathbf{G}\tilde{\boldsymbol{\xi}} - \mathbf{b}\|_2, \quad (4.42)$$

where

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_{11} & \cdots & \mathbf{g}_{1n_e} \\ \vdots & \ddots & \vdots \\ \mathbf{g}_{n_t 1} & \cdots & \mathbf{g}_{n_t n_e} \end{bmatrix} \quad (4.43)$$

and

$$\mathbf{g}_{ie} = \mathbf{V}_e^T \mathbf{f}_e(\mathbf{V}_e \mathbf{q}^{(i)}), \quad \mathbf{b}_i = \tilde{\mathbf{f}}_i(\mathbf{q}^{(i)}) = \sum_{e=1}^{n_e} \mathbf{g}_{ie}, \quad (4.44)$$

where $i = 1, \dots, n_t$, $e = 1, \dots, n_e$. A sparse solution to (4.42) returns a sparse set of elements $E : \{e : \xi_e > 0\}$. An optimally sparse solution to (4.42) can be obtained by using a greedy-approach-based algorithm [50]. For the sake of compactness, this algorithm is not reported here. As an illustrative example, Figure 4.1 shows the ECSW hyper-reduction of the dynamic response of a slightly curved panel, subjected to a bi-harmonic loading (for details, refer to [29]). The ECSW selects only 19 elements out of a mesh of 400, thus decreasing the evaluation cost of the nonlinear terms significantly. The resulting speed-up (ratio between the computing time of the full and reduced solution) is about 50 in this example.

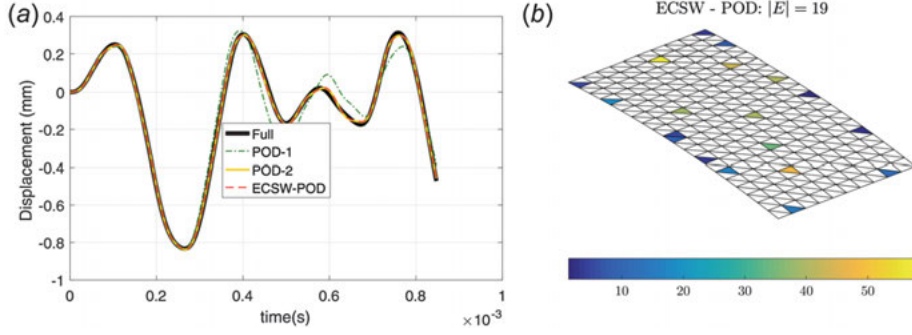


Figure 4.1: (Hyper-)reduction on a simply supported slightly curved panel subject to a time-varying load; see [29] for the details. On the left, the full response (thick black line) is compared to two Proper Orthogonal Decomposition (POD) reductions with 5 and 4 modes in the reduction basis, respectively for POD-1 and POD-2, and to the ECSW response. On the right, corresponding elements and their weights selected by the ECSW. In this case, 19 elements (out of 400) are picked. For a brief discussion of the POD method, see Section 4.5.4.

DEIM

The idea underlying the DEIM is somewhat different. Here, the nonlinear force \mathbf{f} is approximated as a superposition of few force modes collected into the matrix $\mathbf{U} \in \mathbb{R}^{n \times m}$ as

$$\mathbf{f} \approx \mathbf{U}\mathbf{c}, \quad (4.45)$$

where $\mathbf{c} \in \mathbb{R}^m$ are unknown factors. Then a Boolean matrix \mathbf{P} selects m rows of (4.45), so that the factors \mathbf{c} can be found as

$$\mathbf{P}^T \mathbf{f} = \mathbf{P}^T \mathbf{U}\mathbf{c} \Rightarrow \mathbf{c} = (\mathbf{P}^T \mathbf{U})^{-1} \mathbf{P}^T \mathbf{f}. \quad (4.46)$$

The force vector \mathbf{f} can be then approximated as

$$\mathbf{f} \approx \mathbf{U}(\mathbf{P}^T \mathbf{U})^{-1} \mathbf{P}^T \mathbf{f}, \quad (4.47)$$

and therefore the reduced forces are

$$\tilde{\mathbf{f}} = \mathbf{V}^T \mathbf{U}(\mathbf{P}^T \mathbf{U})^{-1} (\mathbf{P}^T \mathbf{f}). \quad (4.48)$$

Note that, in the above, the term $\mathbf{P}^T \mathbf{f}$ is grouped within brackets, to highlight that only the components of \mathbf{f} picked by \mathbf{P} have to be computed. Moreover, the matrix $\mathbf{V}^T \mathbf{U}(\mathbf{P}^T \mathbf{U})^{-1}$ can be pre-computed offline once for all. The issue then shifts to finding \mathbf{U} and \mathbf{P} . Typically, \mathbf{U} is formed by the left vectors of a Single Value Decomposition (SVD) of sampled training forces, i. e.

$$\mathbf{U}\Sigma\mathbf{W}^T = \mathbf{F}, \quad (4.49)$$

where $\mathbf{F} = [\mathbf{f}_1, \dots, \mathbf{f}_{n_t}]$ contains columnwise all the sampled n_t forces, while \mathbf{P} is selected via a greedy procedure; see [11] for details.

While ECSW and DEIM aim both at reducing the evaluation cost of the nonlinear terms, they differ significantly. ECSW operates directly on elements, while DEIM picks DOFs. This feature could lead to a decrease of efficiency for DEIM, as the computation of a specific component of \mathbf{f} requires the evaluation of all the elements connected to such a DOF. A workaround has been proposed in [65], where DEIM is applied on the unassembled mesh. In this manner, each DOF picked by the algorithm maps to one element only. To counteract the increased cost of the SVD of training forces (now unassembled), it is shown that a surrogate quantity per element is sufficient to reproduce the nonlinear contribution. This variant goes under the name of *Surrogate Unassembled* DEIM (SU-DEIM). Note that both methods rely on training data from the HFM. As such, the hyper-reduced NLROM is optimal with respect to the training set used for its generation. At the same time, there is no guaranty that the same NLROM provides an accurate response for a set of parameter values (i. e. load, material properties, etc.) far from those related to the training set. For this reason, hyper-reduction is typically employed to generate NLROM which are local with respect to parameter values. Then there exist techniques that interpolate such NLROM over the domain of the parameters. This topic is the subject of other contributions in this book. In any case, we outline the most common strategies in Section 4.6.

4.5 Reduction methods for geometrically nonlinear systems

As briefly discussed above, geometric nonlinearities induce stretching for bending and twisting deformations. Think of the beam, pinned at both ends (i. e. the extremities of the beam cannot move, both axially and transverse) depicted in Figure 4.2. If it is bent by a pressure load, the bending will cause axial stretching. This effect is not present when the displacements are assumed infinitesimal. As such, a good reduction basis should also include vectors able to produce such effects in the reduced solution. This fact is exemplified when considering the discretized equation of a flat structure, made of linear elastic and isotropic material, as discussed in the next section.

4.5.1 Static condensation

For thin-walled structures excited in the nonlinear range, one can argue that the dominant behavior is still represented by the low-frequency dynamics spanned by bending/twisting vibration modes. Also, the eigenfrequency associated to axially dominated modes is typically of much higher frequency, as the axial to bending stiffness

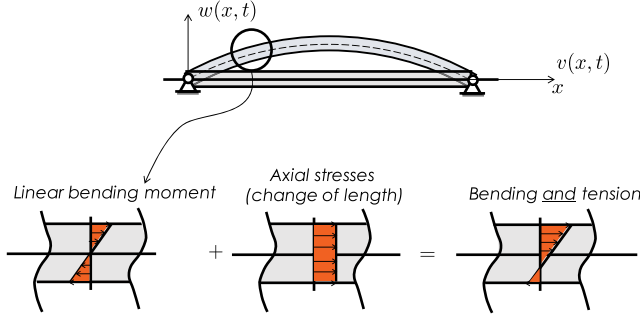


Figure 4.2: Illustration of the bending–stretching coupling due to geometric nonlinearity for a planar beam. An out-of-plane displacement $w(x, t)$ causes the beam to stretch, and induce an in-plane displacement field $v(x, t)$.

ratio scales with h^{-2} , h being the thickness of the component. As such, axial modes are likely not to be excited dynamically, but rather have a nonlinear contribution to the overall displacement. In many cases indeed, it is possible to identify *slow* and *fast* DOFs and have an effective reduction of the problem size. As an example, consider here the discretized FE equations for an Euler–Bernoulli beam modeled with von Karman kinematics. The equations of motion read

$$\begin{bmatrix} \mathbf{M}_{ww} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{vv} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{w}} \\ \ddot{\mathbf{v}} \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{ww} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{vv} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix} + \begin{bmatrix} {}_2\mathbf{f}_w(\mathbf{w}, \mathbf{v}) \\ {}_2\mathbf{f}_v(\mathbf{w}, \mathbf{w}) \end{bmatrix} + \begin{bmatrix} {}_3\mathbf{f}_w(\mathbf{w}, \mathbf{w}, \mathbf{w}) \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_w \\ \mathbf{0} \end{bmatrix}, \quad (4.50)$$

where \mathbf{v} and \mathbf{w} denote the in-planes (axial) and the out-of-plane (bending) displacements. The order of the nonlinear forces ${}_2\mathbf{f}_w$, ${}_2\mathbf{f}_v$ and ${}_3\mathbf{f}_w$ is denoted by the left subscript. As is well known, the in-plane and out-of-plane dynamics is not coupled through the linear operators ($\mathbf{M}_{uw} = \mathbf{M}_{wu}^T = \mathbf{0}$, $\mathbf{K}_{vw} = \mathbf{K}_{wv}^T = \mathbf{0}$). Moreover, for many structural applications, the load is applied only in the transverse direction, i. e. $\mathbf{p}_v = \mathbf{0}$. For slender structures like the beam under consideration, the in-plane dynamics is characterized by much higher frequencies than the bending dynamics. This allows one to neglect the inertial term of the in-plane block, and define \mathbf{v} as a function of \mathbf{w} as

$$\ddot{\mathbf{v}} \approx \mathbf{0} \Rightarrow \mathbf{v} = -\mathbf{K}_{vv}^{-1} {}_2\mathbf{f}_v(\mathbf{w}, \mathbf{w}). \quad (4.51)$$

In other words, the in-plane DOFs \mathbf{v} are a quadratic function of the out-of-plane generalized DOFs \mathbf{w} . (4.51) is usually referred to as *static condensation*. When (4.51) is inserted into (4.50) one obtains

$$\mathbf{M}_{ww} \ddot{\mathbf{w}} + \mathbf{K}_{ww} \mathbf{w} + {}_2\mathbf{f}_w(\mathbf{w}, \mathbf{K}_{vv}^{-1} {}_2\mathbf{f}_v(\mathbf{w}, \mathbf{w})) + {}_3\mathbf{f}_w(\mathbf{w}, \mathbf{w}, \mathbf{w}) = \mathbf{p}_w. \quad (4.52)$$

Note that the static condensation of \mathbf{v} goes into forming a third-order term which “corrects” the cubic stiffness term ${}_3\mathbf{f}_w$. This physically corresponds to allow the beam

to properly compensate for the correct in-plane displacements arising from finite displacements. Clearly, this simple procedure is only effective when one could identify the slow and fast DOFs from the full order model—something possible only in the case of simple geometries [30]. For more complex thin-walled structural components, as curved stiffened panels, the dynamics is still characterized by some low-frequency modes, which statically trigger geometrically nonlinear coupling effects. In these cases, however, it is in general not possible to detect slow and fast DOFs directly from the governing discretized equation. More general methods to tackle these cases are discussed next.

4.5.2 Modal derivatives

Note that equation (4.51) implies a nonlinear (in this case quadratic) constraint between master (i. e. kept) coordinates \mathbf{w} and slave (i. e. condensed) coordinates \mathbf{v} . On the contrary, what proposed in Section 4.4.1 relies on a linear combination of modes over the entire set of DOFs, without partitioning them into slow and fast, which can be difficult for complex geometries. As already discussed, the reduction basis \mathbf{V} should span the solution accurately. For the cases we are tackling, this would mean to properly represent the stretching induced by the out-of-plane dominated modes. One could then think of introducing in \mathbf{V} high-frequency axial modes to enrich the subspace. While straightforward, this strategy presents two issues, namely: (i) ideally, all modes of the system need to be computed, this is computationally very intensive for large systems, (ii) there is no criterion to select the proper vibration modes to reproduce the nonlinear coupling effectively. Typically, a reduction basis of large size results from this approach; see for instance [19].

One alternative way to achieve reduction and properly account for nonlinear stretching effects is through the use of *Modal Derivatives* (MDs) in the reduction basis.

MDs were originally proposed in [26] and [27], and more recently in [67]. Essentially, the MDs are modes shapes stemming from a pre-selected basis of vibration modes, assumed to accurately represent the dynamics of the underlying linearized system. The MDs reproduce the most important deformation shapes resulting from finite deflections in the direction of the dominant vibration modes. As such, they complement the set of vibration modes initially selected in the MDM. At first, MDs are computed by differentiating the eigenvalue problem (4.4) with respect to modal amplitudes, as

$$(\mathbf{K} - \omega^2 \mathbf{M}) \frac{\partial \boldsymbol{\phi}_i}{\partial q_j} + \left(\frac{\partial \mathbf{K}}{\partial q_j} - \frac{\partial \omega^2}{\partial q_j} \mathbf{M} - \frac{\partial \mathbf{M}}{\partial q_j} \omega^2 \right) \boldsymbol{\phi}_i = \mathbf{0}, \quad (4.53)$$

where $\boldsymbol{\theta}_{ij} = \frac{\partial \boldsymbol{\phi}_i}{\partial q_j}$ is the “MD of vibration mode $\boldsymbol{\phi}_i$ with respect to mode $\boldsymbol{\phi}_j$ ”. The above definition requires the computation of the derivative of the eigenfrequency. Moreover,

the system (4.53) is singular and requires special treatment to be solved; see [31] for a detailed discussion. Regardless of the method adopted to solve (4.53), a high dimensional matrix needs to be factorized for each MD, and this leads to high computational cost. A computationally cheaper and more theoretically sound definition of MDs resort to a static version of (4.53), where all the mass related contributions are ignored; see [67] and [69] for details. When the inertial terms are ignored, (4.53) becomes

$$\mathbf{K} \frac{\partial \boldsymbol{\phi}_i}{\partial q_j} = - \frac{\partial \mathbf{K}}{\partial q_j} \boldsymbol{\phi}_i. \quad (4.54)$$

Note that the coefficient matrix \mathbf{K} is already available and therefore each MDs requires only a new right-hand side. MDs computed according to (4.54) are usually referred to as *static* MDs (SMDs). It can be proven that SMDs are symmetric, i. e.

$$\frac{\partial \boldsymbol{\phi}_i}{\partial q_j} = \frac{\partial \boldsymbol{\phi}_j}{\partial q_i}. \quad (4.55)$$

The first five vibration modes and the corresponding MDs for a flat rectangular plate are shown in Figures 4.3 and 4.4, taken from [31]. Note that, while the vibration modes (associated with the lowest eigenfrequencies) feature out-of-plane displacements only, the MDs are in-plane, and represent the displacement field that one has to add to a given vibration mode to account for the geometric nonlinearity. This separation is of course due to the flatness of the system. For more complex systems for which it is not possible to distinguish between in-plane and out-of-plane DOFs, however, the interpretation of MDs still holds. For instance, one could look at [62] for an application of MDs to the case of a shallow arch structure.

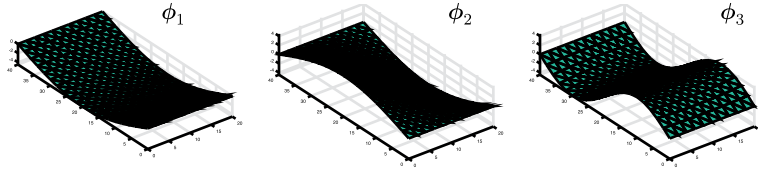


Figure 4.3: First three vibration modes of a rectangular plate, simply supported at its short edges [31]. Note that all modes feature out-of-plane displacement components only.

Once a set of m dominant vibration modes is selected, and the corresponding MDs are computed, the reduction basis \mathbf{V} can be formed as

$$\mathbf{V} = [\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_i, \dots, \boldsymbol{\phi}_m, \boldsymbol{\theta}_{11}, \dots, \boldsymbol{\theta}_{ij}, \dots, \boldsymbol{\theta}_{mm}], \quad (4.56)$$

with $i = 1, \dots, m, j = i, \dots, m$. Due to the symmetry property (4.55), the number of MDs that can be generated is $m(m + 1)/2$. Therefore, the size of the reduction basis grows

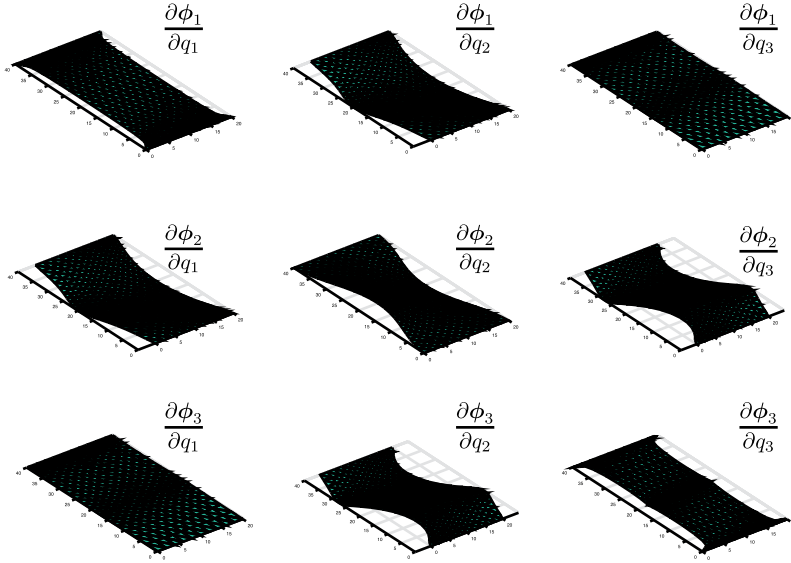


Figure 4.4: Modal derivatives corresponding to the first five vibration modes depicted in Figure 4.3 [31]. Due to the flatness of the structure, the MDs feature in-plane displacements only, and bring the displacement field necessary to account for the geometric nonlinearity.

with m^2 . However, there are heuristic criteria to select the most relevant MDs and reduce the associated NLROM size significantly; see [64] and [31]. Note also that the computation of the MDs requires the Hessian of the stiffness matrix in the direction of the vibration modes. In case the element formulation is accessible, this can be computed at element level analytically and then assembled. This would then classify the MDs method as *intrusive*. However, it is also possible to compute MD through a finite difference procedure (and therefore *non-intrusively*) only by using FE assembly level information typically made available in commercial codes, as outlined in [61]. As a note, MDs have been successfully used for flexible multibody applications, where nonlinear elastic deflections are superimposed to large rigid body motions [68, 69].

4.5.3 Dual modes

Another way to account for the transverse-membrane coupling is by appending the so called *Dual Modes* (DMs) to the dominant linear vibration modes in the reduction basis [38]. The main idea of DMs is based on applying a series of nonlinear static forces to the HFM to obtain related nonlinear displacements. These displacements are then orthogonalized with respect to the linear vibration modes, to yield shapes that represent the nonlinear contribution. The most common procedure for computing the DMs is described in [47, 52] and it is here briefly outlined.

We start from a set of mass normalized vibration modes $\mathbf{V}_{\text{VM}} = [\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_m]$, $m \ll n$, typically selected by the MDM (see Section 4.2). The procedure then selects a *dominant* vibration mode $\boldsymbol{\phi}_d$, where $d \in \mathcal{M}$ is the index relative to the mode with the maximum load participation factor, i. e.

$$d = \operatorname{argmax}_{i \in \mathcal{M}} (|\mathbf{p}^T \boldsymbol{\phi}_i|), \quad (4.57)$$

where \mathbf{p} is the external force in (4.30). Then we form a set of scaling factors $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_r]$, $\boldsymbol{\alpha} \in \mathbb{R}^r$ such that the static solutions \mathbf{x}_i of

$$\mathbf{f}(\mathbf{x}_i) - \alpha_i \mathbf{K} \boldsymbol{\phi}_d = \mathbf{0} \quad (4.58)$$

trigger displacements ranging from linear to strongly nonlinear. That is to say, $\mathcal{O}(\|\mathbf{f}(\mathbf{x}_i) - \mathbf{K} \mathbf{x}_i\|) \ll \mathcal{O}(\|\mathbf{K} \mathbf{x}_i\|)$ for small $|\alpha_i|$ and $\mathcal{O}(\|\mathbf{f}(\mathbf{x}_i) - \mathbf{K} \mathbf{x}_i\|) \approx \mathcal{O}(\|\mathbf{K} \mathbf{x}_i\|)$ for large $|\alpha_i|$, respectively. Let $\mathcal{R} = [1, \dots, r]$ be the set of indices associated to $\boldsymbol{\alpha}$. It is now possible to create load sets,

$$\mathbf{G}^{(j)} = \operatorname{diag}(\boldsymbol{\alpha}) \mathbf{K} \left(\frac{\boldsymbol{\phi}_d + \boldsymbol{\phi}_j}{2} \right), \quad j \in \mathcal{M}, \quad (4.59)$$

and let $\mathbf{g}_i^{(j)}$ be the i th column of $\mathbf{G}^{(j)}$. Now, for each $j \in \mathcal{M}$:

1. Solve the static problems

$$\mathbf{f}(\mathbf{x}_i^{(j)}) - \mathbf{g}_i^{(j)} = \mathbf{0}, \quad i \in \mathcal{R}, \quad (4.60)$$

and collect the obtained solutions in the matrix $\mathbf{X}^{(j)}$ as

$$\mathbf{X}^{(j)} = [\mathbf{x}_1^{(j)}, \dots, \mathbf{x}_r^{(j)}]. \quad (4.61)$$

2. Mass-orthogonalize each column of $\mathbf{X}^{(j)}$ to \mathbf{V}_{VM} , i. e.

$$\mathbf{x}_{i_\perp}^{(j)} = \mathbf{x}_i^{(j)} - \sum_{k=1}^m (\boldsymbol{\phi}_k^T \mathbf{M} \mathbf{x}_i^{(j)}) \boldsymbol{\phi}_k, \quad i \in \mathcal{R}, \quad (4.62)$$

and collect them into $\mathbf{X}_\perp^{(j)}$ as

$$\mathbf{X}_\perp^{(j)} = [\mathbf{x}_{1_\perp}^{(j)}, \dots, \mathbf{x}_{r_\perp}^{(j)}]. \quad (4.63)$$

This step is performed to exclude the linear contribution from the static solutions $\mathbf{x}_i^{(j)}$.

3. Perform an SVD of $\mathbf{X}_\perp^{(j)}$, and retain the singular vectors associated to the lowest $k < r$ singular values, as

$$\bar{\boldsymbol{\Psi}}^{(j)} \mathbf{S}^{(j)} \mathbf{R}^{(j)} = \operatorname{SVD}(\mathbf{X}_\perp^{(j)}), \quad (4.64)$$

$$\bar{\boldsymbol{\Psi}}^{(j)} = [\hat{\boldsymbol{\psi}}_1^{(j)}, \dots, \hat{\boldsymbol{\psi}}_k^{(j)}] \mid \sigma_1 > \sigma_2 > \dots > \sigma_k > \sigma_{k+1} > \dots > \sigma_n, \quad (4.65)$$

where σ_i is the i th diagonal component of $\mathbf{S}^{(j)}$.

4. Compute the linearized strain energy measures $E_i^{(j)}$ of $\mathbf{x}_i^{(j)}$ associated to each column of $\bar{\Psi}^{(j)}$ (i. e. to each dual mode candidate), as

$$E_i^{(j)} = \bar{\psi}_i^{(j)T} \mathbf{K} \bar{\psi}_i^{(j)} \sum_{s=1}^r \beta_s^{(j)2}, \quad \beta_s^{(j)} = \frac{\bar{\psi}_i^{(j)T} \mathbf{x}_s^{(j)}}{\bar{\psi}_i^{(j)T} \bar{\psi}_i^{(j)}}, \quad (4.66)$$

which are collected in a vector $\mathbf{E}^{(j)} = [E_1^{(j)}, \dots, E_r^{(j)}]$.

5. Select the most relevant $s < k$ DMs $\psi_{p_l}^{(j)}$, $l = 1, \dots, s$ as the columns of $\bar{\Psi}^{(j)}$ associated to the largest entries of $\mathbf{E}^{(j)}$, i. e.

$$E_{p_1}^{(j)} > E_{p_2}^{(j)} > \dots > E_{p_s}^{(j)}, \quad (4.67)$$

and let $\mathbf{V}_{\text{DM}}^{(j)} = [\psi_{p_1}^{(j)}, \dots, \psi_{p_s}^{(j)}]$.

6. Move to the next load set: $j \leftarrow j + 1$.

The total collection of selected DMs is then $\mathbf{V}_{\text{DM}} = [\mathbf{V}_{\text{DM}}^{(1)} \dots \mathbf{V}_{\text{DM}}^{(j)} \dots \mathbf{V}_{\text{DM}}^{(m)}]$. The reduction basis \mathbf{V} is formed as

$$\mathbf{V} = [\mathbf{V}_{\text{VM}} \quad \mathbf{V}_{\text{DM}}]. \quad (4.68)$$

Note that the computation of the DMs is intrinsically *non-intrusive*, as only nonlinear static solutions of the HFM are required. Also, the procedure just described is *load-dependent*, meaning that the information of the shape of the external load exerted is needed. Typically, the computation of DMs requires a careful selection of the scaling factors α to trigger the nonlinearity at the desired value.

4.5.4 Proper orthogonal decomposition

As alternative to model-based techniques, a reduction basis for Galerkin projection can be from data obtained by sampling HFM solutions. The most popular of such method is the *Proper Orthogonal Decomposition* (POD) [2, 35, 36, 41], also known as Karhunen–Loeve decomposition or principal component analysis (PCA). This topic is extensively treated in [8, Chapter 2]. Here we just discuss the main idea.

Let us assume to have collected n_t time snapshots of a HFM solution in a matrix $\mathbf{U} \in \mathbb{R}^{n \times n_t}$, $\mathbf{U} := [\mathbf{u}_1, \dots, \mathbf{u}_{n_t}]$. The POD seeks for a lower dimensional representation of \mathbf{U} by a basis $\mathbf{V} \in \mathbb{R}^{n \times m}$, $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_m]$, $m \ll n_t$ by solving the following least squares problem:

$$\min_{\mathbf{v}_i \in \mathbb{R}^n} \sum_{j=1}^{n_t} \left\| \mathbf{u}_j - \sum_{i=1}^m (\mathbf{u}_j^T \mathbf{v}_i) \mathbf{v}_i \right\|_2^2. \quad (4.69)$$

It can be shown that the vectors \mathbf{v}_i are the left singular vectors of \mathbf{U} , obtained by the Singular Value Decomposition (SVD) of \mathbf{U} as

$$\mathbf{U} = \mathbf{L}\mathbf{S}\mathbf{R}^T, \quad (4.70)$$

where $\mathbf{L} = [\mathbf{l}_1, \dots, \mathbf{l}_{n_t}]$, $\mathbf{R} = [\mathbf{r}_1, \dots, \mathbf{r}_n]$, $\mathbf{L}\mathbf{L}^T = \mathbf{I} \in \mathbb{R}^{n_t \times n_t}$, $\mathbf{R}\mathbf{R}^T = \mathbf{I} \in \mathbb{R}^{n \times n}$ and \mathbf{S} is a rectangular matrix containing the singular values on the diagonal, and zero otherwise. For details on algorithms to compute SVD, refer for instance to [23]. These singular values represent the relative importance of corresponding eigenvectors of \mathbf{L} in forming the basis \mathbf{V} . If the singular values S_{ii} , and the corresponding left and right vectors \mathbf{l}_i and \mathbf{r}_i are sorted in a decreasing order, i. e. $S_{11} > S_{22} > \dots, S_{n_t}$, it can be shown that the error norm is bounded, as

$$\sum_{j=1}^{n_t} \left\| \mathbf{u}_j - \sum_{i=1}^m (\mathbf{u}_j^T \mathbf{l}_i)^T \mathbf{l}_i \right\|_2^2 = \sum_{i=m+1}^{n_t} S_{ii}^2. \quad (4.71)$$

In other words, the left singular vectors $\mathbf{l}_1, \dots, \mathbf{l}_m$ corresponding to the highest singular values are the most significant for constructing a reduction basis. In Figure 4.5, we show an example of a POD analysis for the nonlinear dynamic response of a plate. Note that, in this context, the left vectors \mathbf{r}_i represent the (normalized) time evolution of the corresponding left singular vector. Since an SVD can be performed for any set of snapshots of any general nonlinear problem, POD is widespread as a versatile method to form a reduction basis. It should be noted, however, that the POD basis is optimal only for the set of snapshots which were used to generate it, while there is no guarantee on the accuracy of the NLROM outside the training data. For this reason, an NLROM based on POD are used in parametric contexts, i. e. when the HFM is parametrized with respect to material, geometric and load properties. In these cases, POD bases are obtained for different instances of the parameter set. Then parametric NLROMs are formed by either building a larger POD basis that spans the solution within the parameters range of interest [5], or by interpolating local NLROMs online for the desired value of the parameters which were not previously sampled [3]. In this framework, the high computational cost associated to the HFM solution required by the POD are amortized by the large number of queries to the NR0M. For further discussion of this matter, we refer to Section 4.6.

4.5.5 Non-intrusive methods

As already mentioned, non-intrusive model order reduction approaches are beneficial when a HFM model is developed by means of a FE model with no access to element forces and Jacobians required for the analytical computations of reduced nonlinear internal forces. There are various strategies to non-intrusively develop a nonlinear reduced order model. We outline here the most common methods.

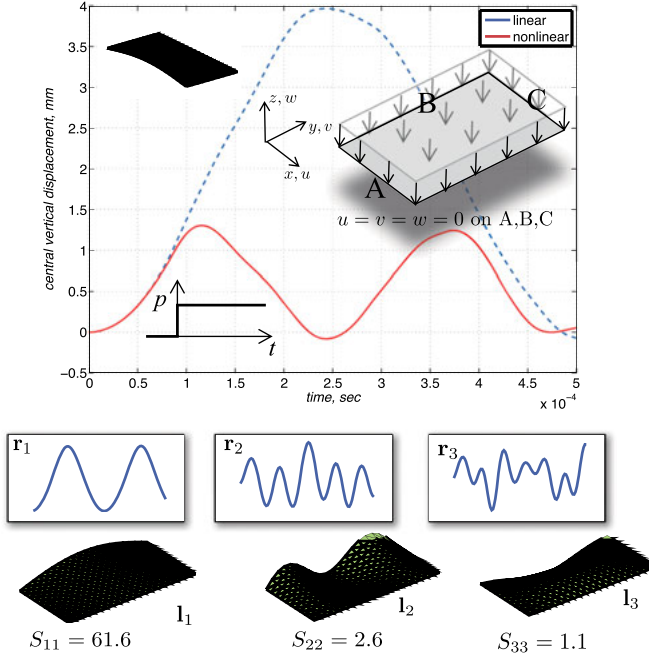


Figure 4.5: Example of a POD applied to a nonlinear structural dynamics case. A flat plate is simply supported on three sides, and loaded with step pressure, as shown. The HFM linear and nonlinear response the mid node of the free side of the plate are shown. The marked difference of the two responses highlight the significance of the nonlinear term. The first three left and right singular vectors, and the corresponding singular values are shown on the bottom of the figure. Note the resemblance of the dominant first left vector l_1 and its corresponding time evolution r_1 , with a random snapshot of the HFM (top left corner).

4.5.5.1 Enforced displacement

One way to identify the nonlinear stiffness coefficients of a reduced order model is to prescribe a set of nonlinear static displacements to the HFM and solve for the corresponding reaction forces that generate them. This method, which is known as *Enforced Displacement* (ED) or *Stiffness Evaluation Procedure* (STEP), was first developed by Muravyov et al. [48] and later extended by Kim et al. [38] for the case of unknown linear stiffness components. Essentially, ED first collects the reaction forces due to statically enforced displacements, projects them onto the modal space, and set linear equations to obtain the nonlinear stiffness coefficients $K_{ijl}^{(2)}$ and $K_{ijlp}^{(3)}$, $i, j, l, p = 1, \dots, m$.

Let us consider the reduced nonlinear restoring forces equation (4.40) already introduced in Section 4.4.1.3, which reads

$$\tilde{f}_i = \tilde{K}_{ij}^{(1)} q_j + \tilde{K}_{ijl}^{(2)} q_j q_l + \tilde{K}_{ijlp}^{(3)} q_j q_l q_p, \quad (4.72)$$

with $i = 1, 2, \dots, m$, where summation over repeated indices is implied. The second-order tensor $\tilde{K}_{ij}^{(1)}$ can be immediately found to be

$$\tilde{K}_{ij}^{(1)} = \mathbf{V}_i^T \mathbf{K} \mathbf{V}_j, \quad (4.73)$$

where \mathbf{K} is the stiffness matrix of the HFM, usually obtainable from any FE program. Clearly, $\tilde{K}_{ij}^{(1)} = \tilde{K}_{ji}^{(1)}$.

The ED method sequentially set linear systems of equation whose unknowns are $K_{ijl}^{(2)}$ and $K_{ijlp}^{(3)}$. The first step consists of choosing a modal amplitude $\pm q_r$, $r = 1, \dots, m$ and assign it to the reduced static problem as

$$\tilde{K}_{ir}^{(1)} q_r + \tilde{K}_{irr}^{(2)} q_r^2 + \tilde{K}_{irr}^{(3)} q_r^3 = \tilde{p}_i^{(r_+)}, \quad (4.74a)$$

$$-\tilde{K}_{ir}^{(1)} q_r + \tilde{K}_{irr}^{(2)} q_r^2 - \tilde{K}_{irr}^{(3)} q_r^3 = \tilde{p}_i^{(r_-)}, \quad (4.74b)$$

with $i = 1, \dots, m$. The right-hand sides $\tilde{p}_i^{(r_+)}$ and $\tilde{p}_i^{(r_-)}$ are given by

$$\tilde{p}_i^{(r_+)} = \mathbf{V}_i^T \mathbf{f}(\mathbf{u}^{(r_+)}), \quad (4.75)$$

$$\tilde{p}_i^{(r_-)} = \mathbf{V}_i^T \mathbf{f}(\mathbf{u}^{(r_-)}), \quad (4.76)$$

where $\mathbf{u}^{(r_+)} = \mathbf{V}_r q_r$ and $\mathbf{u}^{(r_-)} = -\mathbf{V}_r q_r$. In words, $\mathbf{u}^{(r_+)}$ and $\mathbf{u}^{(r_-)}$ are imposed statically to the HFM, and the resulting reaction forces are projected onto \mathbf{V}_i .

Likewise, we can select pairs of modal amplitudes (q_r, q_s) , $(-q_r, -q_s)$ and $(q_r, -q_s)$ and impose it to the reduced static problem as

$$\begin{aligned} \tilde{K}_{ir}^{(1)} q_r + \tilde{K}_{is}^{(1)} q_s + \tilde{K}_{irr}^{(2)} q_r^2 + \tilde{K}_{irs}^{(2)} q_r q_s + \tilde{K}_{iss}^{(2)} q_s^2 \\ + \tilde{K}_{irr}^{(3)} q_r^3 + q_r^2 q_s + 3\tilde{K}_{irss}^{(3)} q_r q_s^2 + \tilde{K}_{iss}^{(3)} q_s^3 = \tilde{p}_i^{(rs_{++})} \end{aligned} \quad (4.77a)$$

$$\begin{aligned} -\tilde{K}_{ir}^{(1)} q_r - \tilde{K}_{is}^{(1)} q_s + \tilde{K}_{irr}^{(2)} q_r^2 + \tilde{K}_{irs}^{(2)} q_r q_s + \tilde{K}_{iss}^{(2)} q_s^2 \\ - \tilde{K}_{irr}^{(3)} q_r^3 - \tilde{K}_{irs}^{(3)} q_r^2 q_s - \tilde{K}_{irss}^{(3)} q_r q_s^2 - \tilde{K}_{iss}^{(3)} q_s^3 = \tilde{p}_i^{(rs_{--})} \end{aligned} \quad (4.77b)$$

$$\begin{aligned} \tilde{K}_{ir}^{(1)} q_r - \tilde{K}_{is}^{(1)} q_s + \tilde{K}_{irr}^{(2)} q_r^2 - \tilde{K}_{irs}^{(2)} q_r q_s + \tilde{K}_{iss}^{(2)} q_s^2 \\ + \tilde{K}_{irr}^{(3)} q_r^3 - \tilde{K}_{irs}^{(3)} q_r^2 q_s + \tilde{K}_{irss}^{(3)} q_r q_s^2 - \tilde{K}_{iss}^{(3)} q_s^3 = \tilde{p}_i^{(rs_{+-})}, \end{aligned} \quad (4.77c)$$

where we restrict the case to $s \geq r$. The right-hand sides of (4.77a)–(4.77c) are given by

$$\tilde{p}_i^{(rs_{++})} = \mathbf{V}_i^T \mathbf{f}(\mathbf{u}^{(rs_{++})}), \quad (4.78a)$$

$$\tilde{p}_i^{(rs_{--})} = \mathbf{V}_i^T \mathbf{f}(\mathbf{u}^{(rs_{--})}), \quad (4.78b)$$

$$\tilde{p}_i^{(rs_{+-})} = \mathbf{V}_i^T \mathbf{f}(\mathbf{u}^{(rs_{+-})}), \quad (4.78c)$$

where, analogously to (4.75), we statically impose to the HFM $\mathbf{u}^{(rs_{++})} = \mathbf{V}_r q_r + \mathbf{V}_s q_s$, $\mathbf{u}^{(rs_{--})} = -\mathbf{V}_r q_r - \mathbf{V}_s q_s$ and $\mathbf{u}^{(rs_{+-})} = \mathbf{V}_r q_r - \mathbf{V}_s q_s$. Lastly, we need to compute $\tilde{K}_{irsp}^{(3)}$,

$r \neq s \neq p$. Similarly to the previous steps, we determine a triple (q_r, q_s, q_p) and insert it to the reduced static problem. This results in a single equation for $\tilde{K}_{irsp}^{(3)}$, as

$$\begin{aligned} & \tilde{K}_{ir}^{(1)} q_r + \tilde{K}_{is}^{(1)} q_s + \tilde{K}_{ip}^{(1)} q_p + \tilde{K}_{irr}^{(2)} q_r^2 + \tilde{K}_{iss}^{(2)} q_s^2 + \tilde{K}_{ipp}^{(2)} q_p^2 \\ & + \tilde{K}_{irs}^{(2)} q_r q_s + \tilde{K}_{irp}^{(2)} q_r q_p + \tilde{K}_{isp}^{(2)} q_s q_p + \tilde{K}_{irr}^{(3)} q_r^3 + \tilde{K}_{iss}^{(3)} q_s^3 + \tilde{K}_{ipp}^{(3)} q_p^3 \\ & + \tilde{K}_{irrs}^{(3)} q_r^2 q_s + \tilde{K}_{irrp}^{(3)} q_r^2 q_p + \tilde{K}_{irss}^{(3)} q_r q_s^2 + \tilde{K}_{irpp}^{(3)} q_r q_p^2 + \tilde{K}_{ipss}^{(3)} q_p q_s^2 + \tilde{K}_{ispp}^{(3)} q_s q_p^2 \\ & + \tilde{K}_{irsp}^{(3)} q_r q_s q_p = \tilde{p}_i^{(rsp)}, \end{aligned} \quad (4.79)$$

with $p \geq s \geq r$, and

$$\tilde{p}_i^{(rsp)} = \mathbf{V}_i^T \mathbf{f}(\mathbf{u}^{(rsp)}), \quad (4.80)$$

where $\mathbf{u}^{(rsp)} = \mathbf{V}_r q_r + \mathbf{V}_s q_s + \mathbf{V}_p q_p$. Note that the restriction $s > r$ in (4.77a)–(4.77c) and $p > s > r$ in (4.79) is in fact setting $\tilde{K}_{irs}^{(2)} = 0 \forall s < r$ and $\tilde{K}_{irsp}^{(2)} = 0 \forall p < s < r$. This is done to minimize the number of coefficients multiplying the same combinations of q_r, q_s and $q_r q_s q_p$, $\forall r, s, p = 1, \dots, m$. Alternatively, one could set symmetry properties as

$$K_{ijl}^{(2)} = K_{ijl}^{(2)} \quad (4.81)$$

and

$$K_{ijlp}^{(3)} = K_{ijpl}^{(3)} = K_{iljp}^{(3)} = K_{ilpj}^{(3)} = K_{ipjl}^{(3)} = K_{iplj}^{(3)}, \quad (4.82)$$

and modify (4.77a)–(4.77c) and (4.79) accordingly. The total number N_{ED} of the required nonlinear static evaluations to fully construct the nonlinear reduced restoring forces is

$$N_{ED} = 2m + 3 {}_m C_2 + {}_m C_3, \quad (4.83)$$

where

$${}_m C_r = \frac{m!}{(m-r)! r!}. \quad (4.84)$$

Since this number is $\mathcal{O}(m^3)$, the computational cost associated to ED grows quickly with the number of retained modes and severely hinders the applicability of the method.

4.5.5.2 Enhanced enforced displacement

To reduce the computational costs of the ED method, Perez et al. [52] proposed a way to identify the NLROM coefficients in the case that the tangent stiffness matrix is released

by the FE program at hand. We refer to this method as *Enhanced Enforced Displacements* (EED). The tangent stiffness matrix $\mathbf{K}^t(\mathbf{u})$ is the Jacobian of the nonlinear forces \mathbf{f} with respect to the nodal displacements, as

$$\mathbf{K}^t(\mathbf{u}) = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}. \quad (4.85)$$

Let us assume that $\mathbf{K}^t(\mathbf{u})$ is available. Then the reduced tangent stiffness matrix $\tilde{\mathbf{K}}^t$ can be obtained from \mathbf{K}^t as

$$\tilde{\mathbf{K}}^t = \mathbf{V}^T \mathbf{K}^t(\mathbf{V}\mathbf{q})\mathbf{V}, \quad (4.86)$$

whose ir component is given by

$$\tilde{K}_{ir}^t = \mathbf{V}_i^T \mathbf{K}^t(\mathbf{V}\mathbf{q})\mathbf{V}_r, \quad (4.87)$$

and we pose $\mathbf{u} = \mathbf{V}\mathbf{q}$. The reduced tangent stiffness matrix $\tilde{\mathbf{K}}^t$ is the Jacobian of the reduced nonlinear internal force vector, $\tilde{\mathbf{f}}$, namely

$$\tilde{\mathbf{K}}^t(\mathbf{q}) = \begin{bmatrix} \frac{\partial \tilde{f}_1}{\partial q_1} & \dots & \frac{\partial \tilde{f}_1}{\partial q_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial \tilde{f}_m}{\partial q_1} & \dots & \frac{\partial \tilde{f}_m}{\partial q_m} \end{bmatrix}. \quad (4.88)$$

Then, following from (4.40), \tilde{K}_{ir} reads

$$\begin{aligned} \tilde{K}_{ir}^t &= \frac{\partial \tilde{f}_i}{\partial q_r} = \frac{\partial}{\partial q_r} [\tilde{K}_{ij}^{(1)} q_j + \tilde{K}_{ijl}^{(2)} q_j q_l + \tilde{K}_{ijlp}^{(3)} q_j q_l q_p] \\ &= \tilde{K}_{ir}^{(1)} + [\tilde{K}_{ijr}^{(2)} + \tilde{K}_{irj}^{(2)}] q_j + [\tilde{K}_{ijlr}^{(3)} + \tilde{K}_{ijrl}^{(3)} + \tilde{K}_{irjl}^{(3)}] q_j q_l. \end{aligned} \quad (4.89)$$

The EED method equates (4.89) and (4.87) for as many combinations of modal amplitudes as those required to solve for the reduced stiffness coefficients $\tilde{K}_{ijr}^{(2)}$ and $\tilde{K}_{ijrl}^{(3)}$, $i, j, r, l = 1, \dots, m$. Similarly to the ED method, first two static displacement vectors for each column of the reduction basis are constructed and inserted in (4.86). We can then set two equations as

$$\tilde{K}_{ir}^{t(a)} = \tilde{K}_{ir}^{(1)} + [\tilde{K}_{ijr}^{(2)} + \tilde{K}_{irj}^{(2)}] q_j^{(a)} + [\tilde{K}_{ijlr}^{(3)} + \tilde{K}_{ijrl}^{(3)} + \tilde{K}_{irjl}^{(3)}] (q_j^{(a)})^2, \quad a = 1, 2. \quad (4.90)$$

Since the elements $\tilde{K}_{ijl}^{(2)}$ and $\tilde{K}_{ijlp}^{(3)}$ were assumed to be zero unless $p \geq l \geq j$, (4.90) splits into three different cases, namely

$$\tilde{K}_{ir}^{t(a)} = \tilde{K}_{ir}^{(1)} + \tilde{K}_{ijr}^{(2)} q_j^{(a)} + \tilde{K}_{ijlr}^{(3)} (q_j^{(a)})^2, \quad \text{if } j < r, \quad (4.91a)$$

$$\tilde{K}_{ir}^{t(a)} = \tilde{K}_{ir}^{(1)} + 2\tilde{K}_{irr}^{(2)} q_j^{(a)} + 3\tilde{K}_{irrr}^{(3)} (q_j^{(a)})^2, \quad \text{if } j = r, \quad (4.91b)$$

$$\tilde{K}_{ir}^{(a)} = \tilde{K}_{ir}^{(1)} + \tilde{K}_{irj}^{(2)} q_j^{(a)} + \tilde{K}_{irjj}^{(3)} (q_j^{(a)})^2, \quad \text{if } j > r. \quad (4.91c)$$

The solutions of (4.90) and (4.91a)–(4.91c) yield the coefficients $\tilde{K}_{ijr}^{(2)}$, $\tilde{K}_{ijjr}^{(3)}$, $\tilde{K}_{ijrr}^{(3)}$, and $\tilde{K}_{ijjj}^{(3)}$, $j, r = 1, \dots, m$. At this point, the only set of coefficients to be identified is $\tilde{K}_{ijlr}^{(3)}$, $j \neq l \neq r$, $j, l, r = 1, \dots, m$. To calculate these coefficients one can use displacement vectors which are formed from combinations of two different columns of \mathbf{V} by

$$\mathbf{u} = \mathbf{V}_j q_j + \mathbf{V}_l q_l, \quad j, l = 1, \dots, m, \quad (4.92)$$

where q_j and q_l are prescribed. These displacements are again inserted into (4.86) to obtain their corresponding tangent stiffness matrices. The ir th component of the associated reduced tangent stiffness matrix is then given by

$$\tilde{K}_{ir}^t = \tilde{K}_{ir}^{(1)} + [\tilde{K}_{ijr}^{(2)} q_j + \tilde{K}_{ilr}^{(2)} q_l] + [\tilde{K}_{ijlr}^{(3)} q_j q_l + \tilde{K}_{ijjr}^{(3)} q_j^2 + \tilde{K}_{illr}^{(3)} q_l^2]. \quad (4.93)$$

The only unknown in (4.93) is $\tilde{K}_{ijlr}^{(3)}$, which can be found without needing combinations of three columns of \mathbf{V} as for the case of the ED method. As a result, EED is computationally more efficient than ED. In fact, the total number of required static solutions N_{EED} is then

$$N_{\text{EED}} = 2m + {}_m C_2, \quad (4.94)$$

which is $\mathcal{O}(m^2)$, as opposed to $\mathcal{O}(m^3)$ for the ED method. Note that the amplitudes q_i , $i = 1, \dots, m$ are user defined. Typically, they are selected such that the resulting displacements generate nonlinear forces of the desired magnitude. For shell-like structures, it is usually suggested to select q_i , $i = 1, \dots, m$ in the order of one thickness of the structure for transverse-dominated modes and 0.1 to 0.01 of that for membrane-dominated ones. A flowchart of the ED and EED method is shown in Figure 4.6.

4.5.5.3 Implicit condensation

In order to identify the coefficients of (4.40) via either the ED or the EED method, the reduction basis \mathbf{V} must contain vibration modes and corresponding membrane-dominated vectors (e. g. MDs, DMs or manually selected high-frequency VMs). This increases the size of the basis significantly and impacts the efficiency of the method. However, it is possible to avoid the use of membrane-dominated modes by using the method of *Implicit Condensation* (IC) or *Applied Force* method, developed by McEwan et al. [45, 46]. Let $\mathbf{p}^{(s)}$ be a generic load obtained by combining one, two and three columns of \mathbf{V} with chosen scaling factors \bar{q}_i , \bar{q}_j , \bar{q}_k as

$$\mathbf{p}^{(s)} = \mathbf{K}(\mathbf{V}_i(\pm\bar{q}_i) + \mathbf{V}_j(\pm\bar{q}_j) + \mathbf{V}_k(\pm\bar{q}_k)), \quad i, j, k = 0, 1, 2, \dots, m, \quad i \neq j \neq k, \quad (4.95)$$

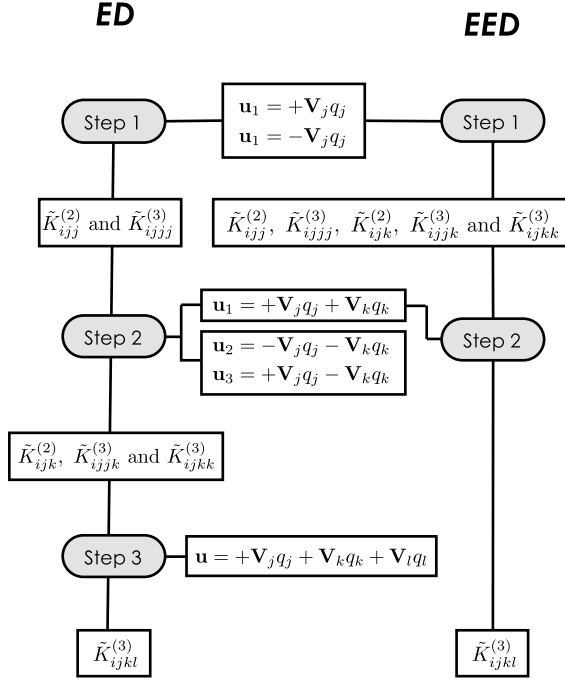


Figure 4.6: Flowchart of comparison between ED and EED steps to identify nonlinear stiffness coefficients. The EED method manages to identify all the nonlinear stiffness coefficients in two steps only, because of the availability of the full order tangent stiffness matrix [34].

where the index 0 implies that the corresponding term is ignored, i. e.

$$\mathbf{K}(\mathbf{V}_0 \bar{q}_0 + \mathbf{V}_1 \bar{q}_1 + \mathbf{V}_2 \bar{q}_2) = \mathbf{K}(\mathbf{V}_1 \bar{q}_1 + \mathbf{V}_2 \bar{q}_2).$$

One can show that the total number of cases N_{IC} that can be generated from (4.95) is

$$N_{IC} = 2m + 4_m C_2 + 8_m C_3, \quad (4.96)$$

and therefore $s = 1, \dots, N_{IC}$. The scaling factors \bar{q}_i , $i = 1, \dots, m$ should be chosen such that the resulting displacements are in the nonlinear regime. Gordon and Hollkamp [20] propose that a scaling factor \bar{q}_i for (4.95) should generate a force that induces a desired maximum displacement $W_{i_{\max}}$ as

$$\bar{q}_i = \frac{W_{i_{\max}}}{V_{i_{\max}}} \omega_i^2, \quad (4.97)$$

where $V_{i_{\max}}$ denotes the component of \mathbf{V}_i mode which has the maximum out-of-plane displacement, and ω_i is the eigenfrequency of mode \mathbf{V}_i . The maximum desired displacement $W_{i_{\max}}$, is usually chosen in the order of the thickness of the structure to properly exercise the nonlinearity.

Each load $\mathbf{p}^{(s)}$ is applied to the static HFM as

$$\mathbf{f}(\mathbf{u}^{(s)}) = \mathbf{p}^{(s)}. \quad (4.98)$$

The resulting displacement $\mathbf{u}^{(s)}$ is then expressed in the subspace spanned by \mathbf{V} through the modal amplitudes $\mathbf{q}^{(s)}$ as

$$\mathbf{u}^{(s)} = \mathbf{V}\mathbf{q}^{(s)} + \hat{\mathbf{u}}^{(s)} \Rightarrow \mathbf{q}^{(s)} = \mathbf{V}^T \mathbf{M}\mathbf{u}^{(s)}, \quad (4.99)$$

where $\hat{\mathbf{u}}^{(s)}$ is the remainder and it is assumed that $\mathbf{V}^T \mathbf{M}\hat{\mathbf{u}}^{(s)} = \mathbf{0}$. Likewise, $\mathbf{p}^{(s)}$ is projected on \mathbf{V} to give the modal loads $\tilde{\mathbf{p}}^{(s)}$ as

$$\tilde{\mathbf{p}}^{(s)} = \mathbf{V}^T \mathbf{p}^{(s)}. \quad (4.100)$$

Then each pair $(\tilde{\mathbf{p}}^{(s)}, \mathbf{q}^{(s)})$ is inserted in the static reduced order problem to obtain

$$\tilde{K}_{ijl}^{(2)} q_j^{(s)} q_l^{(s)} + \tilde{K}_{ijlp}^{(3)} q_j^{(s)} q_l^{(s)} q_p^{(s)} = \tilde{p}_i^{(s)} - \omega_i^2 q_i^{(s)}, \quad i = 1, 2, \dots, m, \quad (4.101)$$

where it is further assumed that the symmetry properties (4.81) and (4.82) hold. All equations resulting from (4.101) can be written in matrix form as

$$\mathbf{G}_{nl} \mathbf{K}_{nl} = \mathbf{P}_{nl}, \quad (4.102)$$

where

$$\mathbf{G}_{nl}^T = \begin{bmatrix} (q_1^{(1)})^2 & \dots & (q_1^{(N_{lc})})^2 \\ \vdots & \ddots & \vdots \\ q_i^{(1)} q_j^{(1)} & \ddots & q_i^{(N_{lc})} q_j^{(N_{lc})} \\ \vdots & \ddots & \vdots \\ (q_m^{(1)})^2 & \dots & (q_m^{(N_{lc})})^2 \\ (q_1^{(1)})^3 & \dots & (q_1^{(N_{lc})})^3 \\ \vdots & \ddots & \vdots \\ q_i^{(1)} q_j^{(1)} q_k^{(1)} & \ddots & q_i^{(N_{lc})} q_j^{(N_{lc})} q_k^{(N_{lc})} \\ \vdots & \ddots & \vdots \\ (q_m^{(1)})^3 & \dots & (q_m^{(N_{lc})})^3 \end{bmatrix}, \quad (4.103)$$

$$\mathbf{K}_{nl}^T = \begin{bmatrix} \tilde{K}_{111}^{(2)} & \dots & \tilde{K}_{1ij}^{(2)} & \dots & \tilde{K}_{1mm}^{(2)} & \tilde{K}_{1111}^{(3)} & \dots & \tilde{K}_{1ijk}^{(3)} & \dots & \tilde{K}_{1mmmm}^{(3)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \tilde{K}_{m11}^{(2)} & \dots & \tilde{K}_{mij}^{(2)} & \dots & \tilde{K}_{mmm}^{(2)} & \tilde{K}_{m111}^{(3)} & \dots & \tilde{K}_{mijk}^{(3)} & \dots & \tilde{K}_{mmmm}^{(3)} \end{bmatrix}, \quad (4.104)$$

and

$$\mathbf{P}_{nl} = \begin{bmatrix} \tilde{p}_1^{(1)} - \omega_1^2 q_1^{(1)} & \dots & \tilde{p}_1^{(N_{IC})} - \omega_1^2 q_1^{(N_{IC})} \\ \vdots & \ddots & \vdots \\ \tilde{p}_m^{(1)} - \omega_m^2 q_1^{(1)} & \dots & \tilde{p}_m^{(N_{IC})} - \omega_m^2 q_1^{(N_{IC})} \end{bmatrix}. \quad (4.105)$$

The system of equations (4.102) can be solved by means of any regression technique, for instance the least square method.

The main advantage of the IC method is the fact that the resulting NROM is based on vibration modes only, and thus is very compact as compared to the ED and EED method, which require also modes (e. g. DMs or MDs,) that represent the membranal behavior due to the large displacements. In the case of ED and EED is then possible to directly retrieve a full displacement field, and from it compute strain and stresses. This feature is crucial when the final aim of the analysis is to perform stress/strain investigations (for instance for fatigue life estimation). On the contrary, the IC method does not offer this possibility, as membranal behavior is not present in the basis.

To overcome this issue, Holkkamp and Gordon [24] developed a post-processing expansion procedure, which approximate the response of these DOFs without increasing the number of required static solutions in IC. In this method, the total displacement vector $\mathbf{u}^{(s)}$ relative to load case s can be written as

$$\mathbf{u}^{(s)} = \mathbf{V}\mathbf{q}^{(s)} + \hat{\mathbf{u}}^{(s)}, \quad (4.106)$$

where $\mathbf{u}^{(s)}$ is the solution of (4.98), and $\hat{\mathbf{u}}^{(s)}$ is the correction vector for in-plane DOFs, to be found by the expansion procedure. We can express $\hat{\mathbf{u}}^{(s)}$ in a modal fashion as

$$\hat{\mathbf{u}}^{(s)} = \hat{\mathbf{V}}\hat{\mathbf{q}}^{(s)} \quad (4.107)$$

where $\hat{\mathbf{V}}$ is the transformation matrix and $\hat{\mathbf{q}}^{(s)}$ is the vector of modal membrane coordinates. The whole displacement set for $s = 1, \dots, N_{IC}$ can be compactly written as

$$\mathbf{U} = \mathbf{V}\mathbf{Q} + \hat{\mathbf{V}}\hat{\mathbf{Q}}, \quad (4.108)$$

where $\mathbf{U} = [\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(N_{IC})}]$, $\mathbf{Q} = [\mathbf{q}^{(1)}, \dots, \mathbf{q}^{(N_{IC})}]$ while $\hat{\mathbf{V}}$ and $\hat{\mathbf{Q}}$ are yet to be determined. It is assumed that $\hat{\mathbf{V}}^T \mathbf{M}\mathbf{V} = \mathbf{0}$, that is to say, the in-plane displacement to be retrieved lies in a subspace which is \mathbf{M} -orthogonal to \mathbf{V} . Therefore,

$$\mathbf{Q} = \mathbf{V}^T \mathbf{M} \mathbf{U}. \quad (4.109)$$

It is then assumed that each column $\hat{\mathbf{q}}^{(s)}$ of $\hat{\mathbf{Q}}$ is given by

$$\hat{\mathbf{q}}^{(s)} = [(q_1^{(s)})^2, q_1^{(s)} q_2^{(s)}, \dots, q_i^{(s)} q_j^{(s)}, \dots, (q_m^{(s)})^2]^T; \quad (4.110)$$

see [49] for more details. Then, by virtue of (4.110) and (4.109), we finally obtain

$$\hat{\mathbf{V}} = (\mathbf{I} - \mathbf{V}\mathbf{V}^T \mathbf{M}) \mathbf{U} \hat{\mathbf{Q}}^+, \quad (4.111)$$

where $\hat{\mathbf{Q}}^+$ is the pseudo-inverse of $\hat{\mathbf{Q}}$ and $\hat{\mathbf{V}} \in \mathbb{R}^{n \times \frac{m(m+1)}{2}}$.

In practice, the NLROM obtained by IC is integrated in time and the vectors of modal coordinates \mathbf{q} at time samples t_1, t_2, \dots, t_{n_T} are collected in \mathbf{Q}_T as

$$\mathbf{Q}_T = [\mathbf{q}(t_1), \dots, \mathbf{q}(t_{n_T})]. \quad (4.112)$$

The total displacement $\mathbf{U}_T = [\mathbf{u}(t_1), \dots, \mathbf{u}(t_{n_T})]$ is retrieved by

$$\mathbf{U}_T = \mathbf{V}\mathbf{Q}_T + \hat{\mathbf{V}}\hat{\mathbf{Q}}_T, \quad (4.113)$$

where the column $\hat{\mathbf{q}}(t_i)$ of $\hat{\mathbf{Q}}_T$ relative to time step t_i is given by

$$\hat{\mathbf{q}}(t_i) = [(q_1(t_i))^2, q_1(t_i)q_2(t_i), \dots, q_i(t_i)q_j(t_i), \dots, (q_m(t_i))^2]. \quad (4.114)$$

A comparison of the number of evaluations required by ED, EED and IC is shown in Figure 4.7.

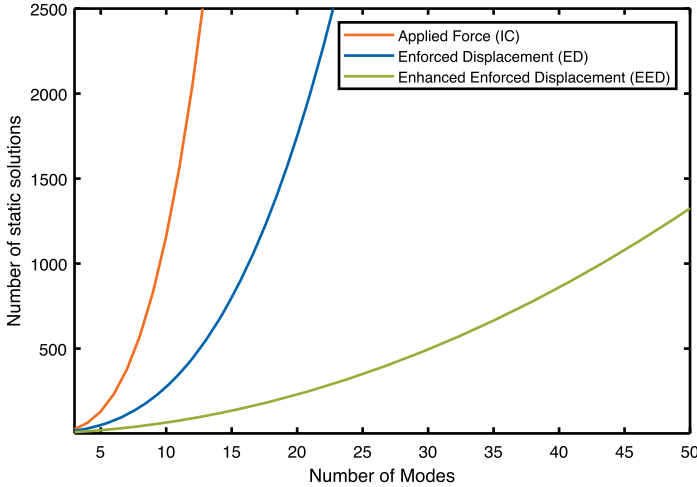


Figure 4.7: Number of required nonlinear static solution for ED, EED and IC versus the number of modes in the reduction basis [34].

4.5.6 Nonlinear normal modes

As briefly discussed, the presence of nonlinearity may alter the stiffness of the system (for geometric nonlinearities), the damping forces (i. e. with friction), or both. In fact, it was shown that linear reduction techniques based on vibration modes only are not sufficient to capture the nonlinear behavior, and therefore other vectors (e. g. MDs or DMs) have to be added to the reduction basis. One could also wonder how a certain eigensolution (i. e. a vibration mode oscillating at its own eigenfrequency) changes

when the displacements trigger the nonlinearity. For conservative autonomous systems (i. e. systems with no damping and no external excitation), there still exist periodic solutions, usually called *Nonlinear Normal Modes* (NNMs), which can be thought of as the nonlinear equivalent of a vibration mode for a linear system. This concept was first defined by Rosenberg [55] as “any vibration-in-unison of a conservative nonlinear system, i. e. where the coordinates of the system pass through the equilibrium and reach their extrema simultaneously”, and later generalized to include any non-necessarily synchronous periodic motion [37]. The latter definition allows the system to exhibit internal resonance, i. e. the coexistence of two or more modes which feature periodic motions at periods at commensurate ratios. Pioneering work on NNMs was done by Shaw and Pierre [59, 60].

NNMs are instrumental in assessing the impact of the nonlinearity on the dynamic response. In [40], NNMs are proposed as tool to assess the quality of a NLROM. In this work, it is in fact suggested to directly compare NNMs computed with NLROMs, as the NNMs of the HFM are too costly to be computed for large systems. Usually, the effect of a NNM is summarized by means of a *Frequency–Energy Plot* (FEP), where the frequency of the periodic motion is plotted against the energy associated to the motion itself, for instance resulting from initial conditions. Note that, in such representation, the actual shape of the motion throughout the period is hidden. In Figure 4.8 (taken from [62]), the NNM associated to the first vibration mode of a shallow arch pinned at both ends is shown. The response stays linear (i. e. of constant frequency) for a wide range of energy. Then the frequency decreases due to a softening¹ behavior, followed by an internal resonance tongue due to the interaction with the NNM associated to the third vibration mode. At this point, in fact, the two NNMs evolve with a period ratio of 4:1, enabling the existence of a periodic solution at which both contributions can coexist. The motion triggers then a hardening behavior at higher energy. A FEP as the one shown in Figure 4.8 is computationally expensive to construct. In fact, every point of the FEP requires the knowledge of a periodic motion for a given energy level. The frequency and the initial conditions of such motion are not known a priori. Numerical techniques for solving this problem are discussed in detail in [51] and [28]. Here, is it worth to highlight the benefits of resorting to NLROMs when computing NNMs. In [62], for instance, MDs-based ROMs were assessed for the computation of NNMs of FE-discretized planar beam structures. In this work, the shooting method was used to obtain periodic solutions of the autonomous conservative systems of interest. This method essentially integrates the system in time for trial initial conditions (i. e. initial deformed shape imposed to the structure) and a trial period, and then correct them via

¹ The term “softening” (“hardening”) usually refers to a decrease (increase) of the slope of the restoring nonlinear forces. For instance, a flat beam, cantilevered at both ends and loaded by a transverse force at its mid-span features a hardening stiffness: the bending associated to finite displacements induces stretching in the cross-section, which in turn generates an additional restoring bending moment to that due to linear effects.

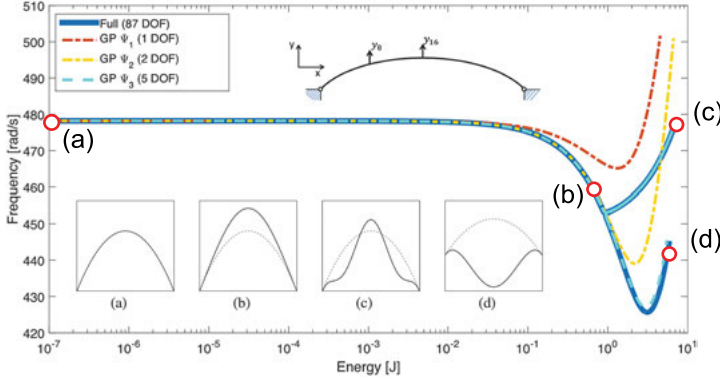


Figure 4.8: First NNM of a shallow arch. Note that the NNM first exhibits softening (i. e. decrease in frequency due to a decrease in stiffness), followed by an internal resonance (4:1 interaction with the third NNM), and then hardening. The FEP computed for the full model (blue solid line), a ROM containing $\mathbf{V} = [\boldsymbol{\phi}_1]$, a ROM containing $\mathbf{V} = [\boldsymbol{\phi}_1 \boldsymbol{\phi}_3]$, and a ROM including $\mathbf{V} = [\boldsymbol{\phi}_1 \boldsymbol{\phi}_3 \boldsymbol{\theta}_{11} \boldsymbol{\theta}_{13} \boldsymbol{\theta}_{33}]$. Plots (a) to (d) show the spatial shape of the response (solid line) for the corresponding points on the FEP. The undeformed configuration is shown with dashed line [62].

Newton–Raphson iterations until a periodicity condition is met with pre-defined accuracy. Once a solution is obtained (i. e. a point of the FEP), continuation is employed to estimate the new guess at higher energy level. It becomes clear then that each point of an FEP requires multiple time integration of the system, and therefore, the tracing of a FEP for each mode of interest can be a daunting task. In [62], it was demonstrated that MD-based NLROMs capture well the nonlinear behavior and internal resonances by only including the vibration modes—and corresponding MDs, which are expected to interact. This is achieved at a smaller computational cost as the one required for the HFM. Moreover, a NLROM so constructed filters a physically meaningless contribution of high-frequency dynamics, which is only due to the resolution of the spatial discretization. In doing so, the convergence of numerical implicit time integration is significantly improved.

4.5.7 Nonlinear manifolds for reduction

Up to now, we discussed techniques that approximate the HFM solution as a linear combination of modes spanning a low-dimensional subspace. This is not the only possibility, as many dynamical systems evolve on *manifolds*, rather than subspaces [44]. This is typically the case for systems characterized by *slow* and *fast* dynamics. An example of this is static condensation (see Section 4.5.1), where the fast dynamics of the in-plane motion is statically enslaved to the slow dynamics of the bending deformations. The resulting mapping between fast and slow dynamics is in this case quadratic, see equation (4.51). Once a condensed system is obtained through slow–fast decom-

position, a further reduction is of course possible. One could apply the techniques discussed in Sections 4.5.2, 4.5.3 and 4.5.4 (i. e. based on linear subspaces for reduction) or seek for a manifold on which the solution is embedded. Such a manifold passes through the equilibrium point and is there tangent to the corresponding linear mode. Recently, the concept of *Spectral Submanifold* (SSM) was introduced by Haller and Ponsioen in [21]. The SSM was introduced as an invariant manifold asymptotic to a NNM, and it is the smoothest nonlinear continuation of a modal subspace emanating from the equilibrium. A discussion on the potential of SSM for model reduction can be found in [53], where SSMs are computed for a linear FE-discretized beam with a cubic nonlinear spring acting at the tip. An interesting application of slow–fast decomposition combined with SSM reduction is presented in [30] for the case of a geometrically nonlinear FE discretized beam.

An approximate method to construct a manifold for reduction, related to the MDs method, is presented in [31] and then generalized in [58]. There, it is postulated that the HFM displacements $\mathbf{u}(t)$ are mapped to the modal coordinates $\mathbf{q}(t)$, i. e. $\mathbf{q} \in \mathbb{R}^m \rightarrow \Gamma(\mathbf{q}) \in \mathbb{R}^n$ as

$$\mathbf{u}(t) \approx \Gamma(\mathbf{q}(t)) := \mathbf{\Phi} \cdot \mathbf{q}(t) + \frac{1}{2}(\mathbf{\Omega} \cdot \mathbf{q}(t) \cdot \mathbf{q}(t)), \quad (4.115)$$

where $\mathbf{\Phi} \in \mathbb{R}^{n \times m}$ contains selected vibration modes, and $\mathbf{\Omega} \in \mathbb{R}^{n \times m \times m}$ is a third-order tensor constructed with the MDs stemming from $\mathbf{\Phi}$. The resulting manifold is therefore quadratic in \mathbf{q} , hence the naming *Quadratic Manifold* (QM). The concept is illustrated in Figure 4.9, for the case of a cantilevered plate. Instead of constituting additional DOFs for the NLRM, the MDs provide the curvature of the manifold. The NLRM is then obtained by inserting (4.115) into (4.30) and projecting onto the tangent manifold $\mathbf{P}_\Gamma(\mathbf{q})$, obtained:

$$\mathbf{P}_\Gamma(\mathbf{q}) = \frac{\partial \Gamma(\mathbf{q})}{\partial \mathbf{q}}. \quad (4.116)$$

Again, the interested reader is referred to [31] for details. The method was tested on a fairly large FE model of a wing subject to representative gust load. This approach outperformed a POD reduction with the same number of modes (5 vibration modes vs 5 POD modes), in terms of accuracy. It is worth mentioning that the QM technique needs to be equipped with a compression technique of the resulting nonlinear reduced terms. An exact tensorial form as outlined in Section 4.4.1.2 is cumbersome, as the resulting nonlinear terms would contain tensors up to seventh order. Likely, the higher order terms are not relevant for accuracy, and yet require the most intensive computational and storage effort. A possible strategy could be therefore to neglect some of the highest terms. Another possibility is hyper-reduction. It was shown in [32] that ECSW can be extended to the case of nonlinear manifolds for reduction. In this work, the same example tackled in [31] is considered, and better speed-ups than traditional, linear basis POD reduction are obtained, as less elements are picked by the algorithm.

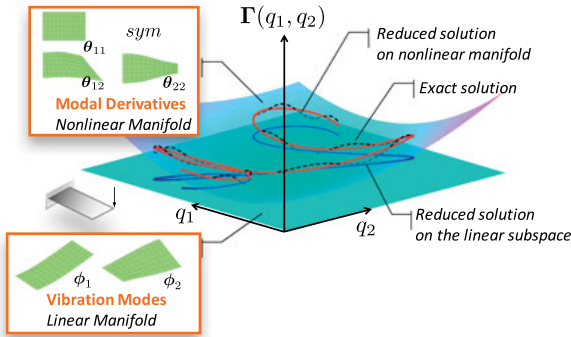


Figure 4.9: Illustrative picture of a quadratic manifold for reduction of a cantilevered plate. The manifold is a function of the amplitudes of the first two vibration modes. The three MDs provide the curvature of the manifold.

Arguably, this is due to the fact that the QM more compactly represents the physics of the problem than the POD basis.

Still, if hyper-reduction is the way to go for reduction with nonlinear manifolds, one would desire to form the training sets without recurring to expensive HFM simulations. Attempts in this sense are made by [57] and [29]. In the latter of these contributions, the QM is in fact used to “lift” inexpensive linear modal solutions obtained as described in Section 4.2. The obtained displacements are then used to generate nonlinear forces to train the ECSW selection. The method proved very effective for geometrically nonlinear problems, for which the QM provides a good description of the physical behavior. The NLROM, though, is still obtained by projection on linear subspaces containing vibration modes and modal derivatives. The extension of this technique to QM-based reduction is yet to be tried, but it is foreseen not to pose major hurdles.

4.6 Overview on parametric nonlinear ROMs

In the previous sections, projection-based NLROMs strategies to select a suitable basis and to evaluate the nonlinear system of equations have been discussed. As it has already been pointed out, most of the showcased methods do provide very high speed-ups in the online phase, but usually they require some kind of overhead costs involving offline training and/or pre-computations. This often heavily reduces the effective gain provided by the ROM. Indeed, calling for convenience t_{on} and t_{off} the online and offline times of a ROM and t_f the time of a full order analysis, the effective speed-up factor can be computed as $SP = (t_{\text{on}} + t_{\text{off}})/t_f$. It is clear that, if the same reduced model is used for multiple (say M) queries requiring different analyses, this factor increases

to $SP_M = (Mt_{\text{on}} + t_{\text{off}})/(Mt_f)$. In other words, the offline costs become less important as the number of evaluations increases. Parametric ROMs (PROMs) are then a natural extension of this concept, as the number M would include the sum of the number of queries for each combination of parameters, making the reduction even more profitable.

For linear systems parametric reduced order modeling (PMOR) techniques are well developed. Loosely speaking, the main challenges in PMOR reside in (i) the construction of the model itself, (ii) identification of a projection basis that can be valid over a *space of parameters*, and (iii) selection of an interpolation and sampling strategies. The model can be constructed following a *local* approach (e. g. building several ROMs for different parameters and interpolating the bases/ROM/solutions) or a *global* approach, where the full order system is parametrized and a single basis is selected. For the latter case, again many options are available. Moment-matching (discussed extensively in [7, Chapter 3]) is a popular strategy, where the basis is constructed requiring the n th derivatives (moments) of the reduced and full order systems with respect to the parameters to match [17]. Another approach is to construct the basis repeatedly applying POD to a set of simulations selected through some kind of parameter-space sampling procedures (e. g. Latin Hyper-cube, Smolyak sparse grid). For an extensive survey of existing methods the reader is referred to [6].

On the other hand, projection-based PMORs for nonlinear systems feature a much less mature theory, and constitute a broad and rapidly evolving area of research. Many of the methods available in the literature try to carry over concepts from linear analysis to the nonlinear case. In this sense, POD represents the most widely exploited method due to its versatility and properties, such as error bound and optimality of the reduction basis it provides. In [4] for instance, a PROM was developed for contact problems by performing a number of full order simulations sampling the parameter space, then taking the POD of each and clustering all the resulting bases. Using a computationally cheap error estimator, this process is repeated in order to refine the approximation. A lot of other strategies involving interpolation of such full order simulations and/or POD bases also exist. In [70] the data coming from the full order simulations is manipulated using a two-level radial basis function interpolation method to obtain hyper-surfaces in the parameters, while in [22] the interpolation step is carried out using neural networks to compute the interpolation coefficients. All the aforementioned strategies belong to the family of (non-intrusive) data-driven methods, and they all demand a huge upfront cost to be paid to develop the ROM itself. Nevertheless, these approaches might probably be the only viable ways to deal with complex problems such as the ones in fluid-mechanics, which often provides the benchmarks for these algorithms.

In the context of geometric nonlinearities in solid mechanics, however, we showed that under some circumstances the nonlinear internal forces take specific shapes (e. g. third-order polynomials), and thus some information from the model can be exploited for the construction of the reduction basis (e. g. MDs) without the

need of full simulations. In [43] for instance, such information is used to construct a PROM to describe *shape defects* of a structure in a parametric fashion and relying solely on the model of the system. Even if probably model-driven approaches seem to go against the trend of a consistent share of the scientific community that gradually, but steadily, moves towards the machine-learning area, we deem such strategies still worth of investigation not only for the possible advantages over other techniques, but also for their engineering value in terms of interpretability and ease of use.

The parametrization could also be intended as time-dependent, as done for instance in [9] and [63] for the reduction of models of meshing gears. Here, a reduction basis which is used to reproduce the contact behavior between gears teeth is made function of a time-dependent load position parameter, and appended to a constant basis that spans the global behavior of the system. The local, parametrized modes can be either residual attachment modes [63], or static solutions at specific configuration [9]. The time variation of the reduction basis is then generating additional, state dependent terms in the ROM, which are shown to be important for stress recovery. In all this contributions, the time variation of the parameter is known a priori. Likewise, a prescribed time-dependent law could parameterize the equilibrium of a given system. This is the case of thermo-mechanical systems, where the temperature field is typically determined by solving the heat equation, and then applied to the mechanical problem. Typically, there is a large characteristic time scale difference between the thermal and mechanical problem, being the latter characterized by much faster dynamics. In this case indeed, the slow variation of the applied thermal field justifies a *parametric equilibrium* and a *parametric reduction basis*, which can be conveniently interpolated online; see [33]. For the same reason, state dependent terms that are otherwise present (see again [9, 63]) can be neglected.

4.7 Conclusions

We briefly overviewed the most popular methods for model order reduction of linear and geometrical nonlinear mechanical systems. The approaches here considered are *system-driven*, meaning with this that the reduced order model is built from quantities that can be derived directly from the discretized equations governing the high fidelity model. This is in contrast with *data-driven* techniques, which in fact build the reduced order model from solutions of the high fidelity model. The main appeal of system-driven method is their independence from computationally expensive solutions and their exploitation of the system characteristics.

In structural dynamics applications, the *modal displacement method* has become a standard already for decades. The full order solution is approximated by a combination of a few carefully selected eigenmodes, which are also used to span a space

where the system is projected. The resulting reduced order model consists of an uncoupled system of modal equations which can be easily solved. Enhancements to the modal displacement method aim at retrieving the contribution of discarded vectors by quasi-static corrections.

The same strategy, namely modal expansion and projection, can be employed also for nonlinear systems. Here, we restricted to systems with geometric nonlinearities, which are typically modeled by smooth nonlinear functions of displacements. Systems affected by geometric nonlinearities usually feature a *slow* dominant behavior, which is complemented by a *fast* dynamics. The slow dynamics—typically featuring out-of-plane motion for the case of thin-walled structures—can still be developed by a set of dominant vibration modes selected by the modal displacement method. The fast dynamics is usually *enslaved* to the slow one, as it is given by the bending–stretching coupling due to the geometric nonlinearity. In fact, for flat systems where the in-plane and out-of-plane dynamics can be clearly separated, a nonlinear mapping between out-of-plane (slow) and in-plane (fast) dynamics—usually referred to as *static condensation*—can be established by neglecting the in-plane inertial terms.

For more geometrically complex systems, this separation of the degrees of freedom might be hard or impossible to perform. In this case, one can of course rely on expansion and projection onto a subspace that is able to reproduce the nonlinear coupling effects. Two methods to form such basis gained popularity over the past two decades, namely *modal derivatives* and *dual modes*. Both methods find the most relevant in-plane contribution to best span the nonlinear behavior. While the dual modes method is inherently *non-intrusive*, as it requires several nonlinear static solutions of the high fidelity model, the modal derivative strategy can be applied also *intrusively* if the directional derivative of the tangent stiffness matrix along the dominant linear modes could be calculated.

Indeed, the wording *non-intrusive* and *intrusive* refer to two distinct classes of reduction methods. The distinction lies on whether or not it is possible to access elemental functions to compute the projected reduced order terms. In non-intrusive methods, the structure of these terms is first postulated and then identified, typically by probing the system with sufficient nonlinear static cases, where either displacements or forces are imposed on the high fidelity model. We discussed the methods of *enforced displacements* and *enhanced enforced displacements*, which identify the coefficients of the projected nonlinear forces by statically applying displacements determined by combinations of the modes used for reduction. The number of such evaluations is $\mathcal{O}(m^3)$ and $\mathcal{O}(m^2)$ for the enforced displacement and the enhanced enforced displacement method, respectively. The latter gain is due to the assumed availability of the tangent stiffness matrix from the high fidelity model. The method of *implicit condensation and expansion* differs from the former two as it relies only on dominant vibration modes—and not on either dual modes or modal derivatives—to construct the reduced order model. A recovery of the full displacement field is obtained through an expansion step that assumes a quadratic dependency of the in-plane modes to the dominant

vibration modes. In intrusive methods, on the contrary, the reduced nonlinear forces are computed exactly by evaluating and projecting at element level. This is efficient if the resulting reduced terms are functions of modal coordinates only—i. e. no access to the mesh is required any longer, once the reduced order model is constructed. This is the case indeed of polynomial terms, resulting from lagrangian formulation and elastic material of continuum elements, or plates modeled with the von Karman kinematic assumptions. In more complex cases, the evaluation at element level is unavoidable, and the speed-up associated to a reduced order model drops significantly. In these cases, one can resort to *hyper-reduction*, which is extensively treated in [8, Chapter 5]. Essentially, hyper-reduction scales the computation of the reduced terms with the size of the reduction basis and not with the one of the high fidelity model, and thus delivers huge speed-ups. Here, we outlined the Discrete Empirical Interpolation Method (DEIM) and the Energy Conserving Sampling and Weighting (ECSW) method. Both methods rely on *training forces*, which typically come from the Proper Orthogonal Decomposition of high fidelity solutions. As such, they could be classified as data-driven methods and would therefore not belong to this chapter. However, recent trends target the construction of such training sets without recurring to expensive full solution. For instance, in the case of geometric nonlinearities, one can *lift* computationally linear modal solutions on a quadratic manifold centered around the equilibrium, and whose curvature is provided by modal derivatives. The resulting displacements generate forces which can be used for training of the hyper-reduction.

The slow–fast dynamics dichotomy of the behavior of nonlinear systems in fact favors reduction methods based on nonlinear manifolds, rather than projection on linear subspaces. In other words, the solution is assumed to evolve on curved, rather than flat, manifolds, which can be constructed via asymptotic expansions around the equilibrium point. Along this direction, the recently proposed *spectral submanifold* is defined as the smoothest nonlinear continuation of a modal subspace at the equilibrium point. Likewise, we briefly discussed the reduction on *quadratic manifolds* constructed with vibration modes and modal derivatives. As the manifold can be seen as a nonlinear constraint to the solution, a configuration-dependent mass matrix and convective generalized forces are generated by this reduction method. Reduction through nonlinear manifolds exacerbates the issue of efficient computation of reduced nonlinear terms. In the case of polynomial nonlinearities for the high fidelity model, the resulting reduced tensors are of higher order as the ones corresponding to linear projection. This could quickly exhaust memory resources and slow down the time integration of the reduced system significantly. Recent contributions are tackling this problem by applying for instance hyper-reduction.

Lastly, we briefly discussed parametric model order reduction. In this context, the general trend is to rely on data-driven methods, usually based on POD, to construct the parametrized reduced order model. In general this could be done by either forming a large reduction basis that spans the parameter space of interest, or by constructing local reduced order models and interpolate across the parameter space. In either case,

a greedy procedure often drives the generation of the parametric reduced order model. However, for the case of geometric nonlinearity, it is possible to exploit the structure of the governing equations and devise a reduction basis containing *sensitivities* of modes with respect to parameters—for instance, in the case of shape imperfections. This procedure is completely system-driven and it is shown to span the parameter space in the neighborhood of the nominal parameter values.

Bibliography

- [1] M. S. Allen, D. Rixen, M. van der Seijs, P. Tiso, T. Abrahamsson, and R. L. Mayes. *Substructuring in Engineering Dynamics: Emerging Numerical and Experimental Techniques*. CISM International Centre for Mechanical Sciences (Courses and Lectures). Springer, Cham, 2020.
- [2] M. Amabili, A. Sarkar, and M. P. Paidoussis. Reduced-order models for nonlinear vibrations of cylindrical shells via the proper orthogonal decomposition method. *J. Fluids Struct.*, 18(2):227–250, 2003. Axial and Internal Flow Fluid-Structure Interactions.
- [3] D. Amsallem, J. Cortial, K. Carlberg, and C. Farhat. A method for interpolating on manifolds structural dynamics reduced-order models. *Int. J. Numer. Methods Eng.*, 80(9):1241–1258, 2009.
- [4] M. Balajewicz, D. Amsallem, and C. Farhat. Projection-based model reduction for contact problems. *Int. J. Numer. Methods Eng.*, 106:644–663, 2015.
- [5] M. Balajewicz, D. Amsallem, and C. Farhat. Projection-based model reduction for contact problems. *Int. J. Numer. Methods Eng.*, 106(8):644–663, 2016. nme.5135.
- [6] P. Benner, S. Gugercin, and K. Willcox. A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems. *SIAM Rev.*, 57(4):483–531, 2015.
- [7] P. Benner, S. Griwet-Talocia, A. Quarteroni, G. Rozza, W. H. A. Schilders and L. M. Silveira. *Model Order Reduction. Volume 1: System- and Data-Driven Methods and Algorithms*. De Gruyter, Berlin, 2020.
- [8] P. Benner, S. Griwet-Talocia, A. Quarteroni, G. Rozza, W. H. A. Schilders and L. M. Silveira. *Model Order Reduction. Volume 2: Snapshot-Based Methods and Algorithms*. De Gruyter, Berlin, 2020.
- [9] B. Blockmans, T. Tamarozzi, F. Naets, and W. Desmet. A nonlinear parametric model reduction method for efficient gear contact simulations. *Int. J. Numer. Methods Eng.*, 102(5):1162–1191, 2015.
- [10] J. Bonet, A. J. Gil, and R. D. Wood. *Nonlinear Solid Mechanics for Finite Element Analysis: Statics*. Cambridge University Press, 2016.
- [11] S. Chaturantabut and D. C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM J. Sci. Comput.*, 32(5):2737–2764, 2010.
- [12] R. R. Craig and M. C. C. Bamton. Coupling of substructures for dynamic analyses. *AIAA J.*, 6(7):1313–1319, 1968.
- [13] R. de Borst, M. A. Crisfield, J. J. C. Remmers, and C. V. Verhoosel. *Non-Linear Finite Element Analysis of Solids and Structures*, 2nd edition, 2012.
- [14] J. M. Dickens, J. M. Nakagawa, and M. J. Wittbrodt. A critique of mode acceleration and modal truncation augmentation methods for modal response analysis. *Comput. Struct.*, 62(6):985–998, 1997.
- [15] J. M. Dickens and K. V. Pool. Modal truncation vectors and periodic time domain analysis applied to a cyclic symmetry structure. *Comput. Struct.*, 45(4):685–696, 1992.

- [16] C. Farhat, P. Avery, T. Chapman, and J. Cortial. Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency. *Int. J. Numer. Methods Eng.*, 98(9):625–662, 2014.
- [17] B. Fröhlich, J. Gade, F. Geiger, M. Bischoff, and P. Eberhard. Geometric element parameterization and parametric model order reduction in finite element based shape optimization. *Comput. Mech.*, 2018.
- [18] M. Geradin and D. J. Rixen. *Mechanical Vibrations: Theory and Application to Structural Dynamics*. Wiley, 2015.
- [19] A. Givois, A. Grolet, O. Thomas, and J.-F. Deü. On the frequency response computation of geometrically nonlinear flat structures using reduced-order finite element models. *Nonlinear Dyn.*, 97(2):1747–1781, 2019.
- [20] R. W. Gordon and J. J. Hollkamp. Reduced-Order Models for Acoustic Response Prediction. Technical Report July, AFRL-RB-WP-TR-2011-3040, 2011.
- [21] G. Haller and S. Ponsioen. Nonlinear normal modes and spectral submanifolds: existence, uniqueness and use in model reduction. *Nonlinear Dyn.*, 86(3):1493–1534, 2016.
- [22] J. S. Hesthaven and S. Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *J. Comput. Phys.*, 363:55–78, 2018.
- [23] L. Hogben. *Handbook of Linear Algebra*. Chapman & Hall/CRC, 2007.
- [24] J. J. Hollkamp and R. W. Gordon. Reduced-order models for nonlinear response prediction: Implicit condensation and expansion. *J. Sound Vib.*, 318(4–5):1139–1153, 2008.
- [25] W. C. Hurty. Dynamic analysis of structural systems using component modes. *AIAA J.*, 3(4):678–685, 1965.
- [26] S. R. Idelsohn and A. Cardona. A load-dependent basis for reduced nonlinear structural dynamics. In A. K. Noor and R. J. Hayduk, editors, *Advances and Trends in Structures and Dynamics*, pages 203–210. Pergamon, 1985.
- [27] S. R. Idelsohn and A. Cardona. A reduction method for nonlinear structural dynamic analysis. *Comput. Methods Appl. Mech. Eng.*, 49(3):253–279, 1985.
- [28] S. Jain, T. Breunung, and G. Haller. Fast computation of steady-state response for high-degree-of-freedom nonlinear systems. *Nonlinear Dyn.*, 97(1):313–341, 2019.
- [29] S. Jain and P. Tiso. Simulation-Free Hyper-Reduction for Geometrically Nonlinear Structural Dynamics: A Quadratic Manifold Lifting Approach. *J. Comput. Nonlinear Dyn.*, 13(7):071003, 2018.
- [30] S. Jain, P. Tiso, and G. Haller. Exact nonlinear model reduction for a von kármán beam: Slow-fast decomposition and spectral submanifolds. *J. Sound Vib.*, 423:195–211, 2018.
- [31] S. Jain, P. Tiso, J. B. Rutzmoser, and D. J. Rixen. A quadratic manifold for model order reduction of nonlinear structural dynamics. *Comput. Struct.*, 188:80–94, 2017.
- [32] S. Jain and P. Tiso. Hyper-Reduction Over Nonlinear Manifolds for Large Nonlinear Mechanical Systems. *J. Comput. Nonlinear Dyn.*, 14(8):081008, 2019.
- [33] S. Jain and P. Tiso. Model order reduction for temperature-dependent nonlinear mechanical systems: A multiple scales approach. *J. Sound Vib.*, 465:115022, 2020.
- [34] M. Karamooz Mahdiabadi. Nonlinear Model Order Reduction and Substructuring for Structural Dynamics Analysis: Non-intrusive Methods. Phd thesis, Technical University of Munich 2019.
- [35] P. Kerfriden, P. Gosselet, S. Adhikari, and S. P. A. Bordas. Bridging proper orthogonal decomposition methods and augmented newton–krylov algorithms: An adaptive model order reduction for highly nonlinear mechanical problems. *Comput. Methods Appl. Mech. Eng.*, 200(5):850–866, 2011.
- [36] G. Kerschen, J. Golinval, A. F. Vakakis, and L. A. Bergman. The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: An overview. *Nonlinear Dyn.*, 41(1):147–169, 2005.

- [37] G. Kerschen, M. Peeters, J. C. Golinval, and A. F. Vakakis. Nonlinear normal modes, part I: A useful framework for the structural dynamicist. *Mech. Syst. Signal Process.*, 23(1):170–194, 2009. Special Issue: Non-linear Structural Dynamics.
- [38] K. Kim, A. G. Radu, X. Q. Wang, and M. P. Mignolet. Nonlinear reduced order modeling of isotropic and functionally graded plates. *Int. J. Non-Linear Mech.*, 49(2013):100–110, 2013.
- [39] D. Krattiger, L. Wu, M. Zacharczuk, M. Buck, R. J. Kuether, M. S. Allen, P. Tiso, and M. R. W. Brake. Interface reduction for hurty/craig-bampton substructured models: Review and improvements. *Mech. Syst. Signal Process.*, 114:579–603, 2019.
- [40] R. J. Kuether, B. J. Deaner, J. J. Hollkamp, and M. S. Allen. Evaluation of geometrically nonlinear reduced-order models with nonlinear normal modes. *AIAA J.*, 53(11):3273–3285, 2015.
- [41] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for parabolic problems. *Numer. Math.*, 90(1):117–148, 2001.
- [42] R. H. MacNeal. A hybrid method of component mode synthesis. *Comput. Struct.*, 1(4):581–601, 1971. Special Issue on Structural Dynamics.
- [43] J. Marconi, P. Tiso, and F. Braghin. A nonlinear reduced order model with parametrized shape defects. *Comput. Methods Appl. Mech. Eng.*, 360:112785, 2020.
- [44] H. G. Matthies and M. Meyer. Nonlinear galerkin methods for the model reduction of nonlinear dynamical systems. *Comput. Struct.*, 81(12):1277–1286, 2003. Advanced Computational Models and Techniques in Dynamics.
- [45] M. I. McEwan, J. R. Wright, J. E. Cooper, and A. Y. T. Leung. A combined modal/finite element analysis technique for the dynamic response of a non-linear beam to harmonic excitation. *J. Sound Vib.*, 243(4):601–624, 2001.
- [46] M. I. McEwan, J. R. Wright, J. E. Cooper, and A. Y. T. Leung. A finite element/modal technique for nonlinear plate and stiffened panel response prediction. In *Collection of Technical Papers – AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, volume 5, pages 3061–3070, 2001.
- [47] M. P. Mignolet, A. Przekop, S. A. Rizzi, and S. M. Spottswood. A review of indirect/non-intrusive reduced order modeling of nonlinear geometric structures. *J. Sound Vib.*, 332(10):2437–2460, 2013.
- [48] A. A. Muravyov and S. A. Rizzi. Determination of nonlinear stiffness with application to random vibration of geometrically nonlinear structures. *Comput. Struct.*, 81(15):1513–1523, 2003.
- [49] M. Nash. Nonlinear Structure Dynamics by Finite Element Modal Synthesis. PhD thesis, Imperial College, London 1977.
- [50] K. O’Hanlon and M. D. Plumbley. Learning overcomplete dictionaries with l0-sparse non-negative matrix factorisation. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 977–980, 2013.
- [51] M. Peeters, R. Vigné, G. Sérandour, G. Kerschen, and J.-C. Golinval. Nonlinear normal modes, part ii: Toward a practical computation using numerical continuation techniques. *Mech. Syst. Signal Process.*, 23(1):195–216, 2009. Special Issue: Non-linear Structural Dynamics.
- [52] R. Perez, X. Q. Wang, and M. P. Mignolet. Nonintrusive Structural Dynamic Reduced Order Modeling for Large Deformations: Enhancements for Complex Structures. *J. Comput. Nonlinear Dyn.*, 9:031008, 2014.
- [53] S. Ponsioen and G. Haller. Exact model reduction and fast forced response calculation in high-dimensional nonlinear mechanical systems, 2019. arXiv:1912.11399v1.
- [54] J. N. Reddy. *Introduction to the Finite Element Method*, 3rd edition. McGraw-Hill Education, New York, 2006.
- [55] R. M. Rosenberg. Normal Modes of Nonlinear Dual-Mode Systems. *J. Appl. Mech.*, 27(2):263–268, 1960.
- [56] S. Rubin. Improved component-mode representation for structural dynamic analysis. *AIAA J.*, 13(8):995–1006, 1975.

- [57] J. B. Rutzmoser and D. J. Rixen. A lean and efficient snapshot generation technique for the hyper-reduction of nonlinear structural dynamics. *Comput. Methods Appl. Mech. Eng.*, 325:330–349, 2017.
- [58] J. B. Rutzmoser, D. J. Rixen, P. Tiso, and S. Jain. Generalization of quadratic manifolds for reduced order modeling of nonlinear structural dynamics. *Comput. Struct.*, 192:196–209, 2017.
- [59] S. W. Shaw and C. Pierre. Non-linear normal modes and invariant manifolds. *J. Sound Vib.*, 150(1):170–173, 1991.
- [60] S. W. Shaw and C. Pierre. Normal modes of vibration for non-linear continuous systems. *J. Sound Vib.*, 169(3):319–347, 1994.
- [61] P. M. A. Slaats, J. de Jongh, and A. A. H. J. Sauren. Model reduction tools for nonlinear structural dynamics. *Comput. Struct.*, 54(6):1155–1171, 1995.
- [62] C. S. M. Sombroek, P. Tiso, L. Renson, and G. Kerschen. Numerical computation of nonlinear normal modes in a modal derivative subspace. *Comput. Struct.*, 195:34–46, 2018.
- [63] T. Tamarozzi, G. H. K. Heirman, and W. Desmet. An on-line time dependent parametric model order reduction scheme with focus on dynamic stress recovery. *Comput. Methods Appl. Mech. Eng.*, 268:336–358, 2014.
- [64] P. Tiso. Optimal second order reduction basis selection for nonlinear transient analysis. In T. Proulx, editor, *Modal Analysis Topics*, volume 3, pages 27–39. Springer New York, New York, NY, 2011.
- [65] P. Tiso, R. Dedden, and D. J. Rixen. A modified discrete empirical interpolation method for reducing non-linear structural finite element models. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Volume 7B: 9th International Conference on Multibody Systems, Nonlinear Dynamics, and Control*, 2013.
- [66] C. Touzé, M. Vidrascu, and D. Chapelle. Direct finite element computation of non-linear modal coupling coefficients for reduced-order shell models. *Comput. Mech.*, 54(2):567–580, 2014.
- [67] O. Weeger, U. Wever, and B. Simeon. On the use of modal derivatives for nonlinear model order reduction. *Int. J. Numer. Methods Eng.*, 108(13):1579–1602, 2016.
- [68] L. Wu and P. Tiso. Nonlinear model order reduction for flexible multibody dynamics: a modal derivatives approach. *Multibody Syst. Dyn.*, 36(4):405–425, 2016.
- [69] L. Wu, P. Tiso, K. Tatsis, E. Chatzi, and A. van Keulen. A modal derivatives enhanced rubin substructuring method for geometrically nonlinear multibody systems. *Multibody Syst. Dyn.*, 45(1):57–85, 2019.
- [70] D. Xiao, F. Fang, C. C. Pain, and I. M. Navon. A parameterized non-intrusive reduced order model and error analysis for general time-dependent nonlinear partial differential equations and its applications. *Comput. Methods Appl. Mech. Eng.*, 317:868–889, 2017.

5 Post-processing methods for passivity enforcement

Abstract: Many physical systems are passive (or dissipative): they are unable to generate energy on their own, but they can store energy in some form while exchanging power with the surrounding environment. This chapter describes the most prominent approaches for ensuring that Reduced Order Models are passive, so that their mathematical representation satisfies an appropriate dissipativity condition. The main focus is on Linear and Time-Invariant (LTI) systems in state-space form. Different conditions for testing passivity of a given LTI model are discussed, including Linear Matrix Inequalities (LMIs), Frequency-Domain Inequalities, and spectral conditions on associated Hamiltonian matrices. Then we describe common approaches for perturbing a given non-passive system to enforce its passivity. Various examples from electronic applications are used to demonstrate both theory and algorithm performance.

Keywords: passivity, dissipativity, positive real lemma, bounded real lemma, Hamiltonian matrices, state-space systems, descriptor systems, eigenvalue perturbation

5.1 Introduction and motivations

Let us consider the problem of designing a complete electronic product, such as a smartphone or a high-end computing server. The complexity of such a system is overwhelming: a single microprocessor might include several billions transistors, and this is just one component. All components are tightly interconnected to exchange signals and power: they interact both through electrical connections as well as (unwanted) electromagnetic couplings, which are inevitable due to the close proximity of components in tightly integrated systems. A proper design flow must ensure that all signals behave as expected during real operation, which requires accounting for all interactions between components and subsystems. A first-pass design can only be achieved by extensive numerical simulation at the system level, in order to verify full compliance with specifications.

All of us would agree that a direct, brute-force simulation of the complete system is totally unrealistic. This is why common engineering practices partition a given complex system into several simpler subsystems, which are modeled independently. All models are then interconnected to obtain a system description that is amenable for numerical simulation. These individual models are very often obtained through

Stefano Grivet-Talocia, Dept. Electronics and Telecommunications, Politecnico di Torino, Turin, Italy
Luis Miguel Silveira, INESC-ID/DEEC, IST Técnico Lisboa, Universidade de Lisboa, Lisbon, Portugal

some Model Order Reduction scheme applied to some initial device-level characterization.

Suppose now that one of the above individual models represents some signal or power interconnect network. Such interconnect structure is intended to feed signal and power supply to all elements of the system, in the form of electrical current flowing through metal wires. The interconnect network is unable to generate energy on its own, but rather redistributes the energy that it receives from its input signals to its output signals. It may store energy through electric and magnetic field densities in the physical space surrounding the interconnect, and it may dissipate energy as heat through metal and dielectric losses, but no more energy can be supplied to the environment than the amount of energy previously stored. Such a system is called dissipative (or passive). The concept of dissipativity naturally arises from energy conservation principles and is therefore ubiquitous in several engineering fields.

A (reduced order) model of the interconnect must respect such property: the simulation model must not be able to release more energy than previously stored. This requirement is not just for self-consistency with fundamental physical principles, but for a very practical reason: a non-passive model may trigger instabilities during system-level simulation. An example is provided in Figure 5.1, which compares the voltage received by a state of the art (at the time of writing) smartphone through a high-speed interconnect, computed using a passive vs. non-passive model connected to various other system parts, including drivers and receiver circuits that send and receive the signals. The non-passive model triggers a resonance, by injecting a continuous flow of power that is responsible for the instability. Conversely, the passive model provides a well-behaved bounded response.

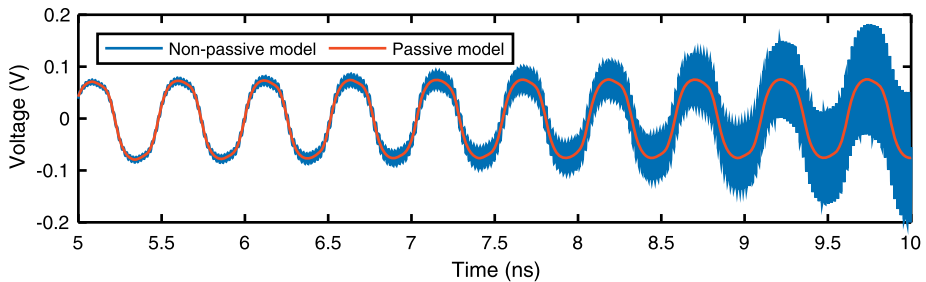


Figure 5.1: Comparison between the responses of passive (thick line) and non-passive (thin line) models of a high-speed smartphone interconnect link.

A fundamental result states that the interconnection of passive subsystems leads to a passive system; see e. g. [74]. Therefore, a guarantee of passivity for all individual

models for which this requirement is adequate¹ is also a guarantee that a model-based system-level simulation will run smoothly. This is a relevant problem not only for electronic applications, but for several applied engineering fields such as, e. g., mechanics or fluid dynamics. Energy conservation or dissipation properties must be preserved in the simulation models.

Figure 5.1 illustrates in a clear manner that any modeling procedure used for analysis of the dynamics of dissipative physical systems should ensure that the resulting model or reduced order model is dissipative. There exist MOR algorithms that are able to preserve dissipativity if applied to an original large-scale dissipative model. Examples are the PRIMA algorithm [63, 67] (see also [6, Chapter 4]) or the PR-TBR algorithm [19, 62, 64, 65, 68] (see also [5, Chapter 2]). Unfortunately, for a variety of reasons, possibly including efficiency considerations, such schemes are not always applicable, and one has to resort to one of the many reduced order modeling techniques that are not able to preserve or enforce dissipativity. Therefore, it is often necessary to perform a-posteriori checks and possibly implement a post-processing procedure that enforces model passivity.

In this Chapter, we review the various forms in which the passivity conditions of a model can be stated. The particular class of systems that we focus on is defined in Section 5.2, although generalizations are discussed in Section 5.6. Different forms of passivity conditions will lead to corresponding different numerical schemes for their verification, discussed in Section 5.3. We then present in Section 5.5 a selection of methods for passivity enforcement, mainly cast as perturbation approaches that, starting from the original non-passive model, update its coefficients in order to achieve passivity. Model accuracy is retained by minimizing the perturbation amount in some norm, as discussed in Section 5.4.

The style of this chapter is informal, with main results being stated with some essential derivation, but without a formal proof. Emphasis is on the practical aspects of the various formulations, which lead to algorithms presented in pseudocode form. Pointers to the relevant literature are provided for additional insight.

5.2 Passivity conditions

In order to keep this chapter self-contained, our discussion is based on the special class of Linear Time-Invariant (LTI), finite-dimensional systems in regular state-space form

$$\mathcal{S} : \begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \\ y(t) = Cx(t) + Du(t), \end{cases} \quad (5.1)$$

¹ Note that not all components are passive: for instance, signal or power sources or amplifier circuits do not and must not be expected to behave as passive elements.

where t denotes time, vectors $u \in \mathcal{U} \subseteq \mathbb{R}^M$ and $y \in \mathcal{Y} \subseteq \mathbb{R}^M$ collect the system inputs and outputs, respectively, and $x \in \mathcal{X} \subseteq \mathbb{R}^N$ is the state vector, with \dot{x} denoting its time derivative. The $M \times M$ transfer function of the system is

$$H(s) = C(sI - A)^{-1}B + D, \quad (5.2)$$

where s is the Laplace variable. We start by assuming the system to be asymptotically stable, with all eigenvalues of A , i. e., the poles of $H(s)$, having a strictly negative real part.

The above assumptions may seem overly restrictive, but most common macro-modeling schemes that are widespread in electronic applications such, e. g. the Vector Fitting algorithm [5, Chapter 8], produce reduced order models in this form. Generalizations will be discussed in Section 5.6.

5.2.1 Dissipative systems

The system \mathcal{S} in (5.1) is *dissipative* [15, 74, 88] with respect to the supply function $s : \mathcal{U} \times \mathcal{Y} \mapsto \mathbb{R}$ if there exists a *storage function* $V : \mathcal{X} \mapsto \mathbb{R}$ such that

$$V(x(t_1)) \leq V(x(t_0)) + \int_{t_0}^{t_1} s(u(t), y(t)) dt \quad (5.3)$$

for all $t_0 \leq t_1$ and all input, state and output signals u, x, y that satisfy the state-space equations (5.1). In the above definition, V represents the internal energy that the system is able to store, and s is the power flow exchanged by the system with the environment. Thus, for a dissipative system the increase in the internal energy that the system undergoes during any time interval (t_0, t_1) cannot exceed the cumulative amount of energy received from the environment, expressed as a time integral of the input power flow. If the storage function is differentiable, the *dissipation inequality* (5.3) can also be cast in the equivalent form

$$\frac{dV(x(t))}{dt} \leq s(u(t), y(t)). \quad (5.4)$$

As a typical example, one may consider an electric RLC circuit made of an arbitrary number of arbitrarily connected resistors $R_k > 0$, inductors $L_k > 0$ and capacitors $C_k > 0$, which interacts with the environment through M ports. Each port defines an input, e. g., the port voltage v_j , with the port current i_j acting as the corresponding output (this representation is called *admittance*). For this example, the state vector comprises all capacitor voltages v_{Ck} and inductor currents i_{Lk} , so that the energy storage function is defined as $V := \frac{1}{2}(\sum_k C_k v_{Ck}^2 + \sum_k L_k i_{Lk}^2)$. The electric power entering the circuit at the j th port is $v_j i_j$, so that the power supply function reads $s(u, y) = u^T y = \sum_j v_j i_j$.

By Tellegen's (power conservation) theorem [4, 24], we have

$$\begin{aligned} s(u, y) &= \sum_j v_j i_j = \sum_k v_{Ck} i_{Ck} + \sum_k v_{Lk} i_{Lk} + \sum_k R_k i_{Rk}^2 \\ &= \frac{dV}{dt} + \sum_k R_k i_{Rk}^2 \geq \frac{dV}{dt} \end{aligned} \quad (5.5)$$

where we used the definition of capacitor currents $i_{Ck} = C_k \frac{dv_{Ck}}{dt}$, inductor voltages $v_{Lk} = L_k \frac{di_{Lk}}{dt}$, and where i_{Rk} are the resistor currents. So, we see that any RLC circuit with positive elements is dissipative.

The system S in (5.1) is called *strictly dissipative* [74, 88] with respect to the supply function s if the stronger condition

$$V(x(t_1)) \leq V(x(t_0)) + \int_{t_0}^{t_1} s(u(t), y(t)) dt - \varepsilon^2 \int_{t_0}^{t_1} \|u(t)\|^2 dt \quad (5.6)$$

holds for some $\varepsilon > 0$ instead of (5.3), which is thus satisfied with a strict inequality.

The following three subsections provide different equivalent passivity conditions that are applicable to linear state-space systems in the form (5.1).

5.2.1.1 Linear matrix inequalities

Building on the above example, we consider for the general system (5.1) a quadratic² storage function $V(x) = \frac{1}{2}x^T P x$ associated to a symmetric and positive definite matrix $P = P^T > 0$. Also, we adopt the same supply function³

$$s(u, y) = u^T y = y^T u = \frac{1}{2}(u^T y + y^T u). \quad (5.7)$$

Imposing the dissipation inequality in differential form (5.4) leads to

$$\frac{d}{dt} \left\{ \frac{1}{2} x^T P x \right\} = \frac{1}{2} (\dot{x}^T P x + x^T P \dot{x}) \leq \frac{1}{2} (u^T y + y^T u) \quad (5.8)$$

and using (5.1) to eliminate \dot{x} and y , we obtain the following condition:

$$\begin{pmatrix} x \\ u \end{pmatrix}^T \begin{pmatrix} A^T P + P A & P B - C^T \\ B^T P - C & -D - D^T \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} \leq 0, \quad P = P^T > 0, \quad (5.9)$$

² It is well known [89] that, if a storage function V satisfying (5.3) for system (5.1) exists, it can be found as a positive definite quadratic form.

³ In many physical systems power is expressed as the product of relevant variables, such as voltage–current in electrical circuits, pressure–flow in hydraulic systems, and force–velocity in mechanical systems.

which provides the passivity condition to the state-space system (5.1). This condition leads to the well-known *Positive Real Lemma (PRL)*, which is a particular case of the *Kalman–Yakubovich–Popov (KYP) lemma* [51, 66, 92] (see also [2, 56, 74]), which states that the state-space system (5.1) is passive if and only if

$$\exists P = P^T > 0 : \begin{pmatrix} A^T P + PA & PB - C^T \\ B^T P - C & -D - D^T \end{pmatrix} \leq 0. \quad (5.10)$$

This class of conditions is generally known as *Linear Matrix Inequalities (LMIs)* [2, 13, 14, 87]. For a strictly passive (dissipative) system (5.10) holds with a strict inequality.

5.2.1.2 Frequency-domain inequalities

An equivalent condition for passivity characterization is provided by a frequency-domain inequality. It is well known [2, 81, 90] that the transfer function of a general passive system must be *Positive Real (PR)*, i. e., the following three conditions must hold:

1. $H(s)$ must be regular in the open right half complex plane $\Re\{s\} > 0$;
2. $H(s^*) = H^*(s)$, where $*$ denotes the complex conjugate;
3. $\Psi(s) = H(s) + H^T(-s) \geq 0$ for $\Re\{s\} > 0$.

Condition 1 is directly related to the stability of $H(s)$, which is here assumed a priori; condition 2 implies that the impulse response of the system is real-valued; condition 3 completes passivity characterization through a Frequency-Domain Inequality. The connection between the PR conditions and the PRL/KYP Lemma are well-developed and proved in [2].

Since by our assumption all the poles of $H(s)$ are strictly stable, and since the adopted state-space realization is real-valued, both conditions 1 and 2 are automatically satisfied, whereas condition 3 can be restricted to the imaginary axis $s = j\omega$ by the minimum principle of harmonic functions [69], showing that

$$\Psi(j\omega) = H(j\omega) + H^H(j\omega) \geq 0 \quad \forall \omega \in \mathbb{R} \quad (5.11)$$

where H denotes Hermitian transpose and j is the imaginary unit. Continuing on the same RLC circuit example above, the latter condition states that the input admittance (matrix) of the circuit block must be nonnegative (Hermitian) definite, which in the scalar case $M = 1$ reduces to the requirement that the real part of the input admittance or impedance of any passive one-port element must be nonnegative at any frequency. Condition (5.11) can be further conveniently rewritten as

$$\lambda_i \geq 0, \quad \forall \lambda_i \in \lambda(\Psi(j\omega)), \quad \forall \omega \in \mathbb{R}, \quad (5.12)$$

where $\lambda(\cdot)$ denotes the set of all eigenvalues of its matrix argument. Nonnegativity can thus be tested for all individual frequency-dependent eigenvalue trajectories $\lambda_i(j\omega)$ of $\Psi(j\omega)$, for $1 \leq i \leq M$. Inequalities (5.11) and (5.12) are strict for $\omega \in \mathbb{R} \cup \{\infty\}$ in the case of strictly passive systems.

5.2.1.3 Hamiltonian matrices

There is a third class of conditions that can be used to characterize a passive system, based on the so-called Hamiltonian matrix associated to (5.1). We introduce this matrix by finding the set of spectral zeros of $\Psi(s)$. Let us assume that $\Psi(s_0)v = 0$ for some vector $v \neq 0$, with $s_0 \in \mathbb{C}$. Using (5.2) we have

$$[C(s_0I - A)^{-1}B + D + B^T(-s_0I - A^T)^{-1}C^T + D^T]v = 0. \quad (5.13)$$

Let us define

$$\begin{aligned} r &= (s_0I - A)^{-1}Bv & \rightarrow & \quad s_0r = Ar + Bv, \\ q &= (-s_0I - A^T)^{-1}C^Tv & \rightarrow & \quad s_0q = -A^Tq - C^Tv. \end{aligned} \quad (5.14)$$

Substituting in (5.13) and solving for v under the assumption that $W_0 = D + D^T$ is nonsingular (see Section 5.6 for a generalization) leads to

$$v = -W_0^{-1}(Cr + B^Tq), \quad (5.15)$$

which, inserted in (5.14), again leads to

$$\underbrace{\begin{pmatrix} A - BW_0^{-1}C & -BW_0^{-1}B^T \\ C^TW_0^{-1}C & -A^T + C^TW_0^{-1}B^T \end{pmatrix}}_{\mathcal{M}} \begin{pmatrix} r \\ q \end{pmatrix} = s_0 \begin{pmatrix} r \\ q \end{pmatrix}. \quad (5.16)$$

The matrix \mathcal{M} in (5.16) has (real) Hamiltonian structure, since

$$(J\mathcal{M})^T = J\mathcal{M} \quad \text{where } J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}. \quad (5.17)$$

It is easily shown that the corresponding eigenspectrum is symmetric with respect to both real and imaginary axis. Condition (5.16) states that the spectral zeros of $\Psi(s)$ are the eigenvalues of the Hamiltonian matrix \mathcal{M} . If one of such eigenvalues is purely imaginary $s_0 = j\omega_0$, then we may have a violation of the frequency-domain inequality (5.12). In fact, if (5.16) holds for some purely imaginary $s_0 = j\omega_0$, then $\Psi(j\omega)$ is singular at ω_0 , implying that one of its eigenvalues $\lambda_i(j\omega)$ vanishes at ω_0 . If this zero is simple, then the eigenvalue trajectory $\lambda_i(j\omega)$ changes sign at ω_0 , thus violating the passivity condition (5.12).

The above observations can be summarized in the following statements [12, 33]. Assuming that A is asymptotically stable, and that $W_0 = D + D^T > 0$ is strictly positive definite, then system (5.1) is strictly passive if and only if the Hamiltonian matrix \mathcal{M} in (5.16) has no purely imaginary eigenvalues. In presence of purely imaginary eigenvalues, the system is passive only if the associated Jordan blocks have even size, in which case it can be shown that the corresponding eigenvalue trajectory $\lambda_i(j\omega)$ does not change sign at ω_0 . A qualitative illustration of the above statements is provided by Figure 5.2. For a more complete treatment, which is outside the scope of this introductory chapter, see [1, 54]. We remark that the condition $W_0 > 0$ is equivalent to requiring that the system is asymptotically passive, so that the transfer function is nonnegative Hermitian for $\omega \rightarrow \infty$.

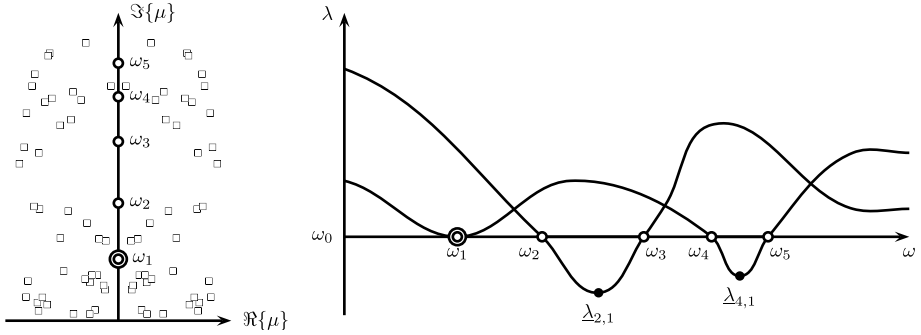


Figure 5.2: Illustration of the relationship between Hamiltonian eigenvalues μ_k (left) and eigenvalues $\lambda_i(j\omega)$ of $\Psi(j\omega)$ (right). In the left panel, purely imaginary Hamiltonian eigenvalues are denoted with circles (number of circles denote multiplicity) to distinguish them from other eigenvalues (squares); only eigenvalues with nonnegative imaginary part are shown. In the right panel, the non-passive frequency bands $\Omega_2 = (\omega_2, \omega_3)$ and $\Omega_4 = (\omega_4, \omega_5)$ are highlighted with a thick line, with corresponding local minima $\lambda_{2,1}$ and $\lambda_{4,1}$.

5.3 Checking passivity

There are two main approaches for checking whether a given state-space model (5.1) is passive. These methods exploit the Positive Real Lemma (5.10) and the properties of the Hamiltonian matrix (5.16), respectively.

5.3.1 Checking passivity via linear matrix inequalities

The Positive Real Lemma discussed in Section 5.2.1.1 states that system (5.1) is passive if and only if (5.10) holds. Note that this condition embeds as a corollary also a stability

check, since restricting (5.10) to its upper-left block, which corresponds to setting $u = 0$, i. e., considering the zero-input response, results in the popular Lyapunov condition for (simple) stability, here restricted to the LTI case, which reads

$$\exists P = P^T > 0 : \quad A^T P + PA \leq 0. \quad (5.18)$$

Both (5.18) and (5.10) are recognized as *LMI feasibility* (convex) problems. As such, they can be readily solved by specialized LMI convex optimization software, such as SeDuMi [77] and Yalmip [55]. For a survey of tools see [49], part 6. If the system is passive, such tools will return a Lyapunov matrix P for which the above conditions are satisfied. Conversely, they will return a certificate of non-feasibility, thus proving existence of passivity violations.

The main advantage of the LMI-based passivity check is simplicity: one does not have to write any particular code, since most LMI solvers have simple-to-use interfaces. This advantage is counterbalanced by two important drawbacks. The first disadvantage is computational complexity. The PRL in (5.10) requires proving nonnegativity of a $M + N$ matrix, with P being unknown. The number of decision variables is $N(N + 1)/2$, i. e., the number of elements of P . A direct implementation thus requires a computational cost that scales as $\mathcal{O}(N^6)$, although advanced solvers exist that can reduce this cost to $\mathcal{O}(N^4)$ [82]. Exploitation of sparsity, structure and symmetries can be used to reduce this cost even further in many practical cases (for an example see [21]).

The second main disadvantage of LMI-based passivity checks is in the binary nature of their output (passive/non-passive). If the system is not passive, no additional information is available from the solver that can be exploited to fix the passivity violation by a suitable perturbation process. Fortunately, this information is available from the Hamiltonian-based passivity check, discussed next.

5.3.2 Checking passivity using Hamiltonian eigenvalues

As discussed in Section 5.2.1.3, an asymptotically stable system (5.1) with $D + D^T > 0$ is passive if the Hamiltonian matrix \mathcal{M} defined in (5.16) does not have purely imaginary eigenvalues with odd-sized Jordan blocks (in the vast majority of cases these eigenvalues, if any, are simple). This suggests a simple algorithm for checking passivity, summarized as pseudocode in Algorithm 5.1. This scheme is formulated so that it provides as output some additional information, in particular the frequency bands where (5.12) is violated [33]. This information will prove very useful in Section 5.5 for removing such passivity violations via perturbation.

As a first step, we form the Hamiltonian matrix and we compute its eigenvalues $\mu_k \in \text{eig}(\mathcal{M})$. If no such eigenvalues are purely imaginary, and if $D + D^T > 0$, then the model is concluded to be strictly passive and the algorithm stops. No eigenvalue trajectory will cross the imaginary axis, otherwise the corresponding intersection would be pinpointed by some imaginary Hamiltonian eigenvalue.

The more interesting case occurs in the presence of purely imaginary eigenvalues $\mu_k = j\omega_k$. Let us extract the subset of these eigenvalues with nonnegative imaginary part (recall that, if $j\omega_k$ is an eigenvalue, also $-j\omega_k$ is an eigenvalue due to the Hamiltonian structure of \mathcal{M}) and sort them in ascending order,

$$0 = \omega_0 < \omega_1 < \omega_2 < \cdots < \omega_K < \omega_{K+1} = +\infty \quad (5.19)$$

where ω_0 is added even if 0 is not an eigenvalue of \mathcal{M} , and where we set $\omega_{K+1} = +\infty$. The frequencies ω_k induce a partition of the frequency axis into disjoint adjacent subbands $\Omega_k = (\omega_k, \omega_{k+1})$ for $k = 0, \dots, K$. From the above discussion, $\Psi(j\omega)$ is nonsingular $\forall \omega \in \Omega_k, \forall k$.

Each subband Ω_k is then flagged as passive or non-passive by assigning k to corresponding index sets \mathcal{K}_p and \mathcal{K}_{np} , respectively, depending on whether (5.12) is verified or not for $\omega \in \Omega_k$. This condition is very easy to check, due to the continuity of all eigenvalue trajectories $\lambda_i(j\omega)$, which is a consequence of the assumed asymptotic stability, so that both $H(s)$ and $\Psi(s)$ are regular on the imaginary axis. It is thus sufficient to check whether

$$\Psi(j\check{\omega}_k) > 0, \quad \text{where } \check{\omega}_k = \frac{\omega_k + \omega_{k+1}}{2}, \quad (5.20)$$

is the midpoint of band Ω_k . If (5.20) is verified, then the model is uniformly passive in Ω_k and $k \in \mathcal{K}_p$. Otherwise, $k \in \mathcal{K}_{np}$ and the number of negative eigenvalues $\lambda_i(j\omega)$ (which is constant in Ω_k), is determined based on their evaluation at the midpoint $\check{\omega}_k$.

As a final optional step, the subbands Ω_k with $k \in \mathcal{K}_{np}$ can be subjected to a local (adaptive) sampling in order to find all local minima of the eigenvalue trajectories $\lambda_i(j\omega)$. These minima, denoted with their frequency location as $(\omega_{kv}, \underline{\lambda}_{kv})$, correspond to the worst-case local passivity violations. See Figure 5.2 for a graphical illustration. See also [20, 34].

The computational cost of the passivity check in Algorithm 5.1 is dominated by the Hamiltonian eigenvalue evaluation. A general-purpose eigenvalue solver scales as $\mathcal{O}(\kappa N^3)$ where κ is a constant, since the size of the Hamiltonian matrix (5.16) is $2N$. Specialized eigensolvers exist that reduce this cost by exploiting the particular matrix structure [7, 10]; see also [1, 9, 79], but still retaining the scaling $\mathcal{O}(\kappa N^3)$ albeit with a smaller constant κ . If the transfer function $H(s)$ of the model is symmetric (which is usually verified in electrical and electronic applications), additional computational savings can be achieved by defining equivalent and smaller-size eigenproblems, often referred to as *half-size passivity tests*; see [23, 36, 47, 48, 75].

When the number of states N is medium-large and the state-space realization is sparse (for instance with A diagonal or quasi-diagonal), then it is more convenient to use eigensolvers based on repeated shift-invert iterations; see e. g. [3, 41, 85]. It has been demonstrated that these methods are able to reduce the scaling law of purely imaginary eigenvalue determination to $\mathcal{O}(\kappa N)$, although with a possibly large κ . See also [8, 60, 85] for details on more general structured eigenproblems.

Algorithm 5.1: Hamiltonian-based passivity check.**Require:** real state-space matrices A, B, C, D **Require:** A asymptotically stable, $D + D^T$ nonsingular

- 1: form the Hamiltonian matrix \mathcal{M} of (5.16) and compute its eigenvalues μ_k
- 2: **if** no eigenvalue is purely imaginary and $D + D^T > 0$ **then**
- 3: system is strictly passive: exit
- 4: **end if**
- 5: extract all imaginary eigenvalues $\mu_k = j\omega_k$ and sort them as in (5.19)
- 6: set $\mathcal{K}_p = \mathcal{K}_{np} = \emptyset$
- 7: **for** $k = 0, \dots, K$ **do**
- 8: form subband $\Omega_k = (\omega_k, \omega_{k+1})$ and its midpoint $\tilde{\omega}_k$
- 9: **if** $\Psi(j\tilde{\omega}_k) > 0$ **then**
- 10: system is locally passive $\forall \omega \in \Omega_k$, add k to \mathcal{K}_p
- 11: **else**
- 12: system is not passive in Ω_k , add k to \mathcal{K}_{np}
- 13: find all local minima $(\underline{\omega}_{kv}, \underline{\lambda}_{kv})$ of the eigenvalues of $\Psi(j\omega)$ in Ω_k
- 14: **end if**
- 15: **end for**

5.4 System perturbation

Assuming that the system (5.1) is detected as non-passive from a passivity check, the main question arises whether we can enforce its passivity through a small perturbation of its coefficients. What is actually important is not the amount of coefficient perturbation, but rather the perturbation in the model response, which should be kept under control in order to maintain model accuracy. Of course, this approach makes sense only if the passivity violations of the initial model are relatively small to enable correction via perturbation. This situation is in fact commonly encountered in applications. Very large passivity violations in models that should represent dissipative systems are a clear indication of poor model quality. Such models should be discarded and regenerated.

Several perturbation approaches are possible for (5.1). In the following, we focus on one particular strategy, which amounts to perturbing only the state-output matrix as $\hat{C} = C + \delta C$ while leaving the other state-space matrices unchanged. This strategy induces the following perturbation in the transfer function:

$$\hat{H}(s) = H(s) + \delta H(s) \quad \text{with } \delta H(s) = \delta C(sI - A)^{-1}B. \quad (5.21)$$

The corresponding impulse response perturbation is thus

$$\hat{h}(t) = h(t) + \delta h(t) \quad \text{with } \delta h(t) = \delta C e^{At} B u(t) \quad (5.22)$$

where u is the Heaviside step function. This approach leaves the state matrix A unchanged, thus preserving the system poles. Allowing for poles perturbation induced by a modification of A would require in fact additional constraints for ensuring that stability is not compromised. Except for very few cases [22, 53], most existing passivity enforcement schemes do not modify matrix A in order to preserve the system poles. This is a common scenario in those applications where passivity enforcement is applied as a post-processing of a model identified from measurements with Vector Fitting (see [5, Chapter 8]). If the system is asymptotically passive with $D + D^T > 0$, there is also no need to modify the direct coupling matrix D . Modification of the input-state map B can be considered as an alternative to (5.21).

Based on (5.21), we need to determine a cost function that measures the perturbation amount in terms of the decision variables, i. e., the elements of δC . A number of popular cost functions are reviewed below.

5.4.1 Gramian-based cost functions

A natural choice for measuring the system perturbation is the L_2 norm. We have

$$\mathcal{E}_2^2 = \|\delta h\|_{L_2}^2 = \int_0^{+\infty} \text{tr}(\delta h(t)\delta h(t)^T) dt = \text{tr}(\delta C \mathcal{G}_c \delta C^T) \quad (5.23)$$

where tr is the trace of its matrix argument, and

$$\mathcal{G}_c = \int_0^{+\infty} e^{At} B B^T e^{A^T t} dt = \frac{1}{2\pi} \int_{-\infty}^{+\infty} (j\omega I - A)^{-1} B B^T (j\omega I - A)^{-H} d\omega \quad (5.24)$$

is the *Controllability Gramian* of the system, which is easily found by solving the Lyapunov equation

$$A \mathcal{G}_c + \mathcal{G}_c A^T = -B B^T. \quad (5.25)$$

Although simple to use, the cost function (5.23) is seldom used in applications. This is due to the fact that most often a reduced order model is obtained from some approximation process that ensures accuracy only in a well-defined frequency band, which usually does not extend up to ∞ . Assuming that the model accuracy is of interest only for $\omega \in [0, \omega_{\max}]$, then it is unnecessary and even detrimental to include any contribution to the Gramian coming from frequencies $|\omega| > \omega_{\max}$. A simple approach to obtain a bandlimited norm is to limit the integration bounds in (5.24) to $\mp \omega_{\max}$. One loses the possibility to compute \mathcal{G}_c through (5.25), so that the corresponding bandlimited Gramian should be obtained by a direct numerical integration of (5.24) through some quadrature rule.

An alternative option is to introduce a nonnegative weighting function $\rho(\omega)$ in the frequency-domain integral (5.24), which allows one to fine-tune the contributions to the Gramian coming from different frequencies. The latter strategy lends itself to a simple algebraic procedure for the weighted Gramian computation, in the case the weight is restricted to be in form of a state-space system applied to the input or to the output of our original transfer function $H(s)$. Some details follow.

Let us consider a weighting function in state-space form with transfer matrix $\Gamma(s) = C_\Gamma(sI - A_\Gamma)^{-1}B_\Gamma + D_\Gamma$ of compatible size, which is applied to defining a *weighted error function*

$$\delta H_\Gamma(s) = \delta H(s) \Gamma(s). \quad (5.26)$$

Instead of (5.23), we measure system perturbation through the Γ -weighted norm defined as

$$\mathcal{E}_\Gamma^2 = \|\delta H\|_\Gamma^2 = \|\delta H_\Gamma\|_{L_2}^2. \quad (5.27)$$

It can be easily shown [99] that this norm can be computed as

$$\|\delta H\|_\Gamma^2 = \text{tr}(\delta C P_\Gamma \delta C^T), \quad (5.28)$$

where P_Γ is the upper-left block of the solution of the following augmented Lyapunov equation:

$$\tilde{A} \tilde{P} + \tilde{P} \tilde{A}^T = -\tilde{B} \tilde{B}^T \quad (5.29)$$

with

$$\tilde{A} = \begin{pmatrix} A & BC_\Gamma \\ 0 & A_\Gamma \end{pmatrix}, \quad \tilde{B} = \begin{pmatrix} BD_\Gamma \\ B_\Gamma \end{pmatrix}, \quad \tilde{P} = \begin{pmatrix} P_\Gamma & P_{12} \\ P_{12}^T & P_{22} \end{pmatrix}. \quad (5.30)$$

We see that this characterization is fully compatible with the standard L_2 norm, as far as the standard Gramian \mathcal{G}_c is replaced by its weighted counterpart P_Γ . This formulation can be adapted to applications that require control over relative error, by choosing $\Gamma(s) = H^{-1}(s)$ or even elementwise relative error [42]. If we are interested in retaining accuracy only in some prescribed frequency band $(\omega_{\min}, \omega_{\max})$, then $\Gamma(s)$ can be defined as a band pass filter matched to this band.

The two Gramian-based error characterizations (5.23) and (5.28) are further simplified as follows. Let us consider (5.23), and let us assume that the initial model is controllable, so that the controllability Gramian \mathcal{G}_c is full-rank and strictly positive definite.⁴ Computing the Cholesky factorization $\mathcal{G}_c = Q_c^T Q_c$ and inserting it into (5.23) leads to

$$\mathcal{E}_2^2 = \text{tr}(\delta C Q_c^T Q_c \delta C^T) = \text{tr}(\Xi \Xi^T) = \|\Xi\|_F^2 = \|\xi\|_2^2 \quad (5.31)$$

⁴ In case \mathcal{G}_c is singular, a preprocessing step based, e. g., on Balanced Truncation [5, Chapter 2] can be applied to remove any uncontrollable states.

where $\|\cdot\|_F$ denotes the Frobenius norm of its matrix argument, $\Xi = \delta C Q_c^T$ and $\xi = \text{vec}(\Xi)$ stacks the columns of Ξ into a single column vector.⁵ We see that the model perturbation is now cast as the Euclidean norm of the (vectorized) decision variables ξ in a new coordinate system induced by the Cholesky factor of the Gramian. Minimization of (5.31) is thus trivial.

5.4.2 Data-based cost functions

A further alternative for defining a cost function that measures the model perturbation error is based on a purely discrete formulation. Let us suppose that the model (5.1) was obtained in first place through a data-driven MOR scheme, starting from a set of frequency-domain measurements of the underlying system response $(\omega_\ell, \check{H}_\ell)$ for $\ell = 1, \dots, L$. A natural choice would be to minimize the error of the perturbed model with respect to these initial data [21, 43, 44, 46]

$$\mathcal{E}^2 = \sum_{\ell=1}^L \rho_\ell^2 \mathcal{E}_\ell^2 \quad \text{with} \quad \mathcal{E}_\ell^2 = \|H(j\omega_\ell) + \delta H(j\omega_\ell) - \check{H}_\ell\|_F^2, \quad (5.32)$$

where we used the Frobenius norm to define the local error \mathcal{E}_ℓ for each frequency point (of course other norm choices are possible), and where ρ_ℓ is a weighting factor to be defined based on the desired approximation criteria. A straightforward derivation shows that, vectorizing the decision variables as $\delta c = \text{vec}(\delta C)$, we can write

$$\mathcal{E}^2 = \|K\delta c - d\|_2^2 \quad (5.33)$$

with

$$K^T = (\rho_1 K_1^T \quad \dots \quad \rho_L K_L^T), \quad d^T = (\rho_1 d_1^T \quad \dots \quad \rho_L d_L^T), \quad (5.34)$$

where the various components are defined using the Kronecker product \otimes as

$$K_\ell = [(j\omega_\ell I - A)^{-1} B]^T \otimes I, \quad d_\ell = \text{vec}(H(j\omega_\ell) - \check{H}_\ell). \quad (5.35)$$

A particular case of (5.32) is obtained by defining

$$\mathcal{E}_\ell^2 = \|\delta H(j\omega_\ell)\|_F^2. \quad (5.36)$$

This choice corresponds to setting the “target” data samples as the responses of the initial model, so that $\check{H}_\ell = H(j\omega_\ell)$ and consequently $d_\ell = 0$. Correspondingly, (5.33) reduces to the simple quadratic form

$$\mathcal{E}^2 = \|K\delta c\|_2^2. \quad (5.37)$$

⁵ We will denote the inverse operation $\Xi = \text{mat}(\xi)$, where the size of Ξ is inferred from the context.

5.5 Passivity enforcement

In Section 5.4, we showed how the perturbation of a model based on a modification of the state-output map can be algebraically characterized as a quadratic form of the decision variables, i. e., the elements of δC , possibly cast in a different coordinate system. The resulting cost function provides an effective control over model perturbation if used within an optimization problem, combined with suitable constraints for passivity enforcement. In this section, we discuss the three most prominent approaches for casting the passivity conditions introduced in Section 5.2.1 as constraints, giving rise to three classes of algorithms for passivity enforcement. An overview of alternative approaches and a more complete treatment is available in [39].

5.5.1 Passivity enforcement via LMI constraints

Let us consider an initial non-passive system (5.1), for which the PRL condition (5.10) is not satisfied. We try to enforce this condition on a perturbed system, where the state-output matrix C is updated as

$$\hat{C} = C + \delta C = C + \Xi Q_c^{-T}, \quad (5.38)$$

where we used the change of variables in (5.31) based on the Cholesky factor Q_c of the controllability Gramian. Enforcing the PRL for the perturbed system while minimizing the perturbation, based, e. g., on the cost function (5.31), amounts to solving the following constrained optimization problem

$$\min_{P, \Xi} \|\Xi\|_F^2 \quad \text{s. t.} \quad P = P^T > 0 \quad \text{and} \quad \mathcal{F}(P, \Xi) \leq 0 \quad (5.39)$$

where

$$\mathcal{F}(P, \Xi) = \begin{pmatrix} A^T P + P A & P B - C^T - Q_c^{-1} \Xi^T \\ B^T P - C - \Xi Q_c^{-T} & -D - D^T \end{pmatrix}. \quad (5.40)$$

The cost function in (5.39) is a quadratic form in the decision variables, and both constraints are of LMI type [21]. Problem (5.39) is known to be convex, therefore there is a theoretical guarantee that a unique optimal solution exists, which can be found in polynomial time. In fact, specialized solvers for this class of problems exist, see e. g. [55, 77], therefore we do not detail any particular algorithm any further (see also section 5.3.1). The reader is referred to standard textbooks on convex optimization for more details [14].

As already discussed in Section 5.3.1, the computational cost that is required to solve (5.39) scales quite badly with the number of decision variables, equivalently with the system size. The main motivation for this high computational requirements is the

presence of the Lyapunov matrix P in the set of decision variables, which is only instrumental to the PRL formulation, but which is not really needed as a result of the optimization process. Therefore, we can think of eliminating P with a suitable pre-processing step, in order to obtain a smaller LMI problem that can be solved more efficiently. The so-called *trace parameterization* provides a solution to this problem; see [25, 21, 18] for details. We now seek alternatives that provide even better scalability. The reader is encouraged to also see [43].

5.5.2 Passivity enforcement via Hamiltonian perturbation

Let us consider the Hamiltonian-based passivity constraints discussed in Section 5.2.1.3. Under the assumptions that A is asymptotically stable and system (5.1) is asymptotically passive with $D + D^T > 0$, then (strict) passivity holds if the Hamiltonian matrix \mathcal{M} in (5.16) has no purely imaginary eigenvalues. If this is not true, as Figure 5.2 shows, the Hermitian part of the frequency response has some negative eigenvalues in some frequency bands, and the system is not passive due to those localized violations.

The main idea of passivity enforcement via Hamiltonian perturbation is to induce a spectral perturbation on the imaginary Hamiltonian eigenvalues, so that they are displaced in the correct direction as to eliminate the local passivity violations [33]. A graphical illustration of this strategy is provided in Figure 5.3, where we show that when two imaginary eigenvalues are displaced along the imaginary axis in a direction that points inward each passivity violation band, the extent of the violation is effectively reduced (top panels). If the perturbation amount is sufficiently large to induce a collision of the two imaginary eigenvalues (bottom panels), then a bifurcation occurs and the two eigenvalues move off the imaginary axis. The passivity violation is thus removed.

The above spectral perturbation is an inverse problem, which requires a precise characterization of the relation between matrix element perturbations and the corresponding induced change in the eigenvalues that we need to displace. The algorithm that we describe below is based on a first-order approximation of this relation.

Let us consider once again a non-passive system which is perturbed by changing the state-output matrix as $\hat{C} = C + \delta C$. A straightforward first-order approximation analysis leads to the following expression for the perturbed Hamiltonian matrix:

$$\hat{\mathcal{M}} = \mathcal{M} + \delta \mathcal{M} \quad \text{with } \delta \mathcal{M} \approx \begin{pmatrix} -BW_0^{-1}\delta C & 0 \\ C^T W_0^{-1}\delta C + \delta C^T W_0^{-1}C & \delta C^T W_0^{-1}B^T \end{pmatrix} \quad (5.41)$$

where $W_0 = D + D^T$. Let us now consider a generic eigenvalue μ_k of \mathcal{M} with unit multiplicity, and let us denote the corresponding right and left eigenvectors as v_k and w_k , normalized such that $\|v_k\| = \|w_k\| = 1$. We have the following first-order eigenvalue

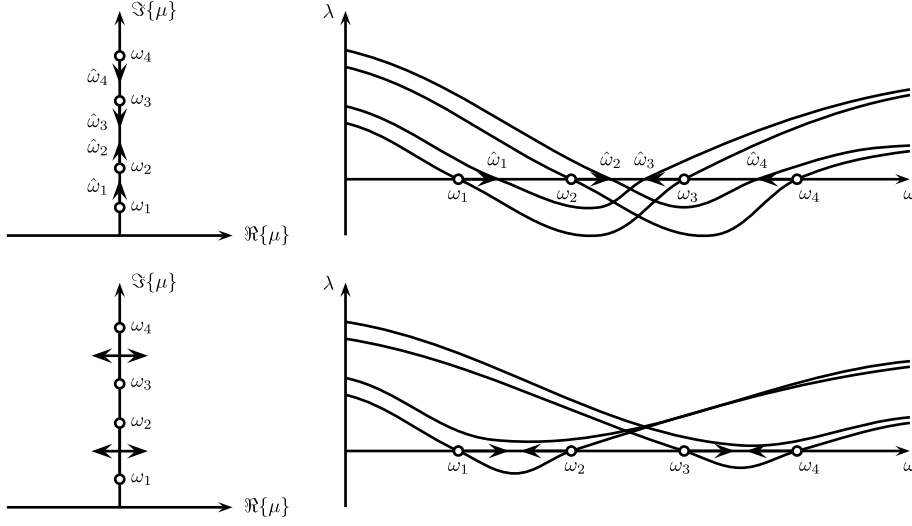


Figure 5.3: Illustration of passivity enforcement via Hamiltonian eigenvalue perturbation. Top and bottom panels refer to two different scenarios that may occur. In the top left panel the purely imaginary Hamiltonian eigenvalues are depicted with empty dots, and their perturbation direction and extent is represented with thick arrows. The corresponding eigenvalue trajectories $\lambda_j(j\omega)$ before (solid lines) and after (dashed lines) perturbation are depicted in the top right panel. Bottom panels show that when two imaginary Hamiltonian eigenvalues collide (left panel), the corresponding intersections of the eigenvalue trajectories $\lambda_j(j\omega)$ with the frequency axis are removed (right panel).

perturbation result [86]:

$$\hat{\mu}_k \approx \mu_k + \delta\mu_k \quad \text{with} \quad \delta\mu_k = \frac{w_k^H \delta\mathcal{M} v_k}{w_k^H v_k}. \quad (5.42)$$

We now particularize (5.42) to the case of a purely imaginary eigenvalue $\mu_k = j\omega_k$. It is well known that, for such eigenvalues, the left and right eigenvectors are related by $w_k = -Jv_k$, where J is defined in (5.17), so that we can write

$$\delta\mu_k = \frac{v_k^H J \delta\mathcal{M} v_k}{v_k^H J v_k}. \quad (5.43)$$

Splitting now the right eigenvector as $v_k^T = (v_{k1}^T, v_{k2}^T)$ according to the block structure of \mathcal{M} , we see that the denominator of (5.43) is purely imaginary

$$v_k^H J v_k = 2J\mathfrak{I}\{v_{k1}^H v_{k2}\} \quad (5.44)$$

whereas the numerator is real-valued since $J\mathcal{M}$ is real and symmetric. A tedious but straightforward calculation leads to

$$v_k^H J \delta\mathcal{M} v_k = 2\Re\{v_{k1}^T \otimes y_k^H\} \delta c \quad (5.45)$$

where $\delta c = \text{vec}(\delta C)$ and the auxiliary vector y_k is defined as

$$y_k = W_0^{-1}(Cv_{k1} + B^T v_{k2}). \quad (5.46)$$

Using the above expressions, we can finally rewrite (5.43) as

$$\Re\{v_{k1}^T \otimes y_k^H\} \delta c \approx (\omega_k - \hat{\omega}_k) \Im\{v_{k1}^H v_{k2}\}, \quad (5.47)$$

where we used the fact that under the adopted first-order approximation also the perturbed eigenvalue is purely imaginary $\hat{\mu}_k = j\hat{\omega}_k$. Furthermore, applying the change of variables (5.31) to (5.47) gives

$$z_k^T \xi \approx \eta_k \quad (5.48)$$

where

$$z_k = \Re\{(Q_c^{-T} v_{k1}) \otimes y_k^*\}, \quad \eta_k = (\omega_k - \hat{\omega}_k) \Im\{v_{k1}^H v_{k2}\}. \quad (5.49)$$

This expression is a linearized constraint that relates the amount of (imaginary) eigenvalue perturbation to the corresponding perturbation on the decision variables ξ .

When using (5.48) as a constraint to determine ξ , the desired location for $\hat{\omega}$ needs to be provided as input. With reference to Figure 5.3, we see that the direction where ω_k should be perturbed is directly related to the slope $\lambda'_{i,k}$ of the eigenvalue trajectory $\lambda_i(j\omega)$ that vanishes at ω_k . A heuristic yet effective choice for $\hat{\omega}_k$ is

$$\begin{cases} \hat{\omega}_k = \omega_k + \alpha(\omega_{k+1} - \omega_k) & \text{for } \lambda'_{i,k} < 0, \\ \hat{\omega}_k = \omega_k - \alpha(\omega_k - \omega_{k-1}) & \text{for } \lambda'_{i,k} > 0, \end{cases} \quad (5.50)$$

where the control parameter $0 < \alpha < 1$ determines the maximum extent of the perturbation amount relative to the size of the violation subband. Additional details on how to determine the slopes $\lambda'_{i,k}$ as well as appropriate values of α can be found in [33].

Supposing now that multiple eigenvalues $\mu_k = j\omega_k$ for $k = 1, \dots, K$ are to be perturbed concurrently, we need to collect all independent constraints (5.48) so that they are enforced simultaneously. The resulting optimization problem to be solved reads

$$\min_{\xi} \|\xi\|_2^2 \quad \text{s. t.} \quad z_k^T \xi = \eta_k, \quad k = 1, \dots, K \quad (5.51)$$

This is a simple linearly constrained minimum norm problem, whose optimal solution is $\xi_{\text{opt}} = Z^\dagger \eta$, where † denotes the pseudoinverse [14], with Z and η collecting z_k^T and η_k as rows. Compared to the evaluation of the Hamiltonian eigenvalues required to set up the constraints (5.48), the solution of (5.51) has a negligible computational cost.

Although the solution of (5.51) is straightforward, its passivity constraint is based on a linearization process and is therefore only accurate up to first order. Therefore,

the perturbation fraction α should be selected to be small enough for the first-order approximation to be accurate, and multiple iterations may be required to displace all imaginary eigenvalues. Figure 5.3 illustrates two scenarios that may typically occur during iterations, whereas Algorithm 5.2 provides the pseudocode of a possible implementation. The computational cost of this implementation is dominated by the Hamiltonian eigensolution; see the discussion in Section 5.3.2. We remark that, despite the fact the optimization problem (5.51) at each iteration has a closed-form solution, the overall iterative scheme in its basic formulation is not guaranteed to converge, since a local perturbation of few eigenvalues does not guarantee that new imaginary eigenvalues will not occur at other locations. The approach that is presented in the next section provides a more robust scheme.

Algorithm 5.2: Passivity enforcement via Hamiltonian perturbation.

Require: real state-space matrices A, B, C, D

Require: A asymptotically stable, $D + D^T > 0$

Require: control parameter $0 < \alpha < 1$ and max iterations i_{\max}

- 1: run Alg. 5.1 to check passivity, store $\{\omega_k\}$ and non-passive bands Ω_k
 - 2: compute Gramian \mathcal{G}_c or weighted Gramian P_Γ and its Cholesky factor Q_c
 - 3: set iteration count $i = 0$
 - 4: **while** (system not passive and $i < i_{\max}$) **do**
 - 5: $i \leftarrow i + 1$
 - 6: compute right eigenvectors v_k and form vectors z_k in (5.49), for all k
 - 7: define $\hat{\omega}_k$ as in (5.50) and form η_k in (5.49) for all k
 - 8: solve optimization problem (5.51) for ξ
 - 9: update state-output map $C \leftarrow C + \Xi Q_c^{-T}$ where $\Xi = \text{mat}(\xi)$
 - 10: run Alg. 5.1 to check passivity, store $\{\omega_k\}$ and non-passive bands Ω_k
 - 11: **end while**
-

Before closing this section we remark that the above discussion was based on the assumption of simple Hamiltonian eigenvalues. A full characterization of the general case with arbitrary higher multiplicity requires knowledge of the complete structure of the possibly multiple Jordan blocks of the Hamiltonian matrix. This discussion is outside the scope of this chapter, the reader is referred to [1, 61] for a complete treatment. We only remark that the presence of defective eigenspaces is structurally unstable to small perturbations, so that the defectivity usually disappears if a small perturbation is applied.

Passivity enforcement via Hamiltonian perturbation was first introduced in [33], followed by various applications [17, 31, 71] and extensions to large-scale systems [41]

with possibly frequency weighted accuracy norms [42]. It is worth mentioning the straightforward extension [11, 57] to so-called *negative imaginary* systems.⁶

5.5.3 Passivity enforcement via local perturbation

Let us consider once again a system that is detected as non-passive from Algorithm 5.1. One of the results that this algorithm provides in addition to the index set $k \in \mathcal{K}_{np}$ that identifies the non-passive bands $\Omega_k = (\omega_k, \omega_{k+1})$ is a set of local minima $(\underline{\omega}_{kv}, \underline{\lambda}_{kv})$ of the eigenvalues of $\Psi(j\omega)$ in each of these subbands. Figure 5.4 depicts these local minima with filled dots.

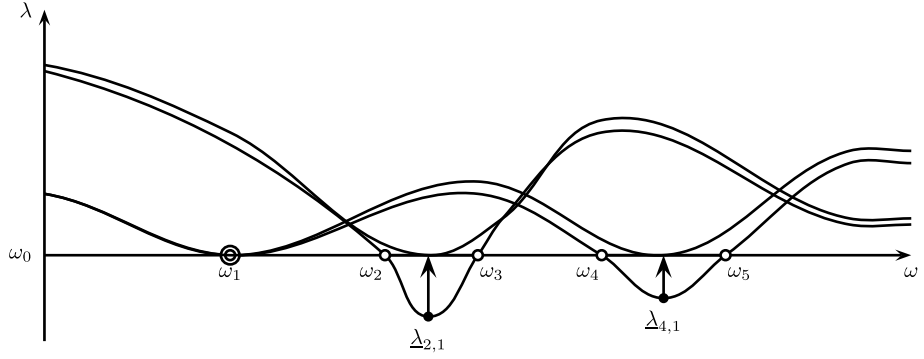


Figure 5.4: Illustration of passivity enforcement via local perturbation. Linearized constraints are used to perturb (thick arrows) the local minima $\underline{\lambda}_{i,k}$ (filled dots) of the eigenvalue trajectories $\lambda_i(j\omega)$ (solid lines) so that they become nonnegative. The resulting perturbed eigenvalue trajectories (dashed lines) are uniformly positive after few iterations.

Assume now to perturb the system through the usual state-output matrix as $\hat{C} = C + \delta C$. This perturbation leads to an induced perturbation on the eigenvalue trajectories $\lambda(j\omega)$, represented in Figure 5.4 by solid lines. We seek a constraint that displaces the local minima to a new nonnegative value [72, 73]. Denoting with v_{kv} the eigenvector of $\Psi(j\underline{\omega}_{kv})$ normalized as $\|v_{kv}\| = 1$ corresponding to the eigenvalue $\underline{\lambda}_{kv}$, we can express the induced eigenvalue perturbation through the following first-order approximation, which results in an inequality constraint after imposing nonnegativity:

$$\hat{\underline{\lambda}}_{kv} = \underline{\lambda}_{kv} + v_{kv}^H \delta \Psi(j\underline{\omega}_{kv}) v_{kv} \geq 0. \quad (5.52)$$

⁶ A system with square, strictly proper and stable transfer matrix $H(s)$ is negative imaginary if and only if $sH(s)$ is Positive Real.

Figure 5.4 provides a graphical illustration of the perturbation, together with the expected perturbed eigenvalue trajectories (dashed lines). Note that these trajectories remain continuous after perturbation, thanks to the assumed asymptotic stability of the model (no poles on the imaginary axis).

The constraint (5.52) can now be readily expressed in terms of our decision variables δC , noting that

$$\delta\Psi(j\omega) = \delta C T(j\omega) + T^H(j\omega) \delta C^T \quad (5.53)$$

where $T(j\omega) = (j\omega I - A)^{-1}B$. Using the vectorized form $\delta c = \text{vec}(\delta C)$ together with the change of variable (5.31) leads to

$$z_{kv}^T \xi \geq -\underline{\lambda}_{kv}, \quad \text{with } z_{kv} = 2\Re\{(Q_c^{-T} T(j\omega_{kv}) v_{kv}) \otimes v_{kv}^*\}. \quad (5.54)$$

As a result, we cast our minimum model perturbation subject to local passivity constraints as

$$\min_{\xi} \|\xi\|_2^2 \quad \text{s. t.} \quad z_{kv}^T \xi \geq -\underline{\lambda}_{kv}, \quad \forall k \in \mathcal{K}_{np}, \forall v. \quad (5.55)$$

This problem is convex and is readily solved through off-the-shelf software. Based on the analysis in [43] the computational cost for solving (5.55) can be reduced to $\mathcal{O}(\kappa NM^2)$.

As for the Hamiltonian perturbation passivity enforcement, the above local perturbation is not guaranteed to achieve a passive model after the solution of (5.55). In fact

- the inequality constraint in (5.54) is only first-order accurate and does not guarantee that the perturbed eigenvalue will be nonnegative after applying the computed model correction;
- it is not guaranteed that a local perturbation of all local eigenvalue minima $(\underline{\omega}_{kv}, \underline{\lambda}_{kv})$ will not induce new passivity violations at new locations, in terms of new negative eigenvalue minima.

The first problem can be easily addressed by embedding (5.55) within an iterative scheme that, after solving (5.55), applies model correction and repeats the perturbation until all local eigenvalue minima are nonnegative. The second problem is also easily addressed by the so-called *robust iterations*, described next.

Assume that after model perturbation a new local eigenvalue minimum $\underline{\lambda}_{\text{new}} < 0$ is detected at some frequency $\underline{\omega}_{\text{new}}$ where the model was locally passive before perturbation. If we could enforce the eigenvalues of $\Psi(j\underline{\omega}_{\text{new}})$ to remain nonnegative through an additional constraint together with those in (5.55), then the new violation would not have arisen. This is exactly the main idea of robust iterations, where problem (5.55) is solved only as a preliminary step. All new violations are collected and nonnegativity constraints are formulated as in (5.54) at the corresponding frequencies and added to

the set of already available constraints. This prediction step is repeated until no new violations are introduced. Then iterations continue after the model is updated.

The passivity enforcement scheme based on local perturbations (without robust iterations) is outlined as pseudocode in Algorithm 5.3. More details on the robust iteration scheme are available in [44, 45].

Algorithm 5.3: Passivity enforcement via local perturbations.

Require: real state-space matrices A, B, C, D

Require: A asymptotically stable, $D + D^T > 0$

Require: max iterations i_{\max}

- 1: run Alg. 5.1 to check passivity, store local eigenvalue minima $(\underline{\omega}_{kv}, \underline{\lambda}_{kv})$
 - 2: compute Gramian \mathcal{G}_c or weighted Gramian P_W and its Cholesky factor Q_c
 - 3: set iteration count $i = 0$
 - 4: **while** (system not passive and $i < i_{\max}$) **do**
 - 5: $i \leftarrow i + 1$
 - 6: compute eigenvectors v_{kv} and form vectors z_{kv} in (5.54), for all k, v
 - 7: solve optimization problem (5.55) for ξ
 - 8: update state-output map $C \leftarrow C + \Xi Q_c^{-T}$ where $\Xi = \text{mat}(\xi)$
 - 9: run Alg. 5.1 to check passivity, store local eigenvalue minima $(\underline{\omega}_{kv}, \underline{\lambda}_{kv})$
 - 10: **end while**
-

5.6 Extensions

The various passivity check and enforcement algorithms discussed in previous sections were restricted to the narrow class of regular state-space systems (5.1), with A asymptotically stable, with $D + D^T > 0$, and with a supply rate defined by (5.7). In this section, we release these assumptions by providing suitable generalizations.

5.6.1 Releasing asymptotic passivity requirements

When $W_0 = D + D^T$ is singular but positive semidefinite, then the system might still be passive (although not strictly passive), with at least one of the eigenvalues $\lambda_i(j\omega)$ of $\Psi(j\omega)$ vanishing for $\omega \rightarrow \infty$. In this scenario, the passivity check based on the Hamiltonian matrix \mathcal{M} in (5.16) cannot be performed, since \mathcal{M} is ill-defined and cannot be constructed.

The Hamiltonian matrix can, however, be generalized [93] by avoiding the inversion of W_0 in (5.15). Retaining the vector v and adding it as an additional block-component to the eigenvector in (5.16) leads to the following generalized eigenvalue

problem:

$$\underbrace{\begin{pmatrix} A & 0 & B \\ 0 & -A^T & -C^T \\ C & B^T & W_0 \end{pmatrix}}_{\mathcal{M}} \underbrace{\begin{pmatrix} r \\ q \\ v \end{pmatrix}}_{\mathcal{N}} = s_0 \underbrace{\begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{\mathcal{N}} \underbrace{\begin{pmatrix} r \\ q \\ v \end{pmatrix}}_{\mathcal{N}}. \quad (5.56)$$

The pencil $(\mathcal{M}, \mathcal{N})$ has at least one infinite eigenvalue due to the singularity of W_0 . However, by the same argument used in Section 5.2.1.3, the finite purely imaginary eigenvalues $\mu_k = j\omega_k$ of the pencil still correspond to the frequencies ω_k where one eigenvalue of Ψ vanishes as $\lambda_i(j\omega_k) = 0$. Therefore, the passivity check detailed in Section 5.3.2 and Algorithm 5.1 can still be applied as far as the Hamiltonian matrix eigenvalue problem (5.16) is replaced by (5.56). Alternative approaches for handling this case, based on frequency transformations, can be found in [23, 72, 76].

5.6.2 Enforcing asymptotic passivity

When $W_0 = D + D^T$ is not sign definite, with at least one negative eigenvalue, most of the foregoing results do not apply if not properly generalized. For instance, the PRL condition (5.10) cannot be satisfied, since the model is not passive at infinite frequency. Therefore, the proposed system perturbation (5.21) for passivity enforcement will not be effective since also matrix D should be modified.

There are two main alternative approaches to recovering strict asymptotic passivity and enable all passivity enforcement schemes discussed in Section 5.5. One approach involves a preprocessing step that first modifies D so that its symmetric part is strictly positive definite. Let us compute the following eigendecomposition:

$$\frac{D + D^T}{2} = V\Lambda V^T \quad (5.57)$$

where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_p)$ collects the eigenvalues and V the corresponding eigenvectors. We can simply redefine the eigenvalues in (5.57) as $\hat{\lambda}_p = \max(\lambda_p, \varepsilon)$ where $\varepsilon > 0$ is a prescribed positive minimum value assigned to the eigenvalues. The resulting model

$$H_{ap}(s) = C(sI - A)^{-1}B + \hat{D}, \quad \hat{D} = V\text{diag}(\hat{\lambda}_1, \dots, \hat{\lambda}_p)V^T + \frac{D - D^T}{2} \quad (5.58)$$

is guaranteed to be asymptotically passive. This new model $H_{ap}(s)$ may exhibit a large deviation with respect to the original model response $H(s)$, since a constant term is added affecting the response at all frequencies. This accuracy loss can be partially compensated by a standard state-output matrix correction $\hat{C} = C + \delta C$, where δC is determined through

$$\min_{\delta C} \|\delta C (j\omega I - A)^{-1}B + \hat{D} - D\|^2 \quad (5.59)$$

where the norm is defined, e. g., as a data-based cost function at discrete frequencies ω_ℓ , as in Section 5.4.2.

A second (preferable) approach for handling models that are not asymptotically passive amounts to:

1. Allowing for a perturbation of the direct coupling matrix $\hat{D} = D + \delta D$ in addition to the usual state-output map. The model perturbation thus becomes

$$\delta H(s) = \delta C(sI - A)^{-1}B + \delta D = \begin{pmatrix} \delta C & \delta D \end{pmatrix} \begin{pmatrix} (sI - A)^{-1}B \\ I \end{pmatrix} \quad (5.60)$$

which is compatible with all previous derivations with obvious modifications. Note that, in this case, the Gramian-based cost functions become ill-defined since the L_2 norm of $\delta H(s)$ is not finite, and a data-based cost function over a limited bandwidth, such as (5.32), should be used during passivity enforcement.

2. Including an explicit local passivity constraint at $\omega = \infty$ during passivity enforcement. This constraint is just a simple particular case of (5.52), where $\delta D + \delta D^T$ replaces $\delta \Psi(j\omega_{kv})$.

We leave details of the above generalization to the reader.

5.6.3 Descriptor systems

Many model order reduction methods lead to systems in descriptor form

$$S: \begin{cases} E\dot{x}(t) = Ax(t) + Bu(t), \\ y(t) = Cx(t) + Du(t), \end{cases} \quad (5.61)$$

with a possibly singular matrix E , and often with $D = 0$, instead of the regular state-space form (5.1). A fundamental requirement to avoid an ill-defined (non-solvable) model is that the pencil (A, E) is regular with $|sE - A| \neq 0$ for some $s \in \mathbb{C}$. In the following, we only discuss the case of *impulse-free* or equivalently *index-one* systems, for which the transfer function

$$H(s) = C(sE - A)^{-1}B + D \quad (5.62)$$

has a finite asymptotic value $H_\infty = \lim_{s \rightarrow \infty} H(s)$. Descriptor systems with higher index require a special treatment⁷ which is outside the scope of this chapter. See [58, 84, 91, 96, 98] for details.

⁷ Index-two systems can be passive with a positive real transfer function $H(s)$ only when the leading asymptotic term $H(s) \sim sL_\infty$ for $s \rightarrow \infty$ is such that $L_\infty = L_\infty^T \geq 0$. Higher index systems are not passive and, in order to recover passivity, the high order impulsive part must be deflated.

For index-one descriptor systems the Hamiltonian-based passivity check is applicable with a minimal modification [91, 96, 98]. In fact, repeating the derivations of Sections 5.2.1.3 and 5.6.1 while using (5.61) as a starting point leads to the same generalized eigenvalue problem (5.56), but with \mathcal{N} redefined as

$$\mathcal{N} = \begin{pmatrix} E & 0 & 0 \\ 0 & E^T & 0 \\ 0 & 0 & 0 \end{pmatrix}. \quad (5.63)$$

Special care should be taken in the (generalized) Hamiltonian eigenvalue computation, for which structured eigensolvers should be preferred to general-purpose eigensolvers; see e. g. the implicitly restarted Krylov method of [59].

Passivity enforcement of descriptor systems via Hamiltonian eigenvalue perturbation is discussed in [83, 84, 91, 96, 97, 98] and further generalized to para-Hermitian pencils in [16]. The Gramian-based cost function for minimizing model perturbation of Section 5.4.1 should also be properly generalized; see [78, 84] and [5, Chapter 2] for details. Finally, we refer the reader to [30] for an extension of the Positive Real Lemma to descriptor systems.

5.6.4 Other supply rates

All above derivations and algorithms assume that the supply rate $s(u, y)$ through which power is delivered to the system from the environment is given by (5.7). However, this is not the only possible choice in general application fields. We review below the notable cases of scattering representations and general quadratic supply rates, discussing the various modifications that are required to define, check, and enforce passivity.

5.6.4.1 Scattering representations and bounded realness

The scattering representation is the most appropriate description of models in several application fields, in particular high-frequency electronics and electromagnetics. This is due to a number of reasons, including regularity and boundedness of the transfer function, as well as the ability to measure it with high accuracy. In scattering representations, the inputs u and outputs y are related to the power flow that is incident and reflected by the structure. In particular, the supply rate is defined as

$$s(u, y) = u^T u - y^T y = \|u\|^2 - \|y\|^2 \quad (5.64)$$

and is interpreted as the net power transferred to the system from the environment, with the term $\|u\|^2$ denoting the power flow incident into the system and $\|y\|^2$ the corresponding power flow that is reflected or scattered back into the environment [2, 4, 90].

The supply rate $s(u, y)$ in (5.64) leads to a set of passivity⁸ conditions that are listed below, and which are obtained by repeating the derivations of Section 5.2.1 while applying the appropriate modifications.

The KYP Lemma for scattering representation is known as *Bounded Real Lemma* (BRL) [2, 74] and states that a scattering state-space system (5.1) is passive if and only if

$$\exists P = P^T > 0 : \begin{pmatrix} A^T P + PA + C^T C & PB + C^T D \\ B^T P + D^T C & -(I - D^T D) \end{pmatrix} \leq 0. \quad (5.65)$$

This lemma can also be stated in the equivalent LMI form

$$\exists P = P^T > 0 : \begin{pmatrix} A^T P + PA & PB & C^T \\ B^T P & -I & D^T \\ C & D & -I \end{pmatrix} \leq 0. \quad (5.66)$$

A scattering system is passive when its transfer function $H(s)$ is *Bounded Real* (BR), i. e., the following three conditions hold [2, 81, 90]:

1. $H(s)$ must be regular in the open right half complex plane $\Re\{s\} > 0$;
2. $H(s^*) = H^*(s)$;
3. $\Psi(s) = I - H^T(-s)H(s) \geq 0$ for $\Re\{s\} > 0$.

These conditions should be compared to the PR conditions of Section 5.2.1.2, noting that the only difference between PR and BR is in the definition of the function $\Psi(s)$. Correspondingly, the frequency-domain inequality conditions for the passivity of a scattering system still require $\Psi(j\omega) \geq 0$ for all $\omega \in \mathbb{R}$, and can be expressed as in (5.12). An equivalent statement is based on the singular values of the transfer function

$$\sigma_i \leq 1, \quad \forall \sigma_i \in \sigma(H(j\omega)), \quad \forall \omega \in \mathbb{R}, \quad (5.67)$$

which implies in turn that passive scattering models must have a bounded and regular transfer function $H(s)$ when restricted to the imaginary axis $s = j\omega$, further requiring that the state-space matrix A must be asymptotically stable. Another yet equivalent condition for passivity is expressed in terms of the superior of the largest singular value throughout the imaginary axis, leading to the well-known \mathcal{H}_∞ norm condition

$$\|H\|_{\mathcal{H}_\infty} = \sup_{\omega \in \mathbb{R}} \sigma_{\max}(H(j\omega)) \leq 1. \quad (5.68)$$

The Hamiltonian matrix associated to a scattering state-space system (5.1) reads

$$\mathcal{M} = \begin{pmatrix} A - B(I - D^T D)^{-1} D^T C & -B(I - D^T D)^{-1} B^T \\ C^T (I - DD^T)^{-1} C & -A^T + C^T D(I - D^T D)^{-1} B^T \end{pmatrix}. \quad (5.69)$$

⁸ We retain the general term *passivity* also in the scattering (and for general quadratic supply rates), as a standard denomination in circuit, electronic and electromagnetic applications, although in some scientific communities this term is dedicated to immittance representations, and the term *dissipative* is used in the more general setting.

The system is passive if \mathcal{M} has no purely imaginary eigenvalues (strictly passive) or at most purely imaginary eigenvalues with even-sized Jordan blocks [12, 33]. The same considerations of Section 5.2.1.3 apply. When the model is not asymptotically passive for $\omega \rightarrow \infty$, then D has one unit singular value and the above Hamiltonian matrix becomes ill-defined. In this case, \mathcal{M} generalizes to the pencil $(\mathcal{M}, \mathcal{N})$ where

$$\mathcal{M} = \begin{pmatrix} A & 0 & B & 0 \\ 0 & -A^T & 0 & -C^T \\ 0 & B^T & -I & D^T \\ C & 0 & D & -I \end{pmatrix}, \quad \mathcal{N} = \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (5.70)$$

which replaces (5.56) for scattering representations [91, 94, 97, 98]. Finally, when the underlying system is in descriptor form (5.61), then we simply replace I with E in \mathcal{N} , as in (5.63).

With all above redefinitions of appropriate passivity conditions for scattering systems, all passivity check and enforcement algorithms discussed in Section 5.3 and Section 5.5 apply with obvious modifications.

5.6.4.2 General quadratic supply rates

Immittance and scattering representations are just particular cases of the more general situation in which the supply rate is a quadratic function of input and output variables. Such case is compactly described by

$$s(u, y) = \begin{pmatrix} u \\ y \end{pmatrix}^T \begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} \begin{pmatrix} u \\ y \end{pmatrix} \quad (5.71)$$

with $Q = Q^T$ and $R = R^T$, from which the immittance case (5.7) and the scattering case (5.64) are obtained by setting $Q = R = 0$, $S = I$ (up to the irrelevant scaling factor 1/2) and $Q = I$, $R = -I$, $S = 0$, respectively.

The LMI condition (KYP lemma) that characterizes a passive (dissipative) state-space system (5.1) with supply rate (5.71) reads

$$\exists P = P^T > 0 : \begin{pmatrix} A^T P + PA - C^T RC & PB - (SC)^T - C^T RD \\ B^T P - SC - D^T RC & -Q - SD - (SD)^T - D^T RD \end{pmatrix} \leq 0, \quad (5.72)$$

and the corresponding Frequency-Domain Inequality reads

$$\Psi(j\omega) = Q + H^H(j\omega)S^T + SH(j\omega) + H^H(j\omega)RH(j\omega) \geq 0, \quad \forall \omega \in \mathbb{R}. \quad (5.73)$$

Finally, the Hamiltonian matrix that generalizes (5.16) and (5.69) reads

$$\mathcal{M} = \begin{pmatrix} A - BW^{-1}Z & -BW^{-1}B^T \\ -C^T RC + Z^T W^{-1}Z & -A^T + Z^T W^{-1}B^T \end{pmatrix} \quad (5.74)$$

where $W = Q + SD + (SD)^T + D^T RD$ and $Z = (SC + D^T RC)$. We leave all details to the reader, pointing to [74, 88, 89] for a complete theoretical discussion. With suitable

modifications, all passivity verification and enforcement schemes of Section 5.3 and Section 5.5 are applicable to this general case as well.

5.6.5 Enforcing stability

All results and algorithms presented up to now are based on the fundamental assumption that the system at hand is asymptotically stable, with all eigenvalues of state matrix A , or pencil (A, E) in the descriptor case, having a strictly negative real part. Many MOR schemes are able to preserve stability if the original model is stable, for instance balanced truncation [5, Chapter 2] or Krylov subspace methods based on split congruence transformations such as PRIMA [6, Chapter 4]. Basic Arnoldi or Lanczos methods are instead not generally able to preserve stability in the reduced order model. Considering data-driven methods, the Vector Fitting algorithm [5, Chapter 8] incorporates a pole-flipping strategy that guarantees stability, whereas basic Loewner interpolation/reduction schemes [5, Chapter 6] do not guarantee stability. Further, even if a stability-preserving MOR method is used, roundoff errors in computer implementations may compromise the stability and may result in some eigenvalue with a positive real part.

Stabilization of a given model or system is a standard problem in Control Theory, where many alternative approaches usually based on feedback are routinely applied. The Reader is referred to any textbook such as [99]. The requirements we have in MOR applications are stronger than simple stabilization, since the final model should be as close as possible to the initial (unstable) model according to a prescribed performance metric or norm. Therefore, the simplistic approach of separating the unstable modes through an eigenvalue or, better, Schur decomposition and simply discarding them is not appropriate. Optimal stabilizing approximations are in fact available through robust and reliable algorithms. As an example, we refer the reader to [52], where some approaches for finding the closest stable system based on \mathcal{H}_2 and \mathcal{H}_∞ norms are introduced; see also [32].

5.6.6 Parameterized systems

Passivity verification and enforcement methods can be extended to parameterized systems, whose response $H(s, \vartheta)$ depends both on frequency s and (multivariate) parameters $\vartheta \in \Theta \subseteq \mathbb{R}^d$. In this framework, many different approaches and solutions have been proposed, depending on how parameters are embedded in the model and on how the model is constructed. A complete treatment would be outside the scope of this chapter, so that we discuss only a specific yet wide class of model parameterizations

$$H(s; \vartheta) = \frac{N(s, \vartheta)}{D(s, \vartheta)} = \frac{\sum_{n=0}^{\bar{n}} \sum_{\ell=1}^{\bar{\ell}} R_{n,\ell} \xi_\ell(\vartheta) \varphi_n(s)}{\sum_{n=0}^{\bar{n}} \sum_{\ell=1}^{\bar{\ell}} r_{n,\ell} \xi_\ell(\vartheta) \varphi_n(s)}, \quad (5.75)$$

where $R_{n,\ell} \in \mathbb{R}^{M \times M}$ and $r_{n,\ell} \in \mathbb{R}$ are the model coefficients, $\varphi_n(s)$, $\xi_\ell(\vartheta)$ are suitable basis functions representing the dependence of model numerator and denominator on frequency and parameters, respectively, and ℓ is a scalar index spanning the parameter basis set through a suitable linear ordering. The parameterization (5.75) includes as particular cases the multivariate barycentric form leading to the (parameterized) Loewner framework [50] (see [5, Chapter 6]) and the generalized Sanathanan–Koerner form [38, 80], which extends to the multivariate setting the Vector Fitting scheme [5, Chapter 8]. In the latter case the frequency basis functions are $\varphi_0(s) = 1$ and $\varphi_n(s) = (s - q_n)^{-1}$ for $n > 0$, where q_n are predefined stable “basis poles”, either real or in complex conjugate pairs, and the parameter-dependent basis functions ξ_ℓ can be orthogonal or trigonometric polynomials, or any other choice that is appropriate for the application at hand. A parameterized (descriptor) realization is easily obtained from (5.75) as (5.61), where

$$A = A(\vartheta) = \sum_{\ell=1}^{\bar{\ell}} A_\ell \xi_\ell(\vartheta), \quad C = C(\vartheta) = \sum_{\ell=1}^{\bar{\ell}} C_\ell \xi_\ell(\vartheta) \quad (5.76)$$

and E, B, D are constant.

One notable and simple approach to obtain a uniformly passive model, so that all passivity conditions discussed in Section 5.2.1 hold $\forall \vartheta \in \Theta$, is to suppress the denominator in (5.75) as $D(s) = 1$ and construct the parameterized system through interpolation of a set of non-parameterized models. This is achieved by choosing ξ_ℓ as interpolating, e. g. Lagrange, basis functions. There exist passivity-preserving interpolation schemes that ensure that, if the individual models being interpolated are passive, then also the interpolated parameterized model is passive $\forall \vartheta \in \Theta$. See [26, 27, 28, 29, 70] and the references therein for details on various alternative approaches within this framework.

A complementary approach is to consider the fully-parameterized model in form (5.75) and extend the passivity verification and enforcement methods of Section 5.3 and Section 5.5 to the multivariate case. The main difficulty that arises in this scenario is that the Hamiltonian matrix, which is the main tool providing localization of the passivity violations, becomes parameter-dependent due to (5.76). The convenience of the purely algebraic test based on its eigenvalues is partially lost, since the purely imaginary eigenvalues (if any) are parameter-dependent. A possible strategy for tracking these eigenvalues based on adaptive sampling in the parameter space is discussed in [95], where a first-order perturbation analysis on the full Hamiltonian eigenspectrum is used to determine the regions in the parameter space that need refined sampling, in order to track the boundaries between the regions defining passive and non-passive models. Figure 5.5 provides an illustration by depicting the results of this adaptive sampling process in a case with two parameters $d = 2$. If any passivity violation region is detected (the red dots in Figure 5.5, left panel), then the worst-case passivity violations are determined as in Algorithm 5.1 and a multivariate extension of Algorithm 5.3 is applied to eliminate them. All details are available in [37, 40, 95].

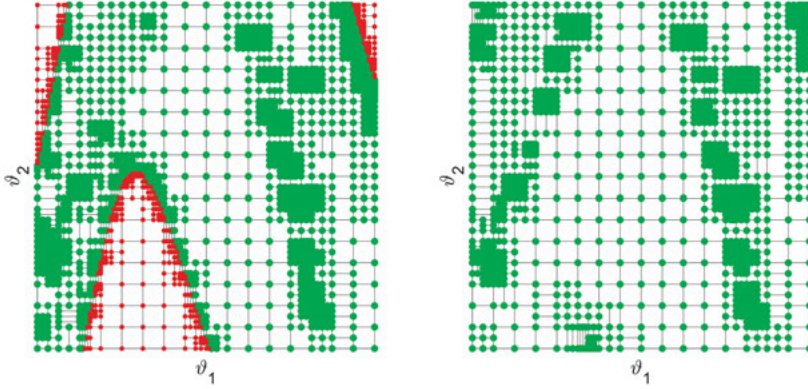


Figure 5.5: Adaptive sampling in a two-dimensional parameter space, applied to a non-passive model (left panel) and to the corresponding passive model after enforcement (right). Each dot represents a non-parameterized model instance obtained by evaluating the parameterized model (5.75) at the corresponding sampling point. Each dot is colored in green/red if the corresponding model instance is locally passive/non-passive, respectively, as resulting from the absence/presence of imaginary Hamiltonian eigenvalues. Iterative refinement leads to tracking the boundaries between passive/non-passive regions. Courtesy of A. Zanco, Politecnico di Torino.

5.7 Examples

5.7.1 A high-speed interconnect in a mobile device

The passivity enforcement process is here applied to a model of a high-speed interconnect providing a data link in a smartphone. An initial characterization of the structure was obtained through a full-wave numerical simulation of the time-harmonic Maxwell's equations, which provided a set of frequency samples of the 4×4 (scattering) transfer function $S(j\omega)$ from 0 to 50 GHz. These samples were processed by Vector Fitting [5, Chapter 8], obtaining a rational approximation of the system responses. This rational approximation was then converted to a state-space realization as in [5, Chapter 8]. The accuracy of the rational approximation is excellent, as depicted in the two top panels of Figure 5.6.

A Hamiltonian-based passivity check on this model reveals the presence of $K = 10$ purely imaginary Hamiltonian eigenvalues $\mu_k = j\omega_k$ (see Figure 5.7, left panel). Correspondingly, a sweep of the model singular values $\sigma_i(H(j\omega))$ (see the top panel of Figure 5.8) up to a maximum frequency slightly beyond ω_K reveals a few evident passivity violations, corresponding to singular value trajectories exceeding the passivity threshold $\sigma = 1$. The local singular value maxima are highlighted with red dots in Figure 5.8.

Figure 5.8 depicts the typical situation that arises when fitting a rational model to response data over a finite frequency band: passivity violations usually occur at frequencies that fall outside the range where data samples are available. The fact that

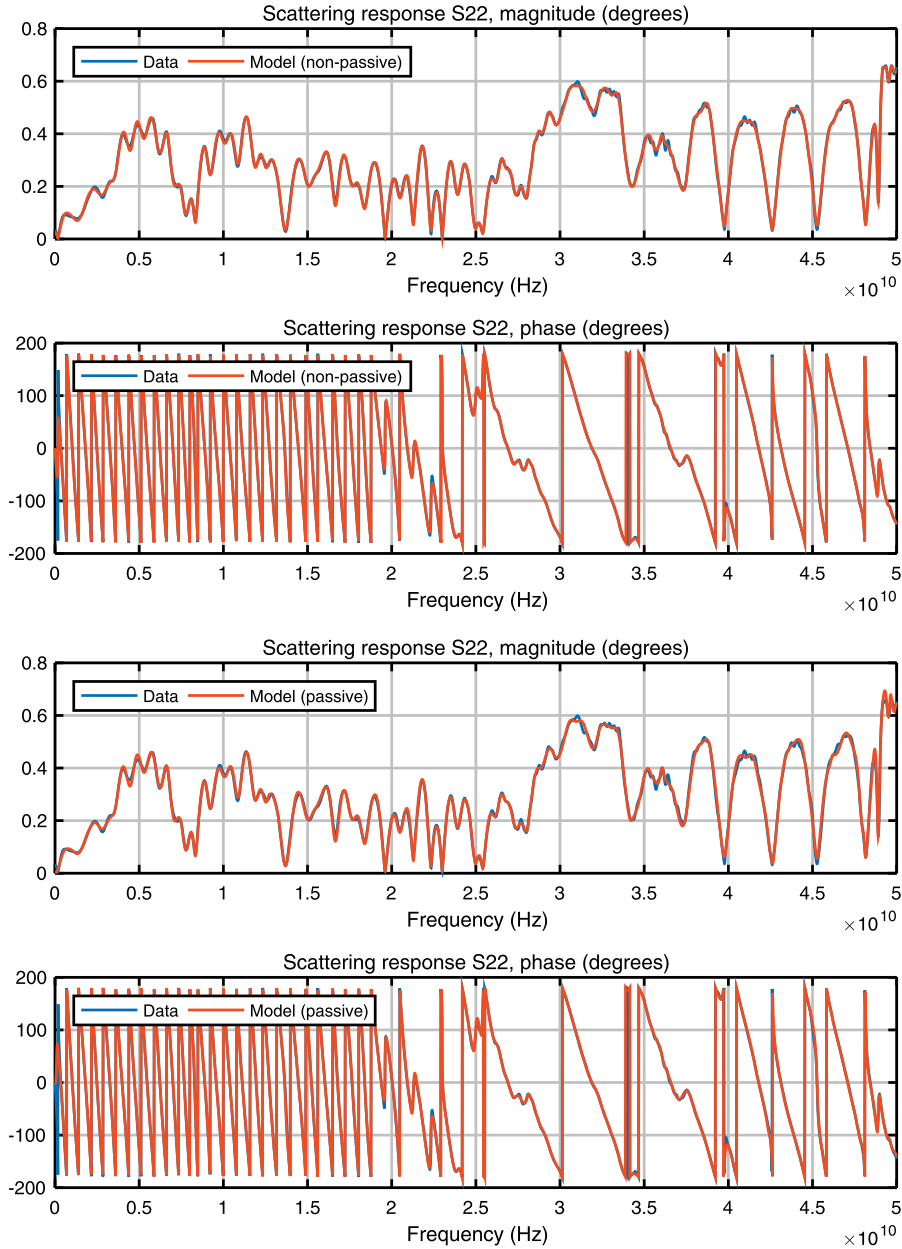


Figure 5.6: Comparison between model and original data used for model extraction for the smart-phone interconnect. For illustration, only response $S_{22}(j\omega)$ of the 4×4 scattering matrix is reported. Top two panels refer to the initial non-passive model, whereas bottom two panels refer to the model after passivity enforcement.

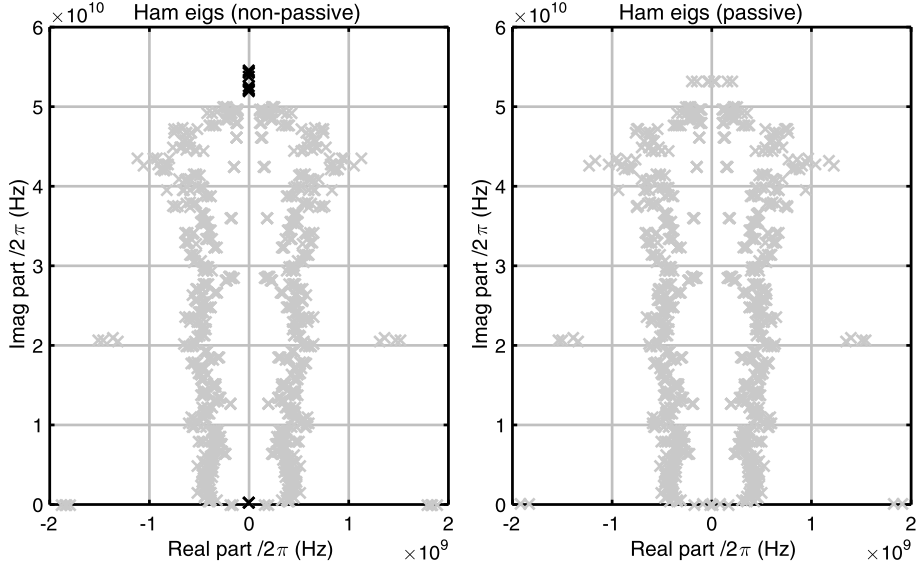


Figure 5.7: Hamiltonian eigenvalues before (left) and after (right) passivity enforcement for the smartphone interconnect model (only eigenvalues with positive imaginary parts are shown). The purely imaginary eigenvalues are highlighted with a darker color in the left panel.

such violations are not located within the modeling bandwidth may induce a false sense of confidence in the model user, who may argue that out-of-band passivity violations are unimportant, since located at frequency ranges that are not of interest. In fact, a time-domain simulation of the model using a transient ODE solver is agnostic whether the passivity violation occurs within or off-band: during time-stepping, numerical approximation errors due to the adopted ODE solver will inevitably excite those frequencies where the model amplifies energy, leading to instability. This is exactly what happens in Figure 5.1, where the thin blue line demonstrates the instability induced by this initial non-passive model. In this simulation scenario, the model was interconnected to a set of other linear (passive) circuits, and it was indeed possible to determine exactly the two poles $p = 2\pi(\alpha \pm j\beta)$ that are responsible for this instability, obtaining $\alpha = +1.13 \times 10^8$ Hz and $\beta = 5.32 \times 10^{10}$ Hz. The real part is positive, and the imaginary part nearly matches the frequency of the singular value peak; see Figure 5.8. This is exactly the frequency where the model injects energy into the system. See [35] for additional details on destabilization of non-passive models.

Enforcing model passivity removes the instability, as we already know from Figure 5.1. Application of Algorithm 5.3 leads to a passive model in 5 iterations, documented by the singular value plots in the various panels of Figure 5.8. The final passive model has no purely imaginary Hamiltonian eigenvalues, as evident from the right panel of Figure 5.7, and its responses still match very accurately the original data samples, as depicted in the bottom panel of Figure 5.6.

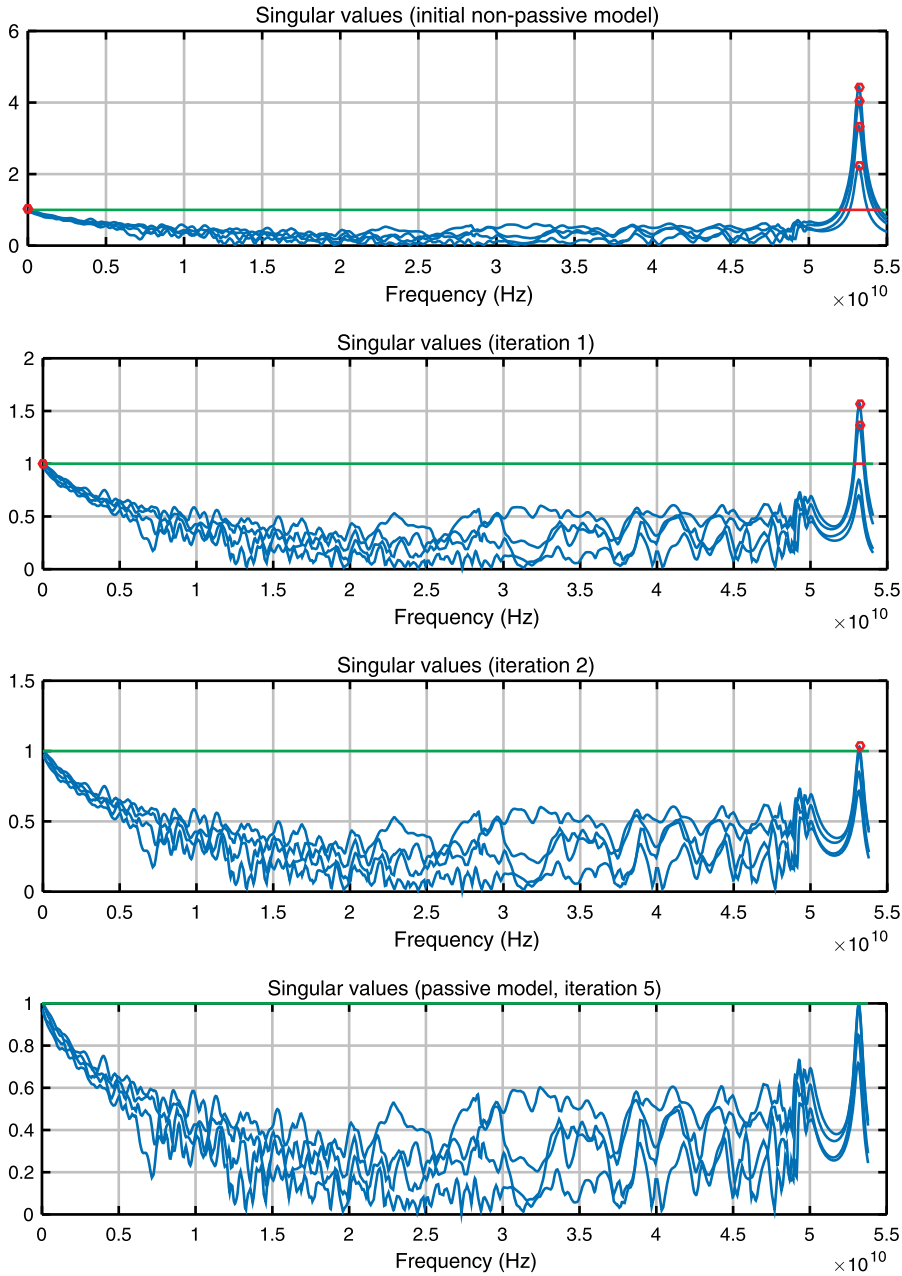


Figure 5.8: Evolution of the singular values (blue solid lines) of the smartphone interconnect during passivity enforcement iterations through Algorithm 5.3. Passive and non-passive frequency bands are highlighted with green and red color, respectively. Local maxima of all singular value trajectories in each non-passive frequency band, which are used to set up local passivity constraints, are highlighted with red dots.

5.7.2 An interconnect link on a high-performance PCB

We consider here the coupled interconnect link on a high-performance Printed Circuit Board (PCB), already discussed in [5, Chapter 8], where an accurate model was extracted using the Vector Fitting algorithm from scattering measurements performed on the real hardware. As depicted in [5, Chapter 8], Figures 6 and 7, the model responses of this initial model are visually undistinguishable from the measured samples, with a model-data error of $1.34 \cdot 10^{-3}$ (worst-case RMS error among all responses).

A passivity check performed on this initial model reveals some small passivity violations at low frequencies. This is actually expected, since the system is almost lossless at low frequency, and passivity violations induced by the rational approximation process of VF are therefore more likely than at high frequency, where energy dissipation is more pronounced. The passivity violations are detected by the presence of eight pairs of purely imaginary Hamiltonian eigenvalues, depicted in Figure 5.9, panels (a), (c) and (d). The corresponding frequencies denote crossings of the singular

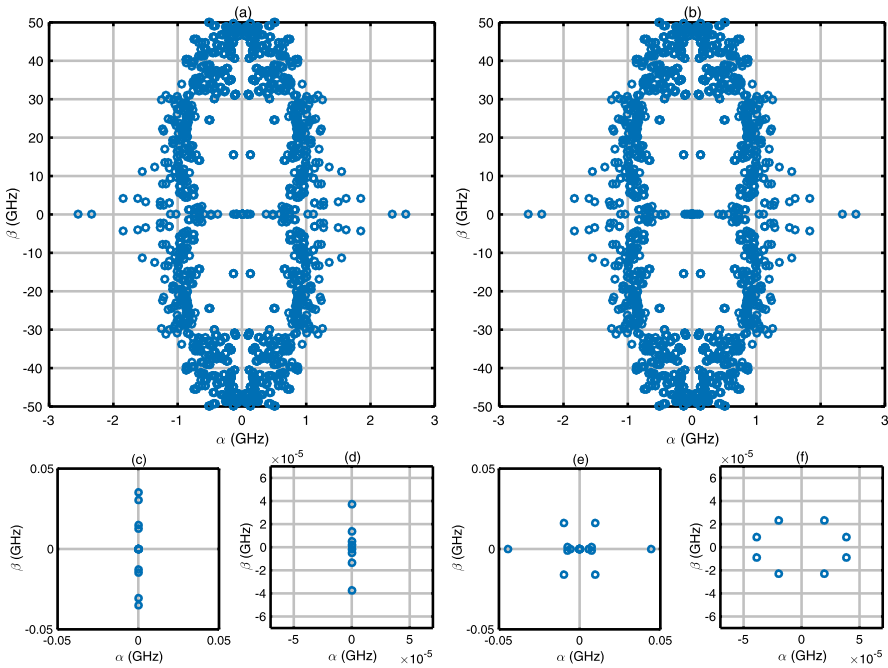


Figure 5.9: Hamiltonian eigenvalues $\alpha + j\beta = \mu/2\pi$ of the PCB interconnect model. Panels (a), (c), (d): original model after Vector Fitting; panels (b), (e), (f): model after passivity enforcement. Top panels (a), (b) depict the full Hamiltonian eigenspectrum. Bottom panels (c), (d) and (e), (f) are enlarged views of the top panels (a) and (b), respectively, at different magnification levels. Panels (e) and (f) show that all purely imaginary Hamiltonian eigenvalues clustered at low frequencies, depicted in panels (c) and (d), are effectively removed by passivity enforcement.

value trajectories $\sigma_i(j\omega)$ of the threshold $\sigma = 1$, as depicted in Figure 5.10, top panel. After few iterations of Algorithm 5.3 all these passivity violations are removed. As the bottom panel of Figure 5.10 shows, all singular value trajectories of the passive model are uniformly bounded by one. This is further confirmed by panels (e) and (f) of Figure 5.9, which show that all purely imaginary eigenvalues of the initial model are now displaced from the imaginary axis.

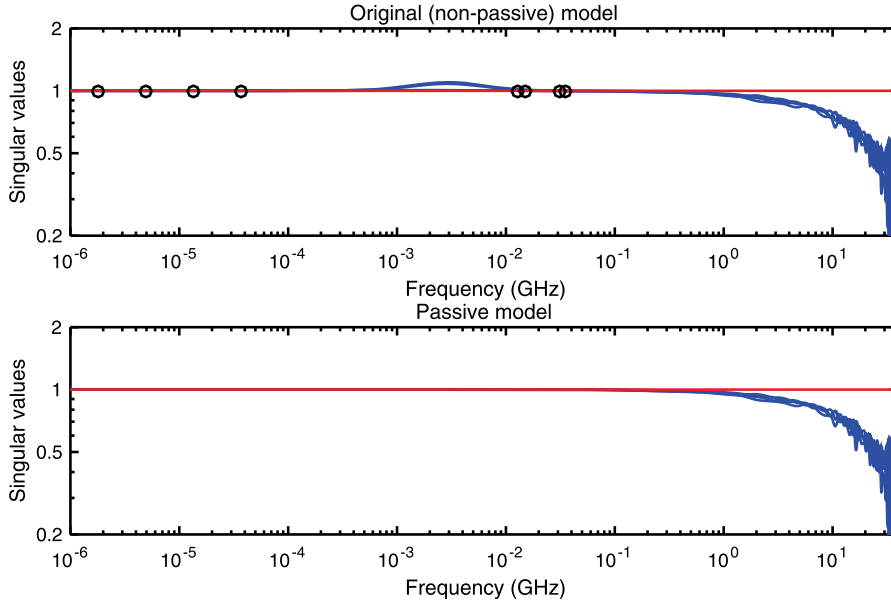


Figure 5.10: Top panel: singular value trajectories (blue lines) of the initial (non-passive) PCB interconnect model, revealing low-frequency passivity violations exceeding the passivity threshold $\sigma = 1$ (red line). Black dots correspond to the frequencies of the purely imaginary Hamiltonian eigenvalues; see Figure 5.9, panels (c) and (d). Bottom panel: singular value trajectories after passivity enforcement, which are uniformly below the passivity threshold.

The passivity enforcement process did not spoil model accuracy. Figure 5.11 compares the scattering responses (1, 2) and (1, 3) of the passive model to the raw measured samples from which the initial model was derived (the same responses already depicted in [5, Chapter 8], Figures 6 and 7). Also for the passive model the responses closely match the measurements, with a worst-case RMS model-data error of $1.38 \cdot 10^{-3}$.

5.8 Conclusions

The goal of this chapter was to survey the most widely used techniques for enforcing passivity of reduced order models. To motivate the ensuing description, a simple

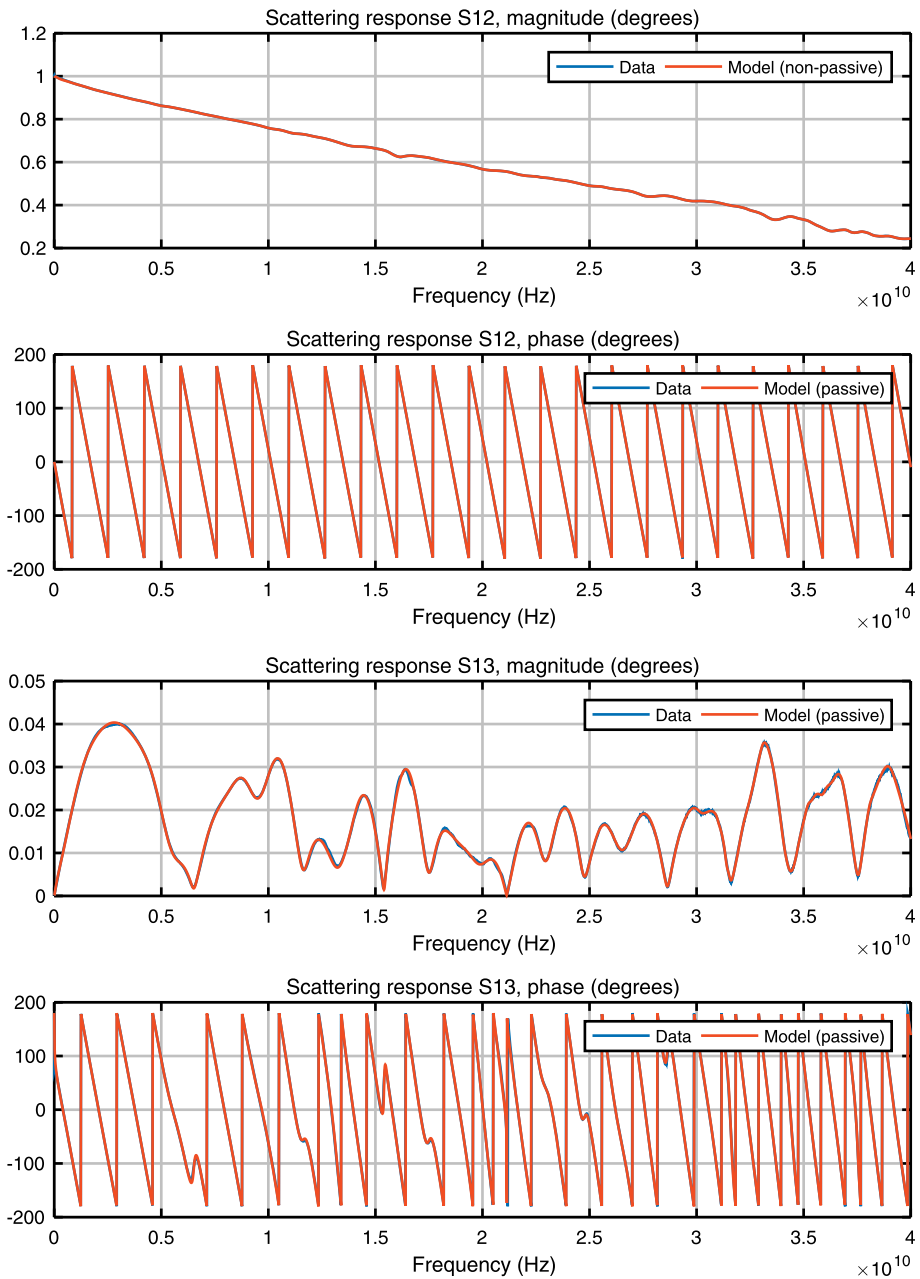


Figure 5.11: Comparison between passive model responses and measured data used for model identification for the high-speed PCB interconnect of Section 5.7.2.

example was shown that illustrates in striking fashion the need for ensuring passivity in models. The focus of the chapter was on Linear Time-Invariant (LTI) systems in state-space form, although the techniques reviewed are applicable in other representations with appropriate modifications. Conditions for testing the passivity of a given LTI model as well as approaches for perturbing non-passive systems in order to enforce passivity were reviewed and examples were shown to demonstrate the application of such techniques to realistic cases.

Bibliography

- [1] R. Alam, S. Bora, M. Karow, V. Mehrmann, and J. Moro. Perturbation theory for Hamiltonian matrices and the distance to bounded-realness. *SIAM J. Matrix Anal. Appl.*, 32(2):484–514, 2011.
- [2] B. D. O. Anderson and S. Vongpanitlerd. *Network analysis and synthesis*. Prentice-Hall, 1973.
- [3] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe and H. van der Vorst. *Templates for the solution of Algebraic Eigenvalue Problems: A Practical Guide*. Society for Industrial and Applied Mathematics, 2000.
- [4] V. Belevitch. *Classical network theory*. Holden-Day, 1968.
- [5] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders and L. Silveira. *Model Order Reduction. Volume 1: System- and Data-Driven Methods and Algorithms*. De Gruyter, Berlin, 2020.
- [6] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders and L. Silveira. *Model Order Reduction. Volume 3: Applications*. De Gruyter, Berlin, 2020.
- [7] P. Benner and D. Kressner. Algorithm 854: Fortran 77 subroutines for computing the eigenvalues of Hamiltonian matrices II. *ACM Trans. Math. Softw.*, 32(2):352–373, 2006.
- [8] P. Benner and D. Kressner. Balancing sparse Hamiltonian eigenproblems. *Linear Algebra Appl.*, 415(1):3–19, 2006. Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems.
- [9] P. Benner, D. Kressner, and V. Mehrmann. Skew-Hamiltonian and Hamiltonian eigenvalue problems: Theory, algorithms and applications. In Z. Drmac, M. Marusic and Z. Tutek, editors, *Proceedings of the Conference on Applied Mathematics and Scientific Computing*, pages 3–39. Springer, Netherlands, 2005.
- [10] P. Benner, V. Mehrmann, V. Sima, S. Huffel, and A. Varga. SLICOT—a subroutine library in systems and control theory. In B. Datta, editor, *Applied and Computational Control, Signals, and Circuits*, pages 499–539. Birkhäuser, Boston, 1999.
- [11] P. Benner and M. Voigt. Spectral characterization and enforcement of negative imaginarity for descriptor systems. *Linear Algebra Appl.*, 439(4):1104–1129, 2013. 17th Conference of the International Linear Algebra Society, Braunschweig, Germany, August 2011.
- [12] S. Boyd, V. Balakrishnan, and P. Kabamba. A bisection method for computing the H_∞ norm of a transfer matrix and related problems. *Math. Control Signals Syst.*, 2(3):207–219, 1989.
- [13] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. Society for Industrial and Applied Mathematics, 1994.
- [14] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.
- [15] B. Brogliato, R. Lozano, B. Maschke, and O. Egeland. *Dissipative systems analysis and control: theory and applications*. Springer, 2007.
- [16] T. Brull and C. Schroder. Dissipativity enforcement via perturbation of para-Hermitian pencils. *IEEE Trans. Circuits Syst. I, Regul. Pap.*, 60(1):164–177, 2013.

- [17] A. Buscarino, L. Fortuna, M. Frasca, and M. G. Xibilia. An analytical approach to one-parameter MIMO systems passivity enforcement. *Int. J. Control*, 85(9):1235–1247, 2012.
- [18] H. Chen and J. Fang. Enforcing bounded realness of S parameter through trace parameterization. In *Electrical Performance of Electronic Packaging, 2003*, pages 291–294, 2003.
- [19] X. Chen and J. T. Wen. Positive realness preserving model reduction with H_∞ norm error bounds. *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, 42(1):23–29, 1995.
- [20] A. Chinae, S. Grivet-Talocia, S. B. Olivadese, and L. Gobbato. High-performance passive macromodeling algorithms for parallel computing platforms. *IEEE Trans. Compon. Packag. Manuf. Technol.*, 3(7):1188–1203, 2013.
- [21] C. P. Coelho, J. Phillips, and L. M. Silveira. A convex programming approach for generating guaranteed passive approximations to tabulated frequency-data. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 23(2):293–301, 2004.
- [22] D. Deschrijver and T. Dhaene. Fast passivity enforcement of S-parameter macromodels by pole perturbation. *IEEE Trans. Microw. Theory Tech.*, 57(3):620–626, 2009.
- [23] D. Deschrijver and T. Dhaene. Modified half-size test matrix for robust passivity assessment of S-parameter macromodels. *IEEE Microw. Wirel. Compon. Lett.*, 19(5):263–265, 2009.
- [24] C. A. Desoer and E. S. Kuh. *Basic circuit theory*. McGraw-Hill, 1984.
- [25] B. Dumitrescu. Parameterization of positive-real transfer functions with fixed poles. *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, 49(4):523–526, 2002.
- [26] F. Ferranti, T. Dhaene, and L. Knockaert. Compact and passive parametric macromodeling using reference macromodels and positive interpolation operators. *IEEE Trans. Compon. Packag. Manuf. Technol.*, 2(12):2080–2088, 2012.
- [27] F. Ferranti, L. Knockaert, and T. Dhaene. Parameterized S-parameter based macromodeling with guaranteed passivity. *IEEE Microw. Wirel. Compon. Lett.*, 19(10):608–610, 2009.
- [28] F. Ferranti, L. Knockaert, and T. Dhaene. Guaranteed passive parameterized admittance-based macromodeling. *IEEE Trans. Adv. Packaging*, 33(3):623–629, 2010.
- [29] F. Ferranti, L. Knockaert, T. Dhaene, G. Antonini, and D. De Zutter. Parametric macromodeling for tabulated data based on internal passivity. *IEEE Microw. Wirel. Compon. Lett.*, 20(10):533–535, 2010.
- [30] R. W. Freund and F. Jarre. An extension of the positive real lemma to descriptor systems. *Optim. Methods Softw.*, 19(1):69–87, 2004.
- [31] S. Gao, Y.-S. Li, and M.-S. Zhang. An efficient algebraic method for the passivity enforcement of macromodels. *IEEE Trans. Microw. Theory Tech.*, 58(7):1830–1839, 2010.
- [32] I. V. Gosea and A. C. Antoulas. Stability preserving post-processing methods applied in the Loewner framework. In *2016 IEEE 20th Workshop on Signal and Power Integrity (SPI)*, pages 1–4, 2016.
- [33] S. Grivet-Talocia. Passivity enforcement via perturbation of Hamiltonian matrices. *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, 51(9):1755–1769, 2004.
- [34] S. Grivet-Talocia. An adaptive sampling technique for passivity characterization and enforcement of large interconnect macromodels. *IEEE Trans. Adv. Packaging*, 30(2):226–237, 2007.
- [35] S. Grivet-Talocia. On driving non-passive macromodels to instability. *Int. J. Circuit Theory Appl.*, 37(8):863–886, 2009.
- [36] S. Grivet-Talocia. On passivity characterization of symmetric rational macromodels. *IEEE Trans. Microw. Theory Tech.*, 58(5):1238–1247, 2010.
- [37] S. Grivet-Talocia. A perturbation scheme for passivity verification and enforcement of parameterized macromodels. *IEEE Trans. Compon. Packag. Manuf. Technol.*, 7(11):1869–1881, 2017.

- [38] S. Grivet-Talocia and E. Fevola. Compact parameterized black-box modeling via Fourier-rational approximations. *IEEE Trans. Electromagn. Compat.*, 59(4):1133–1142, 2017.
- [39] S. Grivet-Talocia and B. Gustavsen. *Passive Macromodeling: Theory and Applications*. John Wiley and Sons, New York, 2016 (published online on Dec 7, 2015).
- [40] S. Grivet-Talocia and R. Trinchero. Behavioral, parameterized, and broadband modeling of wired interconnects with internal discontinuities. *IEEE Trans. Electromagn. Compat.*, 60(1):77–85, 2018.
- [41] S. Grivet-Talocia and A. Ubolli. On the generation of large passive macromodels for complex interconnect structures. *IEEE Trans. Adv. Packaging*, 29(1):39–54, 2006.
- [42] S. Grivet-Talocia and A. Ubolli. Passivity enforcement with relative error control. *IEEE Trans. Microw. Theory Tech.*, 55(11):2374–2383, 2007.
- [43] S. Grivet-Talocia and A. Ubolli. A comparative study of passivity enforcement schemes for linear lumped macromodels. *IEEE Trans. Adv. Packaging*, 31(4):673–683, 2008.
- [44] B. Gustavsen. Computer code for passivity enforcement of rational macromodels by residue perturbation. *IEEE Trans. Adv. Packaging*, 30(2):209–215, 2007.
- [45] B. Gustavsen. Fast passivity enforcement for pole-residue models by perturbation of residue matrix eigenvalues. *IEEE Trans. Power Deliv.*, 23(4):2278–2285, 2008.
- [46] B. Gustavsen and A. Semlyen. Enforcing passivity for admittance matrices approximated by rational functions. *IEEE Trans. Power Syst.*, 16(1):97–104, 2001.
- [47] B. Gustavsen and A. Semlyen. Fast passivity assessment for S-parameter rational models via a half-size test matrix. *IEEE Trans. Microw. Theory Tech.*, 56(12):2701–2708, 2008.
- [48] B. Gustavsen and A. Semlyen. On passivity tests for unsymmetrical models. *IEEE Trans. Power Deliv.*, 24(3):1739–1741, 2009.
- [49] D. Henrion. Course on LMI optimization with applications in control, Czech Technical University, Prague, Czech Republic, April 2013. <http://homepages.laas.fr/henrion/courses/lmi13/>. Accessed: 2018-11-27.
- [50] A. Ionita and A. Antoulas. Data-driven parametrized model reduction in the loewner framework. *SIAM J. Sci. Comput.*, 36(3):A984–A1007, 2014.
- [51] R. E. Kalman. Lyapunov functions for the problem of Lur’e in automatic control. *Proc. Natl. Acad. Sci.*, 49(2):201–205, 1963.
- [52] M. Köhler. On the closest stable descriptor system in the respective spaces RH_2 and RH_∞ . *Linear Algebra Appl.*, 443:34–49, 2014.
- [53] A. Lamecki and M. Mrozowski. Equivalent SPICE circuits with guaranteed passivity from nonpassive models. *IEEE Trans. Microw. Theory Tech.*, 55(3):526–532, 2007.
- [54] P. Lancaster and L. Rodman. Existence and uniqueness theorems for the algebraic Riccati equation. *Int. J. Control*, 32(2):285–309, 1980.
- [55] J. L. Yalmip. A toolbox for modeling and optimization in Matlab. In *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, pages 284–289. IEEE, 2004.
- [56] A. I. Lur’e. Some Non-linear Problems in the Theory of Automatic Control: Nekotorye Nelineinye Zadachi Teorii Avtomaticheskogo Regulirovaniya (Gos. Isdat. Tekh. Teor. Lit., 1951, U. S. S. R.) A Translation from the Russian. H.M. Stationery Office, 1957.
- [57] M. A. Mabrok, A. G. Kallapur, I. R. Petersen, and A. Lanzon. Enforcing a system model to be negative imaginary via perturbation of Hamiltonian matrices. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 3748–3752, 2011.
- [58] R. März. Canonical projectors for linear differential algebraic equations. *Comput. Math. Appl.*, 31(4–5):121–135, 1996. Selected Topics in Numerical Methods.
- [59] V. Mehrmann, C. Schröder, and V. Simoncini. An implicitly-restarted Krylov subspace method for real symmetric/skew-symmetric eigenproblems. *Linear Algebra Appl.*, 436(10):4070–4087, 2012.

- [60] V. Mehrmann and D. Watkins. Structure-preserving methods for computing eigenpairs of large sparse skew-Hamiltonian/Hamiltonian pencils. *SIAM J. Sci. Comput.*, 22(6):1905–1925, 2001.
- [61] V. Mehrmann and H. Xu. Perturbation of purely imaginary eigenvalues of Hamiltonian matrices under structured perturbations. *Electron. J. Linear Algebra*, 17:234–257, 2008.
- [62] R. Ober. Balanced parametrization of classes of linear systems. *SIAM J. Control Optim.*, 29:1251, 1991.
- [63] A. Odabasioglu, M. Celik, and L. T. Pileggi. PRIMA: Passive reduced-order interconnect macromodeling algorithm. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 17(8):645–654, 1998.
- [64] P. C. Opdenacker and E. A. Jonckheere. A contraction mapping preserving balanced reduction scheme and its infinity norm error bounds. *IEEE Trans. Circuits Syst.*, 35(2):184–189, 1988.
- [65] J. R. Phillips, L. Daniel, and L. M. Silveira. Guaranteed passive balancing transformations for model order reduction. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 22(8):1027–1041, 2003.
- [66] V.-M. Popov. Absolute stability of nonlinear systems of automatic control. *Autom. Remote Control*, 22(8):857–875, 1962.
- [67] T. Reis. Circuit synthesis of passive descriptor systems: a modified nodal approach. *Int. J. Circuit Theory Appl.*, 38(1):44–68, 2010.
- [68] T. Reis and T. Stykel. Positive real and bounded real balancing for model reduction of descriptor systems. *Int. J. Control*, 83(1):74–88, 2010.
- [69] W. Rudin. *Real and Complex Analysis*, 3rd edition. McGraw-Hill, Inc., New York, NY, USA, 1987.
- [70] E. R. Samuel, L. Knockaert, F. Ferranti, and T. Dhaene. Guaranteed passive parameterized macromodeling by using Sylvester state-space realizations. *IEEE Trans. Microw. Theory Tech.*, 61(4):1444–1454, 2013.
- [71] D. Saraswat, R. Achar, and M. S. Nakhla. Global passivity enforcement algorithm for macromodels of interconnect subnetworks characterized by tabulated data. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 13(7):819–832, 2005.
- [72] D. Saraswat, R. Achar, and M. S. Nakhla. Fast passivity verification and enforcement via reciprocal systems for interconnects with large order macromodels. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 15(1):48–59, 2007.
- [73] C. S. Saunders, J. Hu, C. E. Christoffersen, and M. B. Steer. Inverse singular value method for enforcing passivity in reduced-order models of distributed structures for transient and steady-state simulation. *IEEE Trans. Microw. Theory Tech.*, 59(4):837–847, 2011.
- [74] C. Scherer and S. Weiland. *Linear matrix inequalities in control. Lecture Notes*. Dutch Institute for Systems and Control, Delft, The Netherlands, 2000.
- [75] A. Semlyen and B. Gustavsen. A half-size singularity test matrix for fast and reliable passivity assessment of rational models. *IEEE Trans. Power Deliv.*, 24(1):345–351, 2009.
- [76] R. N. Shorten, P. Curran, K. Wulff, and E. Zeheb. A note on spectral conditions for positive realness of transfer function matrices. *IEEE Trans. Autom. Control*, 53(5):1258–1261, 2008.
- [77] J. F. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optim. Methods Softw.*, 11(1–4):625–653, 1999.
- [78] T. Stykel. Gramian-based model reduction for descriptor systems. *Math. Control Signals Syst.*, 16(4):297–319, 2004.
- [79] F. Tisseur. A chart of backward errors for singly and doubly structured eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 24(3):877–897, 2003.
- [80] P. Triverio, S. Grivet-Talocia, and M. S. Nakhla. A parameterized macromodeling strategy with uniform stability test. *IEEE Trans. Adv. Packaging*, 32(1):205–215, 2009.
- [81] P. Triverio, S. Grivet-Talocia, M. S. Nakhla, F. Canavero, and R. Achar. Stability, causality, and passivity in electrical interconnect models. *IEEE Trans. Adv. Packaging*, 30(4):795–808, 2007.
- [82] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Rev.*, 38(1):49–95, 1996.

- [83] Y. Wang, Z. Zhang, C.-K. Koh, G. K.-H. Pang, and N. Wong PEDS. Passivity enforcement for descriptor systems via Hamiltonian-symplectic matrix pencil perturbation. In *Computer-Aided Design (ICCAD), 2010 IEEE/ACM International Conference on*, pages 800–807, 2010.
- [84] Y. Wang, Z. Zhang, C.-K. Koh, G. Shi, G. K.-H. Pang, and N. Wong. Passivity enforcement for descriptor systems via matrix pencil perturbation. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 31(4):532–545, 2012.
- [85] D. S. Watkins. On Hamiltonian and symplectic Lanczos processes. *Linear Algebra Appl.*, 385(0):23–45, 2004. Special Issue in honor of Peter Lancaster.
- [86] J. H. Wilkinson. *The algebraic eigenvalue problem*. Clarendon Press, 1965.
- [87] J. C. Willems. Least squares stationary optimal control and the algebraic Riccati equation. *IEEE Trans. Autom. Control*, 16(6):621–634, 1971.
- [88] J. C. Willems. Dissipative dynamical systems part I: General theory. *Arch. Ration. Mech. Anal.*, 45(5):321–351, 1972.
- [89] J. C. Willems. Dissipative dynamical systems part II: Linear systems with quadratic supply rates. *Arch. Ration. Mech. Anal.*, 45(5):352–393, 1972.
- [90] M. R. Wohlers. *Lumped and Distributed Passive Networks*. Academic press, 1969.
- [91] N. Wong and C.-K. Chu. A fast passivity test for stable descriptor systems via skew-Hamiltonian/Hamiltonian matrix pencil transformations. *IEEE Trans. Circuits Syst. I, Regul. Pap.*, 55(2):635–643, 2008.
- [92] V. A. Yakubovich. Solution of certain matrix inequalities encountered in non-linear control theory. In *Doklady Akademii Nauk*, volume 156, pages 278–281. Russian Academy of Sciences, 1964.
- [93] Z. Ye, L. M. Silveira, and J. R. Phillips. Fast and reliable passivity assessment and enforcement with extended Hamiltonian pencil. In *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pages 774–778, 2009.
- [94] Z. Ye, L. M. Silveira, and J. R. Phillips. Extended Hamiltonian pencil for passivity assessment and enforcement for S-parameter systems. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pages 1148–1152, 2010.
- [95] A. Zanco, S. Grivet-Talocia, T. Bradde, and M. De Stefano. Enforcing passivity of parameterized LTI macromodels via hamiltonian-driven multivariate adaptive sampling. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 39(1):225–238, 2020.
- [96] Z. Zhang and N. Wong. An efficient projector-based passivity test for descriptor systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 29(8):1203–1214, 2010.
- [97] Z. Zhang and N. Wong. Passivity check of S-parameter descriptor systems via S-parameter generalized Hamiltonian methods. *IEEE Trans. Adv. Packaging*, 33(4):1034–1042, 2010.
- [98] Z. Zhang and N. Wong. Passivity test of immittance descriptor systems based on generalized Hamiltonian methods. *IEEE Trans. Circuits Syst. II, Express Briefs*, 57(1):61–65, 2010.
- [99] K. Zhou, J. C. Doyle, and K. Glover. *Robust and optimal control*. Prentice Hall, Upper Saddle River, NJ, 1996.

Dimitrios S. Karachalios, Ion Victor Gosea, and
Athanasios C. Antoulas

6 The Loewner framework for system identification and reduction

Abstract: One of the main approaches to model reduction of both linear and nonlinear dynamical systems is by means of interpolation. This approach seeks reduced models whose transfer function matches that of the original system at selected interpolation points. Data-driven methods constitute an important special case. We start with an account of the Loewner framework in the linear case [52]. It constructs models from given data in a straightforward manner. An important attribute is that it provides a trade-off between accuracy of fit and complexity of the model. We compare this approach with other approximation methods and apply it to different test-cases. One of the case studies to which we apply the aforementioned methods is defined by the inverse of the Bessel function. We then turn our attention to the approximation of an Euler–Bernoulli beam model with Rayleigh damping. Further case studies include the approximation of two real valued functions with specific difficulties (discontinuity, sharp peaks). One computational tool is the SVD; its complexity is cubic in the number of data points. For large data sets the CUR factorization is a viable alternative. Note that its complexity is cubic as well but with respect to the dimension of the reduced order model (ROM). Another option is to use stochastic procedures such as randomized singular value decomposition (r-SVD) [41].

Keywords: Loewner framework, rational interpolation, model order reduction, data-driven, system identification, infinite dimensional systems


MSC 2010: 15-00, 37-00, 41-00, 49-00, 65-00, 93-00

6.1 Introduction

A challenging problem that computational linear algebra deals with is that of big data modeling. The problem consists mainly in constructing reduced complexity systems from input/output data. This contribution focuses on reduction via interpolation. The

Dimitrios S. Karachalios, Ion Victor Gosea, Data-Driven System Reduction and Identification (DRI) group, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstrasse 1, 39106 Magdeburg, Germany

Athanasios C. Antoulas, Data-Driven System Reduction and Identification (DRI) group, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstrasse 1, 39106 Magdeburg, Germany; and Electrical and Computer Engineering Department, Rice University Houston, 6100 Main St., Houston, TX 77005, USA; and Baylor College of Medicine, 1 Baylor Plaza, Houston, TX 77030, USA

Open Access. © 2021 Dimitrios S. Karachalios et al., published by De Gruyter.  This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

Loewner framework is a data-driven approach which can construct low order models from measurements. It can be applied to both frequency and time-domain data [56]. Here we will concentrate on frequency domain data. The Loewner framework will be implemented using (a) the SVD (singular value decomposition), (b) the CUR factorization, (c) randomized SVD (r-SVD). Its performance will be compared with that of the recently developed AAA algorithm see [53], the Vector Fitting approach [21, 40] and the IRKA algorithm [13].

The paper is composed of three sections. The first one covers the fundamentals of the Loewner framework starting from left and right interpolatory reduction. It concludes (a) by describing an interpolation property satisfied by reduced systems and (b) by making the procedure of obtaining real reduced models (despite complex interpolation points and values) explicit. Next the description of two algorithms namely, Loewner-SISO and Loewner-MIMO, is given. Finally two simple examples are presented and the role of generalized inverses outlined.

The second chapter describes methods for implementing the Loewner reduction, namely the SVD, the CUR factorization and the role of splitting the interpolation point in left and right sets. The third chapter illustrates the main features of the Loewner approach by means of seven case studies, namely, (a) the CD player, (b) an oscillating function, (c) the inverse of a Bessel function, (d) an Euler–Bernoulli beam, (e) a heat equation, (f) a function with two sharp peaks, and (g) the sign function. An epilogue and references conclude the presentation.

6.2 The Loewner framework and moment matching

The Loewner framework has attracted increased attention of researchers from various fields of applied mathematics and control engineering in the last 13 years. Consequently, a fair amount of contributions that are now available, deal with various aspects on further extending the framework and with its application to different test-cases. Below we provide an account of some of the work related to or inspired by the “Loewner framework” (see Table 6.1).

Consider linear, time-invariant systems with m inputs, n internal variables (states if \mathbf{E} is non-singular) and p outputs:

$$\Sigma : \begin{cases} \mathbf{E}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), & \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t), & \text{where} \\ \mathbf{E}, \mathbf{A} \in \mathbb{R}^{n \times n}, & \mathbf{B} \in \mathbb{R}^{n \times m}, & \mathbf{C} \in \mathbb{R}^{p \times n}. \end{cases} \quad (6.1)$$

We will denote this realization of the system by means of the quadruple $\Sigma = (\mathbf{C}, \mathbf{E}, \mathbf{A}, \mathbf{B})$. The associated transfer function is

$$\mathbf{H}(s) = \mathbf{C}\Phi(s)\mathbf{B} \quad \text{where } \Phi(s) = (s\mathbf{E} - \mathbf{A})^{-1} \in \mathbb{C}^{n \times n}. \quad (6.2)$$

Table 6.1: A collection of contributions related to the Loewner Framework.

Original paper [52] & tutorial paper [6]	Chapters 4 and 7 in the book [9]
Extension to parametrized linear systems [4, 42] bilinear systems [5, 45, 46] quadratic systems [29, 36] quadratic-bilinear systems [32] linear switched systems [34] polynomial systems [11, 16] modeling from noisy data [20, 50] modeling from time-domain data [56]	Application to modeling multi-port linear systems [48] preserving the stability of the ROM [30] the Burgers equations [8] the Oseen equations [10] preserving the structure of DAE systems [37] systems with delay [35, 59] approximating functions [31, 33, 43, 44] singular/rectangular systems [3] genes oscillations [7] and biological rhythms [68]
Perspective based on duality and application to bilinear differential [57, 58]	Interpretation based on interconnection and application to LTV systems [60, 61]

A common way to reduce the complexity of a system is by means of *Petrov–Galerkin projections*. Such projections are defined by means of two matrices $\mathbf{V}, \mathbf{W} \in \mathbb{R}^{n \times k}$, $k < n$, satisfying the condition that $\mathbf{W}^T \mathbf{V} \in \mathbb{R}^{k \times k}$ is invertible.¹

Definition 6.1. Consider $\mathbf{v}_i, \mathbf{w}_i \in \mathbb{R}^n$, $i = 1, \dots, k$, and let $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_k]$, $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_k] \in \mathbb{R}^{n \times k}$. The map defined by $\Pi_1 = \mathbf{V}(\mathbf{V}^T \mathbf{V})^{-1} \mathbf{V}^T$, is an orthogonal projection onto the span of the columns of \mathbf{V} . If $\mathbf{W}^T \mathbf{V}$ is non-singular, $\Pi_2 = \mathbf{V}(\mathbf{W}^T \mathbf{V})^{-1} \mathbf{W}^T$, is an oblique projector onto the span of the columns of \mathbf{V} , along the columns of \mathbf{W} . Π_1 and Π_2 are usually referred to in the model reduction literature as **Galerkin** and **Petrov–Galerkin** projectors, respectively.

Reducing the system $\Sigma = (\mathbf{C}, \mathbf{E}, \mathbf{A}, \mathbf{B})$ defined above, by means of a *Petrov–Galerkin projection*, we obtain the reduced system $\hat{\Sigma} = (\hat{\mathbf{C}}, \hat{\mathbf{E}}, \hat{\mathbf{A}}, \hat{\mathbf{B}})$ with the reduced order matrices given by

$$\hat{\mathbf{C}} = \mathbf{C}\mathbf{V} \in \mathbb{R}^{p \times k}, \quad \hat{\mathbf{E}} = \mathbf{W}^T \mathbf{E} \mathbf{V}, \quad \hat{\mathbf{A}} = \mathbf{W}^T \mathbf{A} \mathbf{V} \in \mathbb{R}^{k \times k}, \quad \hat{\mathbf{B}} = \mathbf{W}^T \mathbf{B} \in \mathbb{R}^{k \times m}. \quad (6.3)$$

There are many ways of choosing Petrov–Galerkin projectors in order to achieve various goals. Here we will restrict our attention to *interpolatory* projections. Such projectors yield reduced models which match moments of the original system. These moments are values of transfer functions at selected frequencies, referred to as *interpolation points*.

Remark 6.1. The **D-term**. In the system representations to follow no explicit **D** terms will be considered. The reason is that such terms can be incorporated in the remaining

¹ The notation $(\cdot)^T$ indicates transposition of (\cdot) , while the notation $(\cdot)^*$ indicates transposition of (\cdot) followed by complex conjugation.

matrices of the realization, thus yielding what is known as a descriptor representation. Consider a rank-revealing factorization

$$\mathbf{D} = \mathbf{D}_1 \mathbf{D}_2 \quad \text{where } \mathbf{D}_1 \in \mathbb{R}^{p \times \rho}, \quad \mathbf{D}_2 \in \mathbb{R}^{\rho \times m},$$

and $\rho = \text{rank } \mathbf{D}$. It readily follows that

$$\hat{\mathbf{E}} = \begin{bmatrix} \mathbf{E} & \\ & \mathbf{0}_{\rho \times \rho} \end{bmatrix}, \quad \hat{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \\ & -\mathbf{I}_\rho \end{bmatrix}, \quad \hat{\mathbf{B}} = \begin{bmatrix} \mathbf{B} \\ \mathbf{D}_2 \end{bmatrix}, \quad \hat{\mathbf{C}} = [\mathbf{C} \quad \mathbf{D}_1],$$

is a descriptor realization of the same system with no explicit \mathbf{D} -term. The reason for not considering explicit \mathbf{D} -terms, comes from the fact that the Loewner framework yields descriptor realizations with the \mathbf{D} -term incorporated in the rest of the realization.

6.2.1 Moments of a system

Given a matrix-valued function of time $\mathbf{h} : \mathbb{R} \rightarrow \mathbb{R}^{p \times m}$, its k th moment is

$$\eta_k = \int_0^\infty t^k \mathbf{h}(t) dt, \quad k = 0, 1, 2, \dots$$

If this function has a Laplace transform defined by $\mathbf{H}(s) = \mathcal{L}(\mathbf{h})(s) = \int_0^\infty \mathbf{h}(t) e^{-st} dt$, the k th moment of \mathbf{h} is, up to a sign, the k th derivative of \mathbf{H} evaluated at $s = 0$:

$$\eta_k = (-1)^k \frac{d^k}{ds^k} \mathbf{H}(s)|_{s=0} \in \mathbb{R}^{p \times m}, \quad k = 0, 1, 2, \dots$$

In the sequel, we will also make use of a generalized notion of moments, namely the *moments of \mathbf{h} around the (arbitrary) point $s_0 \in \mathbb{C}$* :

$$\eta_k(s_0) = \int_0^\infty t^k \mathbf{h}(t) e^{-s_0 t} dt.$$

These generalized moments turn out to be (up to a sign) the derivatives of $\mathbf{H}(s)$ evaluated at $s = s_0$:

$$\eta_k(s_0) = (-1)^k \frac{d^k}{ds^k} \mathbf{H}(s)|_{s=s_0} \in \mathbb{R}^{p \times m}, \quad k = 0, 1, 2, \dots$$

In this context, assuming for simplicity that $\mathbf{E} = \mathbf{I}$, the moments of \mathbf{h} at $s_0 \in \mathbb{C}$ are

$$\eta_k(s_0) = -k \mathbf{C}(s_0 \mathbf{I} - \mathbf{A})^{-(k+1)} \mathbf{B}, \quad k = 0, 1, 2, \dots,$$

provided that s_0 is not an eigenvalue of \mathbf{A} .

Notice that the moments determine the coefficients of the Laurent series expansion of the transfer function $\mathbf{H}(s)$ in the neighborhood of $s_i \in \mathbb{C}$; in particular

$$\begin{aligned}\mathbf{H}(s) &= \mathbf{H}(s_0) + \mathbf{H}^{(1)}(s_0) \frac{(s-s_0)}{1!} + \cdots + \mathbf{H}^{(k)}(s_0) \frac{(s-s_0)^k}{k!} + \cdots \\ &= \eta_0(s_0) + \eta_1(s_0) \frac{(s-s_0)}{1!} + \cdots + \eta_k(s_0) \frac{(s-s_0)^k}{k!} + \cdots.\end{aligned}$$

Approximation by moment matching

Given $\Sigma = (\mathbf{C}, \mathbf{E}, \mathbf{A}, \mathbf{B})$, consider the expansion of the transfer function around s_i , $i = 1, \dots, r$, as above. Approximation by moment matching consists in finding

$$\hat{\Sigma} = (\hat{\mathbf{C}}, \hat{\mathbf{E}}, \hat{\mathbf{A}}, \hat{\mathbf{B}}), \quad \hat{\mathbf{E}}, \hat{\mathbf{A}} \in \mathbb{R}^{k \times k}, \quad \hat{\mathbf{B}} \in \mathbb{R}^{k \times m}, \quad \hat{\mathbf{C}} \in \mathbb{R}^{p \times k}, \quad (6.4)$$

such that the expansion of the transfer function

$$\hat{\mathbf{H}}(s) = \hat{\eta}_0(s_i) + \hat{\eta}_1(s_i) \frac{(s-s_i)}{1!} + \hat{\eta}_2(s_i) \frac{(s-s_i)^2}{2!} + \hat{\eta}_3(s_i) \frac{(s-s_i)^3}{3!} + \cdots,$$

for appropriate $s_i \in \mathbb{C}$, and $\ell_i, r \in \mathbb{N}$, satisfies

$$\eta_j(s_i) = \hat{\eta}_j(s_i), \quad j = 1, 2, \dots, \ell_i \text{ and } i = 1, \dots, r.$$

This problem is also known as *rational interpolation*.

6.2.2 Rational interpolation by Petrov–Galerkin projection

Rational interpolation by projection was originally proposed in the work of Skelton and co-workers; see [65, 66, 67]. Contributions were also made by Grimme, Gallivan and van Dooren [23, 24, 38].

Suppose that we are given a system $\Sigma = (\mathbf{C}, \mathbf{E}, \mathbf{A}, \mathbf{B})$, assumed SISO (single-input single-output, i. e., $m = p = 1$) for simplicity. We wish to find lower dimensional models $\hat{\Sigma} = (\hat{\mathbf{C}}, \hat{\mathbf{E}}, \hat{\mathbf{A}}, \hat{\mathbf{B}})$, defined as in (6.3), $k < n$, such that $\hat{\Sigma}$ approximates the original system in an appropriate way.

Consider k distinct points $s_j \in \mathbb{C}$. Define \mathbf{V} as a *generalized controllability matrix*:

$$\mathbf{V} = [(s_1 \mathbf{E} - \mathbf{A})^{-1} \mathbf{B}, \dots, (s_k \mathbf{E} - \mathbf{A})^{-1} \mathbf{B}] \in \mathbb{C}^{n \times k}, \quad (6.5)$$

and let \mathbf{W}^* be any left inverse of \mathbf{V} . Then we have the following.

Proposition 6.1. $\hat{\Sigma}$ interpolates the transfer function of Σ at the points s_j , that is,

$$\mathbf{H}(s_j) = \mathbf{C}(s_j \mathbf{E} - \mathbf{A})^{-1} \mathbf{B} = \hat{\mathbf{C}}(s_j \hat{\mathbf{E}} - \hat{\mathbf{A}})^{-1} \hat{\mathbf{B}} = \hat{\mathbf{H}}(s_j), \quad j = 1, \dots, k.$$

Proof. Denoting by $\mathbf{e}_j = [0 \cdots 0 \underbrace{1}_j 0 \cdots 0]^T$, the j th unit vector, we obtain the string of equalities below which lead to the desired result:

$$\begin{aligned}
 \hat{\mathbf{C}}(s_j \hat{\mathbf{E}} - \hat{\mathbf{A}})^{-1} \hat{\mathbf{B}} &= \mathbf{C} \mathbf{V} (s_j \mathbf{W}^* \mathbf{E} \mathbf{V} - \mathbf{W}^* \mathbf{A} \mathbf{V})^{-1} \mathbf{W}^* \mathbf{B} \\
 &= \mathbf{C} \mathbf{V} (\mathbf{W}^* (s_j \mathbf{E} - \mathbf{A}) \mathbf{V})^{-1} \mathbf{W}^* \mathbf{B} \\
 &= \mathbf{C} \mathbf{V} ([* \cdots * \mathbf{W}^* \mathbf{B} * \cdots *])^{-1} \mathbf{W}^* \mathbf{B} \\
 &= [\mathbf{C}(s_1 \mathbf{E} - \mathbf{A})^{-1} \mathbf{B}, \dots, \mathbf{C}(s_k \mathbf{E} - \mathbf{A})^{-1} \mathbf{B}] \mathbf{e}_j \\
 &= \mathbf{C}(s_j \mathbf{E} - \mathbf{A})^{-1} \mathbf{B}.
 \end{aligned}$$

□

Next, we are concerned with matching the value of the transfer function at a given point $s_0 \in \mathbb{C}$, together with $k - 1$ derivatives. We define

$$\mathbf{V} = [(s_0 \mathbf{E} - \mathbf{A})^{-1} \mathbf{B}, (s_0 \mathbf{E} - \mathbf{A})^{-2} \mathbf{B}, \dots, (s_0 \mathbf{E} - \mathbf{A})^{-k} \mathbf{B}] \in \mathbb{C}^{n \times k}, \quad (6.6)$$

together with any left inverse \mathbf{W} . The following holds.

Proposition 6.2. *$\hat{\Sigma}$ interpolates the transfer function of Σ at s_0 , together with $k-1$ derivatives at the same point:*

$$\frac{(-1)^j}{j!} \frac{d^j}{ds^j} \mathbf{H}(s)|_{s=s_0} = \mathbf{C}(s_0 \mathbf{E} - \mathbf{A})^{-(j+1)} \mathbf{B} = \hat{\mathbf{C}}(s_0 \hat{\mathbf{E}} - \hat{\mathbf{A}})^{-(j+1)} \hat{\mathbf{B}} = \frac{(-1)^j}{j!} \frac{d^j}{ds^j} \hat{\mathbf{H}}(s)|_{s=s_0},$$

for $j = 0, 1, \dots, k - 1$.

Proof. Let \mathbf{V} be as defined in (6.6), and \mathbf{W} be such that $\mathbf{W}^T \mathbf{V} = \mathbf{I}_k$. It readily follows that the ℓ th power of the projected matrix $s_0 \hat{\mathbf{E}} - \hat{\mathbf{A}}$ satisfies

$$(s_0 \hat{\mathbf{E}} - \hat{\mathbf{A}})^\ell = \left[\underbrace{* \cdots *}_{\ell-1} \mathbf{W}^* \mathbf{B} \underbrace{* \cdots *}_{k-\ell} \right].$$

Consequently $[\mathbf{W}^T (s_0 \mathbf{E} - \mathbf{A}) \mathbf{V}]^{-\ell} \mathbf{W}^T \mathbf{B} = \mathbf{e}_\ell$, which finally implies

$$\hat{\mathbf{C}}(s_0 \hat{\mathbf{E}} - \hat{\mathbf{A}})^{-\ell} \hat{\mathbf{B}} = \mathbf{C} \mathbf{V} [\mathbf{W}^T (s_0 \mathbf{E} - \mathbf{A}) \mathbf{V}]^{-\ell} \mathbf{W}^T \mathbf{B} = \mathbf{C} \mathbf{V} \mathbf{e}_\ell = \mathbf{C}(s_0 \mathbf{E} - \mathbf{A})^{-\ell} \mathbf{B},$$

for $\ell = 1, 2, \dots, k$. □

Since any $\bar{\mathbf{V}}$ that spans the same column space as \mathbf{V} achieves the same objective, projectors composed of combinations of the cases (6.5) and (6.6) achieve matching of an appropriate number of moments. To formalize this we will make use of the following matrices:

$$\mathcal{R}_k(\mathbf{E}, \mathbf{A}, \mathbf{B}; \sigma) = [(\sigma \mathbf{E} - \mathbf{A})^{-1} \mathbf{B} \quad (\sigma \mathbf{E} - \mathbf{A})^{-2} \mathbf{B} \quad \cdots \quad (\sigma \mathbf{E} - \mathbf{A})^{-k} \mathbf{B}].$$

Corollary 6.1.

- (a) If \mathbf{V} as defined above is replaced by $\bar{\mathbf{V}} = \mathbf{R}\mathbf{V}$, $\mathbf{R} \in \mathbb{R}^{k \times k}$, $\det \mathbf{R} \neq 0$, and \mathbf{W} by $\bar{\mathbf{W}} = \mathbf{R}^{-T}\mathbf{W}$, the same matching results hold true.
- (b) Let \mathbf{V} be such that

$$\text{span col } \mathbf{V} = \text{span col } [\mathcal{R}_{m_1}(\mathbf{E}, \mathbf{A}, \mathbf{B}; \sigma_1) \quad \cdots \quad \mathcal{R}_{m_\ell}(\mathbf{E}, \mathbf{A}, \mathbf{B}; \sigma_\ell)],$$

and \mathbf{W} be any left inverse of \mathbf{V} . The reduced system matches m_i moments at $\sigma_i \in \mathbb{C}$, $i = 1, \dots, \ell$.

6.2.3 Two-sided projections

The above results can be strengthened if the row span of the matrix \mathbf{W}^T is chosen to match the row span of a generalized observability matrix. In such a case twice as many moments can be matched with a reduced system of the same dimension. Here, in addition to points s_1, \dots, s_k , and the associated (6.5), we are given k additional distinct points s_{k+1}, \dots, s_{2k} . These points are used to define a *generalized observability matrix*:

$$\mathbf{W} = [(s_{k+1}\mathbf{E}^T - \mathbf{A}^T)^{-1}\mathbf{C}^T \quad \cdots \quad (s_{2k}\mathbf{E}^T - \mathbf{A}^T)^{-1}\mathbf{C}^T] \in \mathbb{C}^{n \times k}. \quad (6.7)$$

Proposition 6.3. Assuming that $\mathbf{W}^T\mathbf{V}$ has full rank, the projected system $\hat{\Sigma}$, interpolates the transfer function of Σ at the points s_i , $i = 1, \dots, 2k$.

Proof. The string of equalities that follows proves the desired result:

$$\begin{aligned} \hat{\mathbf{C}}(s_i\hat{\mathbf{E}} - \hat{\mathbf{A}})^{-1}\hat{\mathbf{B}} &= \mathbf{C}\mathbf{V}(s_i\mathbf{W}^T\mathbf{E}\mathbf{V} - \mathbf{W}^T\mathbf{A}\mathbf{V})^{-1}\mathbf{W}^T\mathbf{B} \\ &= \mathbf{C}\mathbf{V}(\mathbf{W}^T(s_i\mathbf{E} - \mathbf{A})\mathbf{V})^{-1}\mathbf{W}^T\mathbf{B} \\ &= \mathbf{C}\mathbf{V}(\mathbf{W}^T[\cdots \mathbf{B} \cdots])^{-1}\mathbf{W}^T\mathbf{B} \\ &= \mathbf{C}\mathbf{V}\mathbf{e}_i = \mathbf{C}(s_i\mathbf{E} - \mathbf{A})^{-1}\mathbf{B}, \quad \text{for } i = 1, \dots, k, \\ &= \mathbf{C}\mathbf{V} \left(\begin{bmatrix} \vdots \\ \mathbf{C} \\ \vdots \end{bmatrix} \mathbf{V} \right)^{-1} \mathbf{W}^T\mathbf{B} \\ &= \mathbf{e}_i^T \mathbf{W}^T\mathbf{B} = \mathbf{C}(s_i\mathbf{E} - \mathbf{A})^{-1}\mathbf{B}, \quad \text{for } i = k+1, \dots, 2k. \quad \square \end{aligned}$$

The projectors (see [62]) discussed in the previous section satisfy the *Sylvester equations* as shown next.

Proposition 6.4. With $\mathbf{\Lambda} = \text{diag}[\lambda_1, \dots, \lambda_k]$, $\mathbf{M} = \text{diag}[\mu_1, \dots, \mu_q]$, where λ_i and μ_j are mutually distinct, $\mathbf{R} = [1 \quad \cdots \quad 1] \in \mathbb{R}^k$, and $\mathbf{L} = [1 \quad \cdots \quad 1]^T \in \mathbb{R}^q$, the matrices \mathbf{V} and \mathbf{W} satisfy the Sylvester equations:

$$\mathbf{E}\mathbf{V}\mathbf{\Lambda} - \mathbf{A}\mathbf{V} = \mathbf{B}\mathbf{R} \quad \text{and} \quad \mathbf{M}\mathbf{W}^T\mathbf{E} - \mathbf{W}^T\mathbf{A} = \mathbf{L}\mathbf{C}. \quad (6.8)$$

6.2.4 Interpolatory model reduction for MIMO systems

In the general case of MIMO (multi-input multi-output) systems, the moments are $p \times m$ matrices. So, in the case of rational matrix interpolation the most appropriate way to proceed is to interpolate along certain directions. This leads to the so-called *tangential interpolation problem* (see e. g. [6, 21, 25]).

More precisely, we are given a set of *input/output* response measurements specified by left driving frequencies $\{\mu_i\}_{i=1}^q \subset \mathbb{C}$, using left input or tangential directions: $\{\ell_i\}_{i=1}^q \subset \mathbb{C}^p$, producing left responses: $\{\mathbf{v}_i\}_{i=1}^q \subset \mathbb{C}^m$, and right driving frequencies: $\{\lambda_i\}_{i=1}^k \subset \mathbb{C}$, using right input or tangential directions: $\{\mathbf{r}_i\}_{i=1}^k \subset \mathbb{C}^m$, producing right responses: $\{\mathbf{w}_i\}_{i=1}^k$. We are thus given the *left data*: $(\mu_j; \ell_j^T, \mathbf{v}_j^T)$, $j = 1, \dots, q$, and the *right data*: $(\lambda_i; \mathbf{r}_i, \mathbf{w}_i)$, $i = 1, \dots, k$. The problem is to find a rational $p \times m$ matrix $\mathbf{H}(s)$, such that

$$\mathbf{H}(\lambda_i)\mathbf{r}_i = \mathbf{w}_i, \quad i = 1, \dots, k, \quad \ell_j^T \mathbf{H}(\mu_j) = \mathbf{v}_j^T, \quad j = 1, \dots, q. \quad (6.9)$$

The *left data* is rearranged compactly as

$$\mathbf{M} = \begin{bmatrix} \mu_1 & & \\ & \ddots & \\ & & \mu_q \end{bmatrix} \in \mathbb{C}^{q \times q}, \quad \mathbf{L} = \begin{bmatrix} \ell_1^T \\ \vdots \\ \ell_q^T \end{bmatrix} \in \mathbb{C}^{q \times p}, \quad \mathbf{V} = \begin{bmatrix} \mathbf{v}_1^T \\ \vdots \\ \mathbf{v}_q^T \end{bmatrix} \in \mathbb{C}^{q \times m}, \quad (6.10)$$

while the *right data* is rearranged as

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_k \end{bmatrix} \in \mathbb{C}^{k \times k}, \quad \mathbf{R} = [\mathbf{r}_1 \quad \mathbf{r}_2 \quad \dots \quad \mathbf{r}_k] \in \mathbb{C}^{m \times k}, \quad (6.11)$$

$$\mathbf{W} = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \dots \quad \mathbf{w}_k] \in \mathbb{C}^{p \times k}.$$

Interpolation points and tangential directions are determined by the problem or are selected to realize given model reduction goals. For SISO systems, i. e., $m = p = 1$, left and right directions can be taken equal to one ($\ell_j = 1$, $\mathbf{r}_i = 1$) and hence the conditions above become

$$\left. \begin{aligned} \hat{\mathbf{H}}(\mu_j) &= \mathbf{H}(\mu_j) \Rightarrow \hat{\mathbf{H}}(\mu_j) = \mathbf{v}_j, \quad j = 1, \dots, q, \\ \hat{\mathbf{H}}(\lambda_i) &= \mathbf{H}(\lambda_i) \Rightarrow \hat{\mathbf{H}}(\lambda_i) = \mathbf{w}_i, \quad i = 1, \dots, k. \end{aligned} \right\} \quad (6.12)$$

6.2.5 The Loewner framework

Given a row array of complex numbers (μ_j, \mathbf{v}_j) , $j = 1, \dots, q$, and a column array, $(\lambda_i, \mathbf{w}_i)$, $i = 1, \dots, k$, (with λ_i and the μ_j mutually distinct) the associated *Loewner matrix* is

$$\mathbb{L} = \begin{bmatrix} \frac{\mathbf{v}_1 - \mathbf{w}_1}{\mu_1 - \lambda_1} & \dots & \frac{\mathbf{v}_1 - \mathbf{w}_k}{\mu_1 - \lambda_k} \\ \vdots & \ddots & \vdots \\ \frac{\mathbf{v}_q - \mathbf{w}_1}{\mu_q - \lambda_1} & \dots & \frac{\mathbf{v}_q - \mathbf{w}_k}{\mu_q - \lambda_k} \end{bmatrix} \in \mathbb{C}^{q \times k}.$$

Definition 6.2. If \mathbf{g} is rational, i. e., $\mathbf{g}(s) = \frac{\mathbf{p}(s)}{\mathbf{q}(s)}$, for appropriate polynomials \mathbf{p} , \mathbf{q} , the McMillan degree or the complexity of \mathbf{g} is $\deg \mathbf{g} = \max\{\deg(\mathbf{p}), \deg(\mathbf{q})\}$.

Now, if $\mathbf{w}_i = \mathbf{g}(\lambda_i)$, and $\mathbf{v}_j = \mathbf{g}(\mu_j)$, are *samples* of a rational function \mathbf{g} , the *main property* of Loewner matrices asserts the following.

Theorem 6.1 ([52]). *Let \mathbb{L} be as above. If $k, q \geq \deg \mathbf{g}$, then $\text{rank } \mathbb{L} = \deg \mathbf{g}$. In other words the rank of \mathbb{L} encodes the complexity of the underlying rational function \mathbf{g} . Furthermore, the same result holds for matrix-valued functions \mathbf{g} .*

6.2.5.1 The Loewner pencil and interpolatory projectors

In the sequel we denote the *tangential* versions of (6.5) and (6.7) by \mathcal{R} , \mathcal{O} , respectively. For arbitrary k and q , these are defined as

$$\mathcal{R} = [(\lambda_1 \mathbf{E} - \mathbf{A})^{-1} \mathbf{B} \mathbf{r}_1, \dots, (\lambda_k \mathbf{E} - \mathbf{A})^{-1} \mathbf{B} \mathbf{r}_k] \in \mathbb{C}^{n \times k}, \quad (6.13)$$

$$\mathcal{O}^T = [(\mu_1 \mathbf{E}^T - \mathbf{A}^T)^{-1} \mathbf{C}^T \ell_1 \quad \dots \quad (\mu_q \mathbf{E}^T - \mathbf{A}^T)^{-1} \mathbf{C}^T \ell_q] \in \mathbb{C}^{n \times k}. \quad (6.14)$$

It readily follows that the reduced quantities $\hat{\mathbf{E}}$ and $\hat{\mathbf{A}}$ form a *Loewner pencil*:

$$\hat{\mathbf{E}} = -\mathcal{O} \mathbf{E} \mathcal{R} = - \begin{bmatrix} \frac{\mathbf{v}_1^T \mathbf{r}_1 - \ell_1^T \mathbf{w}_1}{\mu_1 - \lambda_1} & \dots & \frac{\mathbf{v}_1^T \mathbf{r}_k - \ell_1^T \mathbf{w}_k}{\mu_1 - \lambda_k} \\ \vdots & \ddots & \vdots \\ \frac{\mathbf{v}_q^T \mathbf{r}_1 - \ell_q^T \mathbf{w}_1}{\mu_q - \lambda_1} & \dots & \frac{\mathbf{v}_q^T \mathbf{r}_k - \ell_q^T \mathbf{w}_k}{\mu_q - \lambda_k} \end{bmatrix} = -\mathbb{L} \in \mathbb{C}^{q \times k}, \quad (6.15)$$

$$\hat{\mathbf{A}} = -\mathcal{O} \mathbf{A} \mathcal{R} = - \begin{bmatrix} \frac{\mu_1 \mathbf{v}_1^T \mathbf{r}_1 - \ell_1^T \mathbf{w}_1 \lambda_1}{\mu_1 - \lambda_1} & \dots & \frac{\mu_1 \mathbf{v}_1^T \mathbf{r}_k - \ell_1^T \mathbf{w}_k \lambda_k}{\mu_1 - \lambda_k} \\ \vdots & \ddots & \vdots \\ \frac{\mu_q \mathbf{v}_q^T \mathbf{r}_1 - \ell_q^T \mathbf{w}_1 \lambda_1}{\mu_q - \lambda_1} & \dots & \frac{\mu_q \mathbf{v}_q^T \mathbf{r}_k - \ell_q^T \mathbf{w}_k \lambda_k}{\mu_q - \lambda_k} \end{bmatrix} = -\mathbb{L}_S \in \mathbb{C}^{q \times k}, \quad (6.16)$$

$$\hat{\mathbf{B}} = \mathcal{O} \mathbf{B} = \begin{bmatrix} \mathbf{v}_1^T \\ \vdots \\ \mathbf{v}_q^T \end{bmatrix} = \mathbb{V} \in \mathbb{C}^{q \times m}, \quad \hat{\mathbf{C}} = \mathbf{C} \mathcal{R} = [\mathbf{w}_1 \quad \dots \quad \mathbf{w}_k] = \mathbb{W} \in \mathbb{C}^{p \times k}. \quad (6.17)$$

The resulting quadruple $(\mathbb{W}, \mathbb{L}, \mathbb{L}_S, \mathbb{V})$ is called the *Loewner quadruple*.

Lemma 6.1. *Upon multiplication of the first equation in (6.8) with \mathcal{O} on the left and the second by \mathcal{R} on the right we obtain*

$$\mathbb{L}_S - \mathbb{L} \mathbf{A} = \mathbb{V} \mathbf{R} \quad \text{and} \quad \mathbb{L}_S - \mathbf{M} \mathbb{L} = \mathbf{L} \mathbb{W}. \quad (6.18)$$

By adding/subtracting appropriate multiples of these expressions it follows that the Loewner quadruple satisfies the Sylvester equations

$$\mathbf{M} \mathbb{L} - \mathbb{L} \mathbf{A} = \mathbb{V} \mathbf{R} - \mathbf{L} \mathbb{W} \quad \text{and} \quad \mathbf{M} \mathbb{L}_S - \mathbb{L}_S \mathbf{A} = \mathbf{M} \mathbb{V} \mathbf{R} - \mathbf{L} \mathbf{W} \mathbf{A}. \quad (6.19)$$

Theorem 6.2. Assume that the pencil $(\mathbb{L}_s, \mathbb{L})$ is regular.² Then $\mathbf{H}(s) = \mathbf{W}(\mathbb{L}_s - s\mathbb{L})^{-1}\mathbf{V}$, satisfies the tangential interpolation condition (6.9).

Proof. Multiplying the first Sylvester equation by s and subtracting it from equation the second one, we get

$$\mathbf{M}(\mathbb{L}_s - s\mathbb{L}) - (\mathbb{L}_s - s\mathbb{L})\mathbf{A} = (\mathbf{M} - s\mathbf{I})\mathbf{V}\mathbf{R} - \mathbf{L}\mathbf{W}(\mathbf{A} - s\mathbf{I}).$$

Multiplying this equation by \mathbf{e}_i on the right and setting $s = \lambda_i$, we obtain

$$\begin{aligned} (\mathbf{M} - \lambda_i\mathbf{I})(\mathbb{L}_s - \lambda_i\mathbb{L})\mathbf{e}_i &= (\mathbf{M} - \lambda_i\mathbf{I})\mathbf{V}\mathbf{r}_i \Rightarrow (\mathbb{L}_s - \lambda_i\mathbb{L})\mathbf{e}_i = \mathbf{V}\mathbf{r}_i \\ &\Rightarrow \mathbf{W}\mathbf{e}_i = \mathbf{W}(\mathbb{L}_s - \lambda_i\mathbb{L})^{-1}\mathbf{V}\mathbf{r}_i. \end{aligned}$$

Thus $\mathbf{w}_i = \mathbf{H}(\lambda_i)\mathbf{r}_i$. Next, we multiply the above equation by \mathbf{e}_j^T on the left and set $s = \mu_j$:

$$\begin{aligned} \mathbf{e}_j^T(\mathbb{L}_s - \mu_j\mathbb{L})(\mathbf{A} - \mu_j\mathbf{I}) &= \mathbf{e}_j^T\mathbf{L}\mathbf{W}(\mathbf{A} - \mu_j\mathbf{I}) \Rightarrow \mathbf{e}_j^T(\mathbb{L}_s - \mu_j\mathbb{L}) = \boldsymbol{\ell}_j^T\mathbf{W} \\ &\Rightarrow \mathbf{e}_j^T\mathbf{V} = \boldsymbol{\ell}_j^T\mathbf{W}(\mathbb{L}_s - \mu_j\mathbb{L})^{-1}\mathbf{V}. \end{aligned}$$

Thus $\mathbf{v}_j^T = \boldsymbol{\ell}_j^T\mathbf{H}(\mu_j)$. □

Remark 6.2 (Parametrization of all interpolants of complexity equal to the size of \mathbb{L}). With $\mathbf{K} \in \mathbb{C}^{p \times m}$, the Sylvester equations can be rewritten as

$$\begin{aligned} \mathbf{M}\mathbb{L} - \mathbb{L}\mathbf{A} &= (\mathbf{V} - \mathbf{L}\mathbf{K})\mathbf{R} - \mathbf{L}(\mathbf{W} - \mathbf{K}\mathbf{R}) \quad \text{and} \\ \mathbf{M}(\mathbb{L}_s + \mathbf{L}\mathbf{K}\mathbf{R}) - (\mathbb{L}_s + \mathbf{L}\mathbf{K}\mathbf{R})\mathbf{A} &= \mathbf{M}(\mathbf{V} - \mathbf{L}\mathbf{K})\mathbf{R} - \mathbf{L}(\mathbf{W} - \mathbf{K}\mathbf{R})\mathbf{A}. \end{aligned}$$

These equations imply that $(\bar{\mathbf{W}}, \mathbb{L}, \bar{\mathbb{L}}_s, \bar{\mathbf{V}})$ is an interpolant for all $\mathbf{K} \in \mathbb{C}^{p \times m}$, where $\bar{\mathbb{L}}_s = \mathbb{L}_s + \mathbf{L}\mathbf{K}\mathbf{R}$, $\bar{\mathbf{V}} = \mathbf{V} - \mathbf{L}\mathbf{K}\mathbf{R}$ and $\bar{\mathbf{W}} = \mathbf{W} - \mathbf{K}\mathbf{R}$.

6.2.5.2 Construction of interpolants

If the pencil $(\mathbb{L}_s, \mathbb{L})$ is regular, then $\mathbf{E} = -\mathbb{L}$, $\mathbf{A} = -\mathbb{L}_s$, $\mathbf{B} = \mathbf{V}$, $\mathbf{C} = \mathbf{W}$, is a minimal interpolant of the data, i. e., $\mathbf{H}(s) = \mathbf{W}(\mathbb{L}_s - s\mathbb{L})^{-1}\mathbf{V}$, interpolates the data. Otherwise, as shown in [52], problem (6.9) has a solution provided that

$$\text{rank}[s\mathbb{L} - \mathbb{L}_s] = \text{rank}[\mathbb{L}, \mathbb{L}_s] = \text{rank} \begin{bmatrix} \mathbb{L} \\ \mathbb{L}_s \end{bmatrix} = r,$$

for all $s \in \{\lambda_j\} \cup \{\mu_i\}$. Consider then the short SVDs:

$$[\mathbb{L}, \mathbb{L}_s] = \mathbf{Y}\hat{\Sigma}_r\tilde{\mathbf{X}}^*, \quad \begin{bmatrix} \mathbb{L} \\ \mathbb{L}_s \end{bmatrix} = \tilde{\mathbf{Y}}\Sigma_r\mathbf{X}^*,$$

where $\hat{\Sigma}_r, \Sigma_r \in \mathbb{R}^{r \times r}$, $\mathbf{Y} \in \mathbb{C}^{q \times r}$, $\mathbf{X} \in \mathbb{C}^{k \times r}$, $\tilde{\mathbf{Y}} \in \mathbb{C}^{2q \times r}$, $\tilde{\mathbf{X}} \in \mathbb{C}^{r \times 2k}$.

² The pencil $(\mathbb{L}_s, \mathbb{L})$ is called regular if there is at least one value of $\lambda \in \mathbb{C}$ such that $\det(\mathbb{L}_s - \lambda\mathbb{L}) \neq 0$.

Remark 6.3. r can be chosen as the *numerical rank* (as opposed to the *exact rank*) of the Loewner pencil. For issues related to the rank, we refer the reader to [2], page 50, for details.

Theorem 6.3. *The quadruple $(\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C})$ of size $r \times r, r \times r, r \times m, p \times r$, given by*

$$\mathbf{E} = -\mathbf{Y}^T \mathbb{L} \mathbf{X}, \quad \mathbf{A} = -\mathbf{Y}^T \mathbb{L}_s \mathbf{X}, \quad \mathbf{B} = \mathbf{Y}^T \mathbb{V}, \quad \mathbf{C} = \mathbb{W} \mathbf{X},$$

is a descriptor realization of an (approximate) interpolant of the data with McMillan degree $r = \text{rank } \mathbb{L}$.

Remark 6.4.

- (a) The Loewner approach constructs a descriptor representation $(\mathbb{W}, \mathbb{L}, \mathbb{L}_s, \mathbb{V})$, of an underlying dynamical system exclusively from the data, with no further manipulations involved (i. e., matrix factorizations or inversions). In general, the pencil $(\mathbb{L}_s, \mathbb{L})$ is singular and needs to be projected to a regular pencil (\mathbf{A}, \mathbf{E}) . However, as shown in the *mass–spring–damper* example in equation (6.22), inversion can be replaced by generalized inversion.
- (b) As already mentioned, in the Loewner framework, by construction, \mathbf{D} terms are absorbed in the other matrices of the realization. Extracting the \mathbf{D} term involves an eigenvalue decomposition of $(\mathbb{L}_s, \mathbb{L})$.

6.2.5.3 Interpolation property of reduced systems

Given a Loewner quadruple and the projection matrices³ $\mathbf{X}, \mathbf{Y} \in \mathbb{C}^{n \times k}$, let the reduced quantities be

$$\hat{\mathbb{L}} = \mathbf{X}^* \mathbb{L} \mathbf{Y}, \quad \hat{\mathbb{L}}_s = \mathbf{X}^* \mathbb{L}_s \mathbf{Y}, \quad \hat{\mathbb{V}} = \mathbf{X}^* \mathbb{V}, \quad \hat{\mathbb{W}} = \mathbb{W} \mathbf{Y}.$$

We also consider the projected \mathbf{L} and \mathbf{R} matrices, namely $\hat{\mathbf{L}} = \mathbf{X}^* \mathbf{L}$, $\hat{\mathbf{R}} = \mathbf{R} \mathbf{Y}$. The question which arises is whether these reduced quantities satisfy interpolation conditions as well. The answer is affirmative and to show this we proceed as follows.

The associated $\hat{\mathbf{A}}$ and $\hat{\mathbf{M}}$ must satisfy the *projected equations* resulting from (6.18), i. e.

$$\hat{\mathbb{L}}_s - \hat{\mathbf{L}} \hat{\mathbf{A}} = \hat{\mathbb{V}} \hat{\mathbf{R}} \quad \text{and} \quad \hat{\mathbb{L}}_s - \hat{\mathbf{M}} \hat{\mathbf{L}} = \hat{\mathbf{L}} \hat{\mathbb{W}}. \quad (6.20)$$

Notice that the projected Loewner pencil is not in Loewner form. To achieve this we proceed as follows. We need to diagonalize $\hat{\mathbf{A}}$ and $\hat{\mathbf{M}}$. For this purpose we compute the following two generalized eigenvalue decompositions:

$$[\mathbf{D}_{\hat{\mathbf{A}}}, \mathbf{T}_{\hat{\mathbf{A}}}] = \text{eig}(\hat{\mathbb{L}}_s - \hat{\mathbb{V}} \hat{\mathbf{R}}, \hat{\mathbf{L}}) \quad \text{and} \quad [\mathbf{D}_{\hat{\mathbf{M}}}, \mathbf{T}_{\hat{\mathbf{M}}}] = \text{eig}(\hat{\mathbb{L}}_s - \hat{\mathbf{L}} \hat{\mathbb{W}}, \hat{\mathbf{L}}).$$

³ We call $\mathbf{X}, \mathbf{Y} \in \mathbb{C}^{n \times k}$ *projection matrices* as they are used for defining the *projector*: $\mathbf{X}(\mathbf{Y}^* \mathbf{X})^{-1} \mathbf{Y}^*$.

These decompositions imply

$$\hat{\mathbf{A}} = \mathbf{T}_{\hat{\mathbf{A}}} \mathbf{D}_{\hat{\mathbf{A}}} \mathbf{T}_{\hat{\mathbf{A}}}^{-1} \quad \text{and} \quad \hat{\mathbf{M}} = \mathbf{T}_{\hat{\mathbf{M}}} \mathbf{D}_{\hat{\mathbf{M}}} \mathbf{T}_{\hat{\mathbf{M}}}^{-1}, \quad (6.21)$$

where for simplicity, it is assumed that the matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{M}}$ are diagonalizable.

It follows that the (diagonal) entries of $\mathbf{D}_{\hat{\mathbf{A}}}$ and $\mathbf{D}_{\hat{\mathbf{M}}}$ are the right frequencies and the left frequencies of the reduced system, respectively. Furthermore, straightforward calculations imply that the remaining quantities are as follows:

$$\begin{cases} \bar{\mathbf{L}}_s = \mathbf{T}_{\hat{\mathbf{M}}}^{-1} \hat{\mathbf{L}}_s \mathbf{T}_{\hat{\mathbf{A}}}, & \bar{\mathbf{L}} = \mathbf{T}_{\hat{\mathbf{M}}}^{-1} \hat{\mathbf{L}} \mathbf{T}_{\hat{\mathbf{A}}}, \\ \bar{\mathbf{V}} = \mathbf{T}_{\hat{\mathbf{M}}}^{-1} \hat{\mathbf{V}}, & \bar{\mathbf{L}} = \mathbf{T}_{\hat{\mathbf{M}}}^{-1} \hat{\mathbf{L}}, \\ \bar{\mathbf{W}} = \hat{\mathbf{W}} \mathbf{T}_{\hat{\mathbf{A}}}, & \bar{\mathbf{R}} = \hat{\mathbf{R}} \mathbf{T}_{\hat{\mathbf{A}}}. \end{cases}$$

Conclusion: the right/left data triples for the reduced system are $(\mathbf{D}_{\hat{\mathbf{A}}}, \bar{\mathbf{W}}, \bar{\mathbf{R}})$, and $(\mathbf{D}_{\hat{\mathbf{M}}}, \bar{\mathbf{V}}, \bar{\mathbf{L}})$, respectively, while the associated Loewner pencil is $(\bar{\mathbf{L}}_s, \bar{\mathbf{L}})$.

6.2.5.4 Real interpolants and reduced models

Most often the data are collected from real systems. In these cases if (s_i, ϕ_i) $s_i, \phi_i \in \mathbb{C}$, is a measurement pair, in order for the interpolants/reduced models to be real, the complex conjugate pair $(\bar{s}_i, \bar{\phi}_i)$, should also be included. Thus the left/right frequencies besides real quantities contain complex ones appearing in complex conjugate pairs. For instance, in the SISO (single-input single-output) case, let the real measurement frequencies be $\sigma_i \in \mathbb{R}$, and the complex ones $\hat{\sigma}_i + j \cdot \hat{\omega}_i$ where j denotes the imaginary unit. We split them in two sets, the left and the right ones, respectively, making sure that each set is closed under complex conjugation:

$$\begin{aligned} \mathbf{M} &= \{\sigma_i, i = 1, \dots, r_1; \hat{\sigma}_i \pm j \cdot \hat{\omega}_i, i = 1, \dots, r_3\}, \\ \mathbf{A} &= \{\sigma_i, i = r_1 + 1, \dots, r_1 + r_2; \hat{\sigma}_i \pm j \cdot \hat{\omega}_i, i = r_3 + 1, \dots, r_3 + r_4\}. \end{aligned}$$

Thus the left set has r_1 real frequencies and r_3 complex frequencies together with their complex conjugates (total $r_1 + 2r_3$ numbers). Similarly the numbers for the right set are r_2 and r_4 , i. e., it consists of $r_2 + 2r_4$ numbers. The quantities \mathbf{W} and \mathbf{V} are assembled in accordance with \mathbf{M} and \mathbf{A} . In addition let us define the matrices:

$$\begin{aligned} \mathbf{J}_\mu &= \text{blkdiag}[\overbrace{\mathbf{I}_{r_1}, \mathbf{J}, \dots, \mathbf{J}}^{r_3 \text{ terms}}] \in \mathbb{C}^{(r_1+2r_3) \times (r_1+2r_3)}, \\ \mathbf{J}_\lambda &= \text{blkdiag}[\overbrace{\mathbf{I}_{r_2}, \mathbf{J}, \dots, \mathbf{J}}^{r_4 \text{ terms}}] \in \mathbb{C}^{(r_2+2r_4) \times (r_2+2r_4)}, \end{aligned}$$

where $\mathbf{J} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -j \\ 1 & j \end{bmatrix}$, where $\text{blkdiag}[\cdot]$ (following Matlab notation) denotes the block diagonal structure. A simple calculation shows then that the matrices

$$\mathbf{M}_R = \mathbf{J}_\mu^* \mathbf{M} \mathbf{J}_\mu, \quad \mathbf{V}_R = \mathbf{J}_\mu^* \mathbf{V}, \quad \mathbf{L}_R = \mathbf{J}_\mu^* \mathbf{L},$$

have real entries. The same happens with the matrices

$$\mathbf{\Lambda}_R = \mathbf{J}_\lambda^* \mathbf{\Lambda} \mathbf{J}_\lambda, \quad \mathbf{W}_R = \mathbf{W} \mathbf{J}_\lambda, \quad \mathbf{R}_R = \mathbf{R} \mathbf{J}_\lambda.$$

Recall equations (6.18). If we now solve the transformed equations for $\mathbb{L}^R, \mathbb{L}_s^R$:

$$\mathbb{L}_s^R - \mathbb{L}^R \mathbf{\Lambda}_R = \mathbf{V}_R \mathbf{R}_R \quad \text{and} \quad \mathbb{L}_s^R - \mathbf{M}_R \mathbb{L}^R = \mathbf{L}_R \mathbf{W}_R,$$

the resulting pencil $(\mathbb{L}_s^R, \mathbb{L}^R)$ has real entries. Hence the algorithms based on \mathbb{L}^R and \mathbb{L}_s^R described below yield *real* reduced order models.

6.2.5.5 The Loewner algorithms for scalar and matrix rational approximation

Next, two algorithms (see Algorithms 6.1 and 6.2) for computing a strictly rational real interpolant for both, the scalar and the matrix interpolation problem are presented.

Algorithm 6.1: Loewner-SISO (Scalar rational approximation) [49].

Input: $\mathbf{S} = [s_1, \dots, s_N] \in \mathbb{C}^N, \mathbf{F} = [\phi_1, \dots, \phi_N] \in \mathbb{C}^N, N \in \mathbb{N}$.

Output: $\hat{\mathbf{E}} \in \mathbb{R}^{r \times r}, \hat{\mathbf{A}} \in \mathbb{R}^{r \times r}, \hat{\mathbf{B}} \in \mathbb{R}^{r \times 1}, \hat{\mathbf{C}} \in \mathbb{R}^{1 \times r}$ with $r \ll N$.

1. Partition the measurements into two disjoint sets and form left and right set as $(\mu_j, \mathbf{v}_j), j = 1, \dots, q$ and $(\lambda_i, \mathbf{w}_i), i = 1, \dots, k$.

$$\text{frequencies : } [s_1, \dots, s_N] \rightarrow [\lambda_1, \dots, \lambda_k], [\mu_1, \dots, \mu_q], \quad k + q = N,$$

$$\text{values : } [\phi_1, \dots, \phi_N] \rightarrow [\mathbf{w}_1, \dots, \mathbf{w}_k] = \mathbf{W}, [\mathbf{v}_1, \dots, \mathbf{v}_q] = \mathbf{V}^T.$$

2. Construct the **Loewner pencil** as

$$\mathbb{L} = \left(\frac{\mathbf{v}_i - \mathbf{w}_j}{\mu_i - \lambda_j} \right)_{i=1, \dots, q}^{j=1, \dots, k}, \quad \mathbb{L}_s = \left(\frac{\mu_i \mathbf{v}_i - \lambda_j \mathbf{w}_j}{\mu_i - \lambda_j} \right)_{i=1, \dots, q}^{j=1, \dots, k}.$$

3. It follows that the **complex raw model** is

$$\{\mathbf{W}, \mathbb{L}, \mathbb{L}_s, \mathbf{V}\}.$$

4. Transform all the complex data to real and there follows the **raw real model**:

$$\{\mathbf{W}_R, \mathbb{L}^R, \mathbb{L}_s^R, \mathbf{V}_R\}.$$

5. Compute the rank-revealing SVDs: $[\mathbf{Y}_1, \Sigma_1, \mathbf{X}_1] = \mathbf{SVD}([\mathbb{L}^R, \mathbb{L}_s^R])$ and $[\mathbf{Y}_2, \Sigma_2, \mathbf{X}_2] = \mathbf{SVD}([\mathbb{L}^R, \mathbb{L}_s^R])$; the decay of the singular values, leads to the choice of the order r of the approximant.

6. The reduced real model is obtained by **projecting** the *raw real model* with $\mathbf{Y} = \mathbf{Y}_1^{n \times r}$ and $\mathbf{X} = \mathbf{X}_2^{n \times r}$ as

$$\underbrace{\{\mathbf{W}_R, \mathbb{L}^R, \mathbb{L}_S^R, \mathbb{V}_R\}}_{\text{singular}} \xrightarrow{\text{SVD}} \underbrace{\{\mathbf{W}_R \mathbf{X}, \mathbf{Y}^T \mathbb{L}^R \mathbf{X}, \mathbf{Y}^T \mathbb{L}_S^R \mathbf{X}, \mathbf{Y}^T \mathbb{V}_R\}}_{\text{regular}} = \{\hat{\mathbf{C}}, -\hat{\mathbf{E}}, -\hat{\mathbf{A}}, \hat{\mathbf{B}}\}.$$

7. A real approximant of the data is then

$$\hat{H}(s) = \hat{\mathbf{C}}(s\hat{\mathbf{E}} - \hat{\mathbf{A}})^{-1}\hat{\mathbf{B}} \approx \phi(s).$$

□

Algorithm 6.2: Loewner-MIMO (Matrix rational approximation).

Input: $\mathbf{S} = [s_1, \dots, s_N] \in \mathbb{C}^N$, $\mathbf{F} = [\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_N] \in \mathbb{C}^{N \times p \times m}$, $N \in \mathbb{N}$.

Output: $\hat{\mathbf{E}} \in \mathbb{R}^{r \times r}$, $\hat{\mathbf{A}} \in \mathbb{R}^{r \times r}$, $\hat{\mathbf{B}} \in \mathbb{R}^{r \times m}$, $\hat{\mathbf{C}} \in \mathbb{R}^{p \times r}$ with $r \ll N$.

1. Partition the measurements into two disjoint sets:

Left data:

$$\mathbf{M} = \begin{bmatrix} \mu_1 & & \\ & \ddots & \\ & & \mu_q \end{bmatrix} \in \mathbb{C}^{q \times q}, \quad \mathbf{L} = \begin{bmatrix} \boldsymbol{\ell}_1^T \\ \vdots \\ \boldsymbol{\ell}_q^T \end{bmatrix} \in \mathbb{C}^{q \times p}, \quad \mathbf{V} = \begin{bmatrix} \mathbf{v}_1^T \\ \vdots \\ \mathbf{v}_q^T \end{bmatrix} \in \mathbb{C}^{q \times m}.$$

Right data:

$$\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_k \end{bmatrix} \in \mathbb{C}^{k \times k}, \quad \mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_k] \in \mathbb{C}^{m \times k},$$

$$\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_k] \in \mathbb{C}^{p \times k}.$$

2. Construct the **Loewner pencil** as

$$\mathbb{L} = \left(\frac{\mathbf{v}_i^T \mathbf{r}_i - \boldsymbol{\ell}_j^T \mathbf{w}_j}{\mu_i - \lambda_j} \right)_{i=1, \dots, q}^{j=1, \dots, k}, \quad \mathbb{L}_S = \left(\frac{\mu_i \mathbf{v}_i^T \mathbf{r}_i - \lambda_j \boldsymbol{\ell}_j^T \mathbf{w}_j}{\mu_i - \lambda_j} \right)_{i=1, \dots, q}^{j=1, \dots, k}.$$

3. It follows that the **complex raw model** is

$$\{\mathbf{W}, \mathbb{L}, \mathbb{L}_S, \mathbf{V}\}.$$

4. Transform all the complex data to real and there follows the **real raw model**:

$$\{\mathbf{W}_R, \mathbb{L}^R, \mathbb{L}_S^R, \mathbf{V}_R\}.$$

5. Compute the rank-revealing SVDs: $[\mathbf{Y}_1, \Sigma_1, \mathbf{X}_1] = \mathbf{SVD}([\mathbb{L}^R, \mathbb{L}_S^R])$ and $[\mathbf{Y}_2, \Sigma_2, \mathbf{X}_2] = \mathbf{SVD}([\mathbb{L}^R; \mathbb{L}_S^R])$; the decay of the singular values, lead to the choice of r .

6. The reduced real model is obtained by **projecting** the *raw real model* with $\mathbf{Y} = \mathbf{Y}_1^{n \times r}$ and $\mathbf{X} = \mathbf{X}_2^{n \times r}$.

$$\underbrace{\{\mathbf{W}_R, \mathbb{L}^R, \mathbb{L}_S^R, \mathbf{V}_R\}}_{\text{singular}} \xrightarrow{\text{SVD}} \underbrace{\{\mathbf{W}_R \mathbf{X}, \mathbf{Y}^T \mathbb{L}^R \mathbf{X}, \mathbf{Y}^T \mathbb{L}_S^R \mathbf{X}, \mathbf{Y}^T \mathbf{V}_R\}}_{\text{regular}} = \{\hat{\mathbf{C}}, -\hat{\mathbf{E}}, -\hat{\mathbf{A}}, \hat{\mathbf{B}}\}.$$

7. A real approximant of the data is

$$\hat{\mathbf{H}}(s) = \hat{\mathbf{C}}(s\hat{\mathbf{E}} - \hat{\mathbf{A}})^{-1}\hat{\mathbf{B}} \approx \boldsymbol{\phi}(s). \quad \square$$

6.2.6 Examples

In this section the theory will be illustrated by means of simple examples.

Example 6.1 (A spring–mass–damper system). Let m , d , and k denote the mass, damping, and stiffness of the spring as in Figure 6.1; let also $x(t)$ denote the displacement and $F(t)$ the force applied; the associated differential equation is

$$m\ddot{x}(t) + d\dot{x}(t) + kx(t) = F(t).$$

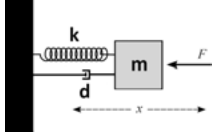


Figure 6.1: A spring–mass–damper system.

This is a SISO (single-input single-output) system. By introducing the state variables $x_1 = x$, $x_2 = \dot{x}$, the input $u = F$, and as output the velocity $y = \dot{x}$, the following state equations result:

$$\dot{x}_1(t) = x_2(t), \quad m\dot{x}_2(t) = -kx_1 - dx_2(t) + u(t), \quad y(t) = x_2(t).$$

The system matrices are thus

$$\mathbf{E} = \begin{bmatrix} 1 & 0 \\ 0 & m \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 0 & 1 \\ -k & -d \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 0 & 1 \end{bmatrix},$$

and the resulting transfer function is

$$\mathbf{H}(s) = \mathbf{C}(s\mathbf{E} - \mathbf{A})^{-1}\mathbf{B} = \frac{s}{ms^2 + ds + k}.$$

In the sequel we will assume for simplicity that all parameters have value one. We now wish to recover state equations equivalent to the ones above from measurements

of the transfer function. Toward this goal we evaluate the transfer function at the real frequencies: $\lambda_1 = \frac{1}{2}$, $\lambda_2 = 1$ (right frequencies), as well as $\mu_1 = -\frac{1}{2}$, $\mu_2 = -1$ (left frequencies). The corresponding values of \mathbf{H} are collected in the matrices

$$\mathbf{W} = \begin{pmatrix} \frac{2}{7} & \frac{1}{3} \end{pmatrix}, \quad \mathbf{V} = \begin{pmatrix} -\frac{2}{3} & -1 \end{pmatrix}^T.$$

Furthermore, with $\mathbf{R} = [1 \ 1] = \mathbf{L}^T$, we construct the *Loewner pencil*:

$$\mathbb{L} = \begin{bmatrix} \frac{20}{21} & \frac{2}{3} \\ \frac{6}{7} & \frac{2}{3} \end{bmatrix}, \quad \mathbb{L}_s = \begin{bmatrix} -\frac{4}{21} & 0 \\ -\frac{4}{7} & -\frac{1}{3} \end{bmatrix}.$$

Since the pencil $(\mathbb{L}_s, \mathbb{L})$ is regular, we recover the original transfer function:

$$\mathbf{H}(s) = \mathbf{W}\Phi(s)^{-1}\mathbf{V} = \frac{s}{s^2 + s + 1}, \quad \text{where } \Phi(s) = \mathbb{L}_s - s\mathbb{L}.$$

Hence, the measurements above yield a minimal (descriptor) realization of the system in terms of the (state) variables $\mathbf{z}_1, \mathbf{z}_2$:

$$\begin{aligned} \frac{20}{21}\dot{\mathbf{z}}_1(t) + \frac{2}{3}\dot{\mathbf{z}}_2(t) &= \frac{4}{21}\mathbf{z}_1(t) + \frac{2}{3}\mathbf{u}(t), \\ \frac{6}{7}\dot{\mathbf{z}}_1(t) + \frac{2}{3}\dot{\mathbf{z}}_2(t) &= -\frac{4}{7}\mathbf{z}_1(t) - \frac{1}{3}\mathbf{z}_2(t) + \mathbf{u}(t), \quad \mathbf{y}(t) = \frac{2}{7}\mathbf{z}_1(t) + \frac{1}{3}\mathbf{z}_2(t), \end{aligned}$$

with

$$\tilde{\mathbf{E}} = \begin{bmatrix} \frac{20}{21} & \frac{2}{3} \\ \frac{6}{7} & \frac{2}{3} \end{bmatrix}, \quad \tilde{\mathbf{A}} = \begin{bmatrix} -\frac{4}{21} & 0 \\ -\frac{4}{7} & -\frac{1}{3} \end{bmatrix}, \quad \tilde{\mathbf{B}} = \begin{bmatrix} \frac{2}{3} \\ 1 \end{bmatrix}, \quad \tilde{\mathbf{C}} = \begin{bmatrix} \frac{2}{7} & \frac{1}{3} \end{bmatrix}.$$

By multiplying with $\tilde{\mathbf{E}}^{-1}$, it yields (id: identified system in state-space form)

$$\tilde{\mathbf{A}}_{id} = \begin{bmatrix} 4 & \frac{7}{2} \\ -6 & -5 \end{bmatrix}, \quad \tilde{\mathbf{B}}_{id} = \begin{bmatrix} -\frac{7}{2} \\ 6 \end{bmatrix}, \quad \tilde{\mathbf{C}}_{id} = \begin{bmatrix} \frac{2}{7} & \frac{1}{3} \end{bmatrix}.$$

Coordinate transformation Let the state vector \mathbf{x} be transformed to the new state vector \mathbf{z} by the non-singular transformation matrix

$$\Psi = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \end{bmatrix}^{-1} \begin{bmatrix} \tilde{\mathbf{C}}_{id} \\ \tilde{\mathbf{C}}_{id}\tilde{\mathbf{A}}_{id} \end{bmatrix},$$

of dimension 2×2 . Then the following hold:

$$\mathbf{z} = \Psi^{-1}\mathbf{x}, \quad \tilde{\mathbf{A}}_{id} = \Psi^{-1}\mathbf{A}\Psi, \quad \tilde{\mathbf{B}}_{id} = \Psi^{-1}\mathbf{B}, \quad \tilde{\mathbf{C}}_{id} = \mathbf{C}\Psi;$$

e. g., $\Psi\tilde{\mathbf{A}}_{id}\Psi^{-1} = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} = \mathbf{A}$.

Remark 6.5. The above result ensures that the Loewner framework constitutes a data-driven system identification method which constructs a realization only from measurements. It is important to mention that under a coordinate transformation, both systems, initial and identified are identical. At the same time, the underlying dynamics is recovered exactly while the corresponding revealing transfer function remains invariant under such a transformation.

The question now arises: what happens if we collect more data than necessary? Let us consider

$$\mathbf{\Lambda} = \text{diag} \left(\frac{1}{2} \quad 1 \quad \frac{3}{2} \quad 2 \right), \quad \mathbf{M} = \text{diag} \left(-\frac{1}{2} \quad -1 \quad -\frac{3}{2} \quad -2 \right).$$

In this case, the associated measurements are

$$\mathbf{W} = \left(\frac{2}{7} \quad \frac{1}{3} \quad \frac{6}{19} \quad \frac{2}{7} \right), \quad \mathbf{V} = \left(-\frac{2}{3} \quad -1 \quad -\frac{6}{7} \quad -\frac{2}{3} \right)^T,$$

and with $\mathbf{R} = [1 \ 1 \ 1 \ 1] = \mathbf{L}^T$, the Loewner pencil is

$$\mathbb{L} = \left[\begin{array}{cc|cc} \frac{20}{21} & \frac{2}{3} & \frac{28}{57} & \frac{8}{21} \\ \frac{6}{7} & \frac{2}{3} & \frac{10}{19} & \frac{3}{7} \\ \hline \frac{4}{7} & \frac{10}{21} & \frac{52}{133} & \frac{16}{49} \\ \frac{8}{21} & \frac{1}{3} & \frac{16}{57} & \frac{5}{21} \end{array} \right], \quad \mathbb{L}_s = \left[\begin{array}{cc|cc} -\frac{4}{21} & 0 & \frac{4}{57} & \frac{2}{21} \\ -\frac{4}{7} & -\frac{1}{3} & -\frac{4}{19} & -\frac{1}{7} \\ \hline -\frac{4}{7} & -\frac{8}{21} & -\frac{36}{133} & -\frac{10}{49} \\ -\frac{10}{21} & -\frac{1}{3} & -\frac{14}{57} & -\frac{4}{21} \end{array} \right].$$

It turns out that we can choose *arbitrary* matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{4 \times 2}$, provided that $\det(\mathbf{Y}^T \mathbf{X}) \neq 0$, e. g.

$$\mathbf{X} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ -2 & 1 \end{bmatrix}, \quad \mathbf{Y}^T = \begin{bmatrix} 0 & 1 & 0 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix},$$

so that the projected quantities

$$\begin{aligned} \widehat{\mathbf{W}} = \mathbf{W}\mathbf{X} &= \begin{bmatrix} -\frac{6}{7} & -\frac{1}{21} \end{bmatrix}, \quad \widehat{\mathbb{L}} = \mathbf{Y}^T \mathbb{L} \mathbf{X} = \begin{bmatrix} -\frac{6}{7} & -\frac{1}{7} \\ \frac{18}{49} & \frac{1}{147} \end{bmatrix}, \\ \widehat{\mathbb{L}}_s &= \mathbf{Y}^T \mathbb{L}_s \mathbf{X} = \begin{bmatrix} 0 & \frac{1}{21} \\ -\frac{48}{49} & -\frac{19}{147} \end{bmatrix}, \quad \widehat{\mathbf{V}} = \mathbf{Y}^T \mathbf{V} = \begin{bmatrix} -\frac{1}{3} \\ \frac{11}{21} \end{bmatrix}, \end{aligned}$$

constitute a minimal realization of $\mathbf{H}(s)$:

$$\mathbf{H}(s) = \widehat{\mathbf{W}}(\widehat{\mathbb{L}}_s - s\widehat{\mathbb{L}})^{-1}\widehat{\mathbf{V}} = \frac{s}{s^2 + s + 1}.$$

It should be stressed that this holds for *arbitrary* projection matrices \mathbf{X}, \mathbf{Y} .

6.2.6.1 The generalized inverse approach

There is another way to express the above relationship avoiding arbitrary projectors. Basic ingredients are *generalized inverses*. This approach will be demonstrated only for the spring–mass–damper example. However, it holds in general (see, e. g., [3]).

In the sequel, we will make use (only) of the *Moore–Penrose* generalized inverse, which is defined as follows. Given the (rectangular) matrix $\mathbf{M} \in \mathbb{R}^{q \times k}$, the Moore–Penrose generalized inverse, denoted by $\mathbf{M}^{\text{MP}} \in \mathbb{R}^{k \times q}$, satisfies:

- (a) $\mathbf{M}\mathbf{M}^{\text{MP}}\mathbf{M} = \mathbf{M}$,
- (b) $\mathbf{M}^{\text{MP}}\mathbf{M}\mathbf{M}^{\text{MP}} = \mathbf{M}^{\text{MP}}$,
- (c) $[\mathbf{M}\mathbf{M}^{\text{MP}}]^T = \mathbf{M}\mathbf{M}^{\text{MP}}$,
- (d) $[\mathbf{M}^{\text{MP}}\mathbf{M}]^T = \mathbf{M}^{\text{MP}}\mathbf{M}$.

This generalized inverse always exists and is unique.

In the sequel we will be concerned with rectangular $q \times k$ *polynomial matrices* which have an explicit (rank-revealing) factorization as follows:

$$\mathbf{M} = \mathbf{X}\mathbf{\Delta}\mathbf{Y}^T,$$

where \mathbf{X} , $\mathbf{\Delta}$, \mathbf{Y} have dimension $q \times n$, $n \times n$, $k \times n$, $n \leq q, k$, and all have full rank n . In such cases, the **Moore–Penrose** generalized inverse is

$$\mathbf{M}^{\text{MP}} = \mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{\Delta}^{-1}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T.$$

Mass–spring–damper example (continued). The quantity needed is the generalized inverse of

$$\Phi(s) = \mathbb{I}_s - s\mathbb{L} = \begin{bmatrix} -\frac{20s}{21} - \frac{4}{21} & -\frac{2s}{3} & \frac{4}{57} - \frac{28s}{57} & \frac{2}{21} - \frac{8s}{21} \\ -\frac{6s}{7} - \frac{4}{7} & -\frac{2s}{3} - \frac{1}{3} & -\frac{10s}{19} - \frac{4}{19} & -\frac{3s}{7} - \frac{1}{7} \\ -\frac{4s}{7} - \frac{4}{7} & -\frac{10s}{21} - \frac{8}{21} & -\frac{52s}{133} - \frac{36}{133} & -\frac{16s}{49} - \frac{10}{49} \\ -\frac{8s}{21} - \frac{10}{21} & -\frac{s}{3} - \frac{1}{3} & -\frac{16s}{57} - \frac{14}{57} & -\frac{5s}{21} - \frac{4}{21} \end{bmatrix}. \quad (6.22)$$

We first notice that $\Phi(s) = \mathbf{X}\mathbf{\Delta}(s)\mathbf{Y}^T$, where \mathbf{X} and \mathbf{Y} can be chosen as follows:

$$\mathbf{X} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -\frac{3}{7} & \frac{8}{7} \\ -\frac{1}{2} & 1 \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} 1 & 0 & -\frac{7}{19} & -\frac{1}{2} \\ 0 & 1 & \frac{24}{19} & \frac{9}{7} \end{bmatrix} \Rightarrow \det(\mathbf{Y}\mathbf{X}) \neq 0.$$

Thus by taking the 2×2 upper-left block as $\Delta(s) = \Phi(1 : 2, 1 : 2)(s)$, it follows that $\Phi(s)^{\text{MP}} = \frac{1}{80989667} \frac{1}{s^2+s+1} \mathbf{Z}(s)$, where

$$\mathbf{Z}(s) = \begin{bmatrix} -28(11610185s + 7274073) & 14(3558666s - 5604037) \\ 294(225182s + 281171) & (-147)(192415s - 19668) \\ 3724(54617s + 48189) & (-1862)(29046s - 17485) \\ 98(2527157s + 2123670) & -49(1250553s - 876439) \\ 6076(32301s - 391) & 14(15168851s + 1670036) \\ -2058(29494s + 15609) & -147(417597s + 261503) \\ -26068(5715s + 1523) & -1862(83663s + 30704) \\ -98(1797669s + 409322) & -49(3777710s + 1247231) \end{bmatrix}$$

In the *rectangular case*, where there are two right measurements less, i. e., we only have $\tilde{\mathbf{A}} = \text{diag}[\frac{1}{2}, 1]$, while \mathbf{M} remains the same, the right values are $\tilde{\mathbf{W}} = \mathbf{W}(:, 1 : 2)$; hence

$$\tilde{\Phi}(s) = \tilde{\mathbf{L}}_s - s\tilde{\mathbf{L}} = \left[\begin{array}{c|c} -\frac{20s}{21} - \frac{4}{21} & -\frac{2s}{3} \\ -\frac{6s}{7} - \frac{4}{7} & -\frac{2s}{3} - \frac{1}{3} \\ -\frac{4s}{7} - \frac{4}{7} & -\frac{10s}{21} - \frac{8}{21} \\ -\frac{8s}{21} - \frac{10}{21} & -\frac{s}{3} - \frac{1}{3} \end{array} \right] = \mathbf{X}\Delta(s)\tilde{\mathbf{Y}}^T,$$

has dimension 4×2 , where $\tilde{\mathbf{Y}} = \mathbf{Y}(1 : 2, 1 : 2)$. In this case the Moore–Penrose inverse is

$$\tilde{\Phi}(s)^{\text{MP}} = \frac{1}{737(s^2 + s + 1)} \times \begin{bmatrix} -4767s - 3402 & \frac{1827}{2}s - \frac{2037}{2} & 3087s + 294 & 3297s + \frac{1365}{2} \\ 5838s + 5250 & -1596s + 903 & -4326s - 1218 & -4515s - 1722 \end{bmatrix},$$

which implies the desired equality

$$\Rightarrow \mathbf{W}\Phi(s)^{\text{MP}}\mathbf{V} = \tilde{\mathbf{W}}\tilde{\Phi}(s)^{\text{MP}}\mathbf{V} = \mathbf{H}(s).$$

Conclusion: the Loewner framework allows the definition of rectangular and singular systems. \square

Example 6.2 (Reduction of a 10th order band-stop filter). The system has two inputs and two outputs (MIMO), state-space dimension 10, and a \mathbf{D} term of rank 2. A state-space representation is as follows:

$$\Sigma: \quad \dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \quad \text{where}$$

$$\mathbf{A} = \left[\begin{array}{ccccc|ccccc} -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -1 & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & 0 & -1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & -1 & 0 & 0 \\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & -1 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & -1 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array} \right], \quad \mathbf{B} = \left[\begin{array}{cc} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \\ \hline 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{array} \right],$$

$$\mathbf{C} = \left[\begin{array}{ccccc|ccccc} -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 \end{array} \right], \quad \mathbf{D} = \left[\begin{array}{cc} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{array} \right].$$

The transfer function is a 2×2 rational matrix given by

$$\mathbf{H}(s) = \frac{1}{\mathbf{d}(s)} \begin{bmatrix} \mathbf{n}_1(s) & \mathbf{n}_2(s) \\ -\mathbf{n}_2(s) & -\mathbf{n}_1(s) \end{bmatrix} + \mathbf{D}, \quad \text{where}$$

$$\mathbf{n}_1(s) = s(s^8 + 7s^6 + 13s^4 + 7s^2 + 1),$$

$$\mathbf{n}_2(s) = s(5s^8 + 6s^7 + 25s^6 + 20s^5 + 41s^4 + 20s^3 + 25s^2 + 6s + 5),$$

$$\mathbf{d}(s) = 2(s^4 + s^3 + 3s^2 + 2s + 1)(2s^6 + 3s^5 + 7s^4 + 7s^3 + 7s^2 + 3s + 2).$$

It readily follows that $\lim_{s \rightarrow \infty} \mathbf{H}(s) = \mathbf{D}$. We take $N = 100$ samples of the transfer function on the imaginary axis (frequency response measurements) between 10^{-1} and 10^1 rad/sec. Figure 6.3 (left) shows the first 20 normalized singular values of the resulting real Loewner pencil (the rest are numerically zero). The rank of \mathbb{L} is 10 (the McMillan degree of the system) while the rank of \mathbb{L}_s is 12 ($= \text{rank } \mathbb{L} + \text{rank } \mathbf{D}$). The right pane in Figure 6.2 shows that we can obtain a perfect fit (total recovery of the model) with the Loewner framework for this MIMO example only by sampling the transfer function. As both Gramians are $\mathcal{P} = \mathcal{Q} = \frac{1}{2}\mathbf{I}_{10}$, i. e., they are equal and a multiple of the identity matrix, the Hankel singular values (see [2]) are all equal; this makes reduction with balanced truncation not feasible.

The right pane in Figure 6.3 shows the poles of the system obtained by means of the Loewner framework along with the zeros for every entry. The right pane in Figure 6.2 shows the band-stop character around frequency $\omega_0 = 1$ rad/s, of entries (1, 2) and (2, 1).

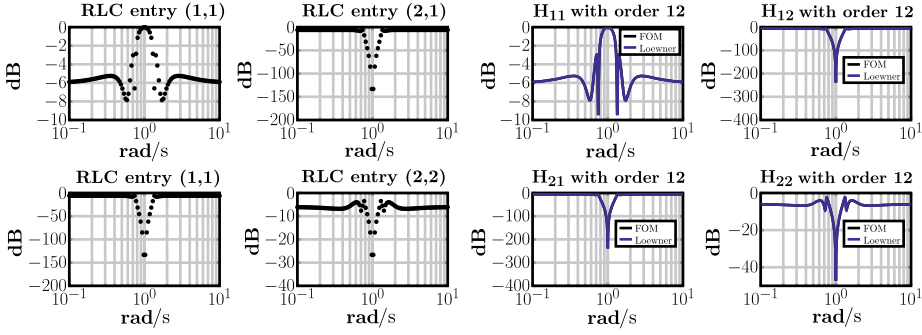


Figure 6.2: Left pane: Shows the 100 measurements sampled with DNS(Direct Numerical Simulations) of the theoretical (2×2) -matrix transfer function. Right pane: Loewner approximants.

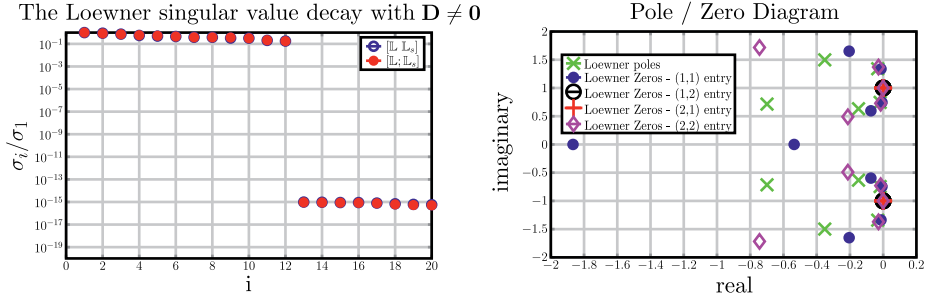


Figure 6.3: Left pane: Shows the first 12 singular values while the rest are numerically zero. Right pane: Pole/Zero diagram.

Computing the poles of the Loewner model confirms the accuracy of the approach. Consider the following:

$\text{eig}(\mathbf{A})$	$\text{eig}(\mathbf{A}_r, \mathbf{E}_r)$
$-0.0181885913675508 - 0.745231200229 i$	$-0.0181885913675508 - 0.745231200229 i$
$-0.0181885913675508 + 0.745231200229 i$	$-0.0181885913675508 + 0.745231200229 i$
$-0.148402943598342 - 0.632502179219046 i$	$-0.148402943598342 - 0.632502179219046 i$
$-0.148402943598342 + 0.632502179219046 i$	$-0.148402943598342 + 0.632502179219046 i$
$-0.699080475814867 - 0.715042997542469 i$	$-0.699080475814867 - 0.715042997542469 i$
$-0.699080475814867 + 0.715042997542469 i$	$-0.699080475814867 + 0.715042997542469 i$
$-0.0327309328175858 - 1.34106659803138 i$	$-0.0327309328175858 - 1.34106659803138 i$
$-0.0327309328175858 + 1.34106659803138 i$	$-0.0327309328175858 + 1.34106659803138 i$
$-0.351597056401658 - 1.49852758300335 i$	$-0.351597056401658 - 1.49852758300335 i$
$-0.351597056401658 + 1.49852758300335 i$	$-0.351597056401658 + 1.49852758300335 i$
	∞
	∞

As can be observed from this table, the Loewner method computes, besides the finite poles, two poles at infinity. This happens because in the Loewner framework the \mathbf{D} -term is incorporated in the remaining matrices of the realization.

6.2.7 Summary

Recall Section 6.2.5. The following result summarizes the cases which arise in the Loewner framework, depending on the amount of data available.

Lemma 6.2. *Given is a scalar transfer function of McMillan degree n .*

1. *Amount of data less than $2n$. For $q = k \leq n$, define the transfer function $\hat{\mathbf{H}}(s) = \hat{\mathbf{C}}(s\hat{\mathbf{E}} - \hat{\mathbf{A}})^{-1}\hat{\mathbf{B}}$, by means of the Loewner procedure. The interpolation conditions below are satisfied:*

$$\hat{\mathbf{H}}(\mu_i) = \mathbf{H}(\mu_i) \quad \text{and} \quad \hat{\mathbf{H}}(\lambda_j) = \mathbf{H}(\lambda_j) \quad \text{for } i = 1, \dots, k.$$

If $k = q = n$, the Loewner quadruple is equivalent to the original one $(\mathbf{C}, \mathbf{E}, \mathbf{A}, \mathbf{B})$.

2. *Arbitrary amount of data, no reduction. For arbitrary k and q (i.e. $k, q \leq n$ or $k, q \geq n$) the Loewner quadruple interpolates the data, even if the pencil $(\mathbb{L}_s, \mathbb{L})$ is singular. This is to be interpreted as follows:*

$$(\mathbb{L}_s - \lambda_i \mathbb{L})\mathbf{e}_i = \mathbb{V} \quad \text{and} \quad \mathbf{e}_j^T (\mathbb{L}_s - \mu_j \mathbb{L}) = \mathbb{W}.$$

Hence $\mathbb{W}\mathbf{e}_i = \mathbf{w}_i$, $i = 1, \dots, k$, and $\mathbf{e}_j^T \mathbb{V} = \mathbf{v}_j$, $j = 1, \dots, q$. Therefore the transfer function of the Loewner pencil interpolates $\mathbf{H}(s)$ at the left and right interpolation points.

3. *Arbitrary amount of data, followed by reduction. If $k, q \geq n$, consider the rank-revealing SVD decompositions:*

$$[\mathbb{L} \quad \mathbb{L}_s] = \hat{\mathbf{Y}}_r \hat{\Sigma}_r \mathbf{X}_r^T \quad \text{and} \quad \begin{bmatrix} \mathbb{L} \\ \mathbb{L}_s \end{bmatrix} = \mathbf{Y}_r \tilde{\Sigma}_r \tilde{\mathbf{X}}_r^T,$$

where $\mathbf{Y}_r \in \mathbb{R}^{q \times r}$, $\mathbf{X}_r \in \mathbb{R}^{k \times r}$, and $r \leq k, q$, is the exact or the numerical rank of the Loewner pencil involved. Let

$$\tilde{\mathbf{E}} = \mathbf{Y}_r^T \mathbb{L} \mathbf{X}_r, \quad \tilde{\mathbf{A}} = \mathbf{Y}_r^T \mathbb{L}_s \mathbf{X}_r \in \mathbb{C}^{r \times r}, \quad \tilde{\mathbf{B}} = \mathbf{Y}_r^T \mathbb{V} \in \mathbb{C}^r, \quad \tilde{\mathbf{C}} = \mathbb{W} \mathbf{X}_r \in \mathbb{C}^{1 \times r}.$$

Then the following approximate interpolation conditions are satisfied:

$$\tilde{\mathbf{H}}(\mu_i) \approx \mathbf{H}(\mu_i), \quad i = 1, \dots, q, \quad \text{and} \quad \tilde{\mathbf{H}}(\lambda_j) \approx \mathbf{H}(\lambda_j), \quad j = 1, \dots, k.$$

In addition, the reduced system satisfies (exact) interpolation conditions as shown in Section 6.2.5.3.

6.3 Practical considerations

This section deals with some key aspects of the Loewner framework, through a practical point of view.

- The factorization of the Loewner matrix into low-rank factor matrices:
 1. Through the singular value decomposition (SVD).
 2. Through the CUR decomposition.
- The choices involving the measurements used in the framework:
 1. Distribution of the interpolation points.
 2. Partition of the interpolation points.

6.3.1 The singular value decomposition

The SVD is arguably one of the most useful and commonly used tools in numerical linear algebra. It is listed as one of the main matrix decompositions and can be efficiently computed through various numerically stable algorithms. It is widely used for different high dimension reduction and approximation methods.

Any complex-valued matrix $\mathbf{A} \in \mathbb{C}^{n \times m}$ has a singular value decomposition given by $\mathbf{A} = \mathbf{Y}\mathbf{\Sigma}\mathbf{X}^*$ where $\mathbf{Y} \in \mathbb{C}^{n \times n}$, $\mathbf{X} \in \mathbb{C}^{m \times m}$ are unitary matrices, i. e., $\mathbf{Y}^*\mathbf{Y} = \mathbf{I}_n$ and $\mathbf{X}^*\mathbf{X} = \mathbf{I}_m$. The left and right singular vectors are collected as columns of matrices \mathbf{X} , and \mathbf{Y} , respectively.

Additionally, the matrix $\mathbf{\Sigma} \in \mathbb{C}^{n \times m}$ is defined as $\Sigma_{i,i} = \sigma_i$ and zero elsewhere. Here, the ordered non-negative scalars $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$ are the singular values (for $n \leq m$).

In what follows, it is assumed that matrix \mathbf{A} has low rank, i. e., $\text{rank}(\mathbf{A}) = r \leq n \leq m$. Let k be a positive integer so that $k < r$. The singular value decomposition of matrix \mathbf{A} can be additively split as follows:

$$\mathbf{A} = \mathbf{Y} \cdot \mathbf{\Sigma} \cdot \mathbf{X}^* = \underbrace{(\mathbf{Y}_k \quad \mathbf{Y}_{n-k})}_{n \times n} \cdot \underbrace{\begin{pmatrix} \mathbf{\Sigma}_k & \mathbf{0}_{k,m-k} \\ \mathbf{0}_{n-k,k} & \mathbf{\Sigma}_{n-k,m-k} \end{pmatrix}}_{n \times m} \cdot \underbrace{\begin{pmatrix} \mathbf{X}_k^* \\ \mathbf{X}_{m-k}^* \end{pmatrix}}_{m \times m} \quad (6.23)$$

$$= \underbrace{\mathbf{Y}_k \mathbf{\Sigma}_k \mathbf{X}_k^*}_{:= \mathbf{A}_k} + \mathbf{Y}_{n-k} \mathbf{\Sigma}_{n-k,m-k} \mathbf{X}_{m-k}^* \quad (6.24)$$

where $\mathbf{Y}_k \in \mathbb{C}^{n \times k}$, $\mathbf{\Sigma}_k \in \mathbb{C}^{k \times k}$ and $\mathbf{X}_k \in \mathbb{R}^{m \times k}$. Note that the matrix $\mathbf{A}_k := \mathbf{Y}_k \mathbf{\Sigma}_k \mathbf{X}_k^* \in \mathbb{C}^{m \times n}$ can be written in terms of the truncated dyadic decomposition, i. e., $\mathbf{A}_k = \sum_{i=1}^k \sigma_i y_i x_i^*$, where y_i and x_i are the i th column of matrices \mathbf{Y} , and \mathbf{X} , respectively.

A problem of interest is to approximate the original matrix \mathbf{A} with a rank k matrix \mathbf{T} , so that the approximation error is minimal with respect to the 2-induced norm or to the Frobenius norm.

From the Schmidt–Eckart–Young–Mirsky theorem (see Theorem 3.6 in [2]), it follows that (given $\sigma_k > \sigma_{k+1}$)

$$\min_{\mathbf{T} \in \mathbb{R}^{n \times m}, \text{rank}(\mathbf{T}) \leq k} \|\mathbf{A} - \mathbf{T}\|_2 = \sigma_{k+1}. \quad (6.25)$$

Moreover, it turns out that the unique solution to the minimization problem in (6.25) is given by $\mathbf{T} = \mathbf{A}_k$. If we replace the 2-induced norm with the Frobenius norm, it follows

that

$$\min_{\mathbf{T} \in \mathbb{R}^{n \times m}, \text{rank}(\mathbf{T}) \leq k} \|\mathbf{A} - \mathbf{T}\|_F = \left(\sum_{i=k+1}^n \sigma_i^2 \right)^{\frac{1}{2}}. \quad (6.26)$$

As before, the unique solution to the minimization problem in (6.26) is again given by $\mathbf{T} = \mathbf{A}_k$. For more details on the singular value decomposition (SVD) we refer the reader to [2], pages 31–41.

The advantage of the SVD is that it offers optimal low-rank solutions in both the 2-induced and the Frobenius norms. At the same time, one disadvantage is given by the fact that the method (full SVD) has cubic complexity with respect to $\min(m, n)$ (in the classical set-up when applied to dense matrices). Taking into account this possible downside, we seek ways of circumventing the usage of the classical SVD and investigate other matrix decompositions. It is worth mentioning that SVD complexity can be faster than cubic for a low-rank approximation with iterative algorithm. In the latter, a randomized version of SVD (r-SVD) will reveal this robust behavior.

6.3.2 The CUR decomposition

A challenging aspect of data-driven approximation methods is the choice of a relevant and meaningful low dimensional subset of the (usually large-scale) data set. In some cases, this subset can be used to preserve relevant characteristics of the dynamics for the model described by the original data. In this framework, it is of interest to devise procedures that can extract relevant information from large-scale sets of measurements. The end goal is to construct reduced order models suitable for tasks such as control, design, and simulation.

Nowadays, the dimension of data sets for various applications can easily reach $\approx \mathcal{O}(10^6)$. In such cases, computing the SVD of large and full matrices becomes prohibitive.

One appealing alternative is the so-called CUR decomposition. As before, the goal is to approximate the original matrix $\mathbf{A} \in \mathbb{C}^{n \times m}$, by a product of three low-rank matrices $\hat{\mathbf{A}} = \mathbf{C}\mathbf{U}\mathbf{R}$. Here, the columns of the matrix $\mathbf{C} \in \mathbb{C}^{n \times c}$ are represent a subset of the columns of \mathbf{A} while the rows of the matrix $\mathbf{R} \in \mathbb{C}^{r \times m}$ form a subset of the rows of \mathbf{A} . Finally, the matrix $\mathbf{U} \in \mathbb{C}^{c \times r}$ is constructed such that the factorization $\hat{\mathbf{A}} = \mathbf{C}\mathbf{U}\mathbf{R}$ holds.

In this new set-up, the left and right singular vectors appearing in the SVD are replaced by columns and rows of the initial matrix \mathbf{A} . Hence, the CUR factorization provides a way of identifying important sets of rows and columns of a matrix \mathbf{A} .

For more details on the CUR decomposition and some of its applications, we refer the reader to [19, 26, 27, 28, 47, 51, 54, 63].

The CUR factorization is hence an important tool for analyzing large-scale data sets which offers the following advantages over SVD:

1. If the matrix \mathbf{A} is sparse, then the matrices \mathbf{C} and \mathbf{R} are also sparse (unlike the matrices \mathbf{X} and \mathbf{Y} in the SVD approach).
2. The CUR factorization computes an approximation of \mathbf{A} in terms of some of the rows and some of the columns of \mathbf{A} . In contrast, the SVD computes approximants in terms of linear combinations of orthonormal bases generated by the rows and columns of \mathbf{A} .
3. Consider $\mathbf{A} \in \mathbb{R}^{m \times n}$, $m > n$. The complexity for computing the full SVD of \mathbf{A} is $O(n^3)$ flops, using for instance the QR factorization, $O(mn^2)$ flops, using iterative methods as in ARPACK, and $O((m+n)k)$ flops per iteration, for approximate incremental methods where the k dominant singular triples are determined approximately (for details see [12]). On the other hand the CUR factorization of order k requires $O(k^3 + k^2(m+n))$ flops per iteration (for details see [47]).

6.3.2.1 CUR approximation of the Loewner matrix

In this section, we apply the CUR factorization to the Loewner matrix. We follow [47], where CUR is applied to Hankel matrices instead.

Definition 6.3. With $\mathbb{L} \in \mathbb{R}^{n \times n}$, let $\mathcal{I} = \{i_1, \dots, i_r\}$ and $\mathcal{J} = \{j_1, \dots, j_r\}$ denote the r -subsets ($r \ll n$) of row and column indices, respectively. If $(\cdot)^{\text{MP}}$ denotes the pseudo inverse, then the CUR factorization of the Loewner matrix \mathbb{L} is given by

$$\mathbb{L}_r := \underbrace{\mathbb{L}(:, \mathcal{J})}_{\mathcal{J}\text{-columns}} \cdot \mathbb{L}(\mathcal{I}, \mathcal{J})^{\text{MP}} \cdot \underbrace{\mathbb{L}(\mathcal{I}, :)}_{\mathcal{I}\text{-rows}}. \quad (6.27)$$

In practical applications, large-scale data matrices are only approximately of low rank (when data can be for instance corrupted by noise). In this case, the sets \mathcal{I} and \mathcal{J} need to be chosen in such a way that the approximation error $\|\mathbb{L} - \mathbb{L}_r\|$ is small. Many approaches for selecting the sets of rows and columns have been proposed. In the following we mention only some of them.

1. Selection based on a maximum volume sub-matrix in [54].
2. Selection based on minimizing the approximation error in the Chebyshev norm (“skeleton” approximation) in [26, 27].
3. Procedure based on the “cross-approximation” algorithm in [55].
4. Selection based on a discrete empirical interpolation method (DEIM) approach in [63].

6.3.2.2 The Loewner CUR algorithm

We introduce a data-driven approximation algorithm for the SISO case based on CUR approach. This constructs a reduced order model by means of an adaptive selection of the rows and columns via the cross-approximation algorithm in [55]. The steps of the procedure are included in Algorithm 6.3.

Algorithm 6.3: Loewner CUR-cross-approximation based – SISO [44].

Input: $\mathbf{S} = [s_1, \dots, s_N] \in \mathbb{C}^N$, $\mathbf{F} = [\phi_1, \dots, \phi_N] \in \mathbb{C}^N$ with $N, r \in \mathbb{N}$, and tolerance values δ, ϵ .

Output: $\hat{\mathbf{E}} \in \mathbb{R}^{r \times r}$, $\hat{\mathbf{A}} \in \mathbb{R}^{r \times r}$, $\hat{\mathbf{B}} \in \mathbb{R}^{r \times 1}$, $\hat{\mathbf{C}} \in \mathbb{R}^{1 \times r}$ with $r \ll N$.

1. Form the left and right sets as (μ_j, \mathbf{v}_j) , $j = 1, \dots, q$ and $(\lambda_i, \mathbf{w}_i)$, $i = 1, \dots, k$
2. Form the Loewner matrices \mathbb{L} and \mathbb{L}_s as in Algorithm 6.1 and step 2.
3. Transform all the complex data to real as explained in Section 6.2.5.4.
4. $\mathcal{J}_0 = [j_1, \dots, j_r] \subset \mathcal{J}_n$ an initial set of column indices.
5. $[\mathcal{I}_r, \sim, \sim] = \text{crossapprox}([\mathbb{L} \ \mathbb{L}_s], \mathcal{J}_0, \delta, \epsilon)$.
6. $[\sim, \mathcal{J}_r, \sim] = \text{crossapprox}([\frac{\mathbb{L}}{\mathbb{L}_s}], \mathcal{I}_r, \delta, \epsilon)$.
7. $\hat{\mathbf{E}} = -\mathbb{L}(\mathcal{I}_r, \mathcal{J}_r)$, $\hat{\mathbf{A}} = -\mathbb{L}_s(\mathcal{I}_r, \mathcal{J}_r)$, $\hat{\mathbf{B}} = \mathbb{V}(\mathcal{I}_r)$, $\hat{\mathbf{C}} = \mathbb{W}(\mathcal{J}_r)$.
8. The rational approximant is given by

$$\mathbf{H}_r(s) = \hat{\mathbf{C}}(s\hat{\mathbf{E}} - \hat{\mathbf{A}})^{-1}\hat{\mathbf{B}}.$$

□

For the practical implementation of the function “*crossapprox*”, used in steps 5 and 6 of the above algorithm, we refer the reader to Algorithm 1 in [47], or to the original reference [55].

Remark 6.6. Instead of using the cross-approximation algorithm, one can use the DEIM (Discrete Empirical Interpolation Method) algorithm from [63]. Hence, steps 5 and 6 in Algorithm 6.3 need to be modified accordingly. As a result, singular value decompositions are performed in order to construct left and right singular vector matrices (for which the DEIM procedure is applied to). In order to avoid the SVD, an incremental QR factorization can be instead used, as proposed in [63].

Remark 6.7. The CUR factorization directly reveals the dominant rows/columns of the data, while the SVD does not. More precisely, the leading singular vectors give only linear combinations of the underlying features. Whereas, with the CUR one gets an actual subset of the initial features (columns) together with the corresponding rows. Consequently, a first benefit of the CUR is that it preserves the physical meaning and structure of the initial data. Additionally, another advantage is that the sparsity is preserved.

6.3.3 Choice of left and right interpolation points

This section deals with the problem of selecting the initial interpolation points in the Loewner framework. More specifically, we investigate how the choice of the initial interpolation points affects the quality of the reduced order model. We take into consideration different point distributions in 1D or in 2D.

Moreover, several splitting techniques are analyzed. These are related to the partition of the data set into two disjoint subsets, which is performed in the beginning of the algorithms in the Loewner framework.

6.3.3.1 Distribution of the interpolation points

We present various distributions of the initial interpolation points for the one-dimensional case (1D) as well as for the two-dimensional case (2D).

1D interpolation grid	2D interpolation grid
equispaced;	equispaced (“same areas”);
logarithmic spaced;	logarithmic spaced;
Chebyshev nodes;	Padua points;
Uniformly random	Uniformly random

In Figure 6.4, we depict different distributions of initial interpolation points. One way of selecting points is that of equispaced or linearly spaced points, commonly used for Fourier analysis. This represents a natural choice because of the usage of trigonometric periodic functions.

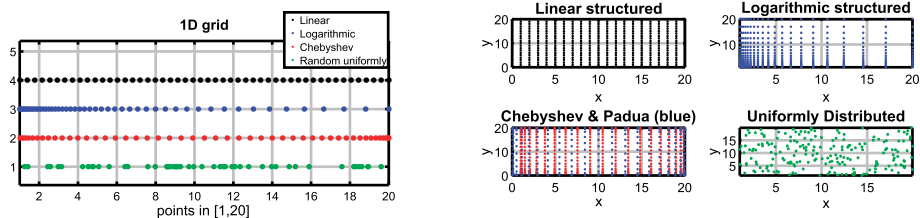


Figure 6.4: A visual representation of different interpolation grids.

In some practical applications, under the assumption that the energy decreases exponentially as time or frequency approach infinity (on an unbounded domain), the choice of logarithmic distributed points is more appropriate.

Naturally, a dense sampling grid can be used in the beginning of the experiment (e. g., for a lower frequency range or for small time instances). The motivation for this approach stems from the assumption that the meaningful quantities (with high energy or with relevant oscillations) appear in the beginning, hence requiring more samples. Afterwards, a more sparse distribution grid of points can be instead chosen as the energy level decays (or as relevant oscillations decay in time).

Additionally, the choice of Chebyshev-type points is motivated by their usage in polynomial-based interpolation on bounded domains due to for example, the elimination of the Runge phenomenon⁴ (high degree polynomials are generally unsuitable for interpolation with equispaced points).

Finally, randomly distributed sampling points often appear in stochastic experiments that are characterized by randomness.

6.3.3.2 Partition of the data points and values

Data splitting is one of the first steps in the classical Loewner algorithm (presented in Section 6.2). In this section, we mention various splitting schemes and how they affect the Loewner matrix singular value decay and also the approximation quality of the Loewner interpolants.

The data set ($n = \text{even}$) is composed of

$$\begin{cases} \text{Sample points : } \mathbf{S} = [\omega_1, \omega_2, \dots, \omega_n] \in \mathbb{R}^n, & \text{with } \omega_1 < \omega_2 < \dots < \omega_n, \\ \text{Sample values : } \mathbf{H} = [H(\omega_1), H(\omega_2), \dots, H(\omega_n)] \in \mathbb{C}^n. \end{cases} \quad (6.28)$$

We analyze four different types of data splitting that are mentioned in the following.

1. **First type:** disjoint splitting.
 - $\boldsymbol{\mu} = [\omega_1, \dots, \omega_{n/2}]$ and $\mathbb{V} = [H(\omega_1), \dots, H(\omega_{n/2})]$,
 - $\boldsymbol{\lambda} = [\omega_{n/2+1}, \dots, \omega_n]$ and $\mathbb{W} = [H(\omega_{n/2+1}), \dots, H(\omega_n)]$.
2. **Second type:** alternate splitting.
 - $\boldsymbol{\mu} = [\omega_1, \omega_3, \dots, \omega_{n-1}]$ and $\mathbb{V} = [H(\omega_1), H(\omega_3), \dots, H(\omega_{n-1})]$,
 - $\boldsymbol{\lambda} = [\omega_2, \omega_4, \dots, \omega_n]$ and $\mathbb{W} = [H(\omega_2), H(\omega_4), \dots, H(\omega_n)]$.
3. **Third type:** magnitude splitting (in this case the set \mathbf{S} is first sorted with respect to the magnitude of the set \mathbf{H}).
 - $\boldsymbol{\mu} = [\omega_1, \dots, \omega_{n/2}]$ and $\mathbb{V} = [H(\omega_1), \dots, H(\omega_{n/2})]$,
 - $\boldsymbol{\lambda} = [\omega_{n/2+1}, \dots, \omega_n]$ and $\mathbb{W} = [H(\omega_{n/2+1}), \dots, H(\omega_n)]$.
4. **Fourth type:** magnitude alternate splitting (in this case, the set \mathbf{S} is first sorted with respect to the magnitude of the set \mathbf{H} and then alternating splitting is applied).
 - $\boldsymbol{\mu} = [\omega_1, \omega_3, \dots, \omega_{n-1}]$ and $\mathbb{V} = [H(\omega_1), H(\omega_3), \dots, H(\omega_{n-1})]$,
 - $\boldsymbol{\lambda} = [\omega_2, \omega_4, \dots, \omega_n]$ and $\mathbb{W} = [H(\omega_2), H(\omega_4), \dots, H(\omega_n)]$.

As observed in practice, when splitting the data as for the first type, the Loewner matrix has a very fast decay of the singular values. Moreover, in this case, the computed reduced models usually provide low approximation quality.

⁴ Runge's phenomenon is a problem of oscillation at the edges of an interval that occurs when using polynomial interpolation with polynomials of high degree over a set of equispaced interpolation points.

On the other hand, for the second separation type (alternate splitting), the left and right sets of sample points can be chosen ϵ -close to one another (element-wise). Hence, as $\epsilon \rightarrow 0$, Hermitian interpolation conditions are enforced (which involve matching the first derivative at those points).

Other observations that hold in the case of second type splitting are that the numerical rank of the Loewner matrix is usually larger than that of the Loewner matrix constructed based on the first type. Additionally, for the second type, the condition number is smaller than that computed for the first type. For the above-mentioned cases, bounds on the singular value decay of the Loewner matrix are provided in [15].

6.4 Case studies

In this section we illustrate the concepts developed in the preceding sections by means of examples. In particular the following seven examples will be analyzed.

1. The benchmark CD player ($n = 120$).
2. The function $f(x) = \exp(-x) \sin(10x)$, $x \in [-1, 1]$.
3. The inverse of the Bessel function of the first kind, in $[0, 10] \times [-1, 1]j$.
4. An Euler–Bernouli beam.
5. A heat equation with transfer function $\mathbf{H}(s) = \exp(-\sqrt{s})$, $s \in [0.01, 100]j$.
6. Approximation of $f = y/\sinh(y)$, $y(x) = 100\pi(x^2 - 0.36)$, $x \in [-1, 1]$.
7. The sign function in the interval $[-b, -a]$ and $[a, b]$, $a > b > 0$.

6.4.1 The CD player

Consider the CD player benchmark example which is a MIMO dynamical system of dimension 120 with 2 inputs and 2 outputs. Here we will consider the $(2, 1)$ sub-system, i. e. the SISO system from the first input to the second output.

The goal is to approximate the transfer function in the Loewner framework. We start by considering 400 interpolation points $\pm j\omega_i$, $i = 1, \dots, 200$, where ω_i are logarithmically spaced in the interval $\Omega = [10^{-1}, 10^5]$. Thus $\Omega = \{\omega_1, \omega_2, \dots, \omega_{200}\}$, where $\omega_i < \omega_{i+1}$, for all i . We now define the left/right interpolation points in four different ways as explained in section 6.3.3.2 and depicted in Figure 6.5 (up).

As can be seen in Figure 6.5 (down), the decay of the Loewner matrix singular values is faster for “half-half” (disjoint) splitting than for “alternating” splitting.

The next step is to choose the truncation order and to determine the level of approximation. We propose two different ways for this purpose.

1. By choosing equal truncation orders r .
2. By choosing for each separation the maximum truncation order so that $\frac{\sigma_r}{\sigma_1}$, is equal to a fixed tolerance value.

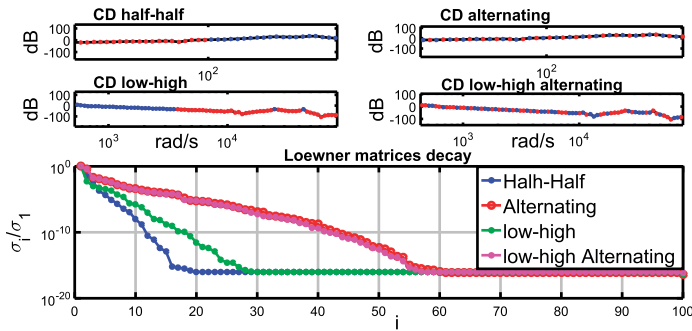


Figure 6.5: The four different splitting schemes (up) and the decay of the singular values ($\frac{\sigma_i}{\sigma_1}, i = 1, \dots, 100$) of the Loewner matrix for each type (down).

First experiment: equal truncation orders

Here, we fix the truncation order to $r = 10$, and compute $\frac{\sigma_r}{\sigma_1}$. The results are presented in Table 6.2.

Table 6.2: Normalized singular values corresponding to $r = 10$ for each splitting.

Case	1st	2nd	3rd	4th
r	10	10	10	10
$\frac{\sigma_r}{\sigma_1}$	$1e-8$	$1e-6$	$1e-4$	$1e-4$

The frequency response of the original system with those of the four reduced systems (corresponding to each different splitting) is shown in Figure 6.6. Note that all methods produce similar approximation quality.

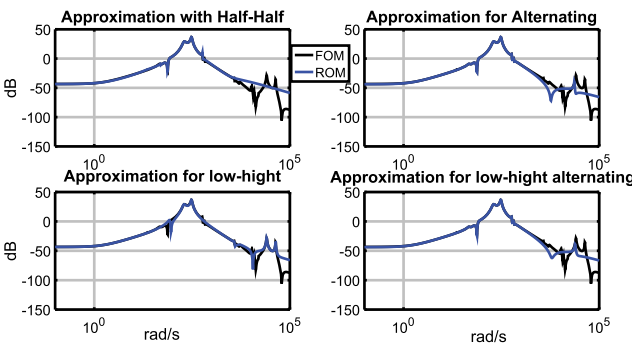


Figure 6.6: Frequency response comparison: original system vs. the reduced ones with equal truncation orders ($r = 10$).

Next, the approximation error for each reduced systems is depicted in Figure 6.7. For the first partition type, the error curve displays a ‘V’ shape form near the middle of the sampling interval. This is where the left and right sampling points are very close to each other.

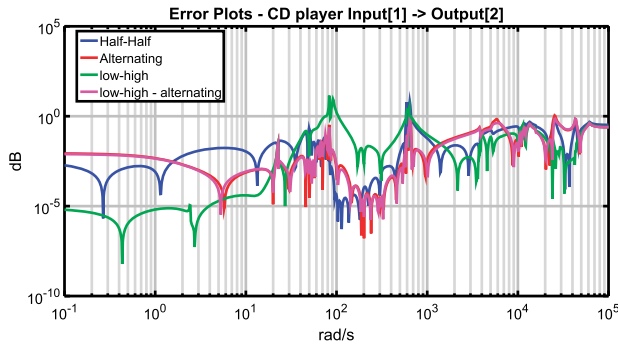


Figure 6.7: Approximation error with the four splitting schemes.

Second experiment: reaching machine precision⁵

The tolerance of normalized singular value $\frac{\sigma_r}{\sigma_1}$ is now fixed (e. g. 10^{-14}). This implies the truncation order r . The results are presented in Table 6.3

Table 6.3: Different truncation orders for all splitting schemes and for a fixed tolerance.

Case	1st	2nd	3rd	4th
r	16	51	23	48
$\frac{\sigma_r}{\sigma_1}$	$1e-14$	$1e-14$	$1e-14$	$1e-14$

The truncation order for the first splitting type is more than three times smaller than that for the second splitting type (16 vs 51).

The frequency response of the original systems with the four reduced systems in depicted in Figure 6.8. All methods produce good approximation quality, with a slight deviation in the high frequency range observed for the first splitting type.

Finally, Figure 6.9 shows the approximation error for each reduced system.

Notice that the blue curve in Figure 6.9 has a ‘V’ shape in the middle of the sampling interval. The lowest approximation error is recorded for the second splitting type (alternate selection).

⁵ Machine precision is the smallest number ϵ such that the difference between 1 and $1 + \epsilon$ is nonzero. This is approximately 10^{-16} .

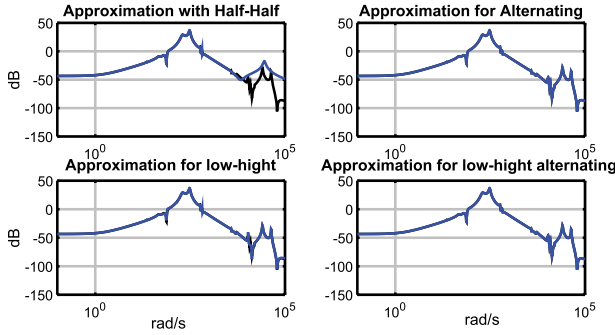


Figure 6.8: Frequency response comparison: original system vs. the reduced ones by reaching machine precision.

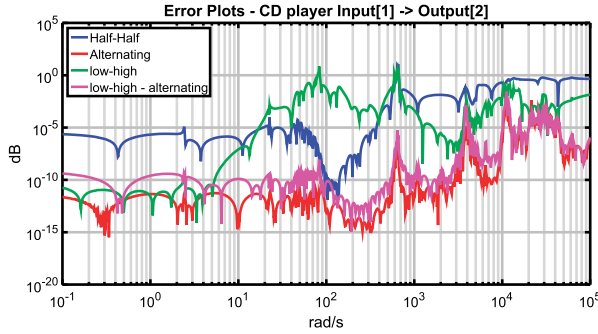


Figure 6.9: Approximation error for the four splitting schemes.

6.4.2 Approximation of an oscillating function

We collect $N = 4,000$ measurements $\{(s_k, \phi_k) : k = 1, \dots, N\}$ of the following function:

$$\phi(x) = e^{-x} \sin(10x), \quad x \in [-1, 1]. \quad (6.29)$$

Assume that the interpolation points $\mathbf{s} = [s_1, s_2, \dots, s_{4000}] \subset [-1, 1]$ are *equispaced*; next we remain with two types of splitting.

1. **First type:** disjoint splitting.
 - Left: $\mu = [s_1, s_2, \dots, s_{2000}] \subset [-1, 0]$
 - Right: $\lambda = [s_{2001}, s_{2002}, \dots, s_{4000}] \subset [0, 1]$

We construct the Loewner pencil and the underlying rank is 11.

2. **Second type:** alternate splitting.
 - Left: $\mu = [s_1, s_3, \dots, s_{3999}] \subset [-1, 1]$
 - Right: $\lambda = [s_2, s_4, \dots, s_{4000}] \subset [-1, 1]$.

We construct the Loewner pencil and the underlying rank is 15.

Figure 6.10 shows the entries of the Loewner matrix in logarithmic scale for the two ways of sampling point separation. Next, the interpolation data is compressed, making use of the following methods: (a) the singular value decomposition SVD, (b) the randomized version rSVD, (c) CUR, implemented with DEIM and (d) CUR implemented with cross approximation. The parameters for the latter two methods are: $\epsilon = 0.001$ and $\delta = 0.01$.

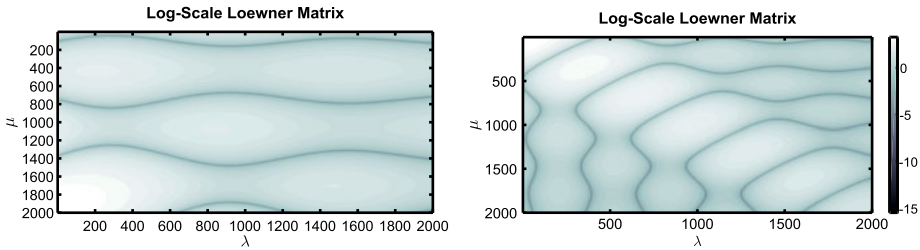


Figure 6.10: Entries of the Loewner matrix for the first splitting (left) and the second splitting (right).

Table 6.4: Results for the first splitting type (disjoint) with an i5-CPU 2.60 GHz.

Reduction – r for $\approx \mathbb{L}$	$\text{rank}(\mathbb{L}_{r \times r})$	$\text{cond}(\mathbb{L}_{r \times r}) = \frac{\sigma_{\max}}{\sigma_{\min}}$	Error $\ \cdot\ _F$	Time (s)
SVD	11	$9.7313e + 10$	$6.7367e - 10$	4.166029
CUR-CrossApprox	11	$7.6582e + 10$	$1.5621e - 09$	0.528352
CUR-DEIM	11	$1.3898e + 11$	$2.2283e - 09$	4.101303
randomized SVD	11	$9.7314e + 10$	$1.1281e - 10$	0.030148

In Figure 6.11 the error curves for the first splitting are shown. The red Xs indicate the selected points with CUR-cross-approximation method while the green crosses (+) indicate the selected points with CUR-DEIM method. In Figure 6.12 the error curves for the second splitting are shown. As opposed to the previously shown results (in Figure 6.11), the error in this case (Figure 6.12) is distributed more uniformly. Additional qualitative measures (e.g., the condition number) under different splitting schemes with the same reduced-order are presented in Tables 6.4 and 6.5.

Table 6.5: Results for the second splitting type (alternate) with an i5-CPU 2.60 GHz.

Reduction – r for $\approx \mathbb{L}$	$\text{rank}(\mathbb{L}_{r \times r})$	$\text{cond}(\mathbb{L}_{r \times r}) = \frac{\sigma_{\max}}{\sigma_{\min}}$	Error $\ \cdot\ _F$	Time (s)
SVD	11	$8.8199e + 4$	0.0020	4.261075
CUR-CrossApprox	11	$1.0228e + 5$	0.0062	0.563411
CUR-DEIM	11	$9.3343e + 4$	0.0245	4.152420
randomized SVD	11	$8.8199e + 4$	0.0020	0.024586

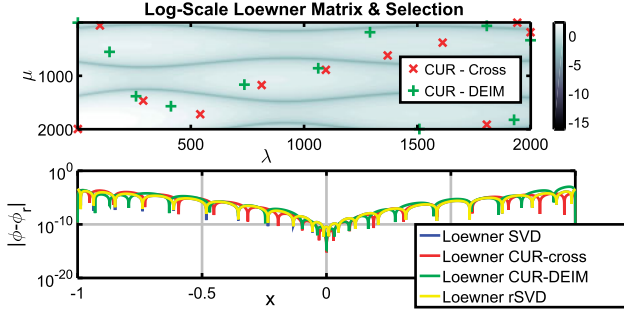


Figure 6.11: Selected points and approximation error for the disjoint splitting.

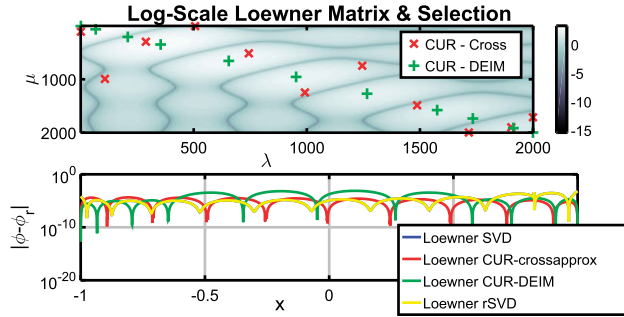


Figure 6.12: Selected points and approximation error for the alternate splitting.

As seen in the above experiments, the splitting of the data influences both the Loewner singular value decay and the quality of approximation. In most of the experiments that follow, we choose the *alternate* way of splitting the data.

6.4.3 Approximation of a Bessel function

In this section we investigate the approximation of the inverse of a Bessel function in a domain in the complex plane. If this function is considered to be the transfer function of a dynamical system, this system is infinite dimensional; furthermore it is not stable as there are poles in the right-half of the complex plane.

In particular we consider the inverse of the Bessel function of the first kind and order $n \in \mathbb{N}$. It is defined by the following contour integral:

$$J_n(s) = \frac{1}{2\pi i} \oint e^{(\frac{s}{2})(t-\frac{1}{t})} t^{-n-1} dt. \quad (6.30)$$

Here, we consider only the case $n = 0$. Our aim is to approximate $H(s) = \frac{1}{J_0(s)}$, $s \in \mathbb{C}$, inside the rectangle $\Omega = [0, 10] \times [-1, 1] \subset \mathbb{C}$. In Figure 6.13 (left pane) the function $H(s)$

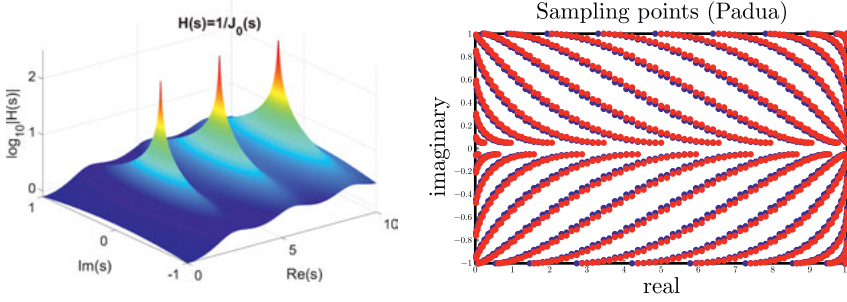


Figure 6.13: Left pane: The inverse of the Bessel function of the 1st kind. Right pane: A subset of 10,000 Padua point grid over $\Omega = [0, 10] \times [-1, 1]$ domain are shown.

is shown in the domain Ω . The three spikes correspond to the unstable poles of the underlying system. These are three of the zeros of the Bessel function. Here we construct approximants $H_r(s)$, of order r , of $H(s)$, using the interpolation points as shown in Figure 6.13 on the right pane. The distribution of the two-dimensional initial grids is 5,000 Padua points with the conjugates. This grid is used to reduce the Runge phenomenon. For more details in approximation theory (i. e. Runge phenomenon, Padua points, barycentric interpolation, etc.), we refer the reader to [64]. In [43, 44], the same experiment with other types of grids (random uniformly, structured) is presented.

In the Loewner framework, the singular value decomposition (SVD) plays a key role. This factorization allows us to extract the numerical order of the rational model which approximates the original non-rational one.

In Figure 6.14 (left pane), we show the distribution of the normalized singular values $\frac{\sigma_j}{\sigma_1}$, $j = 1, \dots, N$, of the augmented matrices $[\mathbb{L} \quad \mathbb{L}_s]$ and $[\mathbb{L} \quad \mathbb{L}_s]$.

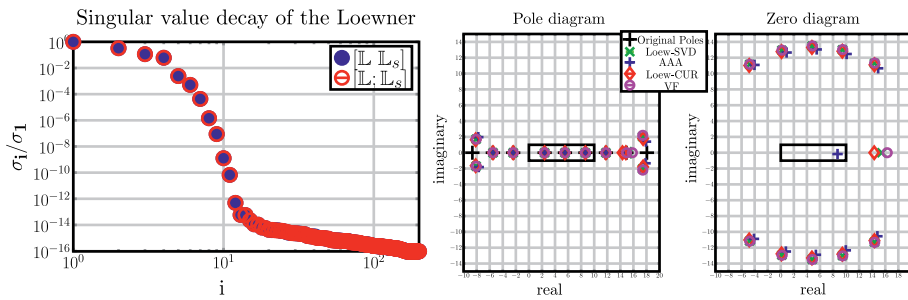


Figure 6.14: Left pane: Singular value decay of 10,000 values. Right pane: Pole/zero diagram with the three original poles (zeros of Bessel) which recovered with 15 digits accuracy.

By taking measurements as in Figure 6.13 (right pane) the decay of the singular values Figure 6.14 – left pane, leads to a reduced order $r = 12$ with $\frac{\sigma_{12}}{\sigma_1} = 4.887 \cdot 10^{-13}$. In Figure 6.14 on the right pane the pole/zero diagram is presented which includes the

results from all methods. Methods VF, and Loewner(SVD or CUR) construct real strictly rational models with degree $(11, 12)^6$ with $D = 0$, as opposed to AAA algorithm which constructs complex proper rational model of degree $(12, 12)$ with a non-zero D term.

By using the methods LoewCUR-cross and AAA, points from the sampling grid are selected. Applying the LoewSVD method the point selection is obtained by compressing the initial grid. This can be achieved by using the first r columns (r : singular vectors) of the singular matrices as projection matrices and by solving two $(r \times r$ -dimension) generalized eigenvalue problems as explained in Section 6.2.5.3. Under this way, we compress the original grid with $N = 10,000$ points into a much smaller set of only $2r = 24$ points which are exact interpolation points for the approximant. As it turns out, the projected points lie in the domain Ω ; see also left pane in Figure 6.15.

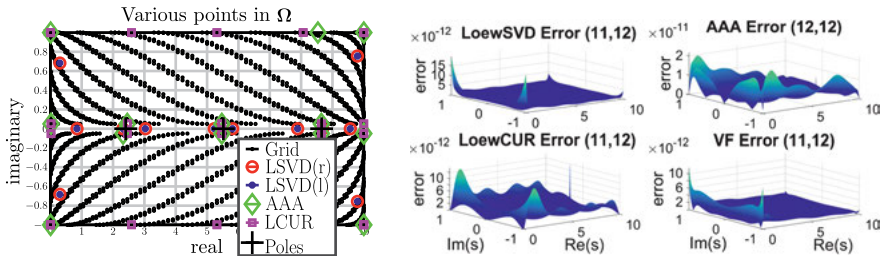


Figure 6.15: Left pane: Support and compressed points for every method over Ω domain with LSVD(r) \rightarrow LoewSVD projected right points, LSVD(l) \rightarrow LoewSVD projected left points. Right pane: The error for every method.

The LoewCUR-cross and AAA methods select points among the initial interpolation points but with different criteria. The AAA algorithm selects support points by minimizing the mean squared error with the rest of the measurements while LoewCUR uses cross approximation, which maximizes the absolute value of the determinant (maximum volume) of the sub-matrix of dimension $(r \times r)$.

In Figure 6.15 on the right pane, the error for each method is shown. The normalized error is computed as $\frac{|H(s) - H_r(s)|}{|H(s)|}$ with 25,000 evaluation points in Ω . It should be mentioned that the above special choice of the original interpolation grid as Padua points, indeed reduced the Runge phenomenon.

Next we wish to visualize the approximation error outside Ω . Towards this goal we chose 25,000 equispaced evaluation points inside the domain $[-3, 13] \times [-3, 3]$. Results with log-contour level error of increasing order $10^{-16}, \dots, 10^{-4}$ are presented in Figure 6.16.

⁶ The notation (m, n) indicates that the order of the numerator polynomial is m and the order of denominator polynomial is n .

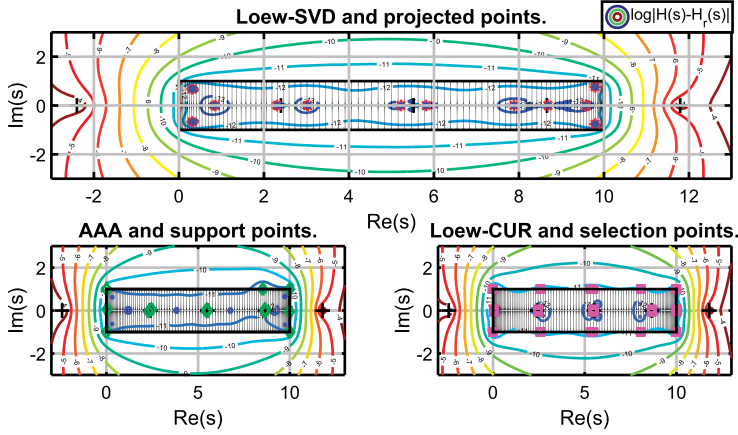


Figure 6.16: Extrapolation error as $\log |H(s) - H_r(s)|$ in $[-3, 13] \times [-3, 3] \subset \mathbb{C}$. The symbol ‘+’ is for the original poles.

All methods constructed accurate rational approximants. Notice, however, that the Loewner approach reaches similar precision with AAA without performing any optimization step. Finally, in terms of computational complexity, the CUR method performed the best.

6.4.4 An Euler–Bernoulli beam

In this subsection we analyse the approximation of an Euler–Bernoulli clamped beam [18]. The underlying PDE describes the oscillation of the free end. As shown in [18], the non-rational transfer function is given by

$$\begin{aligned}
 H(s) &= \frac{sn(s)}{(EI + sc_d I)m^3(s)d(s)}, \quad \text{where} \\
 m(s) &= \left[\frac{-s^2}{EI + c_d Is} \right]^{\frac{1}{4}}, \quad d(s) = 1 + \cosh(Lm(s)) \cos(Lm(s)), \\
 n(s) &= \cosh(Lm(s)) \sin(Lm(s)) - \sinh(Lm(s)) \cos(Lm(s)).
 \end{aligned} \tag{6.31}$$

Usually, the next step consists of a discretization of the PDE involved. We bypass this step and instead take frequency response measurements making use of the transfer function above. The parameter specification is as in [18].⁷ Thus, we have the frequency response of the beam as in Figure 6.17 and on the left pane.

⁷ Young’s modulus (elasticity constant): $E = 69 \text{ GPa} = 6.9 \cdot 10^{10} \text{ N/m}^2$, moment of inertia: $I = 3.58 \cdot 10^{-9} \text{ m}^4$, damping constant: $c_d = 5 \cdot 10^{-4}$, length: $L = 0.7 \text{ m}$, base: $b = 0.07 \text{ m}$, height: $h = 0.0085 \text{ m}$.

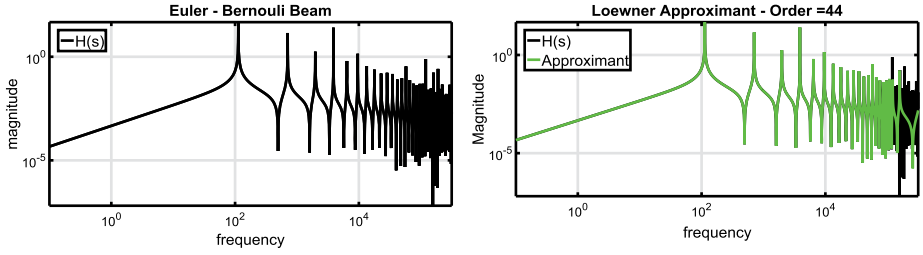


Figure 6.17: Left pane: Original frequency response of the beam. Right pane: The approximant which constructed with the Loewner framework.

The next step is to collect 2,000 measurements on the imaginary axis (frequencies $j\omega_i, i = 1, \dots, 2000$), spaced logarithmically from 1 rad/s to 10^5 rad/s. These points are depicted in the left pane of Figure 6.18.

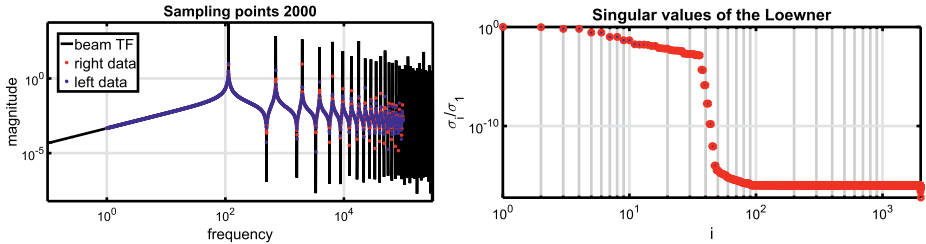


Figure 6.18: Left pane: 2,000 sampling points alternating as left and right. Right pane: The singular value decay.

The singular value of the Loewner matrices decay is as shown in Figure 6.18 on the right pane. Thus, we construct a reduced model with dimension $r = 44$ and the Loewner approximant in Figure 6.17 (right pane) is depicted.

Finally, the poles and zeros for every method are presented in Figure 6.19. The quality of the approximation is given for each method in Figure 6.20 where the evaluation is in the frequency range from 1 to $10^{5.5}$. The error outside the sampling domain increases thus indicating the difficulty of approximation outside of the sampling domain for infinite dimensional systems.

6.4.5 Heat equation

Next, we investigate an one-dimensional heat equation [13]. The corresponding PDE describing the diffusion of heat leads to the following non-rational transfer function:

$$H(s) = e^{-\sqrt{s}}, \quad s \in \mathbb{C}. \quad (6.32)$$

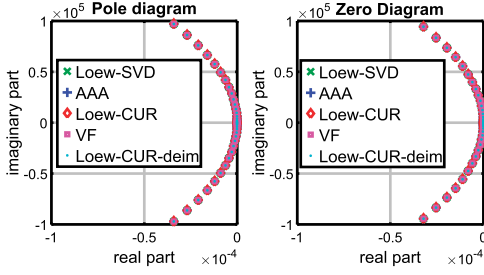


Figure 6.19: Pole/Zero diagram for every method (LoewSVD, LoewCUR-cross, LoewCUR-DEIM, VF and AAA).

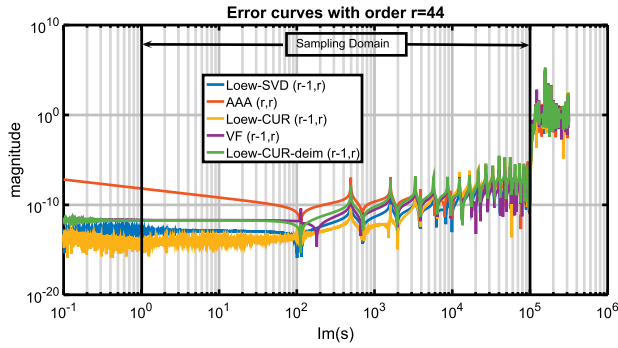


Figure 6.20: The error distribution with 8,000 evaluation points grid.

The aim is to construct reduced models by means of the Loewner framework and compare the results with the TF-IRKA used in [13]. Iterative Rational Krylov Algorithm - IRKA [14] builds optimal reduced models by minimizing the \mathcal{H}_2 error [39].

By collecting 1,000 values of the transfer function on the imaginary axis, the resulting reduced order was chosen to be $r = 6$ (as in [13]). For this truncation order, $\frac{\sigma_6}{\sigma_1} \approx 6 \cdot 10^{-3}$. In Figure 6.21c, the pole/zero distribution for every method is depicted; in Figure 6.21d, the selected points are shown. It is worth mentioning that the Loewner SVD method produced poles near to the optimal set computed by means of IRKA; see Figure 6.21c. Approximation results are in Figure 6.22.

6.4.6 Approximation of a two-peak function

In this section we present an example involving a hyperbolic sine from [22]. The difficulty here results from the two differentiable peaks. More precisely, the function is

$$f(x) = \frac{100\pi(x^2 - 0.36)}{\sinh(100\pi(x^2 - 0.36))}, \quad x \in [-1, 1],$$

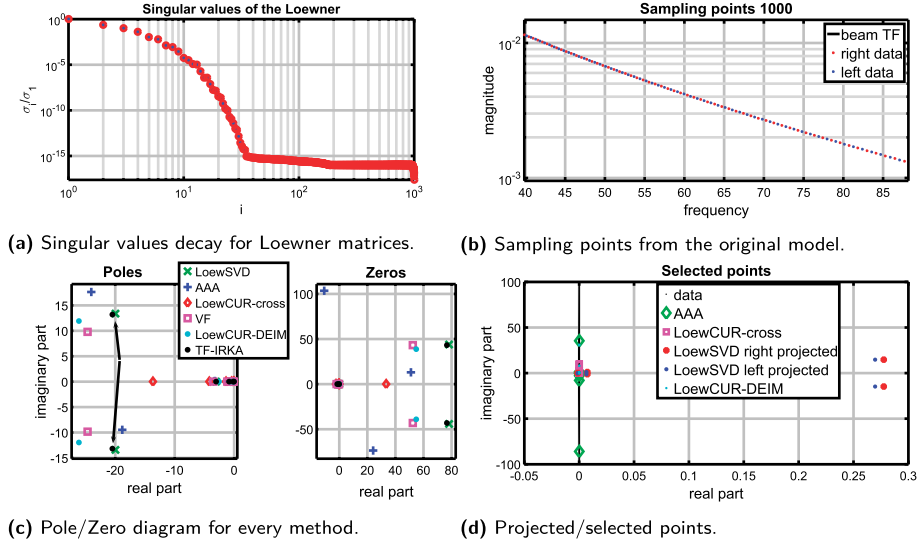


Figure 6.21: Approximation of the heat equation with LoewSVD, LoewCUR, VF, AAA, TF-IRKA.

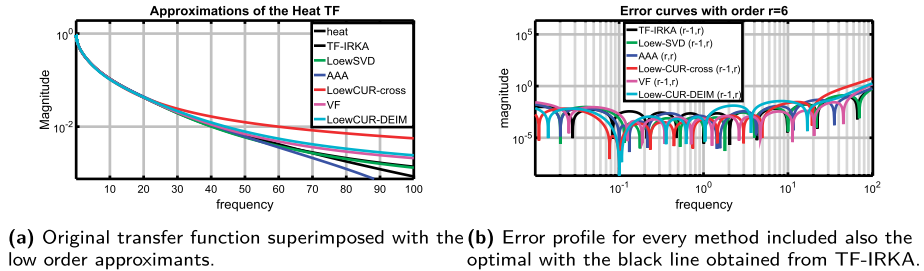


Figure 6.22: Approximation results for the heat equation with various interpolation methods.

and is shown Figure 6.23 (left pane). We approximate this function by choosing 1,000 equispaced points in $[-1, 1]$ as on the right pane in Figure 6.23. The singular values of the Loewner matrix are shown in Figure 6.24 on the left pane while the selected points are shown on the right pane of the same figure. The order is selected to be $r = 38$ with $(\sigma_{38}/\sigma_1 \approx 10^{-12})$. In Figure 6.25, the distribution of the poles and zeros for each method is shown. On the other hand, AAA looks quite different because it does not impose real symmetry.

Remark 6.8. In Figure 6.24, right pane, the different supports points are shown. In the case of the LoewSVD method two almost pure imaginary projected points are obtained even if the initial sampling points were real.

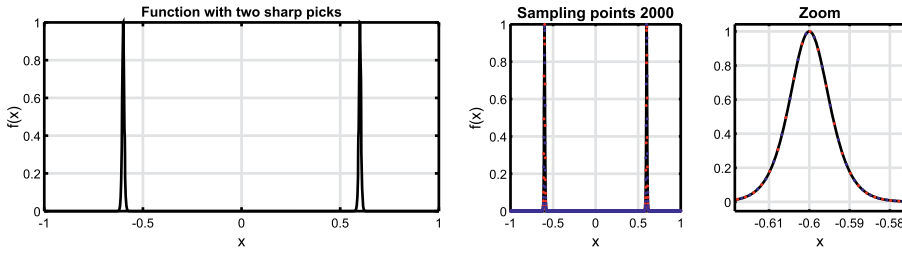


Figure 6.23: Left pane: The function f with two very sharp differentiable picks. Right pane: 1,000 sampling points and zoom in close to one pick.

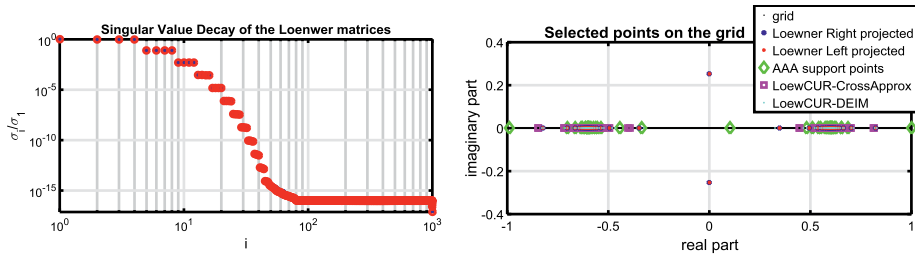


Figure 6.24: Left pane: Singular values decay. Right pane: Various points for every method and the projected points from the Loewner framework.

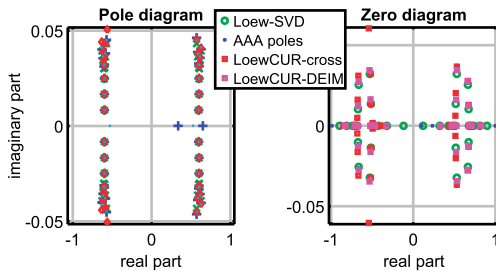


Figure 6.25: The pole/zero diagram.

Finally we observe a good fit for every method, with slightly better performance attained for the Loewner SVD method (see the error plot in Figure 6.26).

6.4.7 Approximation of the sign function

Our final case study problem concerns the approximation of the sign function, known as Zolotarev's fourth problem. Here, we compare the approximation obtained using the Loewner SVD with the optimal solution that is explicitly known [1]. Given two disjoint closed complex sets E and F , Zolotarev's fourth problem is to find the rational

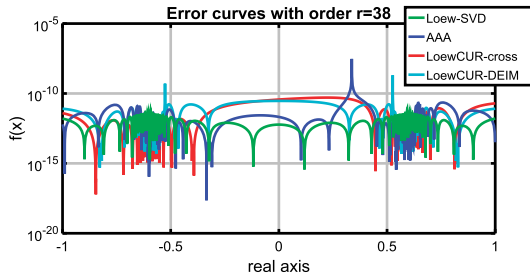


Figure 6.26: The error profile with 5,000 evaluation points over $[-1, 1]$.

function $r(x) = \frac{p(x)}{q(x)}$, where p, q are polynomials of degree k , that deviates least from the sign function

$$\text{sign}(x) = \begin{cases} -1, & x \in E, \\ +1, & x \in F, \end{cases}$$

on $E \cup F$. For general sets E and F , the solution to Zolotarev's fourth problem is not known; however, there are special cases where the rational function can be given explicitly. For the real disjoint intervals, $E = [-b, -1]$, and $F = [1, b]$ with $b > 1$, an explicit (optimal) solution to Zolotarev's fourth problem is known [1]. Here, we investigated how well the Loewner framework can approximate this discontinuous function in two symmetric real intervals. We choose $b = 3$ and $N = 2,000$ initial interpolation points from $[-3, -1] \cup [1, 3]$. We perform two experiments. Firstly, we choose initial interpolation points as equispaced and secondly, as Chebyshev nodes. For each choice, we split the data as "half-half" and "alternating" as discussed previously. The left pane in Figure 6.27 shows the plot of the sign function.

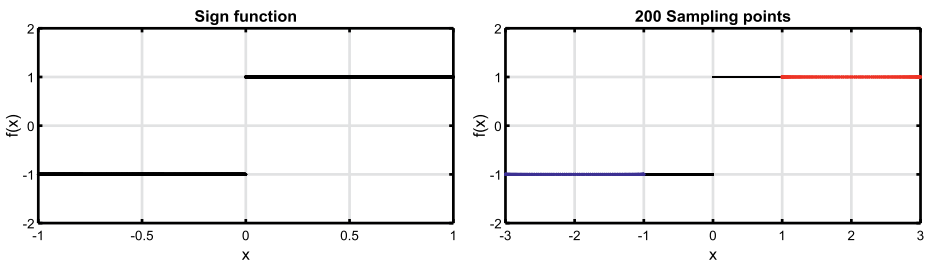


Figure 6.27: Left pane: The sign function. Right pane: 200 Chebyshev points in $[-3, -1] \cup [1, 3]$.

In [17] the explicit solution of this optimization problem is computed. We start by taking $N = 2,000$ measurements as Chebyshev nodes as in Figure 6.27 on the right pane. The above sampling way leads to the following singular value decay of the Loewner matrices as in Figure 6.28 on the left pane.

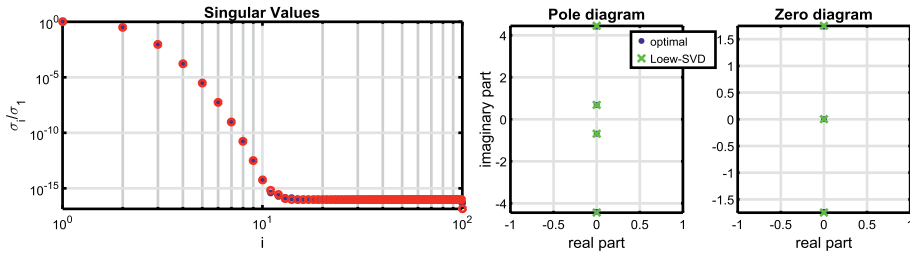


Figure 6.28: Left pane: The singular value decay of the Loewner pencil. Right pane: Pole/Zero diagram for the Loewner and the optimal approximant with order $r = 4$.

From the rank-revealing factorization in the left pane in Figure 6.28, we chose $r = 4$ with $\frac{\sigma_4}{\sigma_1} = 1.657 \cdot 10^{-4}$. In Figure 6.28 on the right pane is the distribution of the pole/zero diagram which is derived from the Loewner SVD method, in comparison with the optimal set is presented.

In Figure 6.29 (left) the Loewner approximant is shown. It is quite close to the optimal one by choosing the Chebyshev nodes and splitting the left and right points as “half–half”. Indeed, the error distribution as presented in the optimal interpolant in Figure 6.30 with the blue line has the equioscillation property of the optimal approximant in the *infinity norm* - $\|\mathbf{x}\|_\infty = \max(|x_1|, \dots, |x_n|)$. Thus the equioscillation of the error $|\text{sign}(x) - r(x)|$ on both intervals shows the optimality of the approximant. The Loewner framework succeeds in constructing an approximant very close to the optimal. Another aspect is shown in Figure 6.29 (right pane). More specifically, note that the projected points are indeed interpolation points.

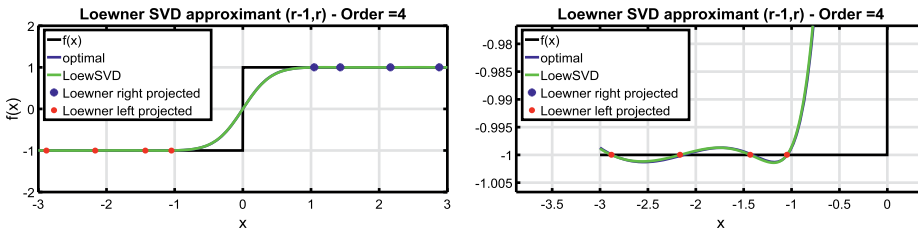


Figure 6.29: Left pane: A comparison between the Loewner approximant with the optimal one order $r = 4$. Right pane: The projected points are approximated interpolation points.

Remark 6.9. If the choice of the splitting is disjoint—“half–half” as in this experiment, the constructed approximant interpolates the data as in Figure 6.29(right pane). If the choice is “alternating” by mixing left and right, then the projected low order model approximates the values and the derivatives at the interpolation points as in Figure 6.31.

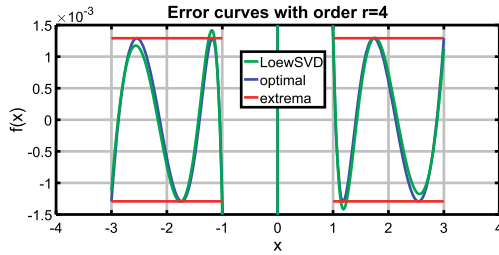


Figure 6.30: Error plot with the Loewner approximant and the optimal solution as well with order $r = 4$.

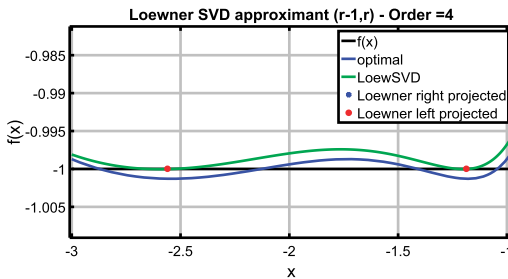


Figure 6.31: By splitting the data as “alternating”, the projected Loewner model approximates the first derivative as well (*Hermite interpolation conditions*).

6.5 Epilogue

Interpolatory methods for model identification and reduction were studied in this contribution. The main focus was on the Loewner framework. The aim was to introduce the Loewner framework by providing results which connect this rational interpolation tool with system theory. At the same time, algorithms that make the Loewner framework a complete numerical tool for approximation with ease of implementation are offered. Several case studies illustrate the effectiveness of the method. Implementation issues like the splitting of the data in left and right were addressed. Finally, connections with the SVD, the r-SVD, CUR, VF and IRKA have been detailed.

Bibliography

- [1] N. I. Akhiezer. *Elements of the theory of elliptic functions. Translations of Mathematical Monographs*, volume 79. American Mathematical Society, Providence, RI, 1990. Translated from the second Russian edition by H. H. McFaden.
- [2] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. Society for Industrial and Applied Mathematics, 2005.

- [3] A. Antoulas. The Loewner framework and transfer functions of singular/rectangular systems. *Appl. Math. Lett.*, 54:36–47, 2016.
- [4] A. Antoulas, A. Ionita, and S. Lefteriu. On two-variable rational interpolation. *Linear Algebra Appl.*, 436(8):2889–2915, 2012. Special Issue dedicated to Danny Sorensen's 65th birthday.
- [5] A. C. Antoulas, I. V. Gosea, and A. C. Ionita. Model reduction of bilinear systems in the Loewner framework. *SIAM J. Sci. Comput.*, 38(5):B889–B916, 2016.
- [6] A. C. Antoulas, S. Lefteriu, and A. C. Ionita. *A Tutorial Introduction to the Loewner Framework for Model Reduction*, pages 335–376. Society for Industrial and Applied Mathematics for Computational Science & Engineering, 2017, Ch. 8.
- [7] A. C. Antoulas, B. Zhu, Q. Zhang, B. York, B. W. O'Malley, and C. C. Dacso. A novel mathematical method for disclosing oscillations in gene transcription: A comparative study. *PLoS ONE*, 13:1–20, 2018.
- [8] A. C. Antoulas, I. V. Gosea, and M. Heinkenschloss. On the Loewner framework for model reduction of Burgers' equation. In R. King, editor, *Active Flow and Combustion Control 2018*, pages 255–270. Springer, Cham, 2019.
- [9] A. C. Antoulas, C. A. Beattie, and S. Gügercin. *Interpolatory Methods for Model Reduction*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2020.
- [10] A. Antoulas, I. Gosea, and M. Heinkenschloss. Data-driven model reduction of the Oseen equations using the Loewner framework. In S. Grundel, T. Reis and S. Schöps, editors, *Progress in Differential-Algebraic Equations II. Differential-Algebraic Equations Forum*. Springer, 2020, accepted for publication. <https://doi.org/10.1109/ECC.2015.7330568>.
- [11] A. Antoulas, I. Gosea, and M. Heinkenschloss. Reduction of systems with polynomial nonlinearities in the Loewner framework. In *Book of Abstracts of XXI Householder Symposium on Numerical Linear Algebra*, Selva di Fasano, Italy, June 14–19, 2020.
- [12] C. Baker, K. Gallivan, and P. V. Dooren. Low-rank incremental methods for computing dominant singular subspaces. *Linear Algebra Appl.*, 436(8):2866–2888, 2012. Special Issue dedicated to Danny Sorensen's 65th birthday.
- [13] C. Beattie and S. Gugercin. Realization-independent H₂-approximation. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 4953–4958, 2012.
- [14] C. Beattie and S. Gugercin. *Model Reduction by Rational Interpolation*, pages 297–334. Society for Industrial and Applied Mathematics for Computational Science & Engineering, 2017, Ch. 7.
- [15] B. Beckermann and A. Townsend. On the singular values of matrices with displacement structure. *SIAM J. Matrix Anal. Appl.*, 38(4):1227–1248, 2017.
- [16] P. Benner and P. Goyal. Interpolation-based model order reduction for polynomial parametric systems, Tech. rep., arXiv preprint available at <https://arxiv.org/abs/1904.11891>, 2019. <https://doi.org/10.1137/19M1259171>.
- [17] M. Berljafa and S. Güttel. A rational Krylov toolbox for Matlab, 2014.
- [18] R. Curtain and K. Morris. Transfer functions of distributed parameter systems: A tutorial. *Automatica*, 45(5):1101–1116, 2009.
- [19] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Relative-error CUR matrix decompositions. *SIAM J. Matrix Anal. Appl.*, 30(2):844–881, 2008.
- [20] Z. Drmač and B. Peherstorfer. Learning low-dimensional dynamical-system models from noisy frequency-response data with Loewner rational interpolation, Tech. rep., arXiv preprint available at <https://arxiv.org/abs/1910.00110>, 2019.
- [21] Z. Drmač, S. Gugercin, and C. Beattie. Vector fitting for matrix-valued rational approximation. *SIAM J. Sci. Comput.*, 37(5):A2346–A2379, 2015.
- [22] S.-I. Filip, Y. Nakatsukasa, L. N. Trefethen, and B. Beckermann. Rational minimax approximation via adaptive barycentric representations. *SIAM J. Sci. Comput.*, 40(4):A2427–A2455, 2018.

- [23] K. Gallivan, A. Vandendorpe, and P. V. Dooren. On the generality of multipoint Padé approximations. *IFAC Proc. Vol.*, 35(1):331–336, 2002. 15th IFAC World Congress.
- [24] K. Gallivan, A. Vandendorpe, and P. V. Dooren. Sylvester equations and projection-based model reduction. *J. Comput. Appl. Math.*, 162(1):213–229, 2004. Proceedings of the International Conference on Linear Algebra and Arithmetic 2001.
- [25] K. Gallivan, A. Vandendorpe, and P. Van Dooren. Model reduction of MIMO systems via tangential interpolation. *SIAM J. Matrix Anal. Appl.*, 26(2):328–349, 2004.
- [26] S. Goreinov and E. Tyrtyshnikov. Quasioptimality of Skeleton approximation of a matrix in the Chebyshev norm. In *Doklady Mathematics*, volume 83, pages 374–375. Springer, 2011.
- [27] S. Goreinov, E. Tyrtyshnikov, and N. Zamarashkin. A theory of pseudoskeleton approximations. *Linear Algebra Appl.*, 261(1):1–21, 1997.
- [28] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. Zamarashkin. *How to find a good submatrix*, 2010.
- [29] I. V. Gosea and A. C. Antoulas. Model reduction of linear and nonlinear systems in the Loewner framework: A summary. In *2015 European Control Conference (ECC)*, pages 345–349, 2015.
- [30] I. V. Gosea and A. C. Antoulas. Stability preserving post-processing methods applied in the Loewner framework. In *2016 IEEE 20th Workshop on Signal and Power Integrity (SPI)*, pages 1–4, 2016.
- [31] I. V. Gosea and A. C. Antoulas. Approximation of a damped Euler-Bernoulli beam model in the Loewner framework, Tech. rep., arXiv preprint available at <https://arxiv.org/abs/1712.06031>, 2017.
- [32] I. V. Gosea and A. C. Antoulas. Data-driven model order reduction of quadratic-bilinear systems. *Numer. Linear Algebra Appl.*, 25(6):e2200, 2018.
- [33] I. V. Gosea and A. C. Antoulas. Rational approximation of the absolute value function from measurements: a numerical study of recent methods, Tech. rep., arXiv preprint available at <https://arxiv.org/abs/2005.02736>, 2020.
- [34] I. V. Gosea, M. Petreczky, and A. C. Antoulas. Data-driven model order reduction of linear switched systems in the Loewner framework. *SIAM J. Sci. Comput.*, 40(2):B572–B610, 2018.
- [35] I. V. Gosea, I. P. Duff, P. Benner, and A. C. Antoulas. Model order reduction of bilinear time-delay systems. In *2019 18th European Control Conference (ECC)*, pages 2289–2294, 2019.
- [36] I. Gosea, D. Karachalios, and A. C. Antoulas. Learning reduced-order models of quadratic control systems from input-output data. In *European Control Conference (ECC21), June 29–July 2, 2021, Rotterdam, The Netherlands (Virtual Conference)*, 2021, accepted for publication, arXiv preprint available at <https://arxiv.org/abs/2012.02075>, 2020.
- [37] I. V. Gosea, Q. Zhang, and A. C. Antoulas. Preserving the DAE structure in the Loewner model reduction and identification framework. *Adv. Comput. Math.*, 46(3), 2020.
- [38] E. Grimme, K. A. Gallivan, and P. V. Dooren. On some recent developments in projection-based model reduction. In *ENUMATH 97*, Heidelberg, pages 98–113. World Sci. Publishing, River Edge, NJ, 1998.
- [39] S. Gugercin, A. C. Antoulas, and C. Beattie. H2 model reduction for large-scale linear dynamical systems. *SIAM J. Matrix Anal. Appl.*, 30(2):609–638, 2008.
- [40] B. Gustavsen and A. Semlyen. Rational approximation of frequency domain responses by vector fitting. *IEEE Trans. Power Deliv.*, 14:1052–1061, 1999.
- [41] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, 2011.
- [42] A. C. Ionita and A. C. Antoulas. Data-driven parametrized model reduction in the Loewner framework. *SIAM J. Sci. Comput.*, 36(3):A984–A1007, 2014.
- [43] D. S. Karachalios, I. V. Gosea, Q. Zhang, and A. C. Antoulas. Case study: Approximations of the Bessel function, Tech. rep., arXiv preprint available at <https://arxiv.org/abs/1801.03390>, 2017.

- [44] D. S. Karachalios, I. V. Gosea, and A. C. Antoulas. Data-driven approximation methods applied to non-rational functions. *PAMM*, 18(1), e201800368, 2018.
- [45] D. S. Karachalios, I. V. Gosea, and A. C. Antoulas. A bilinear identification-modeling framework from time domain data. *PAMM*, 19(1):e201900246, 2019.
- [46] D. Karachalios, I. Gosea, and A. C. Antoulas. On bilinear time domain identification and reduction in the Loewner framework. In P. Benner et al., editors, *Model Reduction of Complex Dynamical Systems. International Series of Numerical Mathematics*. Birkhäuser/Springer Nature, Chur, Switzerland, 2021. https://doi.org/10.1007/978-3-030-72983-7_1
- [47] B. Kramer and A. A. Gorodetsky. System identification via CUR-factored hankel approximation. *SIAM J. Sci. Comput.*, 40(2):A848–A866, 2018.
- [48] S. Lefteriu and A. C. Antoulas. A new approach to modeling multiport systems from frequency-domain data. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 29(1):14–27, 2010.
- [49] S. Lefteriu and A. C. Antoulas. A new approach to modeling multiport systems from frequency-domain data. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 29(1):14–27, 2010.
- [50] S. Lefteriu, A. C. Ionita, and A. C. Antoulas. In *Modeling Systems Based on Noisy Frequency and Time Domain Measurements*, pages 365–378. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [51] M. W. Mahoney and P. Drineas. CUR matrix decompositions for improved data analysis. *Proc. Natl. Acad. Sci.*, 106(3):697–702, 2009.
- [52] A. Mayo and A. Antoulas. A framework for the solution of the generalized realization problem. *Linear Algebra Appl.*, 425(2):634–662, 2007. Special Issue in honor of Paul Fuhrmann.
- [53] Y. Nakatsukasa, O. Sète, and L. N. Trefethen. The AAA algorithm for rational approximation. *SIAM J. Sci. Comput.*, 40(3):A1494–A1522, 2018.
- [54] V. Olshevsky, editor. In *Structured matrices in mathematics, computer science, and engineering. I. Contemporary Mathematics*, volume 280. American Mathematical Society, Providence, RI, 2001.
- [55] I. Oseledets and E. Tyrtysnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra Appl.*, 432(1):70–88, 2010.
- [56] B. Peherstorfer, S. Gugercin, and K. Willcox. Data-driven reduced model construction with time-domain Loewner models. *SIAM J. Sci. Comput.*, 39, 2017.
- [57] P. Rapisarda and A. C. Antoulas. A duality perspective on Loewner rational interpolation and state-space modelling of vector-exponential trajectories. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 2096–2100, 2015.
- [58] P. Rapisarda and A. C. Antoulas. Bilinear differential forms and the Loewner framework for rational interpolation. In M. N. Belur, M. K. Camlibel, P. Rapisarda and J. M. Scherpen, editors, *Mathematical Control Theory II: Behavioral Systems and Robust Control*, volume 462, pages 23–43. Springer, 2015.
- [59] P. Schulze and B. Unger. Data-driven interpolation of dynamical systems with delay. *Syst. Control Lett.*, 97:125–131, 2016.
- [60] J. D. Simard and A. Astolfi. An interconnection-based interpretation of the Loewner matrices*. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 7788–7793, 2019.
- [61] J. D. Simard and A. Astolfi. Loewner functions for linear time-varying systems with applications to model reduction. In *21st IFAC World Congress*, Berlin, Germany, 2020, accepted for publication. <https://doi.org/10.1016/j.ifacol.2020.12.1578>
- [62] D. Sorensen and A. Antoulas. The Sylvester equation and approximate balanced reduction. *Linear Algebra Appl.*, 351–352:671–700, 2002. Fourth Special Issue on Linear Systems and Control.

- [63] D. C. Sorensen and M. Embree. A deim induced CUR factorization. *SIAM J. Sci. Comput.*, 38(3):A1454–A1482, 2016.
- [64] L. N. Trefethen. *Approximation Theory and Approximation Practice (Other Titles in Applied Mathematics)*. Society for Industrial and Applied Mathematics, USA, 2012.
- [65] C. D. Villemagne and R. E. Skelton. Model reductions using a projection formulation. In *26th IEEE Conference on Decision and Control*, volume 26, pages 461–466, 1987.
- [66] A. Yousuff and R. Skelton. Covariance equivalent realizations with application to model reduction of large-scale systems. In C. Leondes, editor, *Decentralized/Distributed Control and Dynamic Systems, Part 1 of 3. Control and Dynamic Systems*, volume 22, pages 273–348. Academic Press, 1985.
- [67] A. Yousuff, D. Wagie, and R. Skelton. Linear system approximation via covariance equivalent realizations. *J. Math. Anal. Appl.*, 106(1):91–115, 1985.
- [68] B. Zhu, Q. Zhang, Y. Pan, E. M. Mace, B. York, C. A. Antoulas, C. C. Dacso, and B. W. O'Malley. A cell-autonomous mammalian 12 hr clock coordinates metabolic and stress rhythms. *Cell Metab.*, 6(25):1305–1319, 2017.

7 Manifold interpolation

Abstract: One approach to parametric and adaptive model reduction is via the interpolation of orthogonal bases, subspaces or positive definite system matrices. In all these cases, the sampled inputs stem from matrix sets that feature a geometric structure and thus form so-called matrix manifolds. This chapter reviews the numerical treatment of the most important matrix manifolds that arise in the context of model reduction. Moreover, the principal approaches to data interpolation and Taylor-like extrapolation on matrix manifolds are outlined and complemented by algorithms in pseudo-code.

Keywords: parametric model reduction, matrix manifold, interpolation, Riemannian computing, Riemannian normal coordinates

MSC 2010: 15-01, 15A16, 15B10, 15B48, 53-04, 65F60, 41-01, 41A05, 65F99, 93A15, 93C30

7.1 Introduction & motivation

This chapter addresses interpolation approaches for parametric model reduction. This includes techniques for

- computing trajectories of parameterized subspaces,
- computing trajectories of parameterized reduced orthogonal bases,
- structure-preserving interpolation.

Mathematically, this requires data processing on nonlinear matrix manifolds. The exposition at hand intends to be an introduction and a reference guide to numerical procedures with matrix manifold-valued data. As such it addresses practitioners and scientists new to the field. It covers the essentials of those matrix manifolds that arise most frequently in practical problems in model reduction. The main purpose is not to discuss concrete model reduction applications, but rather to provide the essential tools, building blocks and background theory to enable the reader to devise her/his own approaches for such applications.

The text was designed such that it works as a commented formula collection, meanwhile giving sufficient context, explanations and, not least, precise references to enable the interested reader to immerse further in the topic.

Acknowledgement: We are grateful to ECOST for pushing us into this.

Ralf Zimmermann, SDU Odense, Odense, Denmark

Open Access. © 2021 Ralf Zimmermann, published by De Gruyter.  This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

7.1.1 Parametric model reduction via manifold interpolation: an introductory example

The basic objective in model reduction is to emulate a large-scale dynamical system with very few degrees of freedom such that its input/output behavior is preserved as well as possible. While classical model reduction techniques aim at producing an accurate low-order approximation to the autonomous behavior of the original system, parametric model reduction (pMOR) tries to account for additional system parameters. If we look for instance at aircraft aerodynamics, an important task is to solve the unsteady Navier–Stokes equations at various flight conditions, which are, amongst others, specified by the altitude, the viscosity of the fluid (i. e. the Reynolds number) and the relative velocity (i. e. the Mach number). We explain the objective of pMOR with the aid of a generic example in the context of proper orthogonal decomposition-based model reduction. Similar considerations apply to frequency domain approaches, Krylov subspace methods and balanced truncation, which are, e. g., discussed Chapters 2 and 3 of this volume and in [19, Chapter 1], [20, Chapter 3].

Consider a spatio-temporal dynamical system in semi-discrete form

$$\frac{\partial}{\partial t}x(t, \mu) = f(x(t, \mu); \mu), \quad x(t_0, \mu) = x_{0, \mu}, \quad (7.1)$$

where $x(t, \mu) \in \mathbb{R}^n$ is the spatially discretized *state vector* of dimension n , the vector $\mu = (\mu_1, \dots, \mu_d) \in \mathbb{R}^d$ accounts for additional system parameters and $f(\cdot; \mu) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is the (possibly nonlinear, parameter-dependent) right hand side function. Projection-based MOR starts with constructing a suitable low-dimensional subspace that acts as a space of candidate solutions.

Subspace construction. One way to construct the required projection subspace is the proper orthogonal decomposition (POD) see [19, Chapter 2]. In its simplest form, the POD can be summarized as follows. For a fixed system parameter $\mu = \mu_0$, let $x^1 := x(t_1, \mu_0), \dots, x^m := x(t_m, \mu_0) \in \mathbb{R}^n$ be a set of state vectors satisfying (7.1) and let $\mathbb{S} := (x^1, \dots, x^m) \in \mathbb{R}^{n \times m}$. The state vectors x^i are called *snapshots* and the matrix \mathbb{S} is called the associated *snapshot matrix*. POD is concerned with finding a subspace \mathcal{V} of dimension $r \leq m$ represented by a column-orthogonal matrix $\mathbb{V}_r \in \mathbb{R}^{n \times r}$ such that the error between the input snapshots and their orthogonal projection onto $\mathcal{V} = \text{ran}(\mathbb{V}_r)$ is minimized:

$$\min_{V \in \mathbb{R}^{n \times r}, V^T V = I} \sum_k \|x^k - VV^T x^k\|_2^2 \quad (\Leftrightarrow \min_{V \in \mathbb{R}^{n \times r}, V^T V = I} \|\mathbb{S} - VV^T \mathbb{S}\|_F^2).$$

The main result of POD is that, for any $r \leq m$, the best r -dimensional approximation of $\text{ran}(x^1, \dots, x^m)$ in the above sense is $\mathcal{V} = \text{ran}(v^1, \dots, v^r)$, where $\{v^1, \dots, v^r\}$ are the eigenvectors of the matrix $\mathbb{S}\mathbb{S}^T$ corresponding to the r largest eigenvalues. The subspace \mathcal{V} is called the *POD subspace* and the matrix $\mathbb{V}_r = (v^1, \dots, v^r)$ is the *POD basis matrix*. The same subspace is obtained via a compact singular value decomposition (SVD) of

the snapshot matrix $\mathbb{S} = \mathbb{V}\Sigma\mathbb{Z}^T$, truncated to the first $r \leq m$ columns of $\mathbb{V} \in \mathbb{R}^{n \times m}$ by setting $\mathcal{V} := \text{ran}(\mathbb{V}_r)$. For more details, see, e. g., [18, §3.3]. In the following, we drop the index r and assume that \mathbb{V} is already the truncated matrix $\mathbb{V} = (v^1, \dots, v^r) \in \mathbb{R}^{n \times r}$.

Since the input snapshots are supplied at a fixed system parameter vector μ_0 , the POD subspace is considered to be an appropriate space of solution candidates $\mathcal{V}(\mu_0) = \text{ran}(\mathbb{V}(\mu_0))$ at μ_0 .

Projection. POD leads to a parameter decoupling

$$\tilde{x}(t, \mu_0) = \mathbb{V}(\mu_0)x_r(t). \quad (7.2)$$

In this way, the time trajectory of the reduced model is uniquely defined by the coefficient vector $x_r(t) \in \mathbb{R}^r$ that represents the reduced state vector with respect to the subspace $\text{ran}(\mathbb{V}(\mu_0))$. Given a matrix $\mathbb{W}(\mu_0)$ such that the matrix pair $\mathbb{V}(\mu_0), \mathbb{W}(\mu_0)$ is bi-orthogonal, i. e. $\mathbb{W}(\mu_0)^T \mathbb{V}(\mu_0) = I$, the original system (7.1) can be reduced in dimension as follows. Substituting (7.2) in (7.1) and multiplying with $\mathbb{W}(\mu_0)^T$ from the left leads to

$$\frac{d}{dt}x_r(t) = \mathbb{W}^T(\mu_0)f(\mathbb{V}(\mu_0)x_r(t); \mu_0), \quad x_r(t_0) = \mathbb{V}^T(\mu_0)x_{0, \mu_0}. \quad (7.3)$$

This approach goes by the name of Petrov–Galerkin projection, if $\mathbb{W}(\mu_0) \neq \mathbb{V}(\mu_0)$ and Galerkin projection if $\mathbb{W}(\mu_0) = \mathbb{V}(\mu_0)$. There are various ways to proceed from (7.3) depending on the nature of the function f and many of them are discussed in other chapters of Model Order Reduction. If $f(\cdot; \mu_0)$ is linear, the reduced operator $\mathbb{W}^T(\mu_0) \circ f(\cdot; \mu_0) \circ \mathbb{V}(\mu_0)$ can be computed a priori (*‘offline’*) and stays fixed throughout the time integration. If $f(\cdot; \mu_0)$ is affine, the same approach can be carried over to the affine building blocks of $f(\cdot; \mu_0)$; see e. g. [45]. For a nonlinear $f(\cdot; \mu_0)$, an affine approximation can be constructed via the empirical interpolation method (EIM, [14]). Other approaches that address nonlinearities include the discrete empirical interpolation method (DEIM, [30]) and the missing point estimation (MPE, [13, 105]).

For illustration purposes, we proceed with $\mathbb{W}(\mu_0) = \mathbb{V}(\mu_0)$ and assume that the right hand side function f splits into a linear and a nonlinear part: $f(x; \mu_0) = A(\mu_0)x + \mathbf{f}(x; \mu_0)$, where $A(\mu_0) \in \mathbb{R}^{n \times n}$ is, say, a symmetric and negative definite matrix to foster stability. Then (7.3) becomes

$$\frac{d}{dt}x_r(t) = \mathbb{V}^T(\mu_0)A(\mu_0)\mathbb{V}(\mu_0)x_r(t) + \mathbb{V}^T(\mu_0)\mathbf{f}(\mathbb{V}(\mu_0)x_r(t); \mu_0).$$

In the discrete empirical interpolation method (DEIM, [30]), the large-scale nonlinear term $\mathbf{f}(\mathbb{V}(\mu_0)x_r(t); \mu_0)$ is approximated via a mask matrix $P = (e_{i_1}, \dots, e_{i_s}) \in \mathbb{R}^{n \times s}$, where $\{i_1, \dots, i_s\} \subset \{1, \dots, n\}$ and $e_j = (\dots, 1, \dots)^T \in \mathbb{R}^n$ is the j th canonical unit vector. The mask matrix P acts as an entry selector on a given n -vector via $P^T v = (v_{i_1}, \dots, v_{i_s})^T \in \mathbb{R}^s$. In addition, another POD basis matrix $\mathbb{U}(\mu_0) \in \mathbb{R}^{n \times s}$ is used, which is obtained from snapshots of the nonlinear term. The matrices P and $\mathbb{U}(\mu_0)$ are combined to form an

oblique projection of the nonlinear term onto the subspace $\text{ran}(\mathbb{U}(\mu_0))$. This leads to the reduced model

$$\begin{aligned} \frac{d}{dt}x_r(t) &= \mathbb{V}^T(\mu_0)A(\mu_0)\mathbb{V}(\mu_0)x_r(t) \\ &\quad + \mathbb{V}^T(\mu_0)\mathbb{U}(\mu_0)(P^T\mathbb{U}(\mu_0))^{-1}P^T\mathbf{f}(\mathbb{V}(\mu_0)x_r(t); \mu_0), \end{aligned} \quad (74)$$

whose computational complexity is formally independent of the full-order dimension n ; see [30] for details. Mind that by assumption, $M(\mu_0) := -\mathbb{V}^T(\mu_0)A(\mu_0)\mathbb{V}(\mu_0)$ is symmetric positive definite and that both $\mathbb{V}(\mu_0)$ and $\mathbb{U}(\mu_0)$ are column-orthogonal. Moreover, for a fixed mask matrix P , coordinate changes of $\mathbb{V}(\mu_0)$ and $\mathbb{U}(\mu_0)$ do not affect the approximated state $\tilde{x}(t, \mu_0) = \mathbb{V}(\mu_0)x_r(t)$, so that essentially, the reduced system (74) depends only on the subspaces $\text{ran}(\mathbb{V}(\mu_0))$ and $\text{ran}(\mathbb{U}(\mu_0))$ rather than the matrices $\mathbb{V}(\mu_0)$ and $\mathbb{U}(\mu_0)$.¹

Solving (73), (74) constitutes the *online stage* of model reduction. The main focus of this chapter is *not* on the efficient solution of the reduced systems (73) or (74) at a fixed μ_0 , but on tackling parametric variations in μ . In view of the associated computational costs, it is important that this can be achieved *without* computing additional snapshots in the online stage.

A straightforward way to achieve this is to extend the snapshot sampling to the μ -parameter range to produce POD basis matrices that are to cover all input parameters. This is usually referred to as the “global approach”. For nonlinear systems, the global approach may suffer from requiring a large number of snapshot samples. Moreover, the snapshot information is blurred in the global POD and features that occur only in a restricted regime affect the ROM predictions everywhere. Therefore, localized approaches are preferable; see e. g. the applications in and the numerical examples in [38, 100].

In this chapter, the focus is on constructing trajectories of functions in the system parameters μ on certain sets of structured matrix spaces. In the above example, these are the symmetric positive definite matrices $\{M \in \mathbb{R}^{r \times r} \mid M^T = M, v^T M v > 0 \forall v \neq 0\}$, the orthonormal basis matrices $\{U \in \mathbb{R}^{n \times s} \mid U^T U = I\}$ or the associated s -dimensional subspaces $\mathcal{U} := \text{ran}(U) \subset \mathbb{R}^n$:

$$\begin{aligned} \mu &\mapsto -\mathbb{V}^T(\mu)A(\mu)\mathbb{V}(\mu) \in \{M \in \mathbb{R}^{r \times r} \mid M^T = M, v^T M v > 0 \forall v \neq 0\}, \\ \mu &\mapsto \mathbb{U}(\mu) \in \{U \in \mathbb{R}^{n \times s} \mid U^T U = I\}, \\ \mu &\mapsto \mathcal{U}(\mu) = \text{ran}(\mathbb{U}(\mu)) \in \{\mathcal{U} \subset \mathbb{R}^n \mid \mathcal{U} \text{ subspace, } \dim(\mathcal{U}) = s\}. \end{aligned}$$

We outline generic methods for constructing such trajectories via interpolation. All the special sets of matrices considered above feature a differentiable structure that

¹ Replacing \mathbb{U} with $\mathbb{U}S$, $S \in \mathbb{R}^{s \times s}$ orthogonal, does not affect (74) at all. Replacing \mathbb{V} with $\mathbb{V}R$, $R \in \mathbb{R}^{r \times r}$ orthogonal, induces a coordinate change on the reduced state $x_r = R\hat{x}_r$ but preserves the output $\tilde{x}(t) = \mathbb{V}x_r(t) = \mathbb{V}R\hat{x}_r(t)$.

allows one to consider them (directly or indirectly) as submanifolds of some Euclidean matrix space, referred to as matrix manifolds. The above example is not exhaustive. Other matrix manifolds may arise in model reduction applications.

To keep the exposition both general and modular, the interpolation techniques will be formulated for arbitrary submanifolds. For working examples that put these techniques into action, the reader is referred to Chapter 5 of Volume 2 and Chapter 9 of Volume 3 of Model Order Reduction. Model reduction literature on manifold interpolation problems includes [8, 9, 18, 32, 34, 67, 74, 76, 78, 94, 100].

7.1.2 Structure and organization

The text is constructed modular rather than consecutive, so that selected reading is enabled. Yet, this entails that the reader will encounter some repetition.

Section 7.2 covers the essential background from differential geometry. Section 7.3 contains generic methods for interpolation and extrapolation on matrix manifolds. In Section 7.4, the geometric and numerical aspects of the matrix manifolds that arise most frequently in the context of model reduction are discussed.

A practitioner that faces a problem in matrix manifold interpolation may skim through the recap on elementary differential geometry in Section 7.2 and then move on to the subsection of Section 7.4 that corresponds to the matrix manifold in the application. This provides the specific ingredients and formulas for conducting the generic interpolation methods of Section 7.3.

7.1.3 Notation & abbreviations

- w. r. t.: with respect to
- EVD: eigenvalue decomposition
- SVD: singular value decomposition
- POD: proper orthogonal decomposition
- LTI: linear time-invariant (system)
- ODE: ordinary differential equation
- PDE: partial differential equation
- ONB: orthonormal basis
- $\mathbb{R}^{n \times r}$: the set of real n -by- r matrices
- I_n : the n -by- n identity matrix; if dimensions are clear, written as I
- $\text{ran}(A)$: the subspace spanned by the columns of $A \in \mathbb{R}^{n \times r}$
- $GL(n)$: the general linear group of real, invertible n -by- n matrices
- $\text{sym}(n) = \{A \in \mathbb{R}^{n \times n} \mid A^T = A\}$: the set of real, symmetric n -by- n matrices
- $\text{skew}(n) = \{A \in \mathbb{R}^{n \times n} \mid A^T = -A\}$: the set of real, skew-symmetric n -by- n matrices

- $\text{SPD}(n) = \{A \in \text{sym}(n) \mid x^T A x > 0 \forall x \in \mathbb{R}^n \setminus \{0\}\}$: the set of real, symmetric positive definite n -by- n matrices
- $O(n) = \{Q \in \mathbb{R}^{n \times n} \mid Q^T Q = I_n = Q Q^T\}$: the orthogonal group
- $SO(n) = \{Q \in O(n) \mid \det(Q) = 1\}$: the special orthogonal group
- $St(n, r) = \{U \in \mathbb{R}^{n \times r} \mid U^T U = I_r\}$: the (compact) Stiefel manifold, $r \leq n$
- $Gr(n, r)$: the Grassmann manifold of r -dimensional subspaces of \mathbb{R}^n , $r \leq n$
- \mathcal{M} : a differentiable manifold
- $\mathcal{D}_p \subset \mathcal{M}$: an open domain around the point p on a manifold \mathcal{M}
- $D_x \subset \mathbb{R}^n$: an open domain in the Euclidean space around a point $x \in \mathbb{R}^n$
- $T_p \mathcal{M}$: the tangent space of \mathcal{M} at a location $p \in \mathcal{M}$
- $\langle A, B \rangle_0 = \text{trace}(A^T B)$: the standard (Frobenius) inner product on $\mathbb{R}^{n \times r}$
- $\langle v, w \rangle_p^{\mathcal{M}}$: the Riemannian metric on $T_p \mathcal{M}$ (the superscript is often omitted)
- \exp_m : standard matrix exponential
- \log_m : standard (principal) matrix logarithm
- $\text{Exp}_p^{\mathcal{M}}$: the Riemannian exponential of a manifold \mathcal{M} at base point $p \in \mathcal{M}$
- $\text{Log}_p^{\mathcal{M}}$: the Riemannian logarithm of a manifold \mathcal{M} at base point $p \in \mathcal{M}$.

7.2 Basic concepts of differential geometry

This section provides the essentials on elementary differential geometry. Established textbook references on differential geometry include [35, 60, 61, 63, 65]; condensed introductions can be found in [49, Appendices C.3, C.4, C.5] and [39]. An account of differential geometry that is tailor-made to matrix manifold applications is given in [3].

The fundamental objects of study in differential geometry are *differentiable manifolds*. Differentiable manifolds are generalizations of curves (one-dimensional) and surfaces (two-dimensional) to arbitrary dimensions. Loosely speaking, an n -dimensional differentiable manifold \mathcal{M} is a topological space that ‘locally looks like \mathbb{R}^n ’ with certain smoothness properties. This concept is rendered precise by postulating that, for every point $p \in \mathcal{M}$, there exists a so-called *coordinate chart* $x : \mathcal{M} \supset \mathcal{D}_p \rightarrow \mathbb{R}^n$ that bijectively maps an open neighborhood $\mathcal{D}_p \subset \mathcal{M}$ of a location p to an open neighborhood $D_{x(p)} \subset \mathbb{R}^n$ around $x(p) \in \mathbb{R}^n$ with the important additional property that the *coordinate change*

$$x \circ \tilde{x}^{-1} : \tilde{x}(\mathcal{D}_p \cap \tilde{\mathcal{D}}_p) \rightarrow x(\mathcal{D}_p \cap \tilde{\mathcal{D}}_p)$$

of two such charts x, \tilde{x} is a diffeomorphism, where their domains of definition overlap; see [39, Figure 18.2, p. 496] or [49, Figure 3.1, p. 342]. Note that the coordinate change $x \circ \tilde{x}^{-1}$ maps from an open domain of \mathbb{R}^n to an open domain of \mathbb{R}^n , so that the standard concepts of multivariate calculus apply. For details, see [3, §3.1.1] or [39, §18.8].

Depending on the context, we will write $x(p)$ for the value of a coordinate chart at p and also $x \in \mathbb{R}^n$ for a point in \mathbb{R}^n .

Of special importance to numerical applications are *embedded submanifolds in the Euclidean space*.

Definition 7.1 (Submanifolds of \mathbb{R}^{n+d}). A *parameterization* is a bijective differentiable function $f : \mathbb{R}^n \supset D \rightarrow f(D) \subset \mathbb{R}^{n+d}$ with continuous inverse such that its Jacobi matrix $Df_x \in \mathbb{R}^{(n+d) \times n}$ has full rank n at every point $x \in D$.

A subset $\mathcal{M} \subset \mathbb{R}^{n+d}$ is called an *n -dimensional embedded submanifold of \mathbb{R}^{n+d}* , if for every $p \in \mathcal{M}$, there exists an open neighborhood $\Omega \subset \mathbb{R}^{n+d}$ such that $\mathcal{D}_p := \mathcal{M} \cap \Omega$ is the image of a parameterization

$$f : \mathbb{R}^n \supset D_x \rightarrow f(D_x) = \mathcal{D}_p = \mathcal{M} \cap \Omega \subset \mathbb{R}^{n+d}.$$

One can show that, if $f : D \rightarrow \mathcal{M} \cap \Omega$ and $\tilde{f} : \tilde{D} \rightarrow \mathcal{M} \cap \tilde{\Omega}$ are two parameterizations, say with $f(x_0) = \tilde{f}(\tilde{x}_0) = p \in \mathcal{M} \cap \Omega \cap \tilde{\Omega}$, then

$$(f^{-1} \circ \tilde{f}) : \tilde{f}^{-1}(\Omega \cap \tilde{\Omega}) \rightarrow f^{-1}(\Omega \cap \tilde{\Omega})$$

is a diffeomorphism (between open sets in \mathbb{R}^n). In this sense, parameterizations f are the inverses of coordinate charts x . In addition to coordinate charts and parameterizations, submanifolds can be characterized via equality constraints. This fact is due to the inverse function theorem of classical multivariate calculus [64, §I.5]. For details, see [39, Thm. 18.7, p. 497].

Theorem 7.1 ([39, Prop. 18.7, p. 500]). Let $h : \mathbb{R}^{n+d} \supset \Omega \rightarrow \mathbb{R}^d$ be differentiable and $c_0 \in \mathbb{R}^d$ be defined such that the differential $Dh_p \in \mathbb{R}^{d \times (n+d)}$ has maximum possible rank d at every point $p \in \Omega$ with $h(p) = c_0$. Then the preimage

$$h^{-1}(c_0) = \{p \in \Omega \mid h(p) = c_0\}$$

is an n -dimensional submanifold of \mathbb{R}^{n+d} .

An obvious application of Theorem 7.1 to the function $h : \mathbb{R}^3 \rightarrow \mathbb{R}, (x_1, x_2, x_3) \mapsto x_1^2 + x_2^2 + x_3^2 - 1$ establishes the unit sphere $S^2 = h^{-1}(0)$ as a 2-dimensional submanifold of \mathbb{R}^{2+1} . As a more sophisticated example, we recognize the orthogonal group as a differentiable (sub)-manifold.

Example 7.1. Consider the orthogonal group $O(n) \subset \mathbb{R}^{n \times n} \simeq \mathbb{R}^{n^2}$ and the set of symmetric matrices $\text{sym}(n) \simeq \mathbb{R}^{n(n+1)/2}$. Define $h : \mathbb{R}^{n \times n} \rightarrow \text{sym}(n), A \mapsto A^T A - I$. Then $Dh_A(B) = A^T B + B^T A$. For $Q \in O(n)$, the differential is indeed surjective: For any $M \in \text{sym}(n)$, we have $Dh_Q(\frac{1}{2}QM) = \frac{1}{2}Q^T QM + \frac{1}{2}M^T Q^T Q = M$. As a consequence, the orthogonal group $O(n)$ is a submanifold of dimension $n^2 - \frac{1}{2}(n(n+1)) = \frac{1}{2}(n(n-1))$ of the Euclidean matrix space $\mathbb{R}^{n \times n}$.

7.2.1 Intrinsic and extrinsic coordinates

As a rule, numerical data processing on manifolds requires calculations in explicit coordinates. For differentiable submanifolds, we distinguish between two types: *extrinsic* and *intrinsic coordinates*. Extrinsic coordinates address points on a submanifold $\mathcal{M} \subseteq \mathbb{R}^n$ with respect to their coordinates in the ambient space \mathbb{R}^n , while intrinsic coordinates are with respect to the local parameterizations. Hence, extrinsic coordinates are what an outside observer would see, while intrinsic coordinates correspond to the perspective of an observer that resides on the manifold. Let us exemplify these concepts on the two-dimensional unit sphere S^2 , embedded in \mathbb{R}^3 . As a point set, the sphere is defined by the equation

$$S^2 = \{(x_1, x_2, x_3)^T \in \mathbb{R}^3 \mid x_1^2 + x_2^2 + x_3^2 = 1\}.$$

Any three-vector $(x_1, x_2, x_3)^T \in S^2$ specifies a point on the sphere in *extrinsic* coordinates. However, it is intuitively clear that S^2 is intrinsically a two-dimensional object. Indeed, S^2 can be parameterized via

$$f : \mathbb{R}^2 \supset [0, 2\pi)^2 \rightarrow S^2 \subset \mathbb{R}^3, \quad (\alpha, \beta) \mapsto \begin{pmatrix} \sin(\alpha) \cos(\beta) \\ \sin(\alpha) \sin(\beta) \\ \cos(\alpha) \end{pmatrix}.$$

The parameter vector $(\alpha, \beta) \in \mathbb{R}^2$ specifies a point on S^2 in *intrinsic* coordinates. Even though intrinsic coordinates directly reflect the dimension of the manifold at hand, they often cannot be calculated explicitly and extrinsic coordinates are the preferred choice in numerical applications [36, §2, p. 305]. Turning back to Example 7.1, we recall that the intrinsic dimension of the orthogonal group is $\frac{1}{2}n(n-1)$. Yet, in practice, one uses the extrinsic representation with $(n \times n)$ -matrices Q , keeping the defining equation $Q^T Q = I$ in mind.

7.2.2 Tangent spaces

We need a few more fundamental concepts.

Definition 7.2 (Tangent space of a differentiable submanifold). Let $\mathcal{M} \subset \mathbb{R}^{n+d}$ be an n -dimensional submanifold of \mathbb{R}^{n+d} . The *tangent space* of \mathcal{M} at a point $p \in \mathcal{M}$, in symbols $T_p \mathcal{M}$, is the space of velocity vectors of differentiable curves $c : t \mapsto c(t)$ passing through p , i. e.,

$$T_p \mathcal{M} = \{\dot{c}(t_0) \mid c : J \rightarrow \mathcal{M}, c(t_0) = p\}.$$

Here, $J \subseteq \mathbb{R}$ is an arbitrarily small open interval with $t_0 \in J$.

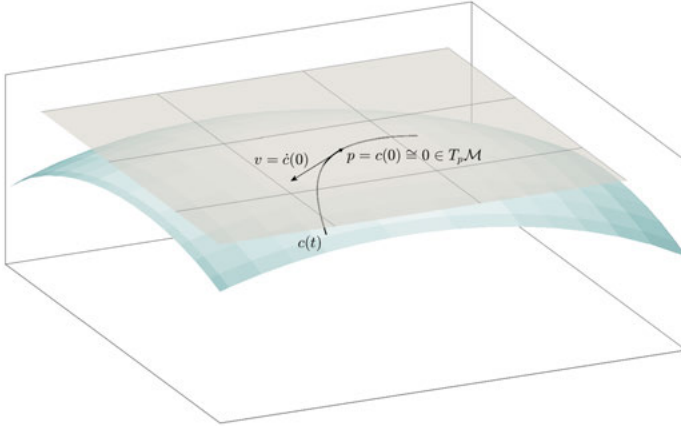


Figure 7.1: Visualization of a manifold (curved surface) with the tangent space $T_p \mathcal{M}$ attached. The tangent vector $v = \dot{c}(0) \in T_p \mathcal{M}$ is the velocity vector of a curve $c : t \mapsto c(t) \in \mathcal{M}$.

The concept is illustrated in Figure 7.1. It is straightforward to show that the tangent space is actually a vector space. Moreover, the tangent space can be characterized both with respect to intrinsic and extrinsic coordinates.

Theorem 7.2 (Tangent space, intrinsic characterization). *Let $\mathcal{M} \subset \mathbb{R}^{n+d}$ be an n -dimensional submanifold of \mathbb{R}^{n+d} and let $f : \mathbb{R}^n \supseteq D \rightarrow f(D) \subseteq \mathcal{M}$ be a parameterization. Then, for $x \in D$ with $p = f(x) \in \mathcal{M}$, we have*

$$T_p \mathcal{M} = \text{ran}(Df_x).$$

Theorem 7.3 (Tangent space, extrinsic characterization). *Let $h : \mathbb{R}^{n+d} \supset \Omega \rightarrow \mathbb{R}^d$ and $c_0 \in \mathbb{R}^d$ be as in Theorem 7.1 and let $\mathcal{M} := h^{-1}(c_0) \subset \mathbb{R}^{n+d}$. Then, for $p \in \mathcal{M}$, we have*

$$T_p \mathcal{M} = \ker(Dh_p).$$

Note that both Theorem 7.2 and Theorem 7.3 immediately show that the tangent space $T_p \mathcal{M}$ is a vector space of the same dimension n as the manifold \mathcal{M} .

Example 7.2. The tangent space of the orthogonal group $O(n)$ at a point Q_0 is

$$T_{Q_0} O(n) = \{\Delta \in \mathbb{R}^{n \times n} \mid \Delta^T Q_0 = -Q_0^T \Delta\}.$$

This fact can be established via considering a matrix curve $Q : t \mapsto Q(t)$ with $Q(0) = Q_0$ and velocity vector $\Delta = \dot{Q}(0) \in T_{Q_0} O(n)$. Then

$$0 = \frac{d}{dt} \Big|_{t=0} I = \frac{d}{dt} \Big|_{t=0} Q^T(t) Q(t) = \Delta^T Q_0 + Q_0^T \Delta.$$

(The claim follows by counting the dimension of the subspace $\{\Delta^T Q_0 = -Q_0^T \Delta\}$.) As an alternative, we can consider $h : \mathbb{R}^{n \times n} \rightarrow \text{sym}(n), A \mapsto A^T A - I$ as in Example 7.1. Then $Dh_{Q_0}(\Delta) = Q_0^T \Delta + \Delta^T Q_0$ and $T_{Q_0} O(n) = \ker(Dh_{Q_0})$.

7.2.3 Geodesics and the Riemannian distance function

One of the most important problems in both general differential geometry and data processing on manifolds is to determine the shortest connection between two points on a given manifold. This requires one to measure the lengths of curves. Recall that the length of a curve $c : [a, b] \rightarrow \mathbb{R}^n$ in the Euclidean space is $L(c) = \int_a^b \|\dot{c}(t)\| dt$. In order to transfer this to the manifold setting, an inner product for tangent vectors is needed that is consistent with the manifold structure.

Definition 7.3 (Riemannian metrics). Let \mathcal{M} be a differentiable submanifold of \mathbb{R}^{n+d} . A *Riemannian metric* on \mathcal{M} is a family $(\langle \cdot, \cdot \rangle_p)_{p \in \mathcal{M}}$ of inner products $\langle \cdot, \cdot \rangle_p : T_p \mathcal{M} \times T_p \mathcal{M} \rightarrow \mathbb{R}$ that is smooth in variations of the base point p .

The *length* of a tangent vector $v \in T_p \mathcal{M}$ is $\|v\|_p := \sqrt{\langle v, v \rangle_p}$.² The length of a curve $c : [a, b] \rightarrow \mathcal{M}$ is defined as

$$L(c) = \int_a^b \|\dot{c}(t)\|_{c(t)} dt = \int_a^b \sqrt{\langle \dot{c}(t), \dot{c}(t) \rangle_{c(t)}} dt.$$

A curve is said to be *parameterized by the arc length*, if $L(c|_{[a,t]}) = t - a$ for all $t \in [a, b]$. Obviously, *unit-speed curves* with $\|\dot{c}(t)\|_{c(t)} \equiv 1$ are parameterized by the arc length. Constant-speed curves with $\|\dot{c}(t)\|_{c(t)} \equiv v_0$ are parameterized proportional to the arc length. The *Riemannian distance* between two points $p, q \in \mathcal{M}$ with respect to a given metric is

$$\text{dist}_{\mathcal{M}}(p, q) = \inf\{L(c) \mid c : [a, b] \rightarrow \mathcal{M} \text{ piecewise smooth, } c(a) = p, c(b) = q\}, \quad (7.5)$$

where, by convention, $\inf\{\emptyset\} = \infty$.

Hence, a shortest path between $p, q \in \mathcal{M}$ is a curve c that connects p and q such that $L(c) = \text{dist}_{\mathcal{M}}(p, q)$. In general, shortest paths on \mathcal{M} do not exist.³ Yet, candidates for shortest curves between points that are sufficiently close to each other can be obtained via a variational principle: Given a parametric family of suitably regular curves $c_s : t \mapsto c_s(t) \in \mathcal{M}$, $s \in (-\varepsilon, \varepsilon)$ that connect the same fixed endpoints $c_s(a) = p$ and $c_s(b) = q$ for all s , one can consider the length functional $s \mapsto L(c_s)$. A curve $c = c_0$ is a first-order candidate for a shortest path between p and q , if it is a critical point of

² This notation should not be confused with the classical p -norm $\sqrt[p]{\sum_i |v_i|^p}$.

³ Consider $\mathbb{R}^{2,*} = \mathbb{R}^2 \setminus \{(0, 0)\}$ with the Euclidean inner product. There is no shortest connection from $(-1, 0)$ to $(1, 0)$ on $\mathbb{R}^{2,*}$. A sequence of curves that is in $\mathbb{R}^{2,*}$ and converges to the curve $c : [-1, 1] \rightarrow \mathbb{R}^2, t \mapsto (t, 0)$ is readily constructed. Hence, the Riemannian distance between $(-1, 0)$ and $(1, 0)$ is 2. Yet, every curve connecting these points must go around the origin. The length-minimizing curve of length 2 crosses the origin and is thus not an admissible curve on $\mathbb{R}^{2,*}$.

the length functional, i. e., if $\frac{d}{ds}|_{s=0}L(c_s) = 0$. Such curves are called *geodesics*. Differentiating the length functional leads to the so-called first variation formula [65, §6], which, in turn, leads to the characterizing equation for geodesics:

Definition 7.4 (Geodesics). A differentiable curve $c : [a, b] \rightarrow \mathcal{M}$ is called a *geodesic* (w. r. t. to a given Riemannian metric), if the *covariant derivative* of its velocity vector field vanishes, i. e.,

$$\frac{D\dot{c}}{dt}(t) = 0 \quad \forall t \in [a, b]. \quad (7.6)$$

Remark 7.1. If a starting point $c(0) = p \in \mathcal{M}$ and a starting velocity $\dot{c}(0) = v \in T_p\mathcal{M}$ are specified, then the geodesic equation (7.6) translates to an initial value problem of second order with guaranteed existence and uniqueness of local solutions, [3, p. 102].

An immediate consequence of (7.6) is that geodesics are constant-speed curves. A formal introduction of the covariant derivative $\frac{D}{dt}$ along a curve is beyond the scope of this contribution, and the interested reader is referred to, e. g., [65, §4, §5]. To get some intuition, we introduce this concept for embedded Riemannian submanifolds $\mathcal{M} \subset \mathbb{R}^{n+d}$, where the metric is the Euclidean metric of \mathbb{R}^{n+d} restricted to the tangent bundle; see also [39, §20.12]:

A *vector field along a curve* $c : [a, b] \rightarrow \mathcal{M}$ is a differentiable map $v : [a, b] \rightarrow \mathbb{R}^{n+d}$ such that $v(t) \in T_{c(t)}\mathcal{M}$.⁴ For every $p \in \mathcal{M}$, the ambient \mathbb{R}^{n+d} decomposes into an orthogonal direct sum

$$\mathbb{R}^{n+d} = T_p\mathcal{M} \oplus T_p\mathcal{M}^\perp,$$

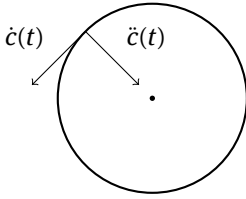
where $T_p\mathcal{M}^\perp$ is the orthogonal complement of $T_p\mathcal{M}$ and orthogonality is w. r. t. the standard Euclidean inner product on \mathbb{R}^{n+d} . Let $\Pi_p : \mathbb{R}^{n+d} \rightarrow T_p\mathcal{M}$ be the (base point-dependent) orthogonal projection onto the tangent space at p . In this setting (and only in this), the covariant derivative of a vector field $v(t)$ along a curve $c(t)$ is the tangent component of $\dot{v}(t)$, i. e., $\frac{Dv}{dt}(t) = \Pi_{c(t)}(\dot{v}(t))$. As a consequence,

$$\frac{D\dot{c}}{dt}(t) = \Pi_{c(t)}(\ddot{c}(t)) \quad (7.7)$$

and the geodesics on Riemannian submanifolds with the metric induced by the ambient Euclidean inner product are precisely the constant-speed curves with acceleration vectors orthogonal to the corresponding tangent spaces, i. e., $\ddot{c}(t) \in T_{c(t)}\mathcal{M}^\perp$.

Example 7.3. On the unit sphere $S^2 \subset \mathbb{R}^3$, the geodesics are great circles. When considered as curves in the ambient \mathbb{R}^3 , their acceleration vector points directly to the origin and is thus orthogonal to the corresponding tangent space, see the cartoon below. When viewed as entities of S^2 , these curves do not experience any acceleration at all.

⁴ The prime example for such a vector field is the curve's own velocity field $v(t) = \dot{c}(t)$.



Mind that a constant-speed curve in \mathbb{R}^n changes its direction only, when it experiences a non-zero acceleration. In this sense, geodesics on manifolds are the counterparts to straight lines in the Euclidean space.

In general, a covariant derivative, also known as a *linear connection*, is a bilinear mapping $(X, Y) \mapsto \nabla_X Y$ that maps two vector fields X, Y to a third vector field $\nabla_X Y$ in such a way that it can be interpreted as the directional derivative of Y in the direction of X . Of importance is the *Riemannian connection* or *Levi-Civita connection* that is compatible with a Riemannian metric [3, Thm 5.3.1], [65, Thm 5.4]. It is determined uniquely by the Koszul formula,

$$\begin{aligned} 2\langle \nabla_X Y, Z \rangle &= X(\langle Y, Z \rangle) + Y(\langle Z, X \rangle) - Z(\langle X, Y \rangle) \\ &\quad - \langle X, [Y, Z] \rangle - \langle Y, [X, Z] \rangle + \langle Z, [X, Y] \rangle, \end{aligned}$$

and is used to define the *Riemannian curvature tensor*

$$(X, Y, Z) \mapsto R(X, Y)Z = \nabla_X \nabla_Y Z - \nabla_Y \nabla_X Z - \nabla_{[X, Y]} Z.^5$$

A Riemannian manifold is flat if and only if it is locally isometric to the Euclidean space, which holds if and only if the Riemannian curvature tensor vanishes identically [65, Thm. 7.3]. Hence, ‘flatness’ depends on the Riemannian metric.

7.2.4 Normal coordinates

The local uniqueness and existence of geodesics allows us to map a tangent vector $v \in T_p \mathcal{M}$ to the endpoint of a geodesic that starts from $p \in \mathcal{M}$ with velocity v . Formalizing this principle gives rise to the *Riemannian exponential*,

$$\text{Exp}_p^{\mathcal{M}} : T_p \mathcal{M} \supset B_\varepsilon(0) \rightarrow \mathcal{M}, \quad v \mapsto q := \text{Exp}_p^{\mathcal{M}}(v) := c_{p,v}(1). \quad (7.8)$$

Here, $t \mapsto c_{p,v}(t)$ is the geodesic that starts from p with velocity v and $B_\varepsilon(0) \subset T_p \mathcal{M}$ is the open ball with radius ε and center 0 in the tangent space;⁶ see Figure 7.2. Note

⁵ In these formulas, $[X, Y] = X(Y) - Y(X)$ is the Lie bracket of two vector fields.

⁶ For technical reasons, $\varepsilon > 0$ must be chosen small enough such that $c_{p,v}(t)$ is defined on the unit interval $[0, 1]$.

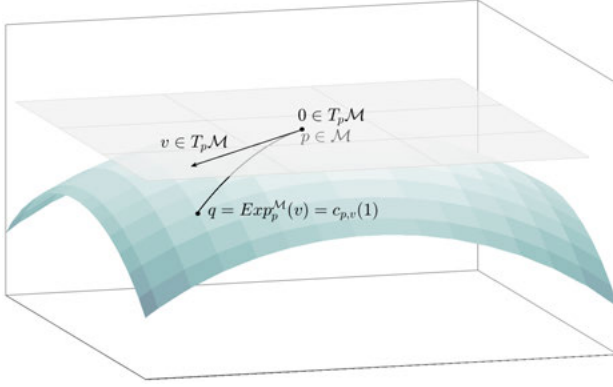


Figure 7.2: The Riemannian exponential sends tangent vectors to end point of geodesic curves.

that we can restrict the considerations to unit-speed geodesics via

$$\text{Exp}_p^{\mathcal{M}}(v) := c_{p,v}(1) = c_{p,v/\|v\|}(t_v) = \text{Exp}_p^{\mathcal{M}}\left(t_v \frac{v}{\|v\|}\right),$$

where $t_v = \|v\|$; see [65, §5., p. 72 ff.] for the details.

For $\varepsilon > 0$ small enough, the Riemannian exponential is a smooth diffeomorphism between $B_\varepsilon(0)$ and an open domain on $\mathcal{D}_p \subset \mathcal{M}$ around the point p . Hence, it is invertible. The smooth inverse map is called the *Riemannian logarithm* and is denoted by

$$\text{Log}_p^{\mathcal{M}} : \mathcal{M} \supset \mathcal{D}_p \rightarrow B_\varepsilon(0) \subset T_p \mathcal{M}, \quad q \mapsto v := (\text{Exp}_p^{\mathcal{M}})^{-1}(q), \quad (7.9)$$

where v satisfies $c_{p,v}(1) = q$.

Thus, the Riemannian logarithm is associated with the *geodesic endpoint problem*: Given $p, q \in \mathcal{M}$, find a geodesic that connects p and q . The Riemannian exponential map establishes a local parametrization of a small region around a location $p \in \mathcal{M}$ in terms of coordinates of the flat vector space $T_p \mathcal{M}$. This is referred to as representing the manifold in *normal coordinates* [60, §III.8], [65, Lem. 5.10]. Normal coordinates are radially isometric in the sense that the Riemannian distance between p and $q = \text{Exp}_p^{\mathcal{M}}(v)$ is exactly the same as the length of the tangent vector $\|v\|_p$ as measured in the metric on $T_p \mathcal{M}$, provided that v is contained in a neighborhood of $0 \in T_p \mathcal{M}$, where the exponential is invertible, [65, Lem. 5.10 & Cor. 6.11].

Mind that the definition of the Riemannian exponential depends on the geodesics, which, in turn, depend on the chosen Riemannian metric—via Definition 7.3. Different metrics lead to different geodesics and thus to different exponential and logarithm maps.

7.2.5 Matrix Lie groups and quotients by group actions

In general, a *Lie group* is a differentiable manifold \mathcal{G} which also has a group structure, such that the group operations ‘multiplication’ and ‘inversion’,

$$\mathcal{G} \times \mathcal{G} \ni (g, \tilde{g}) \mapsto g \cdot \tilde{g} \in \mathcal{G} \quad \text{and} \quad \mathcal{G} \ni g \mapsto g^{-1} \in \mathcal{G}$$

are both smooth [39, 46, 41]. A *matrix Lie group* \mathcal{G} is a subgroup of $GL(n, \mathbb{C})$ that is closed in $GL(n, \mathbb{C})$.⁷ This definition already implies that \mathcal{G} is an embedded submanifold of $\mathbb{C}^{n \times n}$ [46, Corollary 3.45]. Not all matrix groups are Lie groups and not all Lie groups are matrix Lie groups; see [46, §1.1 and §4.8]. However, matrix Lie groups are arguably the most important class of Lie groups when it comes to practical applications and this exposition is restricted to this subclass.

Let \mathcal{G} be an arbitrary matrix Lie group. When endowed with the bracket operator or *matrix commutator* $[V, W] = VW - WV$, the tangent space $T_I \mathcal{G}$ at the identity is called the *Lie algebra* associated with the Lie group \mathcal{G} ; see [46, §3]. As such, it is denoted by $\mathfrak{g} = T_I \mathcal{G}$. For any $A \in \mathcal{G}$, the function “left-multiplication with A ” is a diffeomorphism $L_A : \mathcal{G} \rightarrow \mathcal{G}, L_A(B) = AB$; its differential at a point $M \in \mathcal{G}$ is the isomorphism $d(L_A)_M : T_M \mathcal{G} \rightarrow T_{L_A(M)} \mathcal{G}, d(L_A)_M(V) = AV$. Using this observation at $M = I$ shows that the tangent space at an arbitrary location $A \in \mathcal{G}$ is given by the translates (by left-multiplication) of the tangent space at the identity:

$$T_A \mathcal{G} = T_{L_A(I)} \mathcal{G} = A\mathfrak{g} = \{\Delta = AV \in \mathbb{R}^{n \times n} \mid V \in \mathfrak{g}\}, \quad (7.10)$$

[41, §5.6, p.160]. The Lie algebra $\mathfrak{g} = T_I \mathcal{G}$ of \mathcal{G} can equivalently be characterized as the set of all matrices Δ such that $\exp_m(t\Delta) \in \mathcal{G}$ for all $t \in \mathbb{R}$. The intuition behind this fact is that all tangent vectors are velocity vectors of smooth curves running on \mathcal{G} (Definition 7.2) and that $c(t) = \exp_m(t\Delta)$ is a smooth curve starting from $c(0) = I$ with velocity $\dot{c}(0) = \Delta$; see [46, Def. 3.18 & Cor. 3.46] for the details. By definition, the exponential map⁸ for a matrix Lie group is the matrix exponential restricted to the corresponding Lie algebra, i. e. the tangent space at the identity $\mathfrak{g} = T_I \mathcal{G}$, [46, §3.7],

$$\exp_m|_{\mathfrak{g}} : \mathfrak{g} \rightarrow \mathcal{G}.$$

In general, a Lie algebra is a vector space with a linear, skew-symmetric bracket operation, called *Lie bracket* $[\cdot, \cdot]$, that satisfies the *Jacobi identity*.

$$[X, [Y, Z]] + [Z, [X, Y]] + [Y, [Z, X]] = 0.$$

⁷ But not necessarily in $\mathbb{C}^{n \times n}$.

⁸ The exponential map of a Lie group must not be confused with the Riemannian exponential.

Quotients of Lie groups by closed subgroups

In many settings, it is important or sometimes even necessary to consider certain points p, q on a given differentiable manifold \mathcal{M} as equivalent. Consider the following example.

Example 7.4. Let $U \in \mathbb{R}^{n \times r}$ feature orthonormal columns so that $U^T U = I_r$. We may extend the columns of $U = (u^1, \dots, u^r)$ to an orthogonal matrix $Q = (u^1, \dots, u^r, u^{r+1}, \dots, u^n) \in O(n)$. Define $I_r \times O(n-r) := \left\{ \begin{pmatrix} I_r & 0 \\ 0 & R \end{pmatrix} \mid R \in O(n-r) \right\}$. This is actually a closed subgroup of $O(n)$, in symbols $(I_r \times O(n-r)) \leq O(n)$. The action $\tilde{Q} = Q\Phi$ with any orthogonal matrix $\Phi \in I_r \times O(n-r)$ preserves the first r columns of Q . Hence, we may identify U with the equivalence class $[Q] = \{Q\Phi \mid \Phi \in I_r \times O(n-r)\} \subset O(n)$. In Sections 7.4.4 and 7.4.5, we will see that this example establishes the Stiefel manifold of ONBs and eventually also the Grassmann manifold of subspaces as quotients of the orthogonal group $O(n)$.

Note that in the example, the equivalence relation is induced by actions of the Lie group $I_r \times O(n-r)$. Quotients that arise from such group actions are important examples of *quotient manifolds*. Theorems 7.4 and 7.5 cover this example as well as all other cases of quotient manifolds that are featured in this chapter. First, group actions need to be formalized.

Definition 7.5 (Cf. [66, p. 162, 163]). Let \mathcal{G} be a Lie group, \mathcal{M} be a smooth manifold, and let $\mathcal{G} \times \mathcal{M} \rightarrow \mathcal{M}, (g, p) \mapsto g \cdot p$ be a *left action of \mathcal{G} on \mathcal{M}* .⁹ The *orbit relation* on \mathcal{M} induced by \mathcal{G} is defined by

$$p \simeq q \Leftrightarrow \exists g \in \mathcal{G} : g \cdot p = q.$$

The equivalence classes are the \mathcal{G} -orbits $[p] := \mathcal{G}p := \{g \cdot p \mid g \in \mathcal{G}\}$. The *orbit space* is denoted by $\mathcal{M}/\mathcal{G} := \{[p] \mid p \in \mathcal{M}\}$. The *quotient map* sends a point to its \mathcal{G} -orbit via $\Pi : \mathcal{M} \rightarrow \mathcal{M}/\mathcal{G}, p \mapsto [p]$. The action is *free*, if every isotropy group $\mathcal{G}_p := \{g \in \mathcal{G} \mid g \cdot p = p\}$ is trivial, $\mathcal{G}_p = \{e\}$.

Theorem 7.4 (Quotient manifold theorem, cf. [66, Thm. 21.10]). *Suppose \mathcal{G} is a Lie group acting smoothly, freely, and properly on a smooth manifold \mathcal{M} . Then the orbit space \mathcal{M}/\mathcal{G} is a manifold of dimension $\dim \mathcal{M} - \dim \mathcal{G}$, and has a unique smooth structure such that the quotient map $\Pi : \mathcal{M} \rightarrow \mathcal{M}/\mathcal{G}, p \mapsto [p]$ is a smooth submersion.¹⁰ In this context, \mathcal{M} is called the *total space* and \mathcal{M}/\mathcal{G} is the *quotient (space)*.*

A special case is Lie groups under actions of Lie subgroups.

⁹ The theory for right actions is analogous. In all cases considered in this chapter, \mathcal{M} is a matrix manifold so that “ \cdot ” is the usual matrix product.

¹⁰ I. e. a smooth surjective mapping such that the differential is surjective at every point.

Definition 7.6. [66, §21, p. 551] Let \mathcal{G} be a Lie group and $\mathcal{H} \leq \mathcal{G}$ be a Lie subgroup. For $g \in \mathcal{G}$, a subset of \mathcal{G} of the form $[g] := g\mathcal{H} = \{g \cdot h \mid h \in \mathcal{H}\}$ is called a *left coset* of \mathcal{H} . The left cosets form a partition of \mathcal{G} , and the quotient space determined by this partition is called the *left coset space of \mathcal{G} modulo \mathcal{H}* , and is denoted by \mathcal{G}/\mathcal{H} .

Coset spaces of Lie groups are again smooth manifolds.

Theorem 7.5 (Cf. [66, Thm 21.17, p. 551]). *Let \mathcal{G} be a Lie group and let \mathcal{H} be a closed subgroup of \mathcal{G} . The left coset space \mathcal{G}/\mathcal{H} is a manifold of dimension $\dim \mathcal{G} - \dim \mathcal{H}$ with a unique differentiable structure such that the quotient map $\Pi : \mathcal{G} \rightarrow \mathcal{G}/\mathcal{H}, g \mapsto [g]$ is a smooth submersion.*

In general, if $\pi : \mathcal{M} \rightarrow \mathcal{N}$ is a surjective submersion between two manifolds \mathcal{M} and \mathcal{N} , then for any $q \in \mathcal{N}$, the preimage $\pi^{-1}(q) \subset \mathcal{M}$ is called the *fiber over q* , and is denoted by \mathcal{M}_q . Each fiber \mathcal{M}_q is itself a closed, embedded submanifold by the implicit function theorem. If \mathcal{M} has a Riemannian metric $\langle \cdot, \cdot \rangle_p^{\mathcal{M}}$, then at each point $p \in \mathcal{M}$, the tangent space $T_p\mathcal{M}$ decomposes into an *orthogonal direct sum* $T_p\mathcal{M} = T_p\mathcal{M}_{\pi(p)} \oplus (T_p\mathcal{M}_{\pi(p)})^\perp$. The tangent space of the fiber $T_p\mathcal{M}_{\pi(p)} =: V_p$ is called the *vertical space*, its orthogonal complement $H_p := V_p^\perp$ is the *horizontal space*. The vertical space is the kernel $V_p = \ker(d\pi_p)$ of the differential $d\pi_p : T_p\mathcal{M} \rightarrow T_{\pi(p)}\mathcal{N}$; the horizontal space is isomorphic to $T_{\pi(p)}\mathcal{N}$. This allows one to identify $H_p \cong T_{\pi(p)}\mathcal{N}$; see [3, Figure 3.8., p. 44] for an illustration. This construction helps to compute tangent spaces of quotients, if the tangent space of the total space is known.

If \mathcal{G}/\mathcal{H} is a quotient as in Theorem 7.4 or 7.5 and if $\Pi : \mathcal{G} \rightarrow \mathcal{G}/\mathcal{H}$ is the corresponding quotient map, then Π is a local diffeomorphism. A Riemannian metric on the quotient can be defined by

$$\langle v, w \rangle_{[g]}^{\mathcal{G}/\mathcal{H}} := \langle (d\Pi_g)^{-1}(v), (d\Pi_g)^{-1}(w) \rangle_g^{\mathcal{G}}, \quad v, w \in T_{[g]}(\mathcal{G}/\mathcal{H}). \quad (7.11)$$

For this (and only this) metric, the quotient map is a *local isometry*.

In fact, Theorem 7.5 additionally establishes \mathcal{G}/\mathcal{H} as a *homogeneous space*, i. e. a smooth manifold \mathcal{M} endowed with a *transitive smooth action* by a Lie group (cf. [66, §21, p. 550]). In the setting of the theorem, the group action is given by the left action of \mathcal{G} on \mathcal{G}/\mathcal{H} given by $g_1 \cdot [g_2] := [g_1 \cdot g_2]$. A transitive action allows us to transport a location $p \in \mathcal{M}$ to any other location $q \in \mathcal{M}$.

7.3 Interpolation on non-flat manifolds

When working with matrix manifolds, the data is usually given in extrinsic coordinates; see Section 7.2. For example, data on the compact Stiefel manifold $St(n, r) = \{U \in \mathbb{R}^{n \times r} \mid U^T U = I_r\}$, $r \leq n$, is given in form of n -by- r matrices. These matrices

feature nr entries while the intrinsic number of degrees of freedom, i. e., the intrinsic dimension is $nr - \frac{1}{2}r(r+1)$; see Section 7.4.4. Essentially, the practical obstacle associated with data interpolation on matrix manifolds arises from this fact. Given, say, k matrices on $St(n, r)$ in extrinsic coordinates, interpolating entry-by-entry will most certainly lead to interpolants that do not feature orthogonal columns and thus are *not* points on the Stiefel manifold. Likewise, entry-by-entry interpolation of positive definite matrices is not guaranteed to produce another positive definite matrix.

There are essentially two different approaches to address this issue: Performing the interpolation on the tangent space of the manifold and using the Riemannian barycenter or Riemannian center of mass as an interpolant. Both will be explained in more detail in the next two subsections.¹¹

7.3.1 Interpolation in normal coordinates

As outlined in Section 7.2, every location $p \in \mathcal{M}$ on an n -dimensional differentiable manifold features a small neighborhood \mathcal{D}_p that is the domain of a coordinate chart $x : \mathcal{M} \supset \mathcal{D}_p \rightarrow D_{x(p)} \subset \mathbb{R}^n$ that maps bijectively onto an open set $D_{x(p)} \subset \mathbb{R}^n$. Therefore, for a sample data set $\{p_1, \dots, p_k\} \subset \mathcal{D}_p$ that is completely contained in the domain of a single coordinate chart x , interpolation can be performed as follows:

1. Map the data set to $D_{x(p)}$: Calculate $v_1 = x(p_1), \dots, v_k = x(p_k) \in D_{x(p)}$.
2. Interpolate in $D_{x(p)}$ to produce the interpolant $v^* \in D_{x(p)}$.
3. Map back to manifold: compute $p^* = x^{-1}(v^*) \in \mathcal{D}_p$.

In principle, any coordinate chart may be applied. In practice, the challenge is to find a suitable coordinate chart that can be evaluated efficiently. Moreover, it is desirable that the chosen chart preserves the geometry of the original data set as well as possible.¹² The standard choice is to use *normal coordinates* as introduced in Section 7.2.4. This means that the Riemannian logarithm is used as the coordinate chart

$$\text{Log}_p^{\mathcal{M}} : \mathcal{M} \supset \mathcal{D}_p \rightarrow B_\varepsilon(0) \subset T_p\mathcal{M}$$

with the Riemannian exponential

$$\text{Exp}_p^{\mathcal{M}} : T_p\mathcal{M} \supset B_\varepsilon(0) \rightarrow \mathcal{D}_p \subset \mathcal{M}$$

as the corresponding parameterization. The general procedure of data interpolation via the tangent space is formulated as Algorithm 7.1.

¹¹ The German speaking reader may find an introduction that addresses a general scientific audience in [90].

¹² There are no isometric coordinate charts on a non-flat manifold; see [65, Thm 7.3].

Algorithm 7.1: Interpolation in normal coordinates.**Input:** Data set $\{p_1, \dots, p_k\} \subset \mathcal{M}$.

- 1: Choose $p_i \in \{p_1, \dots, p_k\}$ as a base point.
- 2: Check that $\text{Log}_{p_i}^{\mathcal{M}}(p_j)$ is well-defined for all $j = 1, \dots, k$.
- 3: **for** $j = 1, \dots, k$ **do**
- 4: Compute $v_j := \text{Log}_{p_i}^{\mathcal{M}}(p_j) \in T_{p_i}\mathcal{M}$.
- 5: **end for**
- 6: Compute v^* via Euclidean interpolation of $\{v_1, \dots, v_k\}$.
- 7: Compute $p^* := \text{Exp}_{p_i}^{\mathcal{M}}(v^*)$

Output: $p^* \in \mathcal{M}$.**Remark 7.2.** There are a few facts that the practitioner needs to be aware of:

1. The interpolation procedure of Algorithm 7.1 depends on which sample point is selected to act as the base point. Different choices may lead to different interpolants.¹³
2. For *matrix manifolds*, the tangent space is often also given in *extrinsic coordinates*. This means that an entry-by-entry interpolation of the matrices that represent the tangent vectors may lead to an interpolant that is not in the tangent space. As an illustrative example, consider the Grassmannian $Gr(n, r)$. Matrices $\Delta_1, \dots, \Delta_k \in T_{[U]}Gr(n, r)$ are characterized by $U^T \Delta_j = 0$. Entry-by-entry interpolation in the tangent space may potentially result in a matrix Δ^* that is not orthogonal to the base point U , i. e. $U^T \Delta^* \neq 0$; see [100, §2.4].

In general, because of the vector space structure of the tangent space of any manifold \mathcal{M} , it is sufficient to use an interpolation method that expresses the interpolant in $T_p\mathcal{M}$ as a weighted linear combination of the sampled tangent vectors $v_1, \dots, v_k \in T_p\mathcal{M}$,

$$v^* = \sum_{j=1}^k \omega_j v_j.$$

Amongst others, *linear interpolation*, *Lagrange and Hermite interpolation*, *spline interpolation* and interpolation via *radial basis functions* fulfill this requirement. As an aside, the interpolation procedure is computationally less expensive, since it works on the weight coefficients ω_j rather than on every single entry.

Quasi-linear interpolation of trajectories via geodesics

In this paragraph, we address applications, where the sampled manifold data features a univariate parametric dependency. The setting is as follows. Let \mathcal{M} be a Riemannian

¹³ In the practical applications considered in [8], it was observed that the base point selection has only a minor impact on the final result.

manifold and suppose that there is a trajectory

$$c : [a, b] \rightarrow \mathcal{M}, \quad \mu \mapsto c(\mu)$$

on \mathcal{M} that is sampled at k instants $\mu_1, \dots, \mu_k \in [a, b]$. Then an interpolant \hat{c} for c can be computed via Algorithm 7.2.

Algorithm 7.2: Geodesic interpolation.

Input: Data set $\{c(\mu_1), \dots, c(\mu_k)\} \subset \mathcal{M}$ sampled from a curve $c : \mu \rightarrow c(\mu)$, unsampled instant $\mu^* \in [\mu_j, \mu_{j+1}]$.

- 1: Compute $v_{j+1} := \text{Log}_{c(\mu_j)}^{\mathcal{M}}(c(\mu_{j+1})) \in T_{c(\mu_j)}\mathcal{M}$.
- 2: Compute $\hat{c}(\mu^*) := \text{Exp}_{c(\mu_j)}^{\mathcal{M}}(\frac{\mu^* - \mu_j}{\mu_{j+1} - \mu_j} v_{j+1})$

Output: $\hat{c}(\mu^*) \in \mathcal{M}$ interpolant of $c(\mu^*)$.

The interpolants at $\mu \in [\mu_j, \mu_{j+1}]$ that are output by Algorithm 7.2 lie on the unique geodesic connection between the points $c(\mu_j)$ and $c(\mu_{j+1})$. Hence, it is the straightforward manifold analogue of linear interpolation and is base-point independent.

The generic formulation of Algorithm 7.1 allows one to employ higher-order interpolation methods. However, this does not necessarily lead to more accurate results: the overall error depends not only on the interpolation error within the tangent space but also on the distortion caused by mapping the data to a selected (fixed) tangent space; see Figure 7.3.

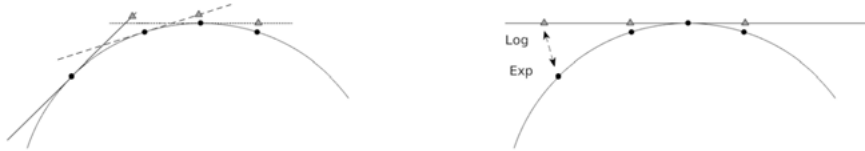


Figure 7.3: Illustration of the course of action of Algorithms 7.1 and 7.2. Algorithm 7.1 (right) first maps all data points to a selected fixed tangent space. In Algorithm 7.2 (left), two points $p_j = c(\mu_j)$ and $p_{j+1} = c(\mu_{j+1})$ are connected by a geodesic line, then the base is shifted to point p_{j+1} and the procedure is repeated.

Algorithms 7.1 and 7.2 can be applied in practical applications, where the Riemannian exponential and logarithm mappings are known in explicit form. Applications in parametric model reduction that consider matrix manifolds include [34] ($GL(n)$ -data), [8, 76, 100] (Grassmann data), [104] (Stiefel data) and [9, 82] (SPD(n) data).

7.3.2 Interpolation via the Riemannian center of mass

As pointed out in Remark 7.2, interpolation of manifold data via the back and forth mapping of a complete data set of sample points between the manifold and its tan-

gent space depends on the chosen base point. As a consequence, sample points may experience an uneven distortion under the projection onto the tangent space; see Figure 7.3 (right). An approach that avoids this issue is to interpret interpolation as the task of finding suitably weighted Riemannian centers of mass. This concept was introduced in the context of geodesic finite elements in [44, 91].

The idea is as follows: The Riemannian center of mass¹⁴ or Fréchet mean of a sample data set $\{p_1, \dots, p_k\} \in \mathcal{M}$ on a manifold with respect to the scalar weights $w_i \geq 0$, $\sum_{i=1}^k w_i = 1$ is defined as the minimizer(s) of the Riemannian objective function

$$\mathcal{M} \ni q \mapsto f(q) = \frac{1}{2} \sum_{i=1}^k w_i \operatorname{dist}(q, p_i)^2 \in \mathbb{R},$$

where $\operatorname{dist}(q, p_i)$ is the Riemannian distance of (7.5). This definition generalizes the notion of the barycentric mean in Euclidean spaces. However, on curved manifolds, the global center might not be unique. Moreover, local minimizers may appear. For more details, see [58] and [4], which also give uniqueness criteria.

Interpolation is now performed by computing weighted Riemannian centers. More precisely, let $\mu_1, \dots, \mu_k \in \mathbb{R}^d$ be sampled parameter locations and let $p_i = p(\mu_i) \in \mathcal{M}$, $i = 1, \dots, k$ be the corresponding sample locations on \mathcal{M} . Interpolation is within the convex hull $\operatorname{conv}\{\mu_1, \dots, \mu_k\} \subset \mathbb{R}^d$ of the samples.

Let $\{\varphi_i : \mu \mapsto \varphi_i(\mu) \mid i = 1, \dots, k\}$ be a suitable set of interpolation functions with $\varphi_i(\mu_j) = \delta_{ij}$, say Lagrangians [91], splines [44] or radial basis functions [26]. Then the interpolant $p^* \approx p(\mu^*) \in \mathcal{M}$ at an unsampled parameter location $\mu^* \in \operatorname{conv}\{\mu_1, \dots, \mu_k\}$ is defined as the minimizer of

$$p^* = \arg \min_{q \in \mathcal{M}} f(q) = \frac{1}{2} \sum_{i=1}^k \varphi_i(\mu^*) \operatorname{dist}(q, p_i)^2. \quad (7.12)$$

At a sample location μ_j , one has indeed that $\sum_{i=1}^k \varphi_i(\mu_j) \operatorname{dist}(q, p_i)^2 = \sum_{i=1}^k \delta_{ij} \operatorname{dist}(q, p_i)^2 = \operatorname{dist}(q, p_j)^2$, which has the unique global minimum at $q = p_j$.

Computing p^* requires one to solve a Riemannian optimization problem. The simplest approach is a gradient descent method [3, 4]. The gradient of the objective function f in (7.12) is

$$\nabla f_q = - \sum_{i=1}^k \varphi_i(\mu^*) \operatorname{Log}_q^{\mathcal{M}}(p_i) \in T_q \mathcal{M}, \quad (7.13)$$

see [58, Thm 1.2], [4, §2.1.5], [91, eq. (2.4)]. Hence, just like interpolation in the tangent space, the interpolation via the Riemannian center can be pursued only in applications, where the Riemannian logarithm can be computed. A generic gradient descent

¹⁴ Here, we introduce this for discrete data sets; for centers w. r. t. a general mass distribution; see the original paper [58], Section 1.

algorithm to compute the barycentric interpolant for a function $p : \mathbb{R}^d \ni \mu \mapsto p(\mu) \in \mathcal{M}$ reads as follows.

Algorithm 7.3: Interpolation via the weighted Riemannian center [4, 84].

Input: Sample data set $\{p_1 = p(\mu_1), \dots, p_k = p(\mu_k)\} \subset \mathcal{M}$, unsampled parameter location $\mu^* \in \text{conv}(\mu_1, \dots, \mu_k) \subset \mathbb{R}^d$, initial guess q_0 , convergence threshold τ .

- 1: $k := 0$
- 2: Compute ∇f_{q_k} according to (7.13)
- 3: **while** $\|\nabla f_{q_k}\|_q > \tau$ **do**
- 4: select a step size α_k
- 5: $q_{k+1} := \text{Exp}_{q_k}^{\mathcal{M}}(-\alpha_k \nabla f_{q_k})$
- 6: $k := k + 1$
- 7: **end while**

Output: $p^* := q_k \in \mathcal{M}$ interpolant of $p(\mu^*)$.

An implementation of this (type of) method for finding the Karcher mean in $SO(3)$ is discussed in [84]. Of course, Riemannian analogues to more sophisticated nonlinear optimization methods may also be employed; see [3].

In the context of model reduction, the benefits of interpolation via weighted Riemannian centers and the computational costs of solving the associated Riemannian optimization problem must be juxtaposed.

7.3.3 Additional approaches

A large variety of sophisticated ideas and further manifold interpolation techniques exist in the literature: The acceleration-minimizing property of cubic splines in the Euclidean space can be generalized to Riemannian manifolds in the form of a variational problem [24, 27, 33, 57, 77, 88, 93]; see also [81] and the references therein. Moreover, the construction concepts of Bézier curves and the De Casteljau algorithm [15] can be transferred to Riemannian manifolds [1, 62, 81, 75, 89]. Bézier curves in Euclidean spaces are polynomial splines that rely on a number of so-called control points. To obtain the value of a Bézier curve at time t , a recursive sequence of straight-line convex combinations between pairs of control points must be computed. The transition of this technique to Riemannian manifolds is via replacing the inherent straight lines with geodesics [81]. Another option is to conduct the Bézier–De Casteljau algorithm in the tangent space and to transfer the results to the manifold via a geodesic averaging of the spline arcs that were constructed in the tangent spaces at the first and the last control point, respectively; see [43].

Derivative information may also be incorporated in manifold interpolation schemes. A Hermite-type method that is specifically tailored for interpolation prob-

lems on the Grassmann manifold is sketched in [7, §3.7.4]. General Hermitian manifold interpolation in compact, connected Lie groups with a bi-invariant metric has been considered in [55]. A practical approach to first-order Hermite interpolation of data on arbitrary Riemannian manifolds is discussed in [103].

7.3.4 Quasi-linear extrapolation on matrix manifolds

In application scenarios, where both snapshot data of the full-order model and derivative information are at hand, various approaches have been suggested to exploit the latter. On the one hand, derivatives can be used for improving the ROMs accuracy and approximation quality by constructing POD bases that incorporate snapshots and snapshot derivatives [28, 51, 54, 99]. On the other hand, snapshot derivatives enable to parameterize the ROM bases and subspaces or to perform sensitivity analyses [48, 47, 97, 101]. In this section, we outline an approach to transferring the idea of extrapolation and parameterization via local linearizations to manifold-valued functions. The underlying idea is comparable to the trajectory piece-wise linear (TPWL) method; see [85] and Chapter 3 of this volume. Yet, TPWL linearizes the full-order model prior to the ROM projection, whereas here we consider linearizing ROM building blocks like the reduced orthogonal bases, reduced subspaces or reduced system matrices.

A geometric first-order Taylor approximation

Any differentiable function $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ can be linearized via a first-order Taylor expansion. A step ahead of size t in direction $d \in \mathbb{R}^n$ gives $f(x_0 + td) = f(x_0) + tDf_{x_0}(d) + \mathcal{O}(t^2)$. When considering $t \mapsto c(t) := f(x_0 + td)$ as a curve, then the first-order Taylor approximant is the straight line $g : t \mapsto c(0) + \dot{c}(0)t$. Such a first-order linearization often serves for extrapolating a given nonlinear function in a neighborhood of a selected expansion point. For doing so, the starting point $c(0)$ and the starting velocity $\dot{c}(0)$ must be available. This procedure translates to the manifold setting, when straight lines are replaced with geodesics.

Let $\mu \in \mathbb{R}$ be a scalar parameter and let $c : \mu \mapsto c(\mu) \in \mathcal{M}$ be a curve on a submanifold \mathcal{M} . For given initial values $c(\mu_0) = p_0 \in \mathcal{M}$ and $\dot{c}(\mu_0) = v_0 \in T_{p_0}\mathcal{M}$, the corresponding unique geodesic c_{p_0, v_0} is expressed via the Riemannian exponential as

$$c_{p_0, v_0} : \mu \rightarrow \mathcal{M}, \quad \mu \mapsto \text{Exp}_{p_0}^{\mathcal{M}}(\mu v_0).$$

Algorithm 7.4: Geodesic extrapolation.

Input: Scalar parameter $\mu_0 \in \mathbb{R}$, initial values $c(\mu_0) \in \mathcal{M}$, $\dot{c}(\mu_0) \in T_{c(\mu_0)}\mathcal{M}$ sampled from a curve $c : \mu \mapsto c(\mu) \in \mathcal{M}$, parameter value $\mu^* > 0$.

1: Compute $\hat{c}(\mu_0 + \mu^*) := \text{Exp}_{c(\mu_0)}^{\mathcal{M}}(\mu^* \dot{c}(\mu_0))$

Output: $\hat{c}(\mu_0 + \mu^*) \in \mathcal{M}$ extrapolant of $c(\mu_0 + \mu^*)$.

Example: extrapolating POD basis matrices

As outlined in Section 7.1.1, snapshot POD works by collecting state vector snapshots, $x^1 := x(t_1, \mu_0), \dots, x^m := x(t_m, \mu_0) \in \mathbb{R}^n$ followed by an SVD of the snapshot matrix $(x^1, \dots, x^m)(\mu_0) =: \mathbb{S}(\mu_0) = \mathbb{U}(\mu_0)\Sigma(\mu_0)\mathbb{Z}^T(\mu_0)$. Here, the matrix dimensions are $\mathbb{U}(\mu_0) \in \mathbb{R}^{n \times m}$, $\Sigma(\mu_0) \in \mathbb{R}^{m \times m}$, $\mathbb{Z}(\mu_0) \in \mathbb{R}^{m \times m}$. The objective is to approximate $\mathbb{U}(\mu_0 + \mu)$ for a small $\mu > 0$ based on the data $\mathbb{U}(\mu_0), \dot{\mathbb{U}}(\mu_0)$, where $\mathbb{U}(\mu_0)$ is a point on the Stiefel manifold $St(n, m)$ and $\dot{\mathbb{U}}(\mu_0)$ is a tangent vector; see Section 7.4.4.1.

Differentiating the SVD. If the snapshot matrix function $\mu \mapsto \mathbb{S}(\mu) \in \mathbb{R}^{n \times m}$ is smooth in the neighborhood of $\mu_0 \in \mathbb{R}$ and if the singular values of $\mathbb{S}(\mu_0)$ are mutually distinct,¹⁵ then the singular values and both the left and the right singular vectors are differentiable in $\mu \in [\mu_0 - \delta\mu, \mu_0 + \delta\mu]$ for $\delta\mu$ small enough. For brevity, let $\dot{\mathbb{S}} = \frac{d\mathbb{S}}{d\mu}(\mu_0)$ denote the derivative with respect to μ evaluated in μ_0 and so forth. Let $\mu \mapsto \mathbb{S}(\mu) = \mathbb{U}(\mu)\Sigma(\mu)\mathbb{Z}(\mu)^T \in \mathbb{R}^{n \times m}$ and let $C(\mu) = (\mathbb{S}^T \mathbb{S})(\mu)$. Let u^j and v^j , $j = 1, \dots, m$, denote the columns of $\mathbb{U}(\mu_0)$ and $\mathbb{Z}(\mu_0)$, respectively. We have

$$\dot{\sigma}_j = (u^j)^T \dot{\mathbb{S}} v^j, \quad (j = 1, \dots, m), \quad (7.14)$$

$$\dot{\mathbb{Z}} = \mathbb{Z}A, \quad \text{where } A_{ij} = \begin{cases} \frac{\sigma_j(u^j)^T \dot{\mathbb{S}} v^i + \sigma_i(u^i)^T \dot{\mathbb{S}} v^j}{(\sigma_j + \sigma_i)(\sigma_j - \sigma_i)}, & i \neq j \\ 0, & i = j \end{cases} \quad (i, j = 1, \dots, m), \quad (7.15)$$

$$\dot{\mathbb{U}} = \dot{\mathbb{S}}\mathbb{Z}\Sigma^{-1} + \mathbb{S}\dot{\mathbb{Z}}\Sigma^{-1} + \mathbb{S}\mathbb{Z}\dot{\Sigma}^{-1} = (\dot{\mathbb{S}}\mathbb{Z} + \mathbb{U}(\Sigma A - \dot{\Sigma}))\Sigma^{-1}. \quad (7.16)$$

A proof can be found in [48]. Note that $\mathbb{U}^T(\mu_0)\dot{\mathbb{U}}(\mu_0)$ is skew-symmetric so that indeed $\dot{\mathbb{U}}(\mu_0) =: \Delta(\mu_0) \in T_{\mathbb{U}(\mu_0)}St(n, m)$. The above equations hold in approximative form for the truncated SVD. For convenience, assume that $\mathbb{U}(\mu_0) \in St(n, r)$ is now the truncated to $r \leq m$ columns.

Performing the Taylor extrapolation on $St(n, r)$. With $\mathbb{U}(\mu_0), \dot{\mathbb{U}}(\mu_0)$ at hand, $\mathbb{U}(\mu_0 + \mu)$ can be approximated using the Stiefel exponential: $\mathbb{U}(\mu_0 + \mu) \approx \hat{\mathbb{U}}(\mu_0 + \mu) := \text{Exp}_{\mathbb{U}_0}^{St}(\mu \dot{\mathbb{U}}(\mu_0))$; see Algorithm 7.7. The process is illustrated in Figure 7.4.

Note that when the μ -dependency is real-analytic, then the Euclidean Taylor expansion

$$\mathbb{U}(\mu_0 + \mu) = \mathbb{U}(\mu_0) + \mu \dot{\mathbb{U}}(\mu_0) + \frac{\mu^2}{2} \ddot{\mathbb{U}}(\mu_0) + \mathcal{O}(\mu^3) \in St(n, r) \quad (7.17)$$

converges to an orthogonal matrix $\mathbb{U}(\mu_0 + \mu) \in St(n, r)$. Yet, when truncating the Taylor series, we leave the Stiefel manifold. In particular, the columns of the first-order approximation are not orthonormal, i. e. $\mathbb{U}(\mu_0) + \mu \dot{\mathbb{U}}(\mu_0) \notin St(n, r)$ for $\mu \neq 0$. By construction, the Stiefel geodesic features the same starting velocity $\dot{\mathbb{U}}(\mu_0)$ and thus matches the Taylor series up to terms of second order. In addition, it respects the geometric structure of the Stiefel manifold and thus preserves column-orthonormality for every μ .

¹⁵ This condition can be relaxed; see the results of [5, §7].

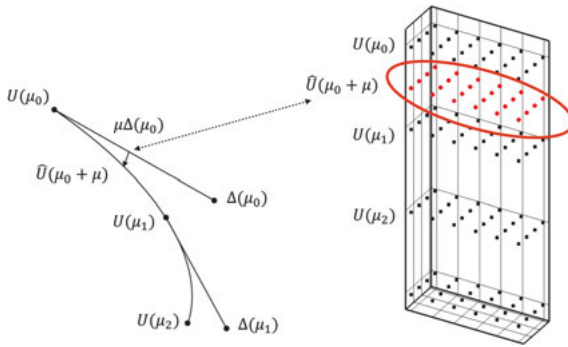


Figure 7.4: Extrapolation of matrix manifold data. Sketched on the right is the sample matrix data in $\mathbb{R}^{n \times r}$. The curved line on the left represents the nonlinear matrix manifold; the straight lines represent the tangent vectors in the tangent space. The matrix curve is linearized at $U(q_0)$, $U(q_1)$, etc.

7.4 Matrix manifolds of practical importance

In this section, we discuss the matrix manifolds that feature most often in practical applications in the context of model reduction. For each manifold under consideration, we recap, if applicable

- the representation of points/locations in numerical schemes.
- the representation of tangent vectors in numerical schemes.
- the most common Riemannian metrics.
- how to compute distances, geodesics and the Riemannian exponential and logarithm mappings.

7.4.1 The general linear group

This section is devoted to the general linear group $GL(n)$ of invertible square matrices. In model reduction, regular matrices appear for example as (reduced) system matrices in LTI and discretized PDE systems [9, 34, 78] and parameterizations have to be such that matrix regularity is preserved. In addition, the discussion of the seemingly simple matrix manifold $GL(n)$ is important, because it is the fundamental matrix Lie group from which all other matrix Lie groups are derived. Moreover, it provides the background for understanding quotient spaces of $GL(n)$; see Subsection 7.2.5 and also [23, 96]. A short summary on the Riemannian geometry of $GL(n)$ is given in [83, §6].

7.4.1.1 Introduction and data representation in numerical schemes

Because $GL(n) = \det^{-1}(\mathbb{R} \setminus \{0\}) = \{A \in \mathbb{R}^{n \times n} \mid \det(A) \neq 0\}$, $GL(n)$ is an open subset of the n^2 -dimensional vector space $\mathbb{R}^{n \times n} \simeq \mathbb{R}^{n^2}$ and is thus an n^2 -dimensional differentiable

manifold; see [66, Examples 1.22–1.27]. The matrix manifold $GL(n)$ is disconnected as it decomposes into two connected components, namely the regular matrices of positive determinant and the regular matrices of negative determinant.

Because $GL(n)$ is an open subset of the vector space $\mathbb{R}^{n \times n}$, the tangent space at a location $A \in GL(n)$ is simply $T_A GL(n) = \mathbb{R}^{n \times n}$. For $GL(n)$, the Lie algebra is $\mathfrak{gl}(n) = \mathbb{R}^{n \times n}$, so that the Lie group exponential is the standard matrix exponential $\exp_m : \mathbb{R}^{n \times n} = \mathfrak{gl}(n) \rightarrow GL(n)$. From the Lie group perspective (7.10), the tangent space at an arbitrary point $A \in GL(n)$ is to be considered as the set $T_A GL(n) = A\mathfrak{gl}(n) = A(\mathbb{R}^{n \times n})$, even though this set coincides with $\mathbb{R}^{n \times n}$.

7.4.1.2 Distances and geodesics

The obvious choice for a Riemannian metric on $GL(n)$ is to use the inner product from the ambient Euclidean matrix space, i. e.,

$$\langle \Delta, \tilde{\Delta} \rangle_A = \langle \Delta, \tilde{\Delta} \rangle_0 = \text{trace}(\Delta^T \tilde{\Delta}),$$

for $A \in GL(n)$ and $\Delta, \tilde{\Delta} \in T_A GL(n) = \mathbb{R}^{n \times n}$.

In many applications, it is more appropriate to consider metrics with certain invariance properties.¹⁶ A *left-invariant metric* can be obtained from the standard metric via

$$\langle \Delta, \tilde{\Delta} \rangle_A = \langle A^{-1}\Delta, A^{-1}\tilde{\Delta} \rangle_0, \quad A \in GL(n), \quad \Delta, \tilde{\Delta} \in T_A GL(n). \quad (7.18)$$

When formally considering $\Delta = AV, \tilde{\Delta} = A\tilde{V} \in T_A GL(n) = A\mathfrak{gl}(n)$ as left-translates of tangent vectors $V, \tilde{V} \in T_I GL(n) = \mathfrak{gl}(n)$, then this metric satisfies $\langle \Delta, \tilde{\Delta} \rangle_A = \langle V, \tilde{V} \rangle_0$. Alternatively, $\langle V, \tilde{V} \rangle_0 = \langle AV, A\tilde{V} \rangle_A$, which explains the name ‘left-invariant’.

The Riemannian exponential and logarithm for the flat metric

When equipped with the Euclidean metric, $GL(n)$ is flat: since the tangent space is the full matrix space $\mathbb{R}^{n \times n}$, the geodesic equation (7.7) requires the acceleration of a geodesic curve to vanish completely. Hence, the geodesic that starts from $A \in GL(n)$ with velocity $\Delta \in \mathbb{R}^{n \times n}$ is the straight line $C(t) = A + t\Delta$. Note that the curve $t \mapsto C(t)$ may leave the manifold $GL(n)$ for some $t \in \mathbb{R}$ as it may hit a matrix with zero determinant. The formulas for the Riemannian exponential and logarithm mapping at a base point $A \in GL(n)$ are

$$\text{Exp}_A^{GL} : T_A GL(n) \supset B_\varepsilon(0) \rightarrow GL(n), \quad \Delta \mapsto \tilde{A} := A + \Delta, \quad (7.19)$$

¹⁶ “Eulerian motion of a rigid body can be described as motion along geodesics in the group of rotations of three-dimensional euclidean space provided with a left-invariant Riemannian metric. A significant part of Euler’s theory depends only upon this invariance, and therefore can be extended to other groups.” [11, Appendix 2, p. 318].

$$\text{Log}_A^{GL} : GL(n) \rightarrow T_A GL(n), \quad \tilde{A} \mapsto \Delta := (\tilde{A} - A). \quad (7.20)$$

In (7.19), $B_\varepsilon(0)$ denotes a suitably small open neighborhood around $0 \in T_A GL(n) \simeq \mathbb{R}^{n \times n}$ such that $A + \Delta \in GL(n)$ for all $\Delta \in B_\varepsilon(0)$.

The Riemannian exponential for the left-invariant metric on $GL(n)$

The left-invariant metric induces a non-flat geometry on $GL(n)$. Formulae for the covariant derivatives and the corresponding geodesics are derived in [10, Thm. 2.14]. The counterparts w. r. t. the right-invariant metrics can be found in [96]. Given a base point $A \in GL(n)$ and a starting velocity $\Delta = AV \in T_A GL(n) = \text{Ag}l(n)$, the associated geodesic is

$$\Gamma_{A,\Delta} : t \mapsto A \exp_m(tV^T) \exp_m(t(V - V^T)). \quad (7.21)$$

The Riemannian exponential is

$$\begin{aligned} \text{Exp}_M^{GL}(\Delta) &= \Gamma_{A,\Delta}(1) = A \exp_m(V^T) \exp_m(V - V^T) \\ &= A \exp_m((A^{-1}\Delta)^T) \exp_m((A^{-1}\Delta) - (A^{-1}\Delta)^T). \end{aligned} \quad (7.22)$$

The author is not aware of a closed formula for the inverse map, i. e., the Riemannian logarithm for the left-invariant metric; see also the discussion in [96, §4.5]. The thesis [83, §6.2] introduces a Riemannian shooting method for computing the Riemannian logarithm w. r. t. the left-invariant metric.

An important special case

For tangent vectors $\Delta = AV \in T_A GL(n)$ with *normal* $V \in \mathbb{R}^{n \times n}$, i. e., $VV^T = V^T V$, we have that the matrices V^T and $(V - V^T)$ commute. Therefore, according to (7.36), $A \exp_m(V^T) \exp_m(V - V^T) = A \exp_m(V^T + V - V^T) = A \exp_m(V)$ and the Riemannian exponential reduces to

$$\text{Exp}_A^{GL} : T_A GL(n) \cap \{\Delta \mid A^{-1}\Delta \text{ normal}\} \rightarrow GL(n), \Delta \mapsto \tilde{A} = A \exp_m(A^{-1}\Delta).$$

The Riemannian logarithm is

$$\text{Log}_A^{GL} : \mathcal{D}_A \cap \{\tilde{A} \mid A^{-1}\tilde{A} \text{ normal}\} \rightarrow T_A GL(n), \quad \tilde{A} \mapsto \Delta = A \log_m(A^{-1}\tilde{A}),$$

where $\mathcal{D}_A \subset GL(n)$ is a domain such that a suitable branch of the matrix logarithm is well-defined. These expressions are sometimes encountered in the literature as the Riemannian exponential and logarithm mappings. Yet, one should be aware of the fact that they hold under special circumstances.

7.4.2 The orthogonal group

This section is devoted to the *orthogonal group* $O(n) \subset \mathbb{R}^{n \times n}$ of orthogonal n -by- n matrices. In parametric model reduction, such matrices may appear as eigenvector matrices in symmetric EVD problems.

7.4.2.1 Introduction and data representation in numerical schemes

The orthogonal group is $O(n) = \{Q \in \mathbb{R}^{n \times n} \mid QQ^T = I = Q^T Q\}$. The manifold structure of $O(n)$ can be established via Theorem 7.1; see also Example 7.1. The orthogonal group decomposes into two connected components, namely the orthogonal matrices with determinant 1 and the orthogonal matrices with determinant -1 . The former constitute the *special orthogonal group* $SO(n) = \{Q \in O(n) \mid \det(Q) = 1\}$. The orthogonal group is a closed subgroup of the Lie group $GL(n)$ and thus itself a Lie group (Section 7.2.5). The tangent space $T_I O(n)$ at the identity forms the Lie algebra associated with the Lie group $O(n)$. It coincides with the Lie algebra of $SO(n)$ and as such is denoted by $\mathfrak{so}(n) = T_I SO(n) = T_I O(n)$, [46, §3.3, 3.4]. The Lie algebra of $SO(n)$ is precisely the vector space of skew-symmetric matrices, $\mathfrak{so}(n) = \text{skew}(n)$. According to (7.10), the tangent space at an arbitrary location Q is given by the translates (by left-multiplication) of the Lie algebra

$$T_Q O(n) = Q\mathfrak{so}(n) = \{\Delta = QV \in \mathbb{R}^{n \times n} \mid V \in \text{skew}(n)\},$$

which is the same as $\{\Delta \in \mathbb{R}^{n \times n} \mid Q^T \Delta = -\Delta^T Q\}$. The Lie exponential is

$$\exp_m|_{\mathfrak{so}(n)} : \mathfrak{so}(n) \rightarrow SO(n). \quad (7.23)$$

This restriction is a surjective map; see Appendix A. The dimensions of both $T_Q O(n)$ and $O(n)$ are $\frac{1}{2}n(n-1)$.

7.4.2.2 Distances and geodesics

We follow up on the discussion in Section 7.4.1.1. For the orthogonal group, the Euclidean metric and the left-invariant metric coincide. Let $\Delta = QV, \tilde{\Delta} = Q\tilde{V} \in T_Q O(n) = Q\mathfrak{so}(n)$. Then

$$\begin{aligned} \langle \Delta, \tilde{\Delta} \rangle_Q &= \langle Q^{-1}\Delta, Q^{-1}\tilde{\Delta} \rangle_0 = \langle V, \tilde{V} \rangle_0 \\ &= \text{trace}(V^T \tilde{V}) = \text{trace}(V^T Q^T Q \tilde{V}) = \langle \Delta, \tilde{\Delta} \rangle_I. \end{aligned}$$

In fact, the metric is also right-invariant, which makes it a *bi-invariant* metric; see [6, §2]. Bi-invariant metrics are important, because for Lie groups endowed with bi-invariant metrics, the Lie exponential map and the Riemannian exponential map at the identity coincide [6, Thm. 2.27, p. 40].

The Riemannian exponential and logarithm maps on $O(n)$

The Riemannian $O(n)$ -exponential at a base point $Q \in O(n)$ sends a tangent vector $\Delta \in T_Q O(n)$ to the endpoint $\tilde{Q} \in O(n)$ of a geodesic that starts from Q with velocity vector Δ . Therefore, it provides at the same time an expression for the geodesic curves on $O(n)$. A formula for computing the Riemannian $O(n)$ -exponential was derived in [36, §2.2.2]. Given $Q \in O(n)$, we have

$$\text{Exp}_Q^{O(n)} : T_Q O(n) \rightarrow O(n), \quad \Delta \mapsto \tilde{Q} := Q \exp_m(Q^T \Delta). \quad (7.24)$$

This result is also immediate from abstract Lie theory; see [6, Eq. (2.2) & Thm. 2.27].¹⁷ The corresponding Riemannian logarithm on $O(n)$ is

$$\text{Log}_Q^{O(n)} : O(n) \supset \mathcal{D}_Q \rightarrow T_Q O(n), \quad \tilde{Q} \mapsto \Delta := Q \log_m(Q^T \tilde{Q}) \quad (7.25)$$

and is well defined on a neighborhood $\mathcal{D}_Q \subset O(n)$ around Q such that, for all $\tilde{Q} \in \mathcal{D}_Q$, the orthogonal matrix $Q^T \tilde{Q}$ does not feature $\lambda = -1$ as an eigenvalue.

The Riemannian distance between orthogonal matrices

For given $Q, \tilde{Q} \in O(n)$ from the same connected component of $O(n)$, consider the EVD $Q^T \tilde{Q} = \Psi \Lambda \Psi^H$. Because $Q^T \tilde{Q}$ is orthogonal, we have $\Lambda = \text{diag}(e^{i\theta_1}, \dots, e^{i\theta_n})$ and we assume that $\theta_1, \dots, \theta_n \in (-\pi, \pi)$. The Riemannian distance is

$$\text{dist}_{O(n)}(Q, \tilde{Q}) = \|\text{Log}_Q^{O(n)}(\tilde{Q})\|_Q = \|\log_m(\Lambda)\|_F = \left(\sum_{k=1}^n \theta_k^2 \right)^{\frac{1}{2}}.$$

The compact Lie group $SO(n)$ is a geodesically complete Riemannian manifold [6, Hopf–Rinow theorem, p. 31], and each two points of $SO(n)$ can be joined by a minimal geodesic.

7.4.3 The matrix manifold of symmetric positive definite matrices

This section is devoted to the matrix manifold $\text{SPD}(n)$ of real, symmetric positive-definite n -by- n matrices. In model reduction, such matrices appear for example as

¹⁷ The Lie exponential is $\exp_m|_{\mathfrak{so}(n)} : \mathfrak{so}(n) \rightarrow SO(n)$, which is in the case at hand the Riemannian exponential at the identity, $\text{Exp}_I^{SO} = \exp_m|_{\mathfrak{so}(n)}$. This translates to any other location via [6, Eq. (2.2)] as follows: Pick any $Q \in SO(n)$ and consider the mapping “left-multiplication by Q ”, i. e., $L_Q : SO(n) \rightarrow SO(n), P \mapsto QP$. Then the differential is $d(L_Q)_I : T_I SO(n) \rightarrow T_{L_Q(I)} SO(n), V \mapsto \Delta := QV$. Because L_Q is an isometry,

$$Q \text{Exp}_I^{SO}(V) = L_Q(\text{Exp}_I^{SO}(V)) = \text{Exp}_{L_Q(I)}^{SO}(d(L_Q)_I(V)) = \text{Exp}_Q^{SO}(QV),$$

which gives $\text{Exp}_Q^{SO}(QV) = Q \text{Exp}_I^{SO}(V) = Q \exp_m(Q^{-1}\Delta)$ and thus (7.24).

(reduced) system matrices in second-order parametric ODEs. For example, in linear structural or electrical dynamical systems, mass, stiffness and damping matrices are usually in $\text{SPD}(n)$, [9, §4.2]. Moreover, positive definite matrices arise as Gramians of reachable and observable LTI systems in the context of balanced truncation; see Chapter 2 of this volume.

Related is the manifold of positive semi-definite matrices of fixed rank. It is investigated in [23, 96, 68]. An application in model reduction features in [67].

7.4.3.1 Introduction and data representation in numerical schemes

The set

$$\text{SPD}(n) = \{A \in \text{sym}(n) \mid x^T A x > 0 \forall x \in \mathbb{R}^n \setminus \{0\}\}$$

is an open subset of the metric Hilbert space $(\text{sym}(n), \langle \cdot, \cdot \rangle_0)$ of symmetric matrices. As such, it is a differentiable manifold [22, §6]. Moreover, it forms a *convex cone* [37, Example 2, p. 8], [71, §2.3], and can be realized as a quotient $\text{SPD}(n) \simeq GL(n)/O(n)$. The latter is based on the fact that, for $A \in \text{SPD}(n)$, matrix factorizations $A = ZZ^T$ with $Z \in GL(n)$ are invariant under orthogonal transformations $Z \mapsto ZQ$, $Q \in O(n)$, [23, §2, p.3].

Since $\text{SPD}(n)$ is an open subset of the vector space $\text{sym}(n)$, the tangent space is simply

$$T_A \text{SPD}(n) = \text{sym}(n). \quad (7.26)$$

The dimensions of both $T_A \text{SPD}(n)$ and $\text{SPD}(n)$ are $\frac{1}{2}n(n+1)$.

There is a smooth one-to-one correspondence between $\text{sym}(n)$ and $\text{SPD}(n)$. That is, every positive definite matrix can be written as the matrix exponential of a unique symmetric matrix, [39, Lem. 18.7, p. 472]. Put in different words, when restricted to $\text{sym}(n)$, the standard matrix exponential

$$\exp_m : \text{sym}(n) \rightarrow \text{SPD}(n)$$

is a diffeomorphism, its inverse is the standard principal matrix logarithm

$$\log_m : \text{SPD}(n) \rightarrow \text{sym}(n);$$

see also [12, Thm. 2.8]. The group $GL(n)$ acts on $\text{SPD}(n)$ via congruence transformations

$$g_X(A) = X^T A X, \quad X \in GL(n), A \in \text{SPD}(n). \quad (7.27)$$

For additional background on $\text{SPD}(n)$; see [72, 73, 79]. Applications in computer vision are presented in [31, 59].

7.4.3.2 Distances and geodesics

The literature knows a large variety of distance measures on $\text{SPD}(n)$; see [56, Table 3.1, p. 56]. Yet, there are essentially two choices that are associated with inner products on the tangent space of $\text{SPD}(n)$ and thus induce Riemannian metrics on the manifold $\text{SPD}(n)$: the so-called *natural metric* and the *log-Euclidean metric*. Let $A \in \text{SPD}(n)$ and let $\Delta, \tilde{\Delta} \in \text{sym}(n)$ be two tangent vectors.

- The natural metric is

$$\langle \Delta, \tilde{\Delta} \rangle_A = \langle A^{-1/2} \Delta A^{-1/2}, A^{-1/2} \tilde{\Delta} A^{-1/2} \rangle_0 = \text{trace}(A^{-1} \Delta A^{-1} \tilde{\Delta}),$$

see [22, §6, p. 201], [23]. It also goes by the name *trace metric*, [64, §XII.1, p. 322]. In statistical applications, it is usually called the *affine-invariant metric* [70, 80].¹⁸

- The log-Euclidean metric is

$$\langle \Delta, \tilde{\Delta} \rangle_A = \langle D(\log_m)_A(\Delta), D(\log_m)_A(\tilde{\Delta}) \rangle_0;$$

see [12, eq. (3.5)].

For the natural metric, it is more appropriate to consider $\text{sym}(n) = T_I \text{SPD}(n)$ as the tangent space at the identity and the tangent space at an arbitrary location $A \in \text{SPD}(n)$ as $T_A \text{SPD}(n) = A^{1/2}(T_I \text{SPD}(n))A^{1/2}$, which, of course, is nothing but a reparameterization of $\text{sym}(n)$. From this perspective, we have for tangent vectors $\Delta = A^{1/2}VA^{1/2}$, $\tilde{\Delta} = A^{1/2}\tilde{V}A^{1/2}$

$$\langle \Delta, \tilde{\Delta} \rangle_A = \langle V, \tilde{V} \rangle_0.$$

The congruence transformations (7.27) are isometries of $\text{SPD}(n)$ with respect to the natural metric, [64, Thm. XII.1.1, p. 324], [22, Lem. 6.1.1, p. 201]. See also the discussion in [80, §3].

By a standard pullback construction from differential geometry [35, Def. 2.2, Example 2.5], the log-Euclidean metric transfers the inner product $\langle \cdot, \cdot \rangle_0$ on $\text{sym}(n)$ to $\text{SPD}(n)$ via the matrix logarithm $\log_m : \text{SPD}(n) \rightarrow \text{sym}(n)$. In [12, eq. (3.5)], the authors take this construction one step further and use the \exp_m - \log_m -correspondence to define a multiplication that turns $\text{SPD}(n)$ into a Lie group and, eventually, into a vector space. As such, it is a *flat* manifold, i. e. a Riemannian manifold with zero curvature. In this way, the computational challenges that come with dealing with data on nonlinear manifolds are circumvented.

¹⁸ The motivation is as follows: if $y = Ax + v_0$, $A \in GL(n)$ is an affine transformation of a random vector x , then the mean is transformed to $\bar{y} := A\bar{x} + v_0$ and the covariance matrix undergoes a congruence transformation $C_{yy} = E[(y - \bar{y})(y - \bar{y})^T] = AC_{xx}A^T$.

Which metric is to be preferred is problem-dependent; see the various contributions in [92] and [69]. Since the natural metric arises canonical both from the geometric approach, [64, §XII.1], and the matrix-algebraic approach [22, §6] and since staying with the standard matrix multiplication is consistent with the setting of solving dynamical systems in model reduction applications, we restrict the discussion of the Riemannian exponential and logarithm to the geometry that is based on the natural metric.

The SPD(n) exponential

The Riemannian SPD(n)-exponential at a base point $A \in \text{SPD}(n)$ sends a tangent vector Δ to the endpoint $\tilde{A} \in \text{SPD}(n)$ of a geodesic that starts from A with velocity vector Δ . Therefore, it provides at the same time an expression for the geodesic curves on SPD(n) with respect to the natural metric. Formulae for computing the SPD(n)-exponential can be found in [23], [80]. The reader preferring a matrix-analytic approach is referred to [22, §6].

Algorithm 7.5: Riemannian SPD(n)-exponential

Input: base point $A \in \text{SPD}(n)$, tangent vector $\Delta \in T_A \text{SPD}(n) = \text{sym}(n)$

Output: $\tilde{A} := \text{Exp}_A^{\text{SPD}}(\Delta) = A^{\frac{1}{2}} \exp_m(A^{-\frac{1}{2}} \Delta A^{-\frac{1}{2}}) A^{\frac{1}{2}}$.

Here, $A^{\frac{1}{2}}$ denotes the matrix square root of A ; see Appendix A.

The SPD(n) logarithm

The Riemannian SPD(n)-logarithm at a base point $A \in \text{SPD}(n)$ finds for another point $\tilde{A} \in \text{SPD}(n)$ an SPD(n)-tangent vector Δ such that the geodesic that starts from A with velocity Δ reaches \tilde{A} after an arc length of $\|\Delta\|_A = \sqrt{\langle \Delta, \Delta \rangle_A}$. Therefore, it provides for two given data points $A, \tilde{A} \in \text{SPD}(n)$

- a solution to the geodesic endpoint problem: a geodesic that starts from A and ends at \tilde{A} ;
- the Riemannian distance between the given points A, \tilde{A} .

Formulas for computing the SPD(n)-logarithm can be found in [23], [80].

Algorithm 7.6: Riemannian SPD(n)-logarithm.

Input: base point $A \in \text{SPD}(n)$, location $\tilde{A} \in \text{SPD}(n)$

Output: $\Delta := \text{Log}_A^{\text{SPD}}(\tilde{A}) = A^{\frac{1}{2}} \log_m(A^{-\frac{1}{2}} \tilde{A} A^{-\frac{1}{2}}) A^{\frac{1}{2}}$.

Both Algorithms 7.5 and 7.6 require one to compute the spectral decomposition of n -by- n -matrices. The computational effort is $\mathcal{O}(n^3)$. In the context of parametric model reduction, the Riemannian exponential and logarithm maps are usually required for

reduced matrix operators [9]. If n denotes the dimension of the full state vectors and $r \ll n$ denotes the dimension of the reduced state vectors, then matrix exponentials for r -by- r -matrices are required, so that the computational effort reduces to $\mathcal{O}(r^3)$.

7.4.4 The Stiefel manifold

This section is devoted to the *Stiefel manifold* $St(n, r) \subset \mathbb{R}^{n \times r}$ of rectangular column-orthogonal n -by- r matrices, $r \leq n$. Points $U \in St(n, r)$ may be considered as orthonormal bases of cardinality r , or r -frames in \mathbb{R}^n . In model reduction, such matrices appear as orthogonal coordinate systems for low-order ansatz spaces that usually stem from a proper orthogonal decomposition or a singular value decomposition of given input solution data. Modeling data on the Stiefel manifold corresponds to data processing for orthonormal bases and thus allows for example for interpolation/parameterization of POD subspace bases. The most important use case in model reduction is where the Stiefel matrices are tall and skinny, i. e., $r \ll n$. Interpolation problems on the Stiefel manifold have not yet been considered in the model reduction context. The reference [62] discusses interpolation of Stiefel data; however, using quasi-geodesics rather than geodesics. Reference [103] includes numerical experiments for interpolating orthogonal frames on the Stiefel manifold that relies on the canonical Riemannian Stiefel logarithm [83, 102].

7.4.4.1 Introduction and data representation in numerical schemes

The *Stiefel manifold* is the compact, homogeneous matrix manifold of column-orthogonal rectangular matrices

$$St(n, r) := \{U \in \mathbb{R}^{n \times r} \mid U^T U = I_r\}.$$

The manifold structure can be directly established via Theorem 7.1 in a similar way as in Example 7.1. An alternative approach is via Example 7.4, where $St(n, r)$ is identified with the quotient space $St(n, r) \cong O(n)/(I_r \times O(n-r))$ under actions of the closed subgroup $I_r \times O(n-r) := \left\{ \begin{pmatrix} I_r & 0 \\ 0 & R \end{pmatrix} \mid R \in O(n-r) \right\} \leq O(n)$. Two square orthogonal matrices in $O(n)$ are identified as the same point on $St(n, r)$, if their first r columns coincide; see [36, §2.4].

For any matrix representative $U \in St(n, r)$, the tangent space of $St(n, r)$ at U is represented by

$$T_U St(n, r) = \{\Delta \in \mathbb{R}^{n \times r} \mid U^T \Delta = -\Delta^T U\} \subset \mathbb{R}^{n \times r}.$$

Every tangent vector $\Delta \in T_U St(n, r)$ may be written as

$$\Delta = UA + (I - UU^T)T, \quad A \in \mathbb{R}^{r \times r} \text{ skew}, \quad T \in \mathbb{R}^{n \times r} \text{ arbitrary}, \quad (7.28)$$

$$\Delta = UA + U^\perp B, \quad A \in \mathbb{R}^{r \times r} \text{ skew}, \quad B \in \mathbb{R}^{(n-r) \times r} \text{ arbitrary}, \quad (7.29)$$

where, in the latter case, $U^\perp \in St(n, n-r)$ is such that $(U, U^\perp) \in O(n)$ is a square orthogonal matrix. The dimension of both $T_U St(n, r)$ and $St(n, r)$ is $nr - \frac{1}{2}r(r+1)$. For additional background and applications, see [3, 21, 29, 36, 52, 95].

7.4.4.2 Distances and geodesics

Let $U \in St(n, r)$ be a point and let $\Delta = UA + U^\perp B$, $\tilde{\Delta} = U\tilde{A} + U^\perp\tilde{B} \in T_U St(n, r)$ be tangent vectors. There are two standard metrics on the Stiefel manifold.

- The *Euclidean metric* on $T_U St(n, r)$ is the one inherited from the ambient $\mathbb{R}^{n \times r}$:

$$\langle \Delta, \tilde{\Delta} \rangle_0 = \text{trace}(\Delta^T \tilde{\Delta}) = \text{trace} A^T \tilde{A} + \text{trace} B^T \tilde{B}.$$

- The *canonical metric* on $T_U St(n, r)$

$$\langle \Delta, \tilde{\Delta} \rangle_U = \text{trace} \left(\Delta^T \left(I - \frac{1}{2} U U^T \right) \tilde{\Delta} \right) = \frac{1}{2} \text{trace} A^T \tilde{A} + \text{trace} B^T \tilde{B}$$

is derived from the quotient representation $St(n, r) = O(n)/(I_r \times O(n-r))$ of the Stiefel manifold.

The canonical metric counts the *independent* coordinates¹⁹ of a tangent vector equally, when measuring the length $\sqrt{\langle \Delta, \Delta \rangle_U}$ of a tangent vector $\Delta = UA + U^\perp B$, while the Euclidean metric disregards the skew-symmetry of A [36, §2.4]. Recall that different metrics entail different measures for the lengths of curves and thus different formulae for geodesics.

The Stiefel exponential

The Riemannian Stiefel exponential at a base point $U \in St(n, r)$ sends a Stiefel tangent vector Δ to the endpoint $\tilde{U} \in St(n, r)$ of a geodesic that starts from U with velocity vector Δ . Therefore, it provides at the same time an expression for geodesic curves on $St(n, r)$.

A closed-form expression for the Stiefel exponential w. r. t. Euclidean metric is included in [36, §2.2.2],

$$\tilde{U} = \text{Exp}_U^{St}(\Delta) = (U, \Delta) \exp_m \left(\begin{pmatrix} U^T \Delta & -\Delta^T \Delta \\ I_p & U^T \Delta \end{pmatrix} \right) \begin{pmatrix} I_p \\ 0 \end{pmatrix} \exp_m(-U^T \Delta).$$

In [53], an alternative formula is derived that features only matrix exponentials of skew-symmetric matrices. An efficient algorithm for computing the Stiefel exponential w. r. t. the canonical metric was derived in [36, §2.4.2]:

¹⁹ That is, the upper triangular entries of the skew-symmetric A and the entries of B of $\Delta = UA + U^\perp B$.

Algorithm 7.7: Stiefel exponential [36].**Input:** base point $U \in St(n, r)$, tangent vector $\Delta \in T_U St(n, r)$

- 1: $A := U^T \Delta$ {horizontal component, skew}
- 2: $QR := \Delta - UA$ {(thin) qr-decomp. of normal component of Δ .}
- 3: $\begin{pmatrix} A & -R^T \\ R & 0 \end{pmatrix} = T \Lambda T^H \in \mathbb{R}^{2r \times 2r}$ {EVD of skew-symmetric matrix}
- 4: $\begin{pmatrix} M \\ N \end{pmatrix} := T \exp_m(\Lambda) T^H \begin{pmatrix} I_r \\ \mathbf{0} \end{pmatrix} \in \mathbb{R}^{2r \times r}$

Output: $\tilde{U} := \text{Exp}_U^{St}(\Delta) = UM + QN \in St(n, r)$

In applications, where $\text{Exp}_U^{St}(\mu\Delta)$ needs to be evaluated for various parameters μ as in the example of Section 7.3.4, steps 1.–3. should be computed a priori (offline). Apart from elementary matrix multiplications, the algorithm requires one to compute the standard matrix exponential of a skew-symmetric matrix. This, however, is for a $2r$ -by- $2r$ -matrix and does not scale in the dimension n . With the usual assumption of model reduction that $n \gg p$, the computational effort is $\mathcal{O}(nr^2)$.

The Stiefel logarithm

The Riemannian Stiefel logarithm at a base point $U \in St(n, r)$ finds for another point $\tilde{U} \in St(n, r)$ a Stiefel tangent vector Δ such that the geodesic that starts from U with velocity Δ reaches \tilde{U} after an arc length of $\|\Delta\|_U = \sqrt{\langle \Delta, \Delta \rangle_U}$. Therefore, it provides for two given data points $U, \tilde{U} \in St(n, r)$

- a solution to the geodesic endpoint problem: a geodesic that starts from U and ends at \tilde{U} ;
- the Riemannian distance between the given points U, \tilde{U} .

An efficient algorithm for computing the Stiefel logarithm w. r. t. the canonical metric was derived in [102].

Algorithm 7.8: Stiefel logarithm [102].**Input:** base point $U \in St(n, r)$, $\tilde{U} \in St(n, r)$ ‘close’ to base point, $\tau > 0$ convergence threshold

- 1: $M := U^T \tilde{U} \in \mathbb{R}^{r \times r}$
- 2: $QN := \tilde{U} - UM \in \mathbb{R}^{n \times r}$ {(thin) qr-decomp. of normal component of \tilde{U} }
- 3: $V_0 := \begin{pmatrix} M & X_0 \\ N & Y_0 \end{pmatrix} \in O(2r)$ {compute orth. completion of the block $\begin{pmatrix} M \\ N \end{pmatrix}$ }
- 4: **for** $k = 0, 1, 2, \dots$ **do**
- 5: $\begin{pmatrix} A_k & -B_k^T \\ B_k & C_k \end{pmatrix} := \log_m(V_k)$ {matrix log of orth. matrix}
- 6: **if** $\|C_k\|_2 \leq \tau$ **then**
- 7: **break**

```

8:  end if
9:   $\Phi_k := \exp_m(-C_k)$  {matrix exp of skew matrix}
10:  $V_{k+1} := V_k W_k$ , where  $W_k := \begin{pmatrix} I_r & 0 \\ 0 & \Phi_k \end{pmatrix}$ 
11: end for
Output:  $\Delta := \text{Log}_U^{\text{St}}(\tilde{U}) = UA_k + QB_k \in T_U \text{St}(n, r)$ 

```

The analysis in [102] shows that the algorithm is guaranteed to converge if the input data points U, \tilde{U} are at most a Euclidean distance of $d = \|U - \tilde{U}\|_2 \leq 0.09$ apart. In this case, the algorithm exhibits a linear rate of convergence that depends on d but is smaller than $\frac{1}{2}$. In practice, the algorithm seems to converge, whenever the initial V_0 is such that its standard matrix logarithm $\log_m(V_0)$ is well-defined. Note that two points on $\text{St}(n, r)$ can at most be a Euclidean distance of 2 away from each other.

Apart from elementary matrix multiplications, the algorithm requires one to compute the standard matrix logarithm of an orthogonal $2r$ -by- $2r$ -matrix and the standard matrix exponential of a skew-symmetric r -by- r -matrix at every iteration k . Yet, these operations are independent of the dimension n . With the usual assumption of model reduction that $r \ll n$, the computational effort is $\mathcal{O}(nr^2)$.

For the Stiefel manifold equipped with the Euclidean metric, methods for calculating the Stiefel logarithm are introduced in [25].

7.4.5 The Grassmann manifold

This section is devoted to the *Grassmann manifold* $Gr(n, r)$ of r -dimensional subspaces of \mathbb{R}^n for $r \leq n$. Every point $\mathcal{U} \in Gr(n, r)$, i. e., every subspace may be represented by selecting a basis $\{u^1, \dots, u^r\}$ with $\text{ran}(u^1, \dots, u^r) = \mathcal{U}$. In numerical schemes, we work exclusively with orthonormal bases. In this way, points \mathcal{U} on the Grassmann manifold are to be represented by points $U \in \text{St}(n, r)$ on the Stiefel manifold via $\mathcal{U} = \text{ran}(U)$. For details and theoretical background, see the references [2, 3, 36]. Modeling data on the Grassmann manifold corresponds to data processing for subspaces and thus allows, for example, for the interpolation/parameterization of POD subspaces see [19, Chapter 5], [19, Chapter 9]. The most important use case in model reduction is where the subspaces are of low dimension when compared to the surrounding state space, i. e., $n \gg p$. Grassmann interpolation problems in the context of projection-based parametric model reduction are considered in [8, 76, 100, 87]. Subspaces also feature in Krylov subspace approaches; see [20, Chapter 3].

7.4.5.1 Introduction and data representation in numerical schemes

The set of all r -dimensional subspaces $\mathcal{U} \subset \mathbb{R}^n$ forms the *Grassmann manifold*

$$Gr(n, r) := \{\mathcal{U} \subset \mathbb{R}^n \mid \mathcal{U} \text{ subspace, } \dim(\mathcal{U}) = r\}.$$

The Grassmann manifold is a quotient of $O(n)$ under the action of the Lie subgroup $O(r) \times O(n-r) = \left\{ \begin{pmatrix} S & 0 \\ 0 & R \end{pmatrix} \mid S \in O(r), R \in O(n-r) \right\} \leq O(n)$. Two matrices $Q, \tilde{Q} \in O(n)$ are in the same $(O(r) \times O(n-r))$ -orbit, if and only if the first r columns of Q and \tilde{Q} span the same subspace and the tailing $n-r$ columns span the corresponding orthogonal complement subspace. Theorem 7.5 applies and shows that $Gr(n, r) = O(n)/(O(r) \times O(n-r))$ is a homogeneous manifold.

Alternatively, the Grassmann manifold can be realized as a quotient manifold of the Stiefel manifold with the help of Theorem 7.4,

$$Gr(n, r) = St(n, r)/O(r) = \{[U] \mid U \in St(n, r)\}, \quad (7.30)$$

where the $O(r)$ -orbits are $[U] = \{UR \mid R \in O(r)\}$. A matrix $U \in St(n, r)$ is called a *matrix representative* of a subspace $\mathcal{U} \in Gr(n, r)$, if $\mathcal{U} = \text{ran}(U)$. The orbit $[U]$ and the subspace $\mathcal{U} = \text{ran}(U)$ are to be considered as the same object. For any matrix representative $U \in St(n, r)$ of $\mathcal{U} \in Gr(n, r)$ the tangent space of $Gr(n, r)$ at \mathcal{U} is represented by

$$T_{\mathcal{U}}Gr(n, r) = \{\Delta \in \mathbb{R}^{n \times r} \mid U^T \Delta = 0\} \subset \mathbb{R}^{n \times r}.$$

Every tangent vector $\Delta \in T_{\mathcal{U}}Gr(n, r)$ may be written as

$$\Delta = (I - UU^T)T, \quad T \in \mathbb{R}^{n \times r} \text{ arbitrary, or,} \quad (7.31)$$

$$\Delta = U^\perp B, \quad B \in \mathbb{R}^{(n-r) \times r} \text{ arbitrary,} \quad (7.32)$$

where in the latter case, $U^\perp \in St(n, n-r)$ is such that $(U, U^\perp) \in O(n)$ is a square orthogonal matrix. The dimension of both $T_{\mathcal{U}}Gr(n, r)$ and $Gr(n, r)$ is $nr - r^2$.

7.4.5.2 Distances and geodesics

A metric on $T_{\mathcal{U}}Gr(n, r)$ can be obtained via making use of the fact that the Grassmannian is a quotient of the Stiefel manifold. Alternatively, one can restrict the standard inner matrix product $\langle A, B \rangle_0 = \text{trace}(A^T B)$ to the Grassmann tangent space. In the case of the Grassmannian, the two approaches lead to the same metric

$$\langle \Delta, \tilde{\Delta} \rangle_{\mathcal{U}} = \text{trace}(\Delta^T \tilde{\Delta}) = \langle \Delta, \tilde{\Delta} \rangle_0;$$

see [36, §2.5].

The Grassmann exponential

The Riemannian Grassmann exponential at a base point $\mathcal{U} \in Gr(n, r)$ sends a Grassmann tangent vector Δ to the endpoint $\tilde{\mathcal{U}} \in Gr(n, r)$ of a geodesic that starts from \mathcal{U} with velocity vector Δ . Therefore, it provides at the same time an expression for the geodesic curves on $Gr(n, r)$. An efficient algorithm for computing the Grassmann exponential was derived in [36, §2.5.1]:

Algorithm 7.9: Grassmann exponential [36].**Input:** base point $\mathcal{U} = [U] \in Gr(n, r)$, where $U \in St(n, r)$, tangent vector $\Delta \in T_{\mathcal{U}}Gr(n, r)$

- 1: $Q\Sigma V^T \stackrel{\text{SVD}}{:=} \Delta$, with $Q \in St(n, r)$ {(thin) SVD of tangent vector}
- 2: $\tilde{U} := UV \cos(\Sigma)V^T + Q \sin(\Sigma)V^T$ {cos and sin act only on diag. entries.}

Output: $\tilde{\mathcal{U}} := \text{Exp}_{\mathcal{U}}^{Gr}(\Delta) = [\tilde{U}] \in Gr(n, r)$.

Apart from elementary matrix multiplications, the algorithm requires one to compute the singular value decomposition of an n -by- r -matrix. The computational effort is $\mathcal{O}(nr^2)$.

The Grassmann logarithm

The Riemannian Grassmann logarithm at a base point $\mathcal{U} \in Gr(n, r)$ finds for another point $\tilde{\mathcal{U}} \in Gr(n, r)$ a Grassmann tangent vector Δ such that the geodesic that starts from \mathcal{U} with velocity Δ reaches $\tilde{\mathcal{U}}$ after an arc length of $\|\Delta\|_{\mathcal{U}} = \sqrt{g_{\mathcal{U}}^C(\Delta, \Delta)}$. Therefore, it provides for two given data points $\mathcal{U}, \tilde{\mathcal{U}} \in Gr(n, r)$

- a solution to the geodesic endpoint problem: a geodesic that starts from \mathcal{U} and ends at $\tilde{\mathcal{U}}$;
- the Riemannian distance between the given points $\mathcal{U}, \tilde{\mathcal{U}}$.

An algorithm for computing the Grassmann logarithm is stated implicitly in [2, §3.8, p. 210]. The reference [40] features expressions for the Grassmann exponential and the corresponding logarithm that formally work with Grassmann representatives in $SO(n)/(SO(r) \times SO(n-r))$ but also keep the computational effort $\mathcal{O}(nr^2)$. Reference [82, §4.3] gives the corresponding mappings after identifying subspaces with orthoprojectors; see also [16].

Algorithm 7.10: Grassmann Logarithm.**Input:** base point $\mathcal{U} = [U] \in Gr(n, r)$ with $U \in St(n, r)$, $\tilde{\mathcal{U}} = [\tilde{U}] \in Gr(n, r)$ with $\tilde{U} \in St(n, r)$.

- 1: $M := U^T \tilde{U}$
- 2: $L := (I - UU^T)\tilde{U}M^{-1} = \tilde{U}M^{-1} - U$
- 3: $Q\Sigma V^T \stackrel{\text{SVD}}{:=} L$ {(thin) SVD }
- 4: $\Delta := Q \arctan(\Sigma)V^T$ {arctan acts only on diag. entries.}

Output: $\Delta = \text{Log}_{\mathcal{U}}^{Gr}(\tilde{\mathcal{U}}) \in T_{\mathcal{U}}Gr(n, r)$

The composition $\text{Exp}_{[U]}^{Gr} \circ \text{Log}_{[U]}^{Gr}$ is the identity on $Gr(n, r)$, wherever it is defined. Yet, on the level of the actual matrix representatives, the operation

$$(\text{Exp}_{[U]}^{Gr} \circ \text{Log}_{[U]}^{Gr})([\tilde{U}_{in}]) = [\tilde{U}_{out}]$$

produces a matrix $\tilde{U}_{out} \neq \tilde{U}_{in}$. Directly recovering the input matrix can be achieved via a Procrustes-type preprocessing step, where \tilde{U} is replaced with $\tilde{U}_* := \tilde{U}\Phi$, $\Phi = \arg \min_{\Phi \in O(r)} \|U - \tilde{U}\Phi\|$. This leads to the following.

Algorithm 7.11: Grassmann Logarithm: modified version.²⁰

Input: base point $\mathcal{U} = [U] \in Gr(n, r)$ with $U \in St(n, r)$, $\tilde{\mathcal{U}} = [\tilde{U}] \in Gr(n, r)$ with $\tilde{U} \in St(n, r)$.

$$1: \Psi SR^T \stackrel{\text{SVD}}{:=} \tilde{U}^T U$$

$$2: \tilde{U}_* := \tilde{U}(\Psi R^T) \quad \{\text{‘Transition to Procrustes representative’}\}$$

$$3: L := (I - UU^T)\tilde{U}_*$$

$$4: Q\Sigma V^T \stackrel{\text{SVD}}{:=} L \quad \{(\text{thin}) \text{ SVD}\}$$

$$5: \Delta := Q \arcsin(\Sigma) V^T \quad \{\arcsin \text{ acts only on diagonal entries.}\}$$

Output: $\Delta = \text{Log}_{\mathcal{U}}^{Gr}(\tilde{\mathcal{U}}) \in T_{\mathcal{U}}Gr(n, r)$

An additional advantage of the modified Grassmann logarithm is that the matrix inversion $M^{-1} = (U^T \tilde{U})^{-1}$ is avoided. In fact, it is replaced by the SVD $\Psi SR^T = \tilde{U}^T U$ that is used to solve the Procrustes problem $\min_{\Phi \in O(r)} \|U - \tilde{U}\Phi\|$. The SVD exists also if $U^T \tilde{U}$ does not have full rank.

Distances between subspaces

The Riemannian logarithm provides the distance between two subspaces $\mathcal{U} = [U], \tilde{\mathcal{U}} = [\tilde{U}] \in Gr(n, r)$ as follows: First, compute $\Delta = \text{Log}_{\mathcal{U}}^{Gr}(\tilde{\mathcal{U}})$, then compute $\|\Delta\|_{\mathcal{U}} = \text{dist}_{Gr}(\mathcal{U}, \tilde{\mathcal{U}})$. In practice, however, this boils down to computing the singular values of the matrix $M = U^T \tilde{U}$, which can be seen as follows. By Algorithm 7.11, $\|\Delta\|_{\mathcal{U}}^2 = \text{trace}(\Delta^T \Delta) = \sum_{k=1}^p \arcsin(\sigma_k)^2$, where the σ_k ’s are the singular values of $L = (I - UU^T)\tilde{U}_*$. These match precisely the square roots of the eigenvalues of $L^T L$. Using the SVD of the square matrix $\tilde{U}^T U = \Psi SR^T$ as in steps 1&2 of Algorithm 7.11, the eigenvalues of $L^T L$ can be read off from

$$L^T L = \tilde{U}_*^T (I - UU^T) \tilde{U}_* = I - RS^2 R^T = R(I - S^2) R^T,$$

so that $\sigma_k^2 = 1 - s_k^2$, when consistently ordered. As a consequence, $s_k = \sqrt{1 - \sigma_k^2} = \cos(\arcsin(\sigma_k))$, which implies

$$\text{dist}_{Gr}(\mathcal{U}, \tilde{\mathcal{U}}) = \left(\sum_{k=1}^p \arcsin(\sigma_k)^2 \right)^{\frac{1}{2}} = \left(\sum_{k=1}^p \arccos(s_k)^2 \right)^{\frac{1}{2}}, \quad (7.33)$$

where $\sigma_1, \dots, \sigma_r$ and s_1, \dots, s_r are the singular values of L and $\tilde{U}^T U$, respectively.

The numerical linear algebra literature knows a variety of distance measures for subspaces. Essentially, all of them are based on the principal angles [36, §2.5.1, §4.3].

²⁰ This is an original contribution to this chapter; for a detailed discussion, see [17, Section 5.2].

The *principal angles* (or canonical angles) $\theta_1, \dots, \theta_r \in [0, \frac{\pi}{2}]$ between two subspaces $[U], [\tilde{U}] \in Gr(n, r)$ are defined recursively by

$$\cos(\theta_k) := u_k^T v_k := \max_{\substack{u \in [U], \|u\| = 1 \\ u \perp u_1, \dots, u_{k-1}}} \max_{\substack{v \in [\tilde{U}], \|v\| = 1 \\ v \perp v_1, \dots, v_{k-1}}} u^T v.$$

The principal angles can be computed via $\theta_k := \arccos(s_k) \in [0, \frac{\pi}{2}]$, where s_k is the k th singular value of $U^T \tilde{U} \in \mathbb{R}^{r \times r}$ [42, §6.4.3]. Hence, the Riemannian subspace distance (7.33) expressed in terms of the principal angles is precisely

$$\text{dist}([U], [\tilde{U}]) := \|\Theta\|_2, \quad \Theta = (\theta_1, \dots, \theta_r) \in \mathbb{R}^r. \quad (7.34)$$

In particular, (7.34) shows that any two points on $Gr(n, r)$ can be connected by a geodesic of length at most $\frac{\sqrt{r}}{2}\pi$; see also [98, Thm 8(b)].

7.5 Conclusion

Interpolation of structured matrices is a viable building block in parametric model reduction approaches. In order to preserve the characteristic features, the matrix sets in question are considered as geometric entities, so-called differentiable manifolds. In this chapter, we exposed how concepts from Riemannian geometry apply in designing manifold counterparts to Euclidean interpolation algorithms. As examples, the generic approach of interpolating in Riemannian normal coordinates, the quasi-linear, geodesic interpolation method and interpolation via the Riemannian center of mass are discussed. All the aforementioned methods share many of their constituent algorithmic units and acquaintance with these units allows one to adapt and modify the established approaches as needed or to design new ones. In this spirit, for a selection of matrix manifolds that feature frequently in practical applications, namely, the general linear group, the orthogonal group, the set of symmetric positive definite matrices, the Stiefel manifold and the Grassmann manifold, we have gathered the essential geometric concepts and formulas necessary to conduct Riemannian interpolation.

Appendix A

The matrix exponential and logarithm

The standard matrix exponential and matrix logarithm are defined via the power series

$$\exp_m(X) := \sum_{j=0}^{\infty} \frac{X^j}{j!}, \quad \log_m(X) := \sum_{j=1}^{\infty} (-1)^{j+1} \frac{(X - I)^j}{j}. \quad (7.35)$$

For $X \in \mathbb{R}^{n \times n}$, $\exp_m(X)$ is invertible with inverse $\exp_m(-X)$. The following restrictions of the exponential map are important:

$$\exp_m|_{\text{sym}(n)} : \text{sym}(n) \rightarrow \text{SPD}(n), \quad \exp_m|_{\text{skew}(n)} : \text{skew}(n) \rightarrow \text{SO}(n).$$

The former is a diffeomorphism [79, Thm. 2.8], the latter is a differentiable, surjective map [41, §. 3.11, Thm. 9]. For additional properties and efficient methods for numerical computation, see [50, §10, 11].

A few properties of the exponential function for real or complex numbers carry over to the matrix exponential. However, since matrices do not commute, the standard exponential law is replaced by

$$\begin{aligned} \exp_m(Z(X, Y)) &= \exp_m(X) \exp_m(Y), \\ Z(X, Y) &= X + Y + \frac{1}{2}[X, Y] \\ &\quad + \frac{1}{12}([X, [X, Y]] + [Y, [Y, X]]) - \frac{1}{24}[Y, [X, [X, Y]]] \dots, \end{aligned} \tag{7.36}$$

where $[X, Y] = XY - YX$ is the commutator bracket, or Lie bracket. This is Dynkin's formula for the Baker–Campbell–Hausdorff series; see [86, §1.3, p. 22]. From a theoretical point of view, it is important that all terms in this series can be expressed in terms of the Lie bracket. A special case is

$$\exp_m(X + Y) = \exp_m(X) \exp_m(Y), \quad \text{if } [X, Y] = 0.$$

Matrix square roots and the polar decomposition

Every $S \in \text{SPD}(n)$ has a unique matrix square root in $\text{SPD}(n)$, i. e., a matrix denoted by $S^{\frac{1}{2}}$ with the property $S^{\frac{1}{2}}S^{\frac{1}{2}} = S$. This square root can be obtained via an EVD $S = Q\Lambda Q^T$ by setting

$$S^{\frac{1}{2}} := Q\sqrt{\Lambda}Q^T,$$

where $Q \in O(n)$, $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ and $\lambda_i > 0$ are the eigenvalues of S . Every $A \in GL(n)$ can be uniquely decomposed into an orthogonal matrix times a symmetric positive definite matrix,

$$A = QP = Q \exp_m(X), \quad Q \in O(n), P \in \text{SPD}(n), X \in \text{sym}(n).$$

The polar factors can be constructed via taking the square root of the assuredly positive definite matrix $A^T A$ and subsequently setting $P := (A^T A)^{\frac{1}{2}}$ and $Q := AP^{-1}$. Because the restriction of \exp_m to the symmetric matrices is a diffeomorphism onto $\text{SPD}(n)$, there is a unique $X \in \text{sym}(n)$ with $P = \exp_m(X)$. For details, see [46, Thm. 2.18].

The Procrustes problem

Let $A, B \in \mathbb{R}^{n \times r}$. The Procrustes problem aims at finding an orthogonal transformation $R^* \in O(r)$ such that R^* is the minimizer of

$$\min_{R \in O(r)} \|A - BR\|_F.$$

The optimal R^* is $R^* = UV^T$, where $B^T A \stackrel{\text{SVD}}{=} U\Sigma V^T \in \mathbb{R}^{r \times r}$; see [42].

Bibliography

- [1] P.-A. Absil, P.-Y. Gousenbourger, P. Striowski, and B. Wirth. Differentiable piecewise-Bézier surfaces on Riemannian manifolds. *SIAM J. Imaging Sci.*, 9(4):1788–1828, 2016.
- [2] P.-A. Absil, R. Mahony, and R. Sepulchre. Riemannian geometry of Grassmann manifolds with a view on algorithmic computation. *Acta Appl. Math.*, 80(2):199–220, 2004.
- [3] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, New Jersey, 2008.
- [4] B. Afsari, R. Tron, and R. Vidal. On the convergence of gradient descent for finding the Riemannian center of mass. *SIAM J. Control Optim.*, 51(3):2230–2260, 2013.
- [5] D. Alekseevsky, A. Kriegl, P. W. Michor, and M. Losik. Choosing roots of polynomials smoothly. *Isr. J. Math.*, 105(1):203–233, 1998.
- [6] M. M. Alexandrino and R. G. Bettiol. *Lie Groups and Geometric Aspects of Isometric Actions*. Springer, Cham, 2015.
- [7] D. Amsallem. Interpolation on Manifolds of CFD-based Fluid and Finite Element-based Structural Reduced-order Models for On-line Aeroelastic Prediction. PhD thesis, Stanford University 2010.
- [8] D. Amsallem and C. Farhat. Interpolation method for adapting reduced-order models and application to aeroelasticity. *AIAA J.*, 46(7):1803–1813, 2008.
- [9] D. Amsallem and C. Farhat. An online method for interpolating linear parametric reduced-order models. *SIAM J. Sci. Comput.*, 33(5):2169–2198, 2011.
- [10] E. Andruchow, G. Laroitonda, L. Recht, and A. Varela. The left invariant metric in the general linear group. *J. Geom. Phys.*, 86:241–257, 2014.
- [11] V. I. Arnol'd. *Mathematical Methods of Classical Mechanics. Graduate Texts in Mathematics*. Springer, New York, 1997.
- [12] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Geometric means in a novel vector space structure on symmetric positive-definite matrices. *SIAM J. Matrix Anal. Appl.*, 29(1):328–347, 2006.
- [13] P. Astrid, S. Weiland, K. Willcox, and T. Backx. Missing points estimation in models described by proper orthogonal decomposition. *IEEE Trans. Autom. Control*, 53(10):2237–2251, 2008.
- [14] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. An “empirical interpolation” method: Application to efficient reduced-basis discretization of partial differential equations. *C. R. Math. Acad. Sci. I*, 339:667–672, 2004.
- [15] R. H. Bartels, J. C. Beatty, and B. A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling. Morgan Kaufmann Series in Comp.* Elsevier Science, 1995.
- [16] E. Batzies, K. Hüper, L. Machado, and F. Silva Leite. Geometric mean and geodesic regression on Grassmannians. *Linear Algebra Appl.*, 466:83–101, 2015.

- [17] T. Bendokat, R. Zimmermann, and P.-A. P.-A.. A Grassmann Manifold Handbook: Basic Geometry and Computational Aspects, 2020. arXiv preprint arXiv:2011.13699.
- [18] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Rev.*, 57(4):483–531, 2015.
- [19] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. H. A. Schilders and L. M. Silveira. *Model Order Reduction. Volume 2: Snapshot-Based Methods and Algorithms*. De Gruyter, Berlin, 2020.
- [20] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. H. A. Schilders and L. M. Silveira. *Model Order Reduction. Volume 3: Applications*. De Gruyter, Berlin, 2020.
- [21] A. V. Bernstein and A. P. Kuleshov. Tangent bundle manifold learning via Grassmann & Stiefel eigenmaps, 2012. arXiv preprint arXiv:1212.6031.
- [22] R. Bhatia. *Positive Definite Matrices. Princeton Series in Applied Mathematics*. Princeton University Press, Princeton, New Jersey, 2007.
- [23] S. Bonnabel and R. Sepulchre. Riemannian metric and geometric mean for positive semidefinite matrices of fixed rank. *SIAM J. Matrix Anal. Appl.*, 31(3):1055–1070, 2009.
- [24] N. Boumal and P.-A. Absil. A discrete regression method on manifolds and its application to data on $SO(n)$. In *IFAC Proceedings Volumes, 18th IFAC World Congress*, volume 44(1), pages 2284–2289, 2011.
- [25] D. Bryner. Endpoint geodesics on the Stiefel manifold embedded in Euclidean space. *SIAM J. Matrix Anal. Appl.*, 38(4):1139–1159, 2017.
- [26] M. D. Buhmann. *Radial Basis Functions. Cambridge Monographs on Applied and Computational Mathematics*, volume 12. Cambridge University Press, Cambridge, UK, 2003.
- [27] M. Camarinha, F. Silva Leite, and P. Crouch. On the geometry of Riemannian cubic polynomials. *Differ. Geom. Appl.*, 15(2):107–135, 2001.
- [28] K. Carlberg and C. Farhat. A low-cost, goal-oriented ‘compact proper orthogonal decomposition’ basis for model reduction of state systems. *Int. J. Numer. Methods Eng.*, 86(3):381–402, 2011.
- [29] R. Chakraborty and B. C. Vemuri. Statistics on the (compact) Stiefel manifold: Theory and applications. *Ann. Stat.*, 47(1):415–438, 2019.
- [30] S. Chaturantabut and D. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM J. Sci. Comput.*, 32(5):2737–2764, 2010.
- [31] A. Cherian and S. Sra. Positive definite matrices: Data representation and applications in computer vision. In H. Q. Minh and V. Murino, editors, *Algorithmic Advances in Riemannian Geometry and Applications: For Machine Learning, Computer Vision, Statistics, and Optimization*, pages 93–114. Springer, Cham, 2016.
- [32] Y. Choi, D. Amsallem, and C. Farhat. Gradient-based constrained optimization using a database of linear reduced-order models arXiv, arXiv:1506.07849v1, pages 1–28, 2015.
- [33] P. Crouch and F. Silva Leite. The dynamic interpolation problem: On Riemannian manifolds, Lie groups, and symmetric spaces. *J. Dyn. Control Syst.*, 1(2):177–202, 1995.
- [34] J. Degroote, J. Vierendeels, and K. Willcox. Interpolation among reduced-order matrices to obtain parameterized models for design, optimization and probabilistic analysis. *Int. J. Numer. Methods Fluids*, 63(2):207–230, 2010.
- [35] M. P. do Carmo. *Riemannian Geometry. Mathematics: Theory & Applications*. Birkhäuser, Boston, 1992.
- [36] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, 20(2):303–353, 1998.
- [37] J. Faraut and A. Koranyi. *Analysis on Symmetric Cones. Oxford Mathematical Monographs*. Oxford University Press, New York, 1994.
- [38] T. Franz, R. Zimmermann, S. Görtz, and N. Karcher. Interpolation-based reduced-order modeling for steady transonic flows via manifold learning. *Int. J. Comput. Fluid Mech.*, 228:106–121, 2014. Special Issue on Reduced Order Modeling.

- [39] J. H. Gallier. *Geometric methods and applications: for computer science and engineering. Texts in Applied Mathematics*. Springer, New York, 2011.
- [40] K. A. Gallivan, A. Srivastava, X. Liu, and P. Van Dooren. Efficient algorithms for inferences on Grassmann manifolds. In *IEEE Workshop on Statistical Signal Processing*, pages 315–318, 2003.
- [41] R. Godement and U. Ray. *Introduction to the Theory of Lie Groups. Universitext*. Springer, 2017.
- [42] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, 4th edition, 2013.
- [43] P.-Y. Gousenbourger, E. Massart, and P.-A. Absil. Data fitting on manifolds with composite Bézier-like curves and blended cubic splines. *J. Math. Imaging Vis.*, online:1–27, 2018.
- [44] P. Grohs. Quasi-interpolation in Riemannian manifolds. *IMA J. Numer. Anal.*, 33(3):849–874, 2013.
- [45] B. Haasdonk and M. Ohlberger. Efficient reduced models and a-posteriori error estimation for parametrized dynamical systems by offline/online decomposition. *Math. Comput. Model. Dyn. Syst.*, 17(2):145–161, 2011.
- [46] B. C. Hall. *Lie Groups, Lie Algebras, and representations: An elementary introduction. Springer Graduate texts in Mathematics*. Springer, New York – Berlin – Heidelberg, 2nd edition, 2015.
- [47] A. Hay, J. Borggaard, I. Akhtar, and D. Pelletier. Reduced-order models for parameter dependent geometries based on shape sensitivity analysis. *J. Comput. Phys.*, 229(4):1327–1352, 2010.
- [48] A. Hay, J. T. Borggaard, and D. Pelletier. Local improvements to reduced-order models using sensitivity analysis of the proper orthogonal decomposition. *J. Fluid Mech.*, 629:41–72, 2009.
- [49] U. Helmke and J. B. Moore. *Optimization and Dynamical Systems. Communications & Control Engineering*. Springer, London, 1994.
- [50] N. J. Higham. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.
- [51] M. Hinze and S. Volkwein. Proper orthogonal decomposition surrogate models for nonlinear dynamical systems: Error estimates and suboptimal control. In *Dimension Reduction of Large-Scale Systems. Lecture Notes in Computational Science and Engineering*, volume 45, pages 261–306. Springer, Berlin–Heidelberg, 2005.
- [52] K. Hüper, M. Kleinstaubert, and F. Silva Leite. Rolling Stiefel manifolds. *Int. J. Syst. Sci.*, 39(9):881–887, 2008.
- [53] K. Hüper and F. Ullrich. Real Stiefel manifolds: An extrinsic point of view. In *2018 13th APCA International Conference on Automatic Control and Soft Computing (CONTROLO)*, pages 13–18, 2018.
- [54] K. Ito and S. S. Ravindran. A reduced-order method for simulation and control of fluid flows. *J. Comput. Phys.*, 143:403–425, 1998.
- [55] J. Jakubiak, F. S. Leite, and R. Rodrigues. A two-step algorithm of smooth spline generation on Riemannian manifolds. *J. Comput. Appl. Math.*, 194:177–191, 2006.
- [56] S. Jayasumana, R. Hartley, and M. Salzmann. Kernels on Riemannian manifolds. In A. Srivastava and P. K. Turaga, editors, *Riemannian computing in computer vision*, pages 45–67. Springer, 2015.
- [57] H. Le, K. R. Kim, and I. L. Dryden. Smoothing splines on Riemannian manifolds, with applications to 3D shape space, 2018. arXiv:1801.04978v2.
- [58] H. Karcher. Riemannian center of mass and mollifier smoothing. *Commun. Pure Appl. Math.*, 30(5):509–541, 1977.
- [59] H. J. Kim, N. Adluru, B. B. Bendlin, S. C. Johnson, B. C. Vemuri, and V. Singh. Canonical correlation analysis on SPD(n) manifolds. In A. Srivastava and P. K. Turaga, editors, *Riemannian computing in computer vision*, pages 69–100. Springer, 2015.

- [60] S. Kobayashi and K. Nomizu. *Foundations of Differential Geometry. Interscience Tracts in Pure and Applied Mathematics, no. 15*, volume I. John Wiley & Sons, New York – London – Sidney, 1963.
- [61] S. Kobayashi and K. Nomizu. *Foundations of Differential Geometry. Interscience Tracts in Pure and Applied Mathematics, no. 15*, volume II. John Wiley & Sons, New York – London – Sidney, 1969.
- [62] K. A. Krakowski, L. Machado, F. Silva Leite, and J. Batista. Solving interpolation problems on Stiefel manifolds using quasi-geodesics. In *Pré-Publicações do Departamento de Matemática*, number 15–36, Universidade de Coimbra, 2015.
- [63] W. Kühnel. *Differential Geometry: Curves – Surfaces – Manifolds. Student mathematical library*. American Mathematical Society, 2006.
- [64] S. Lang. *Fundamentals of Differential Geometry. Graduate Texts in Mathematics*. Springer, New York, 2001.
- [65] J. M. Lee. *Riemannian Manifolds: an Introduction to Curvature*. Springer, New York – Berlin – Heidelberg, 1997.
- [66] J. M. Lee. *Introduction to Smooth Manifolds. Graduate Texts in Mathematics*. Springer, New York, 2012.
- [67] E. Massart, P.-Y. Gousenbourger, T. S. Nguyen, T. Stykel, and P.-A. Absil. Interpolation on the manifold of fixed-rank positive-semidefinite matrices for parametric model order reduction: preliminary results. Technical Report UCL-INMA-2018.13, University of Louvain, 2018.
- [68] E. Massart and P.-A. Absil. Quotient geometry with simple geodesics for the manifold of fixed-rank positive-semidefinite matrices. *SIAM J. Matrix Anal. Appl.*, 41:171, 2020.
- [69] H. Q. Minh and V. Murino. *Algorithmic Advances in Riemannian Geometry and Applications: For Machine Learning, Computer Vision, Statistics, and Optimization. Advances in Computer Vision and Pattern Recognition*. Springer, Cham, 2016.
- [70] H. Q. Minh and V. Murino. From covariance matrices to covariance operators: Data representation from finite to infinite-dimensional settings. In H. Q. Minh and V. Murino, editors, *Algorithmic Advances in Riemannian Geometry and Applications: For Machine Learning, Computer Vision, Statistics, and Optimization*, pages 115–143. Springer, Cham, 2016.
- [71] M. Moakher. A differential geometric approach to the geometric mean of symmetric positive-definite matrices. *SIAM J. Matrix Anal. Appl.*, 26(3):735–747, 2005.
- [72] M. Moakher and P. G. Batchelor. Symmetric positive-definite matrices: From geometry to applications and visualization. In J. Weickert and H. Hagen, editors, *Visualization and Processing of Tensor Fields, Mathematics and Visualization*, pages 285–298. Springer, Berlin – Heidelberg, 2006.
- [73] M. Moakher and M. Zéraï. The Riemannian geometry of the space of positive-definite matrices and its applications to the regularization of positive-definite matrix-valued data. *J. Math. Imaging Vis.*, 40(2):171–187, 2011.
- [74] M. Morzyński, W. Stankiewicz, B. R. Noack, R. King, F. Thiele, and G. Tadmor. Continuous mode interpolation for control-oriented models of fluid flow. In R. King, editor, *Active Flow Control*, pages 260–278. Springer, Berlin – Heidelberg, 2007.
- [75] E. Nava-Yazdani and K. Polthier. De Casteljau’s algorithm on manifolds. *Comput. Aided Geom. Des.*, 30(7):722–732, 2013.
- [76] T. S. Nguyen. A real time procedure for affinely dependent parametric model order reduction using interpolation on Grassmann manifolds. *Int. J. Numer. Methods Eng.*, 93(8):818–833, 2013.
- [77] L. Noakes, G. Heinzinger, and B. Paden. Cubic splines on curved spaces. *IMA J. Math. Control Inf.*, 6(4):465–473, 1989.

- [78] H. Panzer, J. Mohring, R. Eid, and B. Lohmann. Parametric model order reduction by matrix interpolation. *Automatisierungstechnik*, 58(8):475–484, 2010.
- [79] X. Pennec. Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements. *J. Math. Imaging Vis.*, 25(1):127, 2006.
- [80] X. Pennec, P. Fillard, and N. Ayache. A Riemannian framework for tensor computing. *Int. J. Comput. Vis.*, 66(1):41–66, 2006.
- [81] T. Popiel and L. Noakes. Bézier curves and C2 interpolation in Riemannian manifolds. *J. Approx. Theory*, 148(2):111–127, 2007.
- [82] I. U. Rahman, I. Drori, V. C. Stodden, D. L. Donoho, and P. Schröder. Multiscale representations for manifold-valued data. *SIAM J. Multiscale Model. Simul.*, 4(4):1201–1232, 2005.
- [83] Q. Rentmeesters. Algorithms for data fitting on some common homogeneous spaces. PhD thesis, Université Catholique de Louvain, Louvain, Belgium 2013.
- [84] Q. Rentmeesters and P.-A. Absil. Algorithms comparison for Karcher mean computation of rotation matrices and diffusion tensors. In *Proceedings of the 19th European Signal Processing Conference (EUSIPCO 2011)*, Barcelona, Spain, Aug. 29–Sept. 2, 2011.
- [85] M. Rewienski and J. White. Model order reduction for nonlinear dynamical systems based on trajectory piecewise-linear approximations. *Linear Algebra Appl.*, 415(2–3):426–454, 2006. Special Issue on Order Reduction of Large-Scale Systems.
- [86] W. Rossmann. *Lie Groups: An Introduction Through Linear Groups*. Oxford Graduate Texts in Mathematics. Oxford University Press, 2006.
- [87] T. S. Nguyen, P.-Y. Gousenbourger, E. Massart, and P.-A. Absil. Online balanced truncation for linear time-varying systems using continuously differentiable interpolation on grassmann manifold. Technical Report UCL-INMA-2019.01, University of Louvain, 2019.
- [88] C. Samir, P.-A. Absil, A. Srivastava, and E. Klassen. A gradient-descent method for curve fitting on Riemannian manifolds. *Found. Comput. Math.*, 12(1):49–73, 2012.
- [89] C. Samir and I. Adouani. C1 interpolating Bézier path on Riemannian manifolds, with applications to 3D shape space. *Appl. Math. Comput.*, 348:371–384, 2019.
- [90] O. Sander. Interpolation und Simulation mit nichtlinearen Daten. *GAMM Rundbr.*, 1:6–12, 2015.
- [91] O. Sander. Geodesic finite elements of higher order. *IMA J. Numer. Anal.*, 36(1):238–266, 2016.
- [92] A. Srivastava and P. K. Turaga. *Riemannian computing in computer vision*. Springer, 2015.
- [93] F. Steinke, M. Hein, J. Peters, and B. Schoelkopf. *Manifold-valued Thin-Plate Splines with Applications in Computer Graphics*. Computer Graphics Forum, 2008.
- [94] G. Tadmor, O. Lehmann, B. R. Noack, and M. Morzyński. Galerkin models enhancements for flow control. In B. R. Noack, M. Morzyński and G. Tadmor, editors, *Reduced-Order Modelling for Flow Control*, pages 151–252. Springer, Vienna, 2011.
- [95] P. K. Turaga, A. Veeraraghavan, and R. Chellappa. Statistical analysis on Stiefel and Grassmann manifolds with applications in computer vision. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.
- [96] B. Vandereycken, P.-A. Absil, and S. Vandewalle. A Riemannian geometry with complete geodesics for the set of positive semidefinite matrices of fixed rank. *IMA J. Numer. Anal.*, 33(2):481–514, 2012.
- [97] G. Weickum, M. S. Eldred, and K. Maute. Multi-point extended reduced order modeling for design optimization and uncertainty analysis. In *Proceedings of the 2nd AIAA Multidisciplinary Design Optimization Specialist Conference, number AIAA 2006-2145*, Newport, RI, May 1–4, 2006.
- [98] Y.-C. Wong. Differential geometry of Grassmann manifolds. *Proc. Natl. Acad. Sci. USA*, 57:589–594, 1967.

- [99] R. Zimmermann. Gradient-enhanced surrogate modeling based on proper orthogonal decomposition. *J. Comput. Appl. Math.*, 237(1):403–418, 2013.
- [100] R. Zimmermann. A locally parametrized reduced order model for the linear frequency domain approach to time-accurate computational fluid dynamics. *SIAM J. Sci. Comput.*, 36(3):B508–B537, 2014.
- [101] R. Zimmermann. Local parametrization of subspaces on matrix manifolds via derivative information. In B. Karasözen, M. Manguoğlu, M. Tezer-Sezgin, S. Göktepe and Ö. Uğur, editors, *Numerical Mathematics and Advanced Applications ENUMATH 2015*, pages 379–387. Springer, Cham, 2016.
- [102] R. Zimmermann. A matrix-algebraic algorithm for the Riemannian logarithm on the Stiefel manifold under the canonical metric. *SIAM J. Matrix Anal. Appl.*, 38(2):322–342, 2017.
- [103] R. Zimmermann. Hermite interpolation and data processing errors on Riemannian matrix manifolds. *SIAM J. Sci. Comput.*, 42(5):A2593–A2619, 2020.
<https://doi.org/10.1137/19M1282878>.
- [104] R. Zimmermann and K. Debrabant. Parametric model reduction via interpolating orthonormal bases. In F. A. Radu, K. Kumar, I. Berre, D. N. Nordbotten and I. S. Pop, editors, *Numerical Mathematics and Advanced Applications ENUMATH 2017*. Springer, Cham, 2018.
- [105] R. Zimmermann and K. Willcox. An accelerated greedy missing point estimation procedure. *SIAM J. Sci. Comput.*, 38(5):A2827–A2850, 2016.

8 Vector fitting

Abstract: We introduce the Vector Fitting algorithm for the creation of reduced order models from the sampled response of a linear time-invariant system. This data-driven approach to reduction is particularly useful when the system under modeling is known only through experimental measurements. The theory behind Vector Fitting is presented for single- and multiple-input systems, together with numerical details, pseudo-codes, and an open-source implementation [75]. We discuss how the reduced model can be made stable and converted to a variety of forms for use in virtually any modeling context. Finally, we survey recent extensions of the Vector Fitting algorithm geared towards time domain, parametric and distributed systems modeling.

Keywords: vector fitting, data-driven modeling, macromodeling, stability, passivity

MSC 2010: 65D10, 37M05, 93C80, 94C99


8.1 Introduction and motivation

The Vector Fitting (VF) algorithm [42, 35] is one of the most successful techniques for creating reduced order models for linear systems starting from *samples* of their response. Samples may originate from an experimental measurement or from a prior numerical simulation. This need arises in many practical scenarios, and we cite two examples.

A biomedical engineer may need a linear model describing blood flow in a portion of the human cardiovascular system, and have simultaneous in-vivo measurements of pressure and flow rate at the inlets and outlets of the region of interest. With a *data-driven* algorithm for model order reduction, such as VF, the reduced model can be created directly from experimental observations.

As a second example, we consider an electronic engineer that needs a model for a radio-frequency amplifier or an antenna, to be used for design purposes. If the device is provided by a third party, a measurement may be the only way to characterize the system. High-frequency measurements are typically performed in the frequency domain, and return the impedance or admittance seen between the ports of the device, measured at various frequencies ω_k . From these samples, VF can create a reduced model which can be represented as a set of differential equations or as an equivalent

Piero Triverio, Edward S. Rogers Sr. Department of Electrical & Computer Engineering and Institute of Biomedical Engineering, University of Toronto, 10 King's College Rd., M5S 3G4 Toronto, ON, Canada, e-mail: piero.triverio@utoronto.ca

Open Access. © 2021 Piero Triverio, published by De Gruyter.  This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

circuit for use in subsequent simulation, including those performed in the time domain.

The main advantage of a data-driven approach to reduced order modeling is that only samples of the system response are required. This feature makes data-driven reduction a natural choice when experimental measurements are readily available. Furthermore, data-driven reduction can also be applied when samples originate from a numerical simulation based on first-principles equations, such as Maxwell's equations for electromagnetic phenomena. Although in this second scenario one could technically use *equation-driven* methods, the available simulator may not allow the user to export the discretized first-principles equations for reduction. This is the case for most commercial simulators used by industry. The main disadvantage of data-driven reduction is that it offers less physical insight into the system under modeling, since it leads to a “black-box” reduced model. By starting from a first-principles model, equation-driven methods are typically better in this regard, since they can provide to the user more information about which features of the original model were retained, and which features were discarded.

8.2 The Sanathanan–Koerner algorithm

8.2.1 Problem statement

We assume that the system under modeling is linear and time-invariant, with input $u(t) \in \mathbb{R}^{\bar{m}}$ and output $y(t) \in \mathbb{R}^{\bar{q}}$. Because of linearity and time-invariance, the output can be written as a convolution

$$y(t) = \int_{-\infty}^{+\infty} h(t - \tau)u(\tau) d\tau \quad (8.1)$$

between input $u(t)$ and the impulse response $h(t) \in \mathbb{R}^{\bar{q} \times \bar{m}}$ of the system, which is unknown. Applying the Laplace transform to both sides of (8.1), we get

$$Y(s) = H(s)U(s), \quad (8.2)$$

where $s = \sigma + j\omega$ is complex frequency. In (8.2), $U(s) \in \mathbb{C}^{\bar{m}}$ and $Y(s) \in \mathbb{C}^{\bar{q}}$ are the Laplace transforms of $u(t)$ and $y(t)$, respectively, and $H(s) \in \mathbb{C}^{\bar{q} \times \bar{m}}$ is the transfer function of the system. The VF algorithm solves the following problem. Given \bar{k} measurements of the transfer function

$$H_k = H(j\omega_k) \quad k = 1, \dots, \bar{k}, \quad (8.3)$$

determine a rational function $\tilde{H}(s)$ that approximates the given measurements

$$\tilde{H}(j\omega_k) \simeq H_k \quad \forall k = 1, \dots, \bar{k}. \quad (8.4)$$

In VF, $\tilde{H}(s)$ is chosen to be a rational function. Rational functions are universal approximators, and can therefore approximate a wide range of functions with arbitrary accuracy. Moreover, since the transfer function of lumped systems is rational by construction, this is a natural choice to model dynamical systems. Finally, rational functions can be represented as a state-space system, a poles-residue form, a set of differential equations, an equivalent electric circuit and many other forms. This flexibility facilitates the integration of the reduced model into existing software for computational mathematics and system simulation.

8.2.2 The Levy and Sanathanan–Koerner algorithms

The first attempts to solve (8.4) numerically date back at least to the 1950s, with the work of Levy, Sanathanan and Koerner among others. We briefly summarize their work since the VF algorithm can be better understood from that perspective. For simplicity, we initially consider the case of a system with a single input and a single output ($\bar{m} = \bar{q} = 1$). The general case will be discussed in Section 8.3.5.

In order to solve the approximation problem (8.4), we must first choose a suitable parametric form for $\tilde{H}(s)$, which is the model that we want to estimate from the given samples. The most natural choice is to let $\tilde{H}(s)$ be the ratio of two polynomials

$$\tilde{H}(s) = \frac{n(s)}{d(s)} = \frac{\sum_{n=0}^{\bar{n}} a_n s^n}{\sum_{n=0}^{\bar{n}} b_n s^n}, \quad (8.5)$$

where $a_n, b_n \in \mathbb{R}$ are unknowns, and \bar{n} is the order of the desired model. Since one coefficient can be normalized, we let $b_{\bar{n}} = 1$. In (8.5), we chose the same degree \bar{n} for numerator and denominator. This choice is appropriate for transfer functions that are known to be bounded when $s \rightarrow \infty$. This is the case of the scattering coefficients used to model electronic devices at high frequencies, as in the example in Section 8.3.7. In other applications, the transfer function of the system under modeling may grow polynomially as s increases. This is the case, for example, of the impedance and admittance coefficients of passive electrical circuits, which can grow linearly with s . As an example, one can consider the impedance $Z(s) = sL$ of an inductor. In such cases, the degree of the numerator of (8.5) should be increased to $\bar{n} + 1$. This change leads to minor modifications to the algorithms presented in this chapter, which will not be discussed here, but can be found in [35].

After choosing the form of model (8.5), we have to determine its coefficients a_n and b_n in order to satisfy (8.4), minimizing a suitable norm between samples H_k and model response $\tilde{H}(j\omega_k)$. We choose the l_2 norm, and aim to minimize

$$e^2 = \frac{1}{\bar{k}} \sum_{k=1}^{\bar{k}} |H_k - \tilde{H}(j\omega_k)|^2. \quad (8.6)$$

Minimizing (8.6) is a nonlinear least squares problem, due to the unknowns b_n in the denominator. Although nonlinear optimization algorithms can be directly applied to (8.6), experience shows that they can be quite time consuming and prone to local minima. A different approach is preferred, where (8.6) is linearized into a *linear* least squares problem, which can be solved efficiently and robustly with the QR decomposition [28].

We first rewrite (8.6) as

$$e^2 = \frac{1}{\bar{k}} \sum_{k=1}^{\bar{k}} \left| \frac{H_k \sum_{n=0}^{\bar{n}} b_n (j\omega_k)^n - \sum_{n=0}^{\bar{n}} a_n (j\omega_k)^n}{\sum_{n=0}^{\bar{n}} b_n (j\omega_k)^n} \right|^2. \quad (8.7)$$

Levy proposed to linearize (8.7) by simply neglecting the denominator, and minimize [54]

$$(e_L)^2 = \frac{1}{\bar{k}} \sum_{k=1}^{\bar{k}} \left| H_k \sum_{n=0}^{\bar{n}} b_n (j\omega_k)^n - \sum_{n=0}^{\bar{n}} a_n (j\omega_k)^n \right|^2, \quad (8.8)$$

which ultimately boils down to solving a system of linear equations in least squares sense. Unfortunately, this simple trick typically fails to provide an accurate solution of (8.4). Indeed, error functionals (8.7) and (8.8) are equivalent only when $\sum_{n=0}^{\bar{n}} b_n (j\omega)^n$ is approximately constant, which is rarely the case. Furthermore, the monomial terms $(j\omega)^n$ in (8.8) will result in Vandermonde matrices in the least-squares problem to be solved, which are ill-conditioned [28].

To overcome this issue, Sanathanan and Koerner proposed an iterative process to improve the quality of the solution [69]. In the first iteration ($i = 1$), the Levy functional (8.8) is minimized, providing a first estimate of the model coefficients that we denote as $a_n^{(1)}$ and $b_n^{(1)}$. In successive iterations ($i \geq 2$), the following linearization of (8.7) is minimized:

$$(e_{SK}^{(i)})^2 = \frac{1}{\bar{k}} \sum_{k=1}^{\bar{k}} \left| \frac{H_k \sum_{n=0}^{\bar{n}} b_n^{(i)} (j\omega_k)^n - \sum_{n=0}^{\bar{n}} a_n^{(i)} (j\omega_k)^n}{\sum_{n=0}^{\bar{n}} b_n^{(i-1)} (j\omega_k)^n} \right|^2, \quad (8.9)$$

leading to a new estimate of model coefficients $a_n^{(i)}$ and $b_n^{(i)}$. We can see that, in (8.9), the coefficients $b_n^{(i-1)}$ from the previous iteration are used to approximate the “nonlinear” term in (8.7). Since the unknowns $a_n^{(i)}$ and $b_n^{(i)}$ appear only in the numerator, the Sanathanan–Koerner method only requires the solution of linear least squares problems. If the iterative process converges, $b_n^{(i-1)} \rightarrow b_n^{(i)}$, and (8.9) becomes equivalent to (8.7). We can see that the term $\sum_{n=0}^{\bar{n}} b_n^{(i-1)} (j\omega_k)^n$ in the denominator of (8.9) acts as a frequency-dependent weight of the least squares problem. This weight aims to progressively remove the bias introduced in the linearization of (8.6). For discrete-time systems, the counterpart of the Sanathanan–Koerner method was proposed by Steiglitz and McBride [73].

8.2.3 Numerical issues of the Sanathanan–Koerner method

The work of Sanathanan and Koerner solves (8.4) accurately using only linear least squares problems. Unfortunately, this method can still suffer from severe numerical issues when applied to realistic problems, where the required model order \tilde{n} may be quite large and frequency ω may span several orders of magnitude. For example, VF is extensively used in integrated circuit design to model the interconnect network that distributes signals and power across the circuit. In this application, the frequency range of interest typically extends from a few MHz to tens of GHz, for about four decades of variation. The numerical issues associated with the Sanathanan–Koerner method arise from two factors:

- (a) The error (8.9) contains high powers of frequency $(\omega_k)^n$, leading to very poor conditioning. Specifically, the matrix of the least-squares problem to be solved will contain Vandermonde blocks [28], which are known to be ill-conditioned even for relatively modest values of \tilde{n} .
- (b) The weighting term $\sum_{n=0}^{\tilde{n}} b_n^{(i-1)} (j\omega_k)^n$ in the denominator of (8.9) typically exhibits large variations over $[\omega_1, \omega_k]$, which further degrade the conditioning of the least squares problem.

8.3 The Vector Fitting algorithm

The VF algorithm, conceived by Gustavsen and Semlyen [42], addresses both problems with a simple yet brilliant solution.

8.3.1 A new basis function and implicit weighting

In order to avoid the ill-conditioning arising from the s^n terms in (8.5), VF replaces those terms with partial fractions. The numerator and denominator of $\tilde{H}(s)$ are written as

$$n^{(i)} = c_0^{(i)} + \sum_{n=1}^{\tilde{n}} \frac{c_n^{(i)}}{s - p_n^{(0)}}, \quad (8.10)$$

$$d^{(i)} = 1 + \sum_{n=1}^{\tilde{n}} \frac{d_n^{(i)}}{s - p_n^{(0)}}, \quad (8.11)$$

where $p_n^{(0)} \in \mathbb{C}$ are a set of initial poles, whose choice will be discussed later on. We see that, without loss of generality, the constant term in (8.11) has been normalized to one. In comparison to the monomial basis functions s^n used by the Sanathanan–Koerner iteration, which vary wildly as s increases, partial fractions $\frac{1}{s - p_n^{(0)}}$ have more contained variations over frequency if the poles $p_n^{(0)}$ are chosen appropriately [43], as

will be discussed in Section 8.3.2. This feature leads to better conditioning, especially if the poles $p_n^{(0)}$ are distinct and well separated.

The introduction of partial fractions is also crucial to address the second issue discussed in Section 8.2.3, and perform an implicit weighting of (8.9). To understand how VF achieves this, we first give a different interpretation to linearized error (8.9). In terms of (8.10) and (8.11), error (8.9) can be expressed as

$$(e_{SK}^{(i)})^2 = \frac{1}{k} \sum_{k=1}^{\bar{k}} \left| H_k \frac{d^{(i)}(j\omega_k)}{d^{(i-1)}(j\omega_k)} - \frac{n^{(i)}(j\omega_k)}{d^{(i-1)}(j\omega_k)} \right|^2. \quad (8.12)$$

We can see that this expression involves the two new quantities

$$w^{(i)}(s) = \frac{d^{(i)}(s)}{d^{(i-1)}(s)} \quad (8.13)$$

and

$$\bar{H}^{(i)}(s) = \frac{n^{(i)}(s)}{d^{(i-1)}(s)}. \quad (8.14)$$

The function $\bar{H}^{(i)}(s)$ can be interpreted as the model transfer function estimated at iteration i by the minimization of (8.12). Notably, this transfer function is made by the numerator $n^{(i)}(s)$ from the current iteration (to be found), and by the denominator $d^{(i-1)}(s)$ from the previous iteration (already known). This approximation arises from the linearization of the error function, since it indeed avoids the presence of unknowns in the denominator. The function $w^{(i)}(s)$ can be interpreted as a frequency-dependent weight which multiplies the given samples H_k . This weighting function has two purposes:

- providing a new estimate of denominator $d^{(i)}(s)$, and
- compensating for the approximation introduced by fixing the denominator of $\bar{H}^{(i)}(s)$ to the previous iteration value. Indeed, weight $w^{(i)}(s)$ depends on the ratio between new denominator estimate $d^{(i)}(s)$ and previous estimate $d^{(i-1)}(s)$.

Next, we derive alternative expressions for $w^{(i)}(s)$ and $\bar{H}^{(i)}(s)$, which pave the way for an implicit weighting of (8.12). Substituting (8.10) and (8.11) into (8.13) we can derive the following chain of equalities:

$$w^{(i)}(s) = \frac{1 + \sum_{n=1}^{\bar{n}} \frac{d_n^{(i)}}{s - p_n^{(0)}}}{1 + \sum_{n=1}^{\bar{n}} \frac{d_n^{(i-1)}}{s - p_n^{(0)}}} = \frac{\prod (s - p_n^{(i)})}{\prod (s - p_n^{(i-1)})} = 1 + \sum_{n=1}^{\bar{n}} \frac{w_n^{(i)}}{s - p_n^{(i-1)}}, \quad (8.15)$$

where $\prod = \prod_{n=1}^{\bar{n}}$. In (8.15), $p_n^{(i)}$ are the zeros of $d^{(i)}(s)$, and therefore the poles of $\bar{H}^{(i+1)}(s)$. From the second expression in (8.15), we see that $w^{(i)}(s)$ is the ratio of two rational functions with the same poles $p_n^{(0)}$. By factorizing their respective numerators and denominators, as in the third expression, we observe that those common poles

can be eliminated. Finally, we express $w^{(i)}$ in terms of a new set of poles $p_n^{(i-1)}$ that change at every iteration, as in the last expression in (8.15). The same manipulation can be performed on $\tilde{H}^{(i)}(s)$, leading to

$$\tilde{H}^{(i)}(s) = \frac{c_0^{(i)} + \sum_{n=1}^{\bar{n}} \frac{c_n^{(i)}}{s - p_n^{(0)}}}{1 + \sum_{n=1}^{\bar{n}} \frac{d_n^{(i-1)}}{s - p_n^{(0)}}} = r_0^{(i)} + \sum_{n=1}^{\bar{n}} \frac{r_n^{(i)}}{s - p_n^{(i-1)}}. \quad (8.16)$$

Substituting (8.15) and (8.16) into (8.12), we obtain [42, 35]

$$(e_{SK}^{(i)})^2 = \frac{1}{\bar{k}} \sum_{k=1}^{\bar{k}} \left| H_k \left(1 + \sum_{n=1}^{\bar{n}} \frac{w_n^{(i)}}{j\omega_k - p_n^{(i-1)}} \right) - \left(r_0^{(i)} + \sum_{n=1}^{\bar{n}} \frac{r_n^{(i)}}{j\omega_k - p_n^{(i-1)}} \right) \right|^2, \quad (8.17)$$

which is the actual error function used in VF to fit the model to the given samples. The main difference between (8.17) and (8.9) is how the linearized error is iteratively weighted to progressively converge to (8.6). In (8.12), the weight $\frac{1}{d^{(i-1)}(j\omega_k)}$ is applied *explicitly*, which degrades numerical conditioning. In (8.17), instead, the same weight is applied *implicitly* by relocating the poles $p_n^{(i-1)}$ at each iteration.

Once (8.17) has been minimized, the updated poles $p_n^{(i)}$ for the next iteration can be found as the zeros of $d^{(i)}(s)$, as one can see from the third expression in (8.15). It can be shown that such zeros can be calculated as the eigenvalues of a matrix [42]

$$\{p_n^{(i)}\} = \text{eig}(A^{(i-1)} - b_w(c_w^{(i)})^T), \quad (8.18)$$

with $A^{(i-1)} = \text{diag}\{p_1^{(i-1)}, \dots, p_{\bar{n}}^{(i-1)}\}$ being a diagonal matrix formed by poles $p_n^{(i-1)}$. In (8.18) b_w is a $\bar{n} \times 1$ vector of ones, and $(c_w^{(i)})^T = [w_1^{(i)}, \dots, w_{\bar{n}}^{(i)}]$. Upon convergence, $p_n^{(i-1)} \rightarrow p_n^{(i)}$, and they become the poles of the obtained model $H^{(i)}(s)$. When this happens, $w^{(i)} \rightarrow 1$ as we can see from the third expression in (8.15), and the linearized error (8.12) tends to (8.6), as desired.

8.3.2 The Vector Fitting algorithm

We are now ready to present the complete VF algorithm [42, 35], with a pseudo-code implementation available in Algorithm 8.1. The first step is to choose the order \bar{n} of the desired model. This choice will be discussed in Section 8.3.11.1. Next, we set the initial poles $p_n^{(0)}$ of the basis functions in (8.16) and (8.15). Numerical tests [42] showed that a linear distribution of poles with small and negative real part over the bandwidth spanned by samples H_k leads to the best conditioning of the least squares problems to be solved. We assume \bar{n} even, and frequency values ω_k sorted in ascending order. If $\omega_1 = 0$, the initial poles can be set as [35]

$$p_n^{(0)} = \begin{cases} (-\alpha + j) \frac{\omega_k}{\bar{n}/2} n & \text{for } n = 1, \dots, \bar{n}/2 \\ (p_{\bar{n}-\bar{n}/2}^{(0)})^* & \text{for } n = \bar{n}/2 + 1, \dots, \bar{n} \end{cases} \quad (8.19)$$

Algorithm 8.1: Vector Fitting.**Require:** response samples H_k , corresponding frequencies ω_k ($k = 1, \dots, \bar{k}$)**Require:** desired model order \bar{n} **Require:** maximum number of iterations i_{\max}

```

1: set initial poles  $p_n^{(0)}$  according to (8.19) or (8.20).
2:  $i \leftarrow 1$ 
3: while  $i \leq i_{\max}$  do
4:   Solve (8.21) or (8.38) in least squares sense
5:   Compute the new poles estimate  $p_n^{(i)}$  with (8.18)
6:   Enforce poles stability with (8.71), if desired ▷ Stability enforcement
7:   if (8.27) is true then ▷ First convergence test
8:     Solve (8.29) or (8.40) in least squares sense ▷ Tentative final fitting
9:     Compute fitting error  $e$  with (8.6) or (8.34)
10:    if  $e \leq \varepsilon_H$  then ▷ Second convergence test
11:       $\tilde{H}(s) = \tilde{H}^{(i+1)}(s)$ 
12:      return Success!
13:    end if
14:  end if
15:   $i \leftarrow i + 1$ 
16: end while
17: return Failure: maximum number of iterations reached.

```

where $*$ denotes the complex conjugate and α is typically set to 0.01. This rule generates $\bar{n}/2$ pairs of complex conjugate poles, linearly distributed over the frequency range $[0, \omega_{\bar{k}}]$ spanned by samples H_k . The imaginary part of the poles is set to be quite larger than the real part, since this makes the partial fraction basis functions well distinct from each other, which improves numerical conditioning.

When $\omega_1 \neq 0$, the distribution (8.19) can be modified as [35]

$$p_n^{(0)} = \begin{cases} (-\alpha + j)[\omega_1 + \frac{\omega_{\bar{k}} - \omega_1}{\bar{n}/2 - 1}(n - 1)] & \text{for } n = 1, \dots, \bar{n}/2, \\ (p_{n - \bar{n}/2}^{(0)})^* & \text{for } n = \bar{n}/2 + 1, \dots, \bar{n}, \end{cases} \quad (8.20)$$

to linearly spread the poles between $\omega = \omega_1$ and $\omega = \omega_{\bar{k}}$. Rules (8.19) and (8.20) work well for most cases, since the choice of initial poles is typically not critical for VF convergence. When the frequency range of interest spans several decades, and the system frequency response exhibits significant behavior in multiple decades, initial poles can be distributed logarithmically for optimal results [35].

The core of the VF algorithm is an iterative minimization of (8.17), which begins with $i = 1$. Minimizing (8.17) is equivalent to solving, in the least-squares sense, the

system of equations

$$\begin{bmatrix} \Phi_0^{(i)} & -D_H \Phi_1^{(i)} \end{bmatrix} \begin{bmatrix} c_H^{(i)} \\ c_w^{(i)} \end{bmatrix} = V_H \quad (8.21)$$

where $\Phi_0^{(i)}$ and $\Phi_1^{(i)}$ contain the partial fraction basis functions evaluated at the different frequency points ω_k :

$$\Phi_0^{(i)} = \begin{bmatrix} 1 & \frac{1}{j\omega_1 - p_1^{(i-1)}} & \cdots & \frac{1}{j\omega_1 - p_n^{(i-1)}} \\ \vdots & \vdots & & \vdots \\ 1 & \frac{1}{j\omega_k - p_1^{(i-1)}} & \cdots & \frac{1}{j\omega_k - p_n^{(i-1)}} \end{bmatrix}, \quad (8.22)$$

$$\Phi_1^{(i)} = \begin{bmatrix} \frac{1}{j\omega_1 - p_1^{(i-1)}} & \cdots & \frac{1}{j\omega_1 - p_n^{(i-1)}} \\ \vdots & & \vdots \\ \frac{1}{j\omega_k - p_1^{(i-1)}} & \cdots & \frac{1}{j\omega_k - p_n^{(i-1)}} \end{bmatrix}, \quad (8.23)$$

and $D_H = \text{diag}\{H_1, \dots, H_k\}$. The right hand side of (8.21) is a column vector formed by the given samples

$$V_H = [H_1 \quad \dots \quad H_k]^T, \quad (8.24)$$

while $c_H^{(i)}$ and $c_w^{(i)}$ contain the unknown coefficients

$$c_H^{(i)} = [r_0^{(i)} \quad \dots \quad r_n^{(i)}]^T, \quad (8.25)$$

$$c_w^{(i)} = [w_1^{(i)} \quad \dots \quad w_n^{(i)}]^T. \quad (8.26)$$

System (8.21) can be solved in the least-squares sense with a QR decomposition of the coefficient matrix [28]. Once (8.21) has been solved, the new poles estimate $p_n^{(i)}$ is computed with (8.18).

The VF iterative process usually converges very quickly, often in 4–5 iterations, except when the given samples are noisy. The fast and reliable convergence of VF is truly remarkable considering that VF ultimately solves a nonlinear minimization problem. Unfortunately, so far no one has been able to support this experimental evidence with strong theoretical results on VF convergence. Actually, contrived examples show that VF convergence is not guaranteed [53, 72]. However, these examples are quite artificial and far from practical datasets. Two decades of widespread use indeed show that, when properly implemented, VF is a remarkably robust algorithm for the identification of reduced order models from sampled data. In VF, convergence is typically monitored with three conditions:

1. When the poles estimates stabilizes, i. e. $p_n^{(i)} \simeq p_n^{(i-1)}$, performing new iterations will not improve accuracy. When this happens, $w^{(i)}(j\omega) \simeq 1$ for $\omega \in [\omega_1, \omega_k]$. This occurrence can be tested numerically as

$$\sqrt{\frac{1}{k} \sum_{k=1}^k |w^{(i)}(j\omega) - 1|^2} \leq \varepsilon_w, \quad (8.27)$$

where ε_w is a user-defined threshold. The advantage of criterion (8.27) is that it does not require additional computations apart from the calculation of the norm in (8.27). The limitation of this test is that it is based on an empirical condition, which may be satisfied even when the reduced model $H^{(i)}(s)$ still does not fit well the given frequency samples. Conversely, when samples H_k are noisy, test (8.27) may fail even when additional iterations will not significantly improve the poles estimate [33]. Therefore, test (8.27) should be only used as a low-cost check to decide whether it is worth to compute the error between the reduced model response and samples H_k .

2. When condition (8.27) is satisfied, the error between the fitted model and samples H_k should be checked. In principle, this can be done by computing the error between (8.16) and H_k . However, since after solving (8.21) a new estimate of the poles can be found via (8.18), the common practice is to use those poles to fit a new model. This is done by minimizing the exact error (8.6) between the given samples H_k and model

$$\tilde{H}^{(i+1)}(s) = r_0^{(i+1)} + \sum_{n=1}^{\tilde{n}} \frac{r_n^{(i+1)}}{s - p_n^{(i)}}, \quad (8.28)$$

considering only residues $r_0^{(i+1)}, \dots, r_{\tilde{n}}^{(i+1)}$ as unknowns. Since poles $p_n^{(i)}$ are now fixed, this is equivalent to solve, in least squares sense, the linear system

$$\Phi_0^{(i+1)} c_H^{(i+1)} = V_H. \quad (8.29)$$

The VF iteration ends, successfully, when

$$e \leq \varepsilon_H, \quad (8.30)$$

since model (8.28) meets the accuracy threshold ε_H set by the user. The main reason why this additional fitting step is performed is because this step minimizes the exact error (8.6) between model and given samples, rather than a linear approximation like (8.17), which improves accuracy and more reliably detects convergence. Therefore, solving (8.29) serves both as convergence test and as final fitting of the model.

3. In selected circumstances, VF may be unable to reach (8.30) even after many iterations. In this case, the iterative process concludes unsuccessfully when i exceeds the maximum number of iterations i_{\max} allowed by the user.

8.3.3 Example: fitting a rational transfer function

In this example, we apply VF to a set of samples H_k generated from a known rational function of order 10. Its poles were generated randomly, and are reported in Table 8.1. The original transfer function was sampled at $k = 100$ frequency points linearly spaced between $\omega_1 = 0.1$ rad/s and $\omega_{100} = 10$ rad/s. A Matlab implementation of VF [75] was used to fit the samples with a model in the form (8.16) with order $\bar{n} = 10$. The initial distribution of poles $p_n^{(0)}$ set by (8.20) is depicted in the left panel of Figure 8.1. Throughout the VF iterations, poles relocate to the final distribution shown in the right panel of Figure 8.1, which also compares them to the exact poles of the original rational function. We can see that the poles estimated by VF closely match the poles of the original system.

Table 8.1: Example of Section 8.3.3: poles and residues of the transfer function used to generate samples H_k .

Pole	Residue
constant term	$r_0 = 0.1059$
$p_1 = -1.3578$	$r_1 = -0.2808$
$p_2 = -1.2679$	$r_2 = 0.1166$
$p_{3,4} = -1.4851 \pm 0.2443j$	$r_{3,4} = 0.9569 \mp 0.7639j$
$p_{5,6} = -0.8487 \pm 2.9019j$	$r_{5,6} = 0.9357 \mp 0.7593j$
$p_{7,8} = -0.8587 \pm 3.1752j$	$r_{7,8} = 0.4579 \mp 0.7406j$
$p_{9,10} = -0.2497 \pm 6.5369j$	$r_{9,10} = 0.2405 \mp 0.7437j$

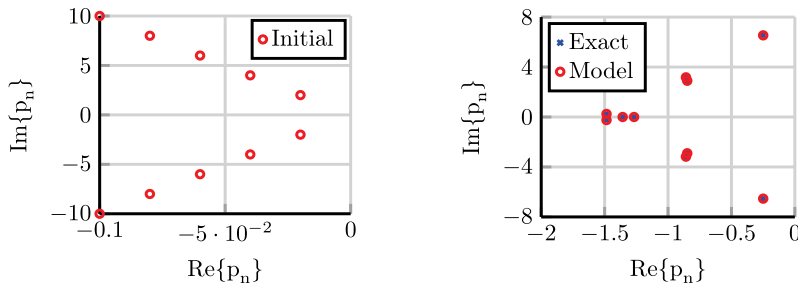


Figure 8.1: Left panel: initial poles $p_n^{(0)}$ used by VF in the first iteration. Right panel: poles of the final model $\tilde{H}(s)$ compared to the exact poles of the original transfer function.

In Figure 8.2, the frequency response $\tilde{H}(j\omega)$ of the VF model is compared to the initial samples. We observe excellent agreement over the entire frequency range of interest. At the conclusion of the VF iterative process, the worst-case error between samples H_k and model response

$$e_\infty = \max_k |H_k - H(j\omega_k)| \quad (8.31)$$

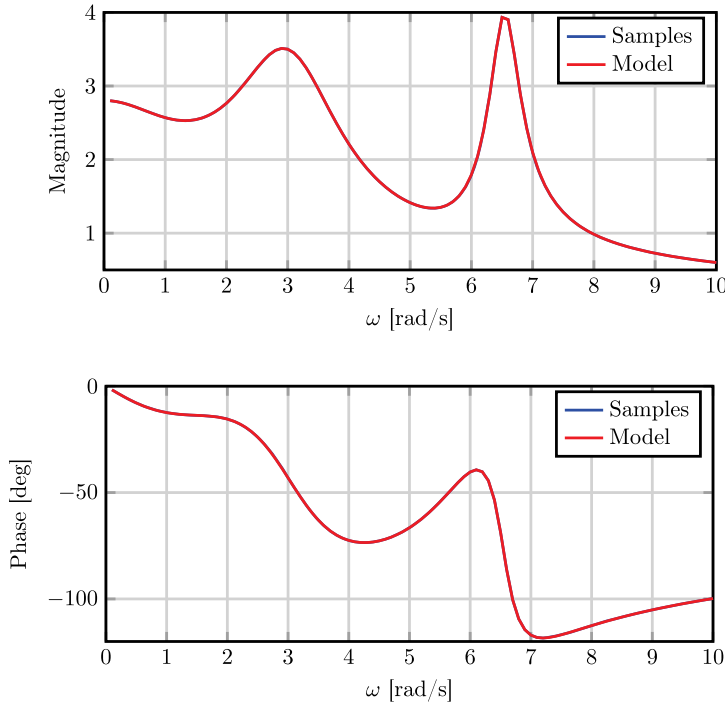


Figure 8.2: Example of Section 8.3.3: magnitude (top) and phase (bottom) of samples H_k and of the model $\hat{H}(\omega)$ identified by VF.

is 2.37×10^{-14} . Figure 8.3 shows the evolution of e_∞ throughout the five iterations performed by VF, plus a final iteration ($i = 6$) where poles were kept fixed and residues were calculated one more time using (8.29). The figure shows that VF converges very quickly, reaching an error below 10^{-8} in only three iterations. We can also observe that the final fitting iteration ($i = 6$) with fixed poles provides a more accurate model. For this example, VF took only 0.2 s of CPU time on a 2.2 GHz mobile processor. The source codes related to this example can be downloaded from [75].

8.3.4 Example: modeling of aortic input impedance

In this example, VF is used to model the relation between pressure $p(t)$ and flow rate $q(t)$ in the ascending aorta of a 1.1-year old patient [71, patient 1]. Simultaneous pressure and flow rate measurements were collected during a surgical procedure. The blood flow rate was measured with an ultrasonic flow probe positioned about 1 cm downstream of the aortic valve. The pressure was acquired using a catheter with a pressure transducer on its tip, positioned in the same location as the flow rate probe. From time domain recordings, the input impedance seen from the aorta was

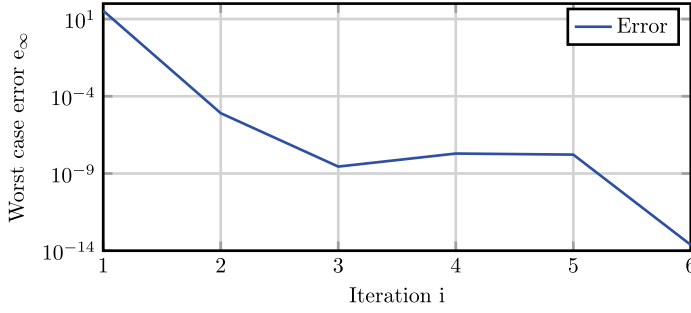


Figure 8.3: Example of Section 8.3.3: worst-case fitting error e_∞ as a function of iteration counter i . The last iteration ($i = 6$) was performed with fixed poles.

obtained:

$$Z(j\omega) = \frac{\mathcal{F}\{p(t)\}}{\mathcal{F}\{q(t)\}}, \quad (8.32)$$

where $\mathcal{F}\{\cdot\}$ denotes the Fourier transform. Impedance was computed at $\bar{k} = 11$ frequency points $\omega_k = 2\pi(k-1)f_0$ for $k = 1, \dots, 11$, where $f_0 = 2.54 \text{ Hz} = 152.4 \text{ beats/min}$ corresponds to the heart rate of the patient. The authors of [71] estimate that the impedance measurements are affected by uncertainty with a relative standard deviation that ranges between 0.66 % to 14.5 % depending on frequency. The relative standard deviation was normalized to $|Z(0)|$.

We apply VF to the impedance samples to obtain a closed-form model relating aortic pressure and flow rate. The limited number of available samples and their uncertainty make the identification of an accurate model challenging. We use this non-trivial scenario to explore the relation between number and quality of the available samples, model order \bar{n} , and accuracy. Vector Fitting was applied to the given samples four times with model order \bar{n} increasing from 2 to 8 in steps of 2. Figure 8.4 compares the magnitude and phase of the identified model to the original impedance samples. We can see that the $\bar{n} = 2$ model captures the overall trend of the impedance. However, it fails to resolve the increase in impedance at $f = 12.7 \text{ Hz}$ and the associated phase variation. Increasing order to 4 or 6 resolves that feature and provides higher accuracy. Further increasing order \bar{n} to 8 leads to a model which matches closely most given samples, but has a sharp and high peak at $f = 12.3 \text{ Hz}$. This unrealistic behavior in-between the given samples is typical of an overfitting scenario, where the sought model has too many degrees of freedom, which can be hardly estimated from the information contained in the available samples. Although still solvable, the conditioning number of (8.21) degrades. The system solution, which gives the model coefficients, becomes very sensitive to the noise superimposed to the given samples. The source codes related to this example can be downloaded from [75].

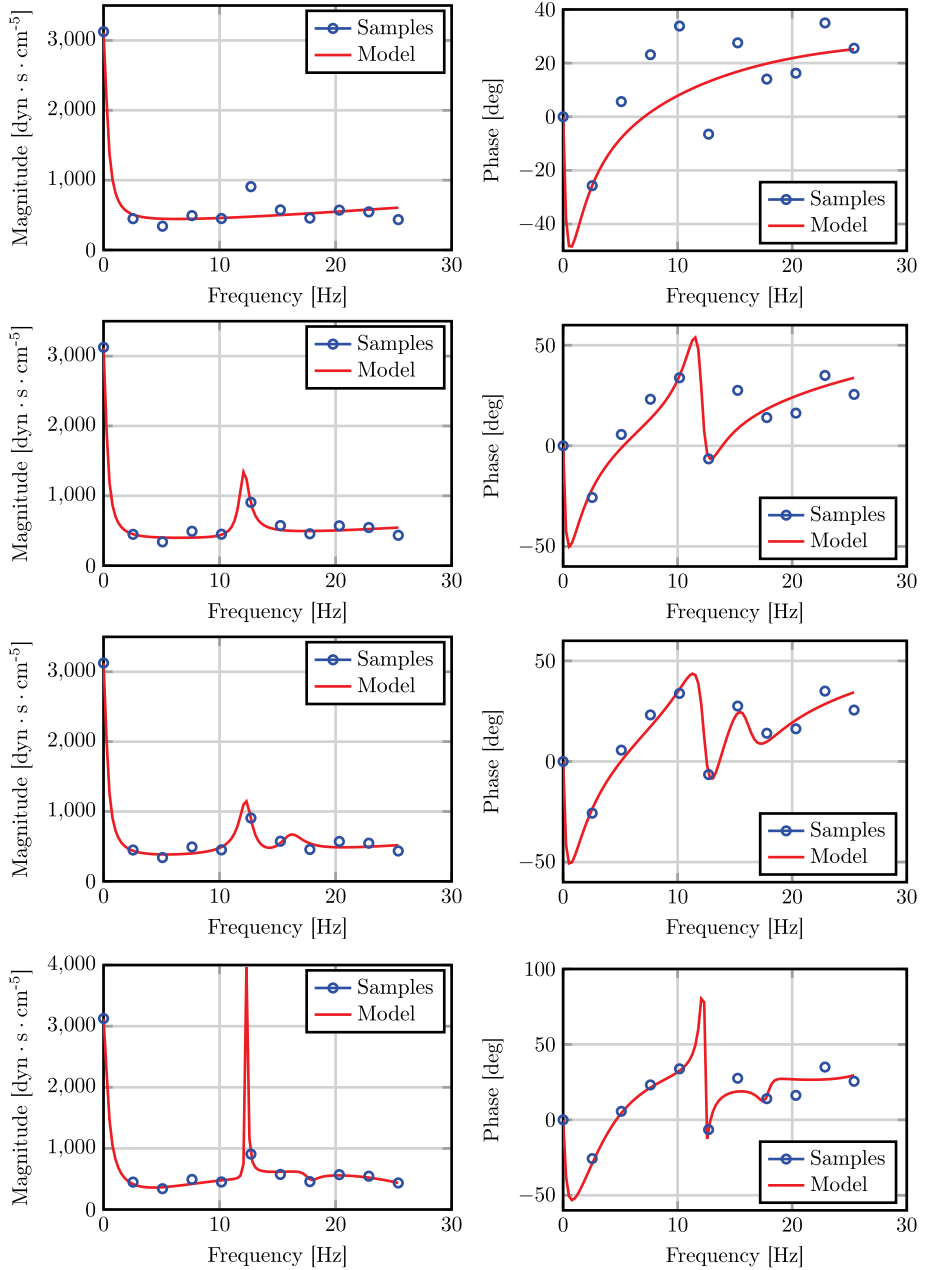


Figure 8.4: Impedance seen into the ascending aorta of the pediatric patient considered in Section 8.3.4: measured samples (circles) and response of four different VF models (dashed lines) of order $\tilde{n} = 2, 4, 6, 8$ (from top to bottom).

8.3.5 The multi-input multi-output case

The VF algorithm presented in Section 8.3.2 for the single-input single-output case can be easily extended to the general case of a system with \bar{m} inputs and \bar{q} outputs. In this case, the given samples are $\bar{q} \times \bar{m}$ complex matrices H_k , and we denote their (q, m) entry as $H_{k,qm}$. The model transfer function is now defined as

$$\bar{H}^{(i)}(s) = R_0^{(i)} + \sum_{n=1}^{\bar{n}} \frac{R_n^{(i)}}{s - p_n^{(i-1)}}, \quad (8.33)$$

where $R_n^{(i)} \in \mathbb{C}^{\bar{q} \times \bar{m}}$. In (8.33), the same poles $p_n^{(i-1)}$ are used for all elements of matrix $\bar{H}^{(i)}(s)$. This choice is appropriate when modeling linear dynamical systems, since it is known that the poles of each transfer function entry are a subset of a common set of poles shared by all transfer function elements. The physical justification of this fact is that poles are related to the natural modes of the system, which are a property of the system itself and not of individual entries of its transfer function. In other communities, natural modes are referred to as resonances or eigenmodes of the system. When VF is applied to model transfer functions not related to the same physical system, one should use distinct poles for different elements of (8.33). This scenario is discussed in [35], which also elaborates on the computational implications of this choice.

In the multi-input multi-output case, weighting function $w^{(i)}(s)$ remains defined by (8.15). Since the transfer function (8.33) is now matrix-valued, VF aims to minimize the error functional

$$e^2 = \frac{1}{\bar{k}\bar{q}\bar{m}} \sum_{k=1}^{\bar{k}} \|H_k - \bar{H}(j\omega_k)\|_F^2, \quad (8.34)$$

where $\|\cdot\|_F$ denotes the Frobenius norm, which for $A \in \mathbb{C}^{\bar{q} \times \bar{m}}$ is defined as

$$\|A\|_F = \sqrt{\sum_{q=1}^{\bar{q}} \sum_{m=1}^{\bar{m}} |A_{qm}|^2}. \quad (8.35)$$

From (8.35), we see that the square of the Frobenius norm is simply equal to the sum of the squared magnitudes of each entry. Therefore, minimizing (8.34) means minimizing the sum of the squared error between each sample $H_{k,qm}$ and the corresponding entry of (8.33).

The minimization of (8.34) is a nonlinear least-squares problem, which VF solves iteratively by working on the linearized error [42]

$$(e_{SK}^{(i)})^2 = \frac{1}{\bar{k}\bar{q}\bar{m}} \sum_{k=1}^{\bar{k}} \left\| H_k \left(1 + \sum_{n=1}^{\bar{n}} \frac{w_n^{(i)}}{j\omega_k - p_n^{(i-1)}} \right) - \left(R_0^{(i)} + \sum_{n=1}^{\bar{n}} \frac{R_n^{(i)}}{j\omega_k - p_n^{(i-1)}} \right) \right\|_F^2. \quad (8.36)$$

As in the single-input single-output case, we can see that (8.36) uses weighting function $w^{(i)}(s)$ to offset the error introduced by using the previous poles estimate in the

denominators. Minimizing (8.36) is equivalent to solving, in the least-squares sense, the system of equations

$$R_{0,qm}^{(i)} + \sum_{n=1}^{\bar{n}} \frac{R_{n,qm}^{(i)}}{j\omega_k - p_n^{(i-1)}} - H_{k,qm} \sum_{n=1}^{\bar{n}} \frac{w_n^{(i)}}{j\omega_k - p_n^{(i-1)}} = H_{k,qm} \quad (8.37)$$

for $k = 1, \dots, \bar{k}$, $q = 1, \dots, \bar{q}$ and $m = 1, \dots, \bar{m}$. In matrix form, equations (8.37) read

$$\begin{bmatrix} \Phi_0^{(i)} & 0 & \dots & 0 & -D_{H_{11}} \Phi_1^{(i)} \\ 0 & \Phi_0^{(i)} & \ddots & \vdots & -D_{H_{21}} \Phi_1^{(i)} \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & \Phi_0^{(i)} & -D_{H_{\bar{q}\bar{m}}} \Phi_1^{(i)} \end{bmatrix} \begin{bmatrix} c_{H_{11}}^{(i)} \\ c_{H_{21}}^{(i)} \\ \vdots \\ c_{H_{\bar{q}\bar{m}}}^{(i)} \\ c_w^{(i)} \end{bmatrix} = \begin{bmatrix} V_{H_{11}} \\ V_{H_{21}} \\ \vdots \\ V_{H_{\bar{q}\bar{m}}} \end{bmatrix}, \quad (8.38)$$

where $D_{H_{qm}}$ and $V_{H_{qm}}$ are, respectively, a diagonal matrix and a column vector formed by all samples $H_{k,qm}$ for $k = 1, \dots, \bar{k}$. In the unknown vector of (8.38),

$$c_{H_{qm}}^{(i)} = [R_{0,qm}^{(i)} \quad \dots \quad R_{\bar{n},qm}^{(i)}]^T, \quad (8.39)$$

and $c_w^{(i)}$ is defined by (8.26). System (8.38) is solved in step 4 of Algorithm 8.1. In step 8, a tentative final fitting of the model is performed, assuming fixed poles and determining only a new estimate of residues $R_n^{(i+1)}$. This step can be achieved by solving

$$\Phi_0^{(i+1)} c_{H_{qm}}^{(i+1)} = V_{H_{qm}}, \quad (8.40)$$

for $q = 1, \dots, \bar{q}$ and $m = 1, \dots, \bar{m}$.

8.3.6 The fast Vector Fitting algorithm

As the number of inputs \bar{m} and outputs \bar{q} increases, the computational cost of solving (8.38) can quickly become unsustainable. As technology evolves, this scenario arises more frequently, as engineers need to model systems of increasing complexity, either in terms of dynamic order or number of inputs and outputs. For example, a modern server processor has about 2,000 pins, which are connected to the motherboard by a dense network of tiny wires realized on the chip package. Seen as an input-output system, this network will have about 4,000 inputs and outputs, half where the network connects to the motherboard, and half where the network is connected to the silicon die. The need to predict electromagnetic interference in this dense and intricate network of wires calls for scalable algorithms to create reduced order models for systems where the number of inputs \bar{m} and outputs \bar{q} can be several thousands [70, 8].

The Fast VF algorithm [22, 47] significantly reduces the cost of solving (8.38) for multi-input and multi-output systems. Savings are achieved by exploiting the block

structure of (8.38) and the fact that, of the solution vector of (8.38), only $c_w^{(i)}$ is actually needed to compute the new poles estimate (8.18). A least-squares problem in the form of (8.38) can be efficiently solved by first performing the QR decompositions [29, 9, 86, 22]

$$\begin{bmatrix} \Phi_0^{(i)} & -D_{H_{qm}} \Phi_1^{(i)} \end{bmatrix} = \begin{bmatrix} \mathcal{Q}_{qm}^1 & \mathcal{Q}_{qm}^2 \end{bmatrix} \begin{bmatrix} \mathcal{R}_{qm}^{11} & \mathcal{R}_{qm}^{12} \\ 0 & \mathcal{R}_{qm}^{22} \end{bmatrix}, \quad (8.41)$$

for $q = 1, \dots, \bar{q}$ and $m = 1, \dots, \bar{m}$. Then a reduced system is formed [22]:

$$\begin{bmatrix} \mathcal{R}_{11}^{22} \\ \mathcal{R}_{21}^{22} \\ \vdots \\ \mathcal{R}_{\bar{q}\bar{m}}^{22} \end{bmatrix} c_w^{(i)} = \begin{bmatrix} (\mathcal{Q}_{11}^2)^H V_{H_{11}} \\ (\mathcal{Q}_{21}^2)^H V_{H_{21}} \\ \vdots \\ (\mathcal{Q}_{\bar{q}\bar{m}}^2)^H V_{H_{\bar{q}\bar{m}}} \end{bmatrix}, \quad (8.42)$$

where H denotes the conjugate transpose, also known as Hermitian transpose. System (8.42) is solved in the least-squares sense to determine $c_w^{(i)}$, and compute the new poles estimate with (8.18). Computational savings arise from the fact that the size of the matrices involved in (8.41) and (8.42) is much lower than the size of the coefficient matrix in (8.38). Furthermore, since the $\bar{q}\bar{m}$ QR decompositions (8.41) are independent, they can be performed in parallel [14]. The Fast VF algorithm with parallelization can identify reduced models for systems with hundreds of inputs and outputs in minutes [35]. A pseudo-code of a real-valued implementation of the Fast VF algorithm will be given in Section 8.3.8.

Several other ideas were proposed to increase VF scalability for large input and output counts. In VF with compression, samples H_k are “compressed” with a singular value decomposition reducing the cost of the subsequent fitting [37] and passivity enforcement steps [63]. The Loewner method [52, 46], which is an alternative to VF for the data-driven modeling of linear systems, was also shown to scale favorably with respect to the number of inputs and outputs. This class of techniques is the subject of Chapter 6 of this volume.

8.3.7 Example: modeling of a multiport interconnect on a printed circuit board

Vector Fitting is extensively used by electronic designers to model how high-speed digital signals propagate over a printed circuit board, and design the system accordingly. We consider the structure shown in Figure 8.5, which consists of several copper traces realized on the top face of a high-performance printed circuit board (Wild River Technology CMP-28 [88]). This structure mimics, in a simplified way, the multiwire buses that may connect the CPU and memory of a high-performance server. At the end of

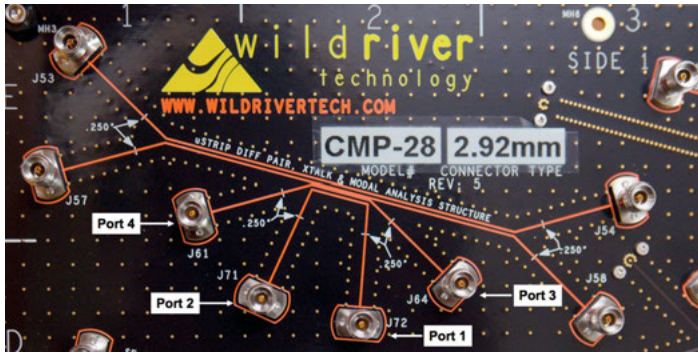


Figure 8.5: Interconnect network on a printed circuit board considered in Section 8.3.7. The four measurement ports of the vector network analyzer were connected as shown in the Figure.

each trace, an electrical port is defined between the trace endpoint and a reference point on the ground plane underneath. The port is defined where the CPU or memory chip would be connected. In the test system, a high-frequency connector was installed at each port allowing the user to inject a signal from each port, and observe the signal received at the other ports.

In this example, we consider the two lower traces in Figure 8.5, which have connectors J72, J71, J64, J61 soldered at their ends. The scattering matrix $H(j\omega)$ of this 4-port device was measured from 10 MHz to 40 GHz in steps of 10 MHz with a Keysight N5227A vector network analyzer (courtesy of Fadime Bekmambetova, University of Toronto). In the scattering representation, input $U_m(j\omega)$ is the amplitude of the electromagnetic wave injected into port m by the instrument. Output $Y_q(j\omega)$ is the amplitude of the wave received at port q . The scattering representation is commonly used at high frequency since it can be measured more accurately compared to the impedance or admittance representations used at low frequency.

A commercial implementation of the VF algorithm (IdEM, Dassault Systemes) was used to generate a reduced order model from the measured samples (courtesy of Prof. Stefano Grivet-Talocia, Politecnico di Torino). Figure 8.6 compares the VF model response to the original samples for the (1, 2) element of the scattering matrix. This response is the ratio between the amplitude of the wave received at one end of the trace (port 1) and the amplitude of the wave injected at the other end (port 2). We see that, as frequency increases, the received signal is progressively weaker, due to higher attenuation. The agreement between the VF model and the samples is excellent over the entire frequency range spanned by the measured data. Figure 8.7 compares the model response to the measured samples for the (1, 3) entry of the scattering matrix, which describes the signal received on the lower copper trace in Figure 8.5 when only the upper trace is excited. This coefficient is about 25 times smaller than the (1, 2) coefficient, since the two traces are not directly connected, and any coupling is due to

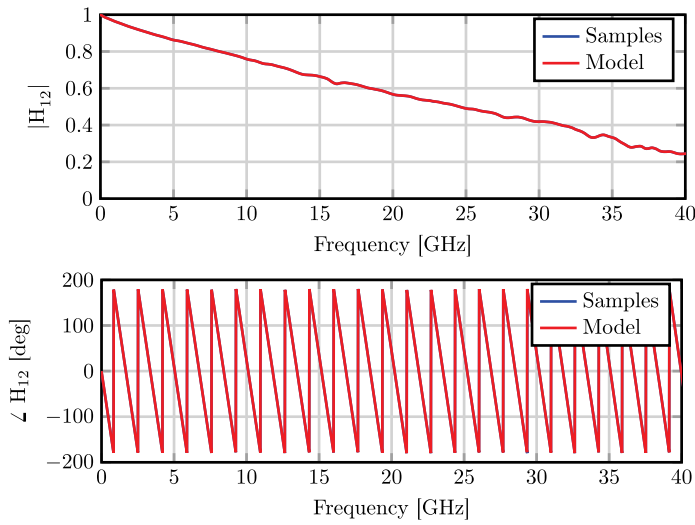


Figure 8.6: Example of Section 8.3.7: comparison between samples $H_{k,12}$ and corresponding VF model response.

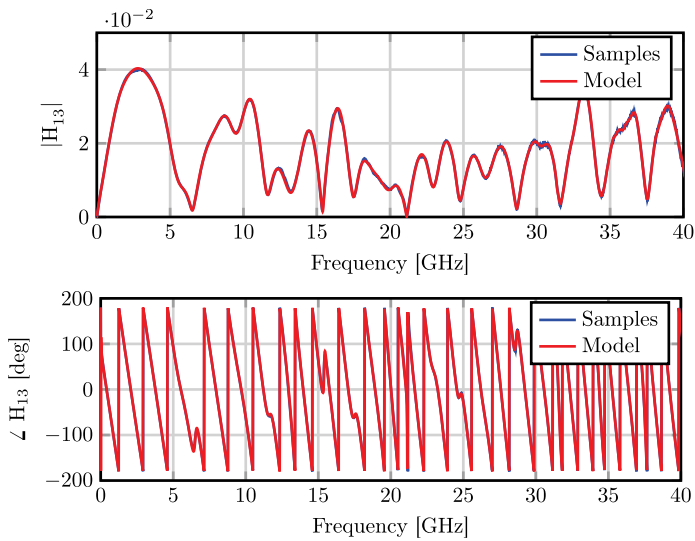


Figure 8.7: Example of Section 8.3.7: comparison between samples $H_{k,13}$ and corresponding VF model response.

electromagnetic interference. We can see that the VF model approximates this small entry very accurately.

Figure 8.8 plots the samples-model error $e_{SK}^{(i)}$ as a function of i , together with the order \tilde{n} used by VF at each iteration. In this example, the order is adapted throughout

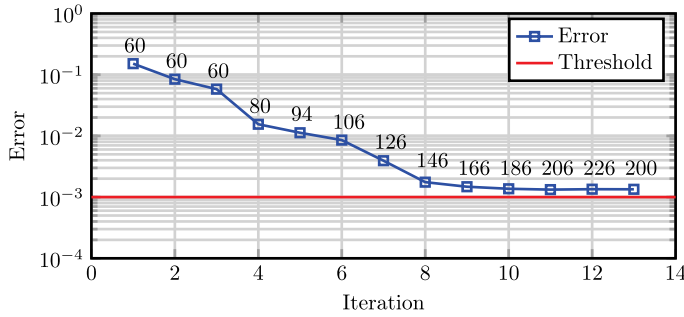


Figure 8.8: Example of Section 8.3.7: VF error as a function of iteration, compared to the desired error level. Labels indicate the order \bar{n} used by VF at each iteration.

iterations with the adding and skimming process [33] described in Section 8.3.11.1. We observe that VF is able to progressively reduce the error throughout iterations, but convergence is slower than in the analytical example of Section 8.3.3. This happens because of two reasons. First, this implementation of VF adaptively determines order \bar{n} in a single run, without requiring the user to determine a suitable \bar{n} with multiple VF runs. Second, some noise is unavoidably present in the experimental measurements, which slows down convergence, and prevents VF from reducing the fitting error below 10^{-3} . Indeed, we can see that VF is unable to increase model accuracy after the 10th iteration. Ultimately, VF delivers a reduced model with an error of $1.34 \cdot 10^{-3}$, which is adequate for most design purposes.

8.3.8 A real-valued formulation of VF and fast VF

In most systems of practical interest, input $u(t)$ and output $y(t)$ are real-valued. Consequently, poles p_n and residues R_n are expected to be either real or in complex conjugate pairs. Because of round-off errors, the VF algorithm described so far may not ensure this realness condition. In this section, we describe a real-valued version of Fast VF which can be implemented in real arithmetics, and will ensure the realness condition by construction. The pseudo-code of the described algorithm is given in Algorithm 8.2. An open-source implementation of this algorithm, which closely follows the notation and pseudo-code in this chapter, can be downloaded from [75].

To ensure complex conjugate poles and residues, we redefine model (8.33) as

$$\tilde{H}^{(i)}(s) = R_0^{(i)} + \sum_{n=1}^{\bar{n}_r} \frac{R_n^{(i)}}{s - p_n^{(i-1)}} + \sum_{n=\bar{n}_r+1}^{\bar{n}_r+\bar{n}_c} \left[\frac{R_n^{(i)}}{s - p_n^{(i-1)}} + \frac{(R_n^{(i)})^*}{s - (p_n^{(i-1)})^*} \right], \quad (8.43)$$

where \bar{n}_r is the number of real poles and \bar{n}_c is the number of pairs of complex conjugate poles, for a total order $\bar{n} = \bar{n}_r + 2\bar{n}_c$. In (8.43), we force $R_n^{(i)} \in \mathbb{R}^{\bar{q} \times \bar{m}}$ for $n = 0, \dots, \bar{n}_r$. The

VF weighting function (8.15) is redefined in a similar fashion as

$$w^{(i)}(s) = 1 + \sum_{n=1}^{\bar{n}_r} \frac{w_n^{(i)}}{s - p_n^{(i-1)}} + \sum_{n=\bar{n}_r+1}^{\bar{n}_r+\bar{n}_c} \left[\frac{w_n^{(i)}}{s - p_n^{(i-1)}} + \frac{(w_n^{(i)})^*}{s - (p_n^{(i-1)})^*} \right], \quad (8.44)$$

where $w_n^{(i)} \in \mathbb{R}$ for $n = 1, \dots, \bar{n}_r$. Using (8.43) and (8.44), and following the steps in Section 8.3.5, one can arrive at a least-squares system in the same form as (8.38)

$$\begin{bmatrix} \Phi_0^{(i)} & 0 & \dots & 0 & -D_{H_{11}} \Phi_1^{(i)} \\ 0 & \Phi_0^{(i)} & \ddots & \vdots & -D_{H_{21}} \Phi_1^{(i)} \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & \Phi_0^{(i)} & -D_{H_{qm}} \Phi_1^{(i)} \end{bmatrix} \begin{bmatrix} c_{H_{11}}^{(i)} \\ c_{H_{21}}^{(i)} \\ \vdots \\ c_{H_{qm}}^{(i)} \\ c_w^{(i)} \end{bmatrix} = \begin{bmatrix} V_{H_{11}} \\ V_{H_{21}} \\ \vdots \\ V_{H_{qm}} \end{bmatrix}, \quad (8.45)$$

but where we take as unknowns the real and imaginary part of each residue in (8.43)

$$c_{H_{qm}}^{(i)} = [R_{0,qm}^{(i)} \quad \dots \quad R_{\bar{n}_r,qm}^{(i)} \quad \text{Re}\{R_{\bar{n}_r+1,qm}^{(i)}\} \quad \text{Im}\{R_{\bar{n}_r+1,qm}^{(i)}\} \quad \dots]^T, \quad (8.46)$$

and the real and imaginary part of each residue of the weighting function (8.44)

$$c_w^{(i)} = [w_1^{(i)} \quad \dots \quad w_{\bar{n}_r}^{(i)} \quad \text{Re}\{w_{\bar{n}_r+1}^{(i)}\} \quad \text{Im}\{w_{\bar{n}_r+1}^{(i)}\} \quad \dots]^T. \quad (8.47)$$

This choice of unknowns will ensure that complex residues always come in conjugate pairs. The coefficient matrices $\Phi_0^{(i)}$ and $\Phi_1^{(i)}$ in (8.45) are given by

$$\Phi_0^{(i)} = [1_{\bar{k}} \quad \Phi_r^{(i)} \quad \Phi_c^{(i)}], \quad (8.48)$$

$$\Phi_1^{(i)} = [\Phi_r^{(i)} \quad \Phi_c^{(i)}], \quad (8.49)$$

where $1_{\bar{k}}$ is a $\bar{k} \times 1$ vector of ones, and

$$\Phi_r^{(i)} = \begin{bmatrix} \frac{1}{j\omega_1 - p_1^{(i-1)}} & \dots & \frac{1}{j\omega_1 - p_{\bar{n}_r}^{(i-1)}} \\ \vdots & \ddots & \vdots \\ \frac{1}{j\omega_{\bar{k}} - p_1^{(i-1)}} & \dots & \frac{1}{j\omega_{\bar{k}} - p_{\bar{n}_r}^{(i-1)}} \end{bmatrix}, \quad (8.50)$$

$$\Phi_c^{(i)} = \begin{bmatrix} \frac{1}{j\omega_1 - p_{\bar{n}_r+1}^{(i-1)}} + \frac{1}{j\omega_1 - (p_{\bar{n}_r+1}^{(i-1)})^*} & \frac{j}{j\omega_1 - p_{\bar{n}_r+1}^{(i-1)}} - \frac{j}{j\omega_1 - (p_{\bar{n}_r+1}^{(i-1)})^*} & \dots \\ \vdots & \vdots & \\ \frac{1}{j\omega_{\bar{k}} - p_{\bar{n}_r+1}^{(i-1)}} + \frac{1}{j\omega_{\bar{k}} - (p_{\bar{n}_r+1}^{(i-1)})^*} & \frac{j}{j\omega_{\bar{k}} - p_{\bar{n}_r+1}^{(i-1)}} - \frac{j}{j\omega_{\bar{k}} - (p_{\bar{n}_r+1}^{(i-1)})^*} & \dots \end{bmatrix}. \quad (8.51)$$

Although (8.45) has real unknowns, its coefficients matrix and right hand side are still complex-valued. To remedy this issue, we write the real and imaginary part of each

equation separately

$$\begin{bmatrix} \text{Re}\{\Phi_0^{(i)}\} & 0 & \dots & 0 & -\text{Re}\{D_{H_{11}} \Phi_1^{(i)}\} \\ \text{Im}\{\Phi_0^{(i)}\} & 0 & \dots & 0 & -\text{Im}\{D_{H_{11}} \Phi_1^{(i)}\} \\ \vdots & & & \vdots & \vdots \\ 0 & \dots & 0 & \text{Re}\{\Phi_0^{(i)}\} & -\text{Re}\{D_{H_{qm}} \Phi_1^{(i)}\} \\ 0 & \dots & 0 & \text{Im}\{\Phi_0^{(i)}\} & -\text{Im}\{D_{H_{qm}} \Phi_1^{(i)}\} \end{bmatrix} \begin{bmatrix} c_{H_{11}}^{(i)} \\ \vdots \\ c_{H_{qm}}^{(i)} \\ c_w^{(i)} \end{bmatrix} = \begin{bmatrix} \text{Re}\{V_{H_{11}}\} \\ \text{Im}\{V_{H_{11}}\} \\ \vdots \\ \text{Re}\{V_{H_{qm}}\} \\ \text{Im}\{V_{H_{qm}}\} \end{bmatrix}. \quad (8.52)$$

The obtained system, which has real coefficients and unknowns will ensure, by construction, that model poles and residues are either real or complex conjugate. Due to its block structure, system (8.52) can be efficiently solved with the Fast VF approach discussed in Section 8.3.6. In step 5 of Algorithm 8.2, the QR decompositions

$$\begin{bmatrix} \text{Re}\{\Phi_0^{(i)}\} & -\text{Re}\{D_{H_{qm}} \Phi_1^{(i)}\} \\ \text{Im}\{\Phi_0^{(i)}\} & -\text{Im}\{D_{H_{qm}} \Phi_1^{(i)}\} \end{bmatrix} = \begin{bmatrix} \mathcal{Q}_{qm}^{11} & \mathcal{Q}_{qm}^{12} \\ \mathcal{Q}_{qm}^{21} & \mathcal{Q}_{qm}^{22} \end{bmatrix} \begin{bmatrix} \mathcal{R}_{qm}^{11} & \mathcal{R}_{qm}^{12} \\ 0 & \mathcal{R}_{qm}^{22} \end{bmatrix}, \quad (8.53)$$

are computed for $q = 1, \dots, \bar{q}$ and $m = 1, \dots, \bar{m}$. Then, in step 5, the reduced system

$$\begin{bmatrix} \mathcal{R}_{11}^{22} \\ \mathcal{R}_{21}^{22} \\ \vdots \\ \mathcal{R}_{\bar{q}\bar{m}}^{22} \end{bmatrix} c_w^{(i)} = \begin{bmatrix} (\mathcal{Q}_{11}^{12})^T \text{Re}\{V_{H_{11}}\} + (\mathcal{Q}_{11}^{22})^T \text{Im}\{V_{H_{11}}\} \\ (\mathcal{Q}_{21}^{12})^T \text{Re}\{V_{H_{21}}\} + (\mathcal{Q}_{21}^{22})^T \text{Im}\{V_{H_{21}}\} \\ \vdots \\ (\mathcal{Q}_{\bar{q}\bar{m}}^{12})^T \text{Re}\{V_{H_{\bar{q}\bar{m}}}\} + (\mathcal{Q}_{\bar{q}\bar{m}}^{22})^T \text{Im}\{V_{H_{\bar{q}\bar{m}}}\} \end{bmatrix} \quad (8.54)$$

is solved in the least-squares sense to determine $c_w^{(i)}$ and compute the new poles estimate with the real-valued counterpart of (8.18), which reads [35]

$$\{p_n^{(i)}\} = \text{eig}(A^{(i-1)} - b_w(c_w^{(i)})^T), \quad (8.55)$$

with $A^{(i-1)} = \text{diag}\{p_1^{(i-1)}, \dots, p_{\bar{n}_r}^{(i-1)}, \Pi_{\bar{n}_r+1}^{(i-1)}, \dots, \Pi_{\bar{n}_r+\bar{n}_c}^{(i-1)}\}$ being a block diagonal matrix formed by the real poles and, for complex conjugate pairs, by the blocks

$$\Pi_n^{(i-1)} = \begin{bmatrix} \text{Re}\{p_n^{(i-1)}\} & \text{Im}\{p_n^{(i-1)}\} \\ -\text{Im}\{p_n^{(i-1)}\} & \text{Re}\{p_n^{(i-1)}\} \end{bmatrix}. \quad (8.56)$$

In (8.55), b_w is a $\bar{n} \times 1$ vector with the first \bar{n}_r entries set to one, followed by a $[2, 0]^T$ block for each pair of complex conjugate poles.

Once poles have been estimated, a first convergence test is performed in step 8 using (8.27). If the test is passed, in step 9 of Algorithm 8.2 we fit the residues of the final model, solving in the least-squares sense

$$\begin{bmatrix} \text{Re}\{\Phi_0^{(i+1)}\} \\ \text{Im}\{\Phi_0^{(i+1)}\} \end{bmatrix} c_{H_{qm}}^{(i+1)} = \begin{bmatrix} \text{Re}\{V_{H_{qm}}\} \\ \text{Im}\{V_{H_{qm}}\} \end{bmatrix}, \quad (8.57)$$

for $q = 1, \dots, \bar{q}$ and $m = 1, \dots, \bar{m}$. The second and final convergence test is performed in step 11 of Algorithm 8.2.

Algorithm 8.2: Fast Vector Fitting, real-valued implementation.

Require: response samples H_k , corresponding frequencies ω_k ($k = 1, \dots, \bar{k}$)

Require: desired model order \bar{n}

Require: maximum number of iterations i_{\max}

- 1: set initial poles $p_n^{(0)}$ according to (8.19) or (8.20).
- 2: $i \leftarrow 1$
- 3: **while** $i \leq i_{\max}$ **do**
- 4: Compute QR decompositions (8.53)
- 5: Solve (8.54) in the least-squares sense
- 6: Compute the new poles estimate $p_n^{(i)}$ with (8.55)
- 7: Enforce poles stability with (8.71), if desired ▷ Stability enforcement
- 8: **if** (8.27) is true **then** ▷ First convergence test
- 9: Solve (8.57) in the least-squares sense ▷ Tentative final fitting
- 10: Compute fitting error e with (8.34)
- 11: **if** $e \leq \varepsilon_H$ **then** ▷ Second convergence test
- 12: $\tilde{H}(s) = \tilde{H}^{(i+1)}(s)$
- 13: **return** Success!
- 14: **end if**
- 15: **end if**
- 16: $i \leftarrow i + 1$
- 17: **end while**
- 18: **return** Failure: maximum number of iterations reached.

8.3.9 Model realization

The real-valued formulation of VF, discussed in Section 8.3.8, produces a reduced model in the form

$$\tilde{H}(s) = R_0 + \sum_{n=1}^{\bar{n}_r} \frac{R_n}{s - p_n} + \sum_{n=\bar{n}_r+1}^{\bar{n}_r+\bar{n}_c} \left[\frac{R_n}{s - p_n} + \frac{R_n^*}{s - p_n^*} \right], \quad (8.58)$$

which can be easily converted into a variety of equivalent representations to facilitate its use in different simulation scenarios. Expression (8.58) is known as pole-residue form of the transfer function. This form is the most convenient when the model will be used in frequency domain analyses, since it minimizes the computational cost of evaluating $H(j\omega)$.

For time domain analyses, such as transient simulations, expression (8.58) can be converted into the time domain with the inverse Laplace transform, which yields

$$\tilde{h}(t) = R_0 + \sum_{n=1}^{\bar{n}_r} R_n e^{p_n t} + \sum_{n=\bar{n}_r+1}^{\bar{n}_r+\bar{n}_c} [2R'_n e^{p'_n t} \cos(p''_n t) - 2R''_n e^{p'_n t} \sin(p''_n t)] \quad (8.59)$$

for $t \geq 0$, where $p'_n = \text{Re}\{p_n\}$, $p''_n = \text{Im}\{p_n\}$, $R'_n = \text{Re}\{R_n\}$ and $R''_n = \text{Im}\{R_n\}$. In (8.59), $\bar{h}(t)$ denotes the impulse response of the model. This form is particularly convenient in transient simulators based on convolutions like (8.1). While computing convolution integrals is in general very expensive, when an impulse response has the form (8.59), convolution can be computed very quickly using recursive formulas [35].

While convolutional simulators are prominent in selected applications, the majority of transient simulators is based on the solution of differential equations, and cannot handle (8.59) directly. To overcome this issue, we can represent (8.58) through a set of differential equations in state-space form

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \\ y(t) = Cx(t) + Du(t). \end{cases} \quad (8.60)$$

System (8.60) is constructed in such a way that the transfer function between input $u(t)$ and output $y(t)$ is (8.58). Given a transfer function, there are infinitely-many systems (8.60) that meet this criterion, known as *realizations* of $H(s)$. We present a popular realization, due to Gilbert [27], and refer the reader to [35] for a comprehensive description of how VF models can be realized.

For reasons that will become clear later on, the Gilbert realization process begins with the truncated singular value decomposition [28] of residues R_n

$$R_n = U_n \Sigma_n V_n^H \quad \text{for } n = 1, \dots, \bar{n}_r + \bar{n}_c, \quad (8.61)$$

where $\Sigma_n = \text{diag}\{\sigma_{n,1}, \dots, \sigma_{n,\rho_n}\}$ is a diagonal matrix collecting all nonzero singular values of R_n , and ρ_n is the rank of R_n . Matrices $U_n \in \mathbb{C}^{\bar{q} \times \rho_n}$ and $V_n \in \mathbb{C}^{\bar{m} \times \rho_n}$ are formed by the left and right singular vectors of R_n , respectively. Given (8.61), we can express the partial fractions in (8.58) associated to real poles as

$$\frac{R_n}{s - p_n} = U_n \Sigma_n \frac{I_{\rho_n}}{s - p_n} V_n^T = C_n (sI_{\rho_n} - A_n)^{-1} B_n, \quad (8.62)$$

for $n = 1, \dots, \bar{n}_r$. In (8.62), I_{ρ_n} is the identity matrix of size $\rho_n \times \rho_n$, $C_n = U_n \Sigma_n$, $A_n = p_n I_{\rho_n}$, and $B_n = V_n^T$. For complex poles, we can derive an equivalent expression for the sum of the two conjugate partial fractions [35]

$$\frac{R_n}{s - p_n} + \frac{R_n^*}{s - p_n^*} = C_n (sI_{2\rho_n} - A_n)^{-1} B_n, \quad (8.63)$$

for $n = \bar{n}_r + 1, \dots, \bar{n}_r + \bar{n}_c$, where

$$A_n = \begin{bmatrix} p'_n I_{\rho_n} & p''_n I_{\rho_n} \\ -p''_n I_{\rho_n} & p'_n I_{\rho_n} \end{bmatrix} \quad B_n = 2 \begin{bmatrix} \text{Re}\{V_n^H\} \\ \text{Im}\{V_n^H\} \end{bmatrix} \quad (8.64)$$

$$C_n = [\text{Re}\{U_n \Sigma_n\} \quad \text{Im}\{U_n \Sigma_n\}]. \quad (8.65)$$

Expressions (8.62) and (8.63) allow us to rewrite (8.58) as

$$\tilde{H}(s) = D + C(sI_N - A)^{-1}B, \quad (8.66)$$

where

$$A = \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_{\tilde{n}_r + \tilde{n}_c} \end{bmatrix}, \quad B = \begin{bmatrix} B_1 \\ \vdots \\ B_{\tilde{n}_r + \tilde{n}_c} \end{bmatrix}, \quad (8.67)$$

$$C = [C_1 \quad \dots \quad C_{\tilde{n}_r + \tilde{n}_c}], \quad D = R_0. \quad (8.68)$$

Since (8.66) is the transfer function of (8.60), equations (8.67) and (8.68) provide the coefficient matrices of a state space realization (8.60) of the transfer function (8.58) produced by VF. The order of (8.66) is

$$N = \sum_{n=1}^{\tilde{n}_r} \rho_n + 2 \sum_{n=\tilde{n}_r+1}^{\tilde{n}_r + \tilde{n}_c} \rho_n \quad (8.69)$$

and can be shown to be minimal [27]. This property stems from the singular value decompositions (8.61), which reveal the rank ρ_n of each residue R_n . If those singular value decompositions are not performed, a realization of order $\tilde{n}\tilde{m}$ is obtained. This realization may not be minimal, and may contain states that are not controllable, not observable, or both, as discussed in Chapter 2 of this volume.

In addition to the forms presented in this section, the VF model (8.58) can be converted to a variety of additional forms, including equivalent electric circuits [5, 35] for seamless integration into any circuit simulator.

8.3.10 Stability, causality and passivity enforcement

Most systems of practical interest are stable, and the real part of their poles is either negative or zero. One would expect that, given noise-free samples of their frequency response, VF will produce a model with stable poles satisfying

$$\text{Re}\{p_n\} \leq 0 \quad \forall n. \quad (8.70)$$

Unfortunately, this is not guaranteed, since round-off errors may indeed push a few poles into the right half of the complex plane, making the VF model unstable.

Condition (8.70) is essentially mandatory for time domain simulations, since otherwise results will diverge. The standard practice is to enforce stability during VF iterations. After computing the new poles estimate $p_n^{(i)}$ with (8.18), the following rule is applied:

$$p_n^{(i)} = \begin{cases} p_n^{(i)} & \text{if } \text{Re}\{p_n^{(i)}\} < 0, \\ -\text{Re}\{p_n^{(i)}\} + j\text{Im}\{p_n^{(i)}\} & \text{if } \text{Re}\{p_n^{(i)}\} > 0, \end{cases} \quad (8.71)$$

for $n = 1, \dots, \bar{n}_r + \bar{n}_c$. We can see that, if a pole $p_n^{(i)}$ is unstable, the sign of its real part is inverted. Since in the tentative final fitting in step 8 of Algorithm 8.1 poles are fixed, condition (8.71) ensures the stability of the final model.

For frequency domain analyses, one may think that (8.70) is not necessary, since stability is not an issue. However, one can show that (8.70), in the frequency domain, becomes a condition for causality [82]. Causality means that the system will react to an excitation only after it has been applied, and not before. In other words, if the system input $u(t)$ begins at $t = t_0$ ($u(t) = 0$ for $t < t_0$), the system output will start varying only at or after $t = t_0$. All systems in nature are obviously causal, since they cannot “anticipate” the application of an excitation. Enforcing (8.70) ensures that VF model (8.58) is causal. If this is not the case, frequency domain analyses will succeed, but results may be inaccurate and unphysical. In particular, the VF model may underestimate the delay between input and output which is present in the real system, which may be important in some applications, such as the timing analysis of digital circuits. A complete discussion of causality is beyond the scope of this chapter, and the reader is referred to [82].

Overall, condition (8.70) simultaneously enforces the stability and causality of the VF model. This condition can be enforced without any accuracy penalty when the given samples H_k are error free, and thus faithfully represent the response of a causal and stable system. When samples are corrupted by noise or measurement errors, VF may be unable to reduce fitting error (8.6) to the desired level if condition (8.70) is enforced. This happens when the noise or errors in samples H_k are not causal functions themselves, and thus cannot be approximated with stable and causal poles [82]. Numerical algorithms exist to verify if the given samples H_k satisfy the causality condition required by VF to fit them with high accuracy [77, 78, 51, 76, 7].

In addition to causality and stability, passivity is another important property that one may want to impose on the VF model (8.58). This property characterizes those physical systems that are unable to generate energy on their own, simply due to the lack of energy sources or gain mechanisms inside them. A circuit made by positive resistors, capacitors and inductors is an example of a passive system, in contrast to an amplifying circuit. When applied to the response of a passive system, VF may still produce a non-passive model, due to approximation and numerical errors. However, passivity can be enforced a-posteriori, with the methods presented in Chapter 5 of this volume.

8.3.11 Numerical implementation

Vector Fitting is easy to implement, and several free codes are available [75, 38]. This section briefly describes a few changes to the basic templates in Algorithms 8.1 and 8.2 that can lead to a more robust and efficient implementation.

8.3.11.1 Order estimation

The VF templates in Algorithms 8.1 and 8.2 require the desired model order \bar{n} as input. Typically, this is not known a priori, but can be determined during the fitting process using the VF algorithm with adding and skimming [33], as shown in the example of Section 8.3.7. In this method, an initial estimate of \bar{n} is derived from the phase of the given samples H_k , and used in the first VF iteration. Then \bar{n} is automatically increased or decreased based on the achieved error, as visible in Figure 8.8. If error e is still too high, the order is increased until either VF converges or it becomes evident that no further error reduction can be achieved, as in the last four iterations in Figure 8.8. Conversely, when the algorithm detects that some partial fractions in (8.58) give a negligible contribution over the frequency range of interest, the order \bar{n} is reduced at the next iteration by removing such terms. This happens in the 13th iteration of the example in Section 8.3.7, where the order is reduced from 226 to 200.

8.3.11.2 Relaxed VF: a better normalization of the weighting function

In the original VF algorithm, the coefficients of weighting function (8.15) are normalized such that $w^{(i)}(j\omega) \rightarrow 1$ when $\omega \rightarrow \infty$. It can be shown that this normalization is not optimal, and can slow down VF convergence when samples H_k are contaminated by noise. The relaxed VF algorithm [41] mitigates this issue by redefining the weighting function as

$$w^{(i)}(s) = w_0^{(i)} + \sum_{n=1}^{\bar{n}} \frac{w_n^{(i)}}{s - p_n^{(i-1)}}, \quad (8.72)$$

where $w_0^{(i)}$ is now free to depart from one. With this change, the fitting equation (8.37) becomes

$$R_{0,qm}^{(i)} + \sum_{n=1}^{\bar{n}} \frac{R_{n,qm}^{(i)}}{j\omega_k - p_n^{(i-1)}} - H_{k,qm} \left(w_0^{(i)} + \sum_{n=1}^{\bar{n}} \frac{w_n^{(i)}}{j\omega_k - p_n^{(i-1)}} \right) = 0. \quad (8.73)$$

Since (8.73) admits a trivial solution ($R_{n,qm}^{(i)} = w_n^{(i)} = 0 \forall n$), the relaxed VF algorithm adds an additional constraint to exclude it [41]

$$\frac{1}{\bar{k}} \sum_{k=1}^{\bar{k}} \operatorname{Re} \left\{ w_0^{(i)} + \sum_{n=1}^{\bar{n}} \frac{w_n^{(i)}}{j\omega_k - p_n^{(i-1)}} \right\} = 1. \quad (8.74)$$

This constraint can be seen as a more relaxed normalization of the weighting function. Equations (8.73) and (8.74) are then jointly solved in the least-squares sense. In the single-input single-output case ($\bar{q} = \bar{m} = 1$), the system to be solved takes the form

$$\begin{bmatrix} \Phi_0^{(i)} & -D_H \Phi_0^{(i)} \\ 0 & \frac{\beta}{\bar{k}} (1_{\bar{k}})^T \Phi_0^{(i)} \end{bmatrix} \begin{bmatrix} c_H^{(i)} \\ c_w^{(i)} \end{bmatrix} = \begin{bmatrix} 0 \\ \beta \end{bmatrix} \quad (8.75)$$

where

$$c_w^{(i)} = \begin{bmatrix} w_0^{(i)} & \dots & w_n^{(i)} \end{bmatrix}^T. \quad (8.76)$$

In (8.75), β is a suitable weight to the last equation, which is typically set to [41]

$$\beta = \sqrt{\sum_{k=1}^{\bar{k}} |H_k|^2}. \quad (8.77)$$

8.4 Generalized and advanced VF algorithms

Since its inception in 1996, VF has inspired a generation of algorithms for the data-driven modeling of linear systems. These extensions either improve the original VF formulation, or extend it to different modeling scenarios. We briefly summarize the most relevant work in this area, and provide several bibliographic references where more details can be found.

8.4.1 Time domain VF algorithms

The original VF algorithm works in the frequency domain, and creates the reduced model from samples of the system frequency response. In some applications, however, it is more convenient to characterize the system in the time domain. For example, one may have simultaneous measurements of the system input $u(t_l)$ and output $y(t_l)$ at several time points t_l for $l = 1, \dots, \bar{l}$, as in the example of Section 8.3.4. In this scenario, one has two options. The first is to estimate the systems' frequency response from the time domain samples with the discrete Fourier transform, and apply VF in the frequency domain. However, the accuracy of the discrete Fourier transform depends significantly on the sampling rate of the given samples, and on their behavior near the boundaries $t = t_1$ and $t = t_{\bar{l}}$ of the acquisition window. These issues, if not well understood and managed, can result in an inaccurate time–frequency conversion, and degrade model quality.

The second option is to use the time domain VF algorithm [30, 31], which directly extracts (8.58) from the time domain samples $u(t_l)$ and $y(t_l)$. This is achieved by rewriting the fitting error (8.17) in the time domain, where multiplication by partial fraction $1/(s - p_n)$ becomes a convolution between $e^{p_n t}$ and the input or output samples. These convolutions can be computed by numerical integrations, leading to a time domain version of the original VF algorithm which closely follows the steps of the original frequency domain VF algorithm [35].

The time domain VF algorithm leads to a model in the continuous time domain. Alternatively, if the sampling period $\Delta t = t_{l+1} - t_l$ is constant, one can also apply the z-domain VF [59], which relies on the z transform as opposed to the Laplace transform.

This latter algorithm leads to a model in the discrete time domain, which can be expressed as a digital filter or as a set of difference equations (as opposed to differential equations).

8.4.2 Improved Vector Fitting formulations

In the QuadVF algorithm [23], a quadrature rule inspired by the H_2 error measure is used in conjunction with a suitable choice of frequency sampling points to improve the fidelity of the reduced model to the given samples. The same work also shows how one can incorporate derivative information, making QuadVF able to minimize a discrete Sobolev norm. In [24], this approach is extended to the multi-input multi-output case, and a way to control the McMillan degree¹ of the approximation is proposed, which helps to achieve smaller reduced models when \bar{q} and \bar{m} are high.

The numerical robustness of VF, which is already quite remarkable in its original formulation, is further improved in the Orthonormal VF algorithm [21]. This algorithm replaces partial fractions $1/(s - p_n)$ in (8.15) and (8.16) with orthonormal rational functions, achieving better numerical conditioning of the linear system (8.38) to be solved.

Another subject that received considerable attention is the robustness of VF against noise in the given samples H_k . Noise may arise from the measurement process or, if samples were obtained with a numerical simulation, from round-off errors, approximations, and convergence issues. The relaxed normalization discussed in Section 8.3.11.2 improves VF convergence in the presence of noise [41]. Furthermore, the VF with adding and skimming includes a mechanism to detect spurious poles caused by noise [33]. Since spurious poles impair VF convergence, they must be removed throughout iterations [33]. This mechanism is coupled with a robust way to adaptively refine model order \bar{n} to maximize accuracy even when noise is significant [33]. Taking into account noise variance in the definition of the VF fitting error was also shown to improve convergence [26]. Finally, instrumental variables can be used to unbiased the VF process from the effects of noise, leading to better accuracy and convergence at no additional cost [10].

8.4.3 VF algorithms for distributed systems

The efficient modeling of distributed systems is an open problem in model order reduction. A system is distributed when the time a signal takes to propagate from an input to an output is not negligible. In systems described by a Helmholtz (wave) equation, this happens when the physical size of the system is not negligible compared to the

¹ The McMillan degree [93] of a matrix transfer function $H(s)$ is the order of a minimal state space realization of $H(s)$, such as the order N of the Gilbert realization discussed in Section 8.3.9.

wavelength. Propagation delays lead to the presence of irrational terms in the transfer function of the underlying system. Typically, these terms are in the form $e^{-s\tau}$ where τ is the propagation delay. Rational functions, including the partial fractions in (8.58) can accurately fit these irrational terms, up to arbitrary accuracy. However, if τ is not negligible, the required order may be large, and will quickly increase as τ grows. This leads to a large model which may burden subsequent simulations.

To overcome this issue, the core idea is to explicitly include exponential terms $e^{-s\tau_l}$ in the reduced model which will be fitted to the given samples. A popular choice is to define each element $\tilde{H}_{qm}(s)$ of the model transfer function as

$$\sum_{l=1}^{\bar{l}} \left(r_{0,l} + \sum_{n=1}^{\bar{n}_l} \frac{r_{n,l}}{s - p_{n,l}} \right) e^{-s\tau_l}, \quad (8.78)$$

where the $_{qm}$ subscript was omitted from all coefficients for clarity. The exponential factors in (8.78) are meant to efficiently capture long propagation delays, while the rational terms between brackets will resolve the residual behavior of the system. Typically, since long propagation delays are already accounted for by the exponential terms, the order \bar{n}_l of the rational factors can be kept quite low.

For systems with uniform cross-section along the direction of propagation, such as electrical transmission lines and fluid pipes, VF is used in conjunction to the method of characteristics to obtain an efficient distributed model [50, 2, 36, 61]. For distributed systems of general shape, several VF algorithms with delay terms have been proposed [15, 16, 13, 79, 67, 58]. In these algorithms, the first step is to identify the values of the relevant propagation delays τ_i present in the system. Given only frequency samples H_k , this is not a trivial task, and the dominant approach is to exploit time–frequency decompositions [39, 32, 67, 48]. Next, the coefficients of the remaining rational factors in the model are determined with a VF-like iterative process [16, 13, 79, 67, 58].

8.4.4 Parametric VF algorithms

The design process of an engineering system typically requires a large number of simulations for different values of design parameters, such as material properties, geometrical dimensions and operating conditions (e. g. bias voltages, temperature, ...). In early design stages, parametric simulations are used to explore the design space. Later on, they may be used to optimize design in order to meet specifications or improve performance. Moreover, parametric simulations also help designers to account for manufacturing variability during design. In the context of parametric simulations, conventional VF models may be inefficient. Indeed, every time a parameter changes, a new set of samples H_k must be obtained, and the fitting process has to be repeated from scratch.

A better solution is to create a parametric VF model which captures the system response with respect to both frequency s and some parameters of interest $\mu^{(1)}, \mu^{(2)}, \dots$. The core idea behind parametric VF techniques [83, 80, 64, 20, 34] is to let residues R_n and poles p_n in (8.58) be parameter-dependent functions, such as polynomials in $\mu^{(1)}, \mu^{(2)}, \dots$. Their coefficients can be determined with an iterative process analogous to the Sanathanan–Koerner iteration in Section 8.2, starting from samples of the system's frequency response obtained for multiple values of parameters $\mu^{(1)}, \mu^{(2)}, \dots$. The main advantage of a parametric model is that, once generated, it can be reused many times for different parameter values within its range of validity. One of the challenges in the generation of parametric VF models is how to guarantee that the model will be stable and passive over the desired parameter range [81, 85, 25]. Recently, systematic solutions to this challenging problem have been proposed [92].

8.5 Conclusion

This chapter introduced the Vector Fitting algorithm, which has become one of the most popular tools for the extraction of linear reduced order models from samples of their response, collected in the frequency or in the time domain. Vector Fitting produces a rational model which approximately minimizes the least-squares error between the given samples and the model response. Determining model coefficients is originally a nonlinear least-squares problem, whose solution is prone to the typical issues of nonlinear minimization: high computational cost and problematic convergence due to local minima. Vector Fitting overcomes these issues by iteratively minimizing a linearization of the original problem, leveraging well-established methods for the solution of linear least-squares problems. Several strategies to obtain a robust and efficient implementation of VF have been reviewed. When properly implemented, Vector Fitting enjoys remarkable robustness, efficiency and versatility, typically converging in a handful of iterations. Finally, we reviewed the most prominent extensions of the original algorithm which have been proposed for data-driven modeling of time domain systems, noisy samples, distributed systems, and parametric systems.

Vector Fitting's superior performance and reliability lead to a widespread use in many different fields. Originally conceived to predict how transients propagate throughout power distribution networks, VF is the method of choice for the wideband modeling of overhead lines, underground cables and power transformers [61, 40, 62, 4, 35]. In electronic engineering, VF is extensively used to model the propagation of high-speed signals through interconnect networks found at the chip, package and printed circuit board level. These models are crucial for system design, and greatly help in preventing signal integrity, power integrity and electromagnetic compatibility issues [2, 68, 55, 74, 64, 1, 90]. The impact of VF in this area is confirmed by the fact that

all leading commercial tools for the design of high-frequency electronic circuits include a VF module. Applications in microwave engineering [56, 84, 19, 18] and digital filter design [89] have also been reported. Within computational electromagnetism, VF can be used to efficiently model the Green function of layered media, which is necessary to solve Maxwell's equations with integral equation methods [49, 11, 65]. The ability of VF to generate models compatible with transient simulations has also been exploited in the finite difference time domain (FDTD) method [57, 60], the finite element time domain method [12, 87], and the discontinuous Galerkin method [91]. Beyond electrical engineering, VF found countless applications in various domains, including acoustics [17, 66], fluid dynamics [3, 45, 35], mechanical engineering [35, 6], and in the thermal modeling of chemical batteries [44]. For a collection of VF applications and additional references, the reader is referred to [35].

Bibliography

- [1] A. Chinea, S. Grivet-Talocia, H. Hu, P. Triverio, D. Kaller, C. Siviero, and M. Kindscher. Signal integrity verification of multi-chip links using passive channel macromodels. *IEEE Trans. Compon. Packag. Manuf. Technol.*, 1(6):920–933, 2011.
- [2] R. Achar and M. S. Nakhla. Simulation of high-speed interconnects. *Proc. IEEE*, 89(5):693–728, 2001.
- [3] A. Almondo and M. Sorli. Time domain fluid transmission line modelling using a passivity preserving rational approximation of the frequency dependent transfer matrix. *Int. J. Fluid Power*, 7(1):41–50, 2006.
- [4] U. Annakkage, N.-K. C. Nair, Y. Liang, A. Gole, V. Dinavahi, B. Gustavsen, T. Noda, H. Ghasemi, A. Monti, M. Matar, and et al.. Dynamic system equivalents: A survey of available techniques. *IEEE Trans. Power Deliv.*, 27(1):411–420, 2012.
- [5] G. Antonini. SPICE equivalent circuits of frequency-domain responses. *IEEE Trans. Electromagn. Compat.*, 45(3):502–512, 2003.
- [6] E. Balmès. GARTEUR Group on Ground Vibration Testing. Results from the Test of a Single Structure by 12 Laboratories in Europe. In *15th International Modal Analysis Conference*, volume 3089, page 1346, 1997.
- [7] L. L. Barannyk, H. A. Aboutaleb, A. Elshabini, and F. D. Barlow. Spectrally accurate causality enforcement using svd-based fourier continuations for high-speed digital interconnects. *IEEE Trans. Compon. Packag. Manuf. Technol.*, 5(7):991–1005, 2015.
- [8] U. Baur, P. Benner, and L. Feng. Model order reduction for linear and nonlinear systems: a system-theoretic perspective. *Arch. Comput. Methods Eng.*, 21(4):331–358, 2014.
- [9] D. S. Bayard. High-order multivariable transfer function curve fitting: Algorithms, sparse matrix methods and experimental results. *Automatica*, 30(9):1439–1444, 1994.
- [10] A. Beygi and A. Dounavis. An instrumental variable vector-fitting approach for noisy frequency responses. *IEEE Trans. Microw. Theory Tech.*, 60(9):2702–2712, 2012.
- [11] R. R. Boix, F. Mesa, and F. Medina. Application of total least squares to the derivation of closed-form Green's functions for planar layered media. *IEEE Trans. Microw. Theory Tech.*, 55(2):268–280, 2007.
- [12] Y. Cai and C. Mias. Faster 3D finite element time domain-floquet absorbing boundary condition modelling using recursive convolution and vector fitting. *IET Microw. Antennas Propag.*, 3(2):310–324, 2009.

- [13] A. Charest, M. S. Nakhla, R. Achar, D. Saraswat, N. Soveiko, and I. Erdin. Time domain delay extraction-based macromodeling algorithm for long-delay networks. *IEEE Trans. Adv. Packaging*, 33(1):219–235, 2010.
- [14] A. Chinae and S. Grivet-Talocia. On the parallelization of vector fitting algorithms. *IEEE Trans. Compon. Packag. Manuf. Technol.*, 1(11):1761–1773, 2011.
- [15] A. Chinae, P. Triverio, and S. Grivet-Talocia. Compact macromodeling of electrically long interconnects. In *Proc. of the 17th Topical Meeting on Electrical Performance of Electronic Packaging (EPEP 2008)*, pages 199–202. IEEE, 2008.
- [16] A. Chinae, P. Triverio, and S. Grivet-Talocia. Delay-based macromodeling of long interconnects from frequency-domain terminal responses. *IEEE Trans. Adv. Packaging*, 33(1):246–256, 2010.
- [17] B. Cotté, P. Blanc-Benon, C. Bogey, and F. Poisson. Time-domain impedance boundary conditions for simulations of outdoor sound propagation. *AIAA J.*, 47(10):2391–2403, 2009.
- [18] D. De Jonghe and G. Gielen. Characterization of analog circuits using transfer function trajectories. *IEEE Trans. Circuits Syst. I, Regul. Pap.*, 59(8):1796–1804, 2012.
- [19] D. Deschrijver, G. Avolio, D. Schreurs, T. Dhaene, G. Crupi, and L. Knockaert. Microwave small-signal modelling of FinFETs using multi-parameter rational fitting method. *Electron. Lett.*, 47(19):1084–1086, 2011.
- [20] D. Deschrijver, T. Dhaene, and D. De Zutter. Robust parametric macromodeling using multivariate orthonormal vector fitting. *IEEE Trans. Microw. Theory Tech.*, 56(7):1661–1667, 2008.
- [21] D. Deschrijver, B. Haegeman, and T. Dhaene. Orthonormal vector fitting: A robust macromodeling tool for rational approximation of frequency domain responses. *IEEE Trans. Adv. Packaging*, 30(2):216–225, 2007.
- [22] D. Deschrijver, M. Mrozowski, T. Dhaene, and D. De Zutter. Macromodeling of multiport systems using a fast implementation of the vector fitting method. *IEEE Microw. Wirel. Compon. Lett.*, 18(6):383–385, 2008.
- [23] Z. Drmac, S. Gugercin, and C. Beattie. Quadrature-based vector fitting for discretized h_2 approximation. *SIAM J. Sci. Comput.*, 37(2):A625–A652, 2015.
- [24] Z. Drmac, S. Gugercin, and C. Beattie. Vector fitting for matrix-valued rational approximation. *SIAM J. Sci. Comput.*, 37(5):A2346–A2379, 2015.
- [25] F. Ferranti, L. Knockaert, and T. Dhaene. Guaranteed passive parameterized admittance-based macromodeling. *IEEE Trans. Adv. Packaging*, 33(3):623–629, 2010.
- [26] F. Ferranti, Y. Rolain, L. Knockaert, and T. Dhaene. Variance weighted vector fitting for noisy frequency responses. *IEEE Microw. Wirel. Compon. Lett.*, 20(4):187–189, 2010.
- [27] E. G. Gilbert. Controllability and observability in multivariable control systems. *J. Soc. Ind. Appl. Math., A, on Control*, 1(2):128–151, 1963.
- [28] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.
- [29] G. H. Golub and R. J. Plemmons. Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition. *Linear Algebra Appl.*, 34:3–28, 1980.
- [30] S. Grivet-Talocia. Package macromodeling via time-domain vector fitting. *IEEE Microw. Wirel. Compon. Lett.*, 13(11):472–474, 2003.
- [31] S. Grivet-Talocia. The time-domain vector fitting algorithm for linear macromodeling. *Int. J. Electron. Commun.*, 58(4):293, 2004.
- [32] S. Grivet-Talocia. Delay-based macromodels for long interconnects via time–frequency decompositions. In *2006 IEEE Electrical Performance of Electronic Packaging*, pages 199–202. IEEE, 2006.
- [33] S. Grivet-Talocia and M. Bandinu. Improving the convergence of vector fitting for equivalent circuit extraction from noisy frequency responses. *IEEE Trans. Electromagn. Compat.*, 48(1):104–120, 2006.

- [34] S. Grivet-Talocia and E. Fevola. Compact parameterized black-box modeling via Fourier-rational approximations. *IEEE Trans. Electromagn. Compat.*, 59(4):1133–1142, 2017.
- [35] S. Grivet-Talocia and B. Gustavsen. *Passive macromodeling: Theory and applications*. John Wiley & Sons, 2015.
- [36] S. Grivet-Talocia, H. -M. Huang, A. E. Ruehli, F. Canavero, and I. Elfadel. Transient analysis of lossy transmission lines: An efficient approach based on the method of characteristics. *IEEE Trans. Adv. Packaging*, 27(1):45–56, 2004.
- [37] S. Grivet-Talocia, S. Olivadese, and P. Triverio. A compression strategy for rational macromodeling of large interconnect structures. In *2011 IEEE Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, pages 53–56. IEEE, 2011.
- [38] B. Gustavsen. The Vector Fitting Website. <https://www.sintef.no/projectweb/vectfit/>. Accessed: 2018-12-06.
- [39] B. Gustavsen. Time delay identification for transmission line modeling. In *8th IEEE Workshop on Signal Propagation on Interconnects*, pages 103–106. IEEE, 2004.
- [40] B. Gustavsen. Wide band modeling of power transformers. *IEEE Trans. Power Deliv.*, 19(1):414–422, 2004.
- [41] B. Gustavsen. Improving the pole relocating properties of vector fitting. *IEEE Trans. Power Deliv.*, 21(3):1587–1592, 2006.
- [42] B. Gustavsen and A. Semlyen. Rational approximation of frequency domain responses by vector fitting. *IEEE Trans. Power Deliv.*, 14(3):1052–1061, 1999.
- [43] W. Hendrickx and T. Dhaene. A discussion of “Rational approximation of frequency domain responses by vector fitting”. *IEEE Trans. Power Syst.*, 21(1):441–443, 2006.
- [44] X. Hu, L. Chaudhari, S. Lin, S. Stanton, S. Asgari, and W. Lian. A state space thermal model for HEV/EV battery using vector fitting. In *2012 IEEE Transportation Electrification Conference and Expo (ITEC)*, pages 1–8. IEEE, 2012.
- [45] S. Jaensch, C. Sovardi, and W. Polifke. On the robust, flexible and consistent implementation of time domain impedance boundary conditions for compressible flow simulations. *J. Comput. Phys.*, 314:145–159, 2016.
- [46] M. T. Kassis, M. Kabir, Y. Q. Xiao, and R. Khazaka. Passive reduced order macromodeling based on loewner matrix interpolation. *IEEE Trans. Microw. Theory Tech.*, 64(8):2423–2432, 2016.
- [47] L. Knockaert. Comments on “macromodeling of multiport systems using a fast implementation of the vector fitting method”. *IEEE Microw. Wirel. Compon. Lett.*, 19(9):602, 2009.
- [48] I. Kocar and J. Mahseredjian. New procedure for computation of time delays in propagation function fitting for transient modeling of cables. *IEEE Trans. Power Deliv.*, 31(2):613–621, 2016.
- [49] V. N. Kourkoulos and A. C. Cangellaris. Accurate approximation of Green’s functions in planar stratified media in terms of a finite sum of spherical and cylindrical waves. *IEEE Trans. Antennas Propag.*, 54(5):1568–1576, 2006.
- [50] D. B. Kuznetsov and J. E. Schutt-Ainé. Optimal transient simulation of transmission lines. *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, 43(2):110–121, 1996.
- [51] S. Lalgudi. On checking causality of tabulated S-parameters. *IEEE Trans. Compon. Packag. Manuf. Technol.*, 3(7):1204–1217, 2013.
- [52] S. Lefteriu and A. C. Antoulas. A new approach to modeling multiport systems from frequency-domain data. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 29(1):14–27, 2010.
- [53] S. Lefteriu and A. C. Antoulas. On the convergence of the vector-fitting algorithm. *IEEE Trans. Microw. Theory Tech.*, 61(4):1435–1443, 2013.
- [54] E. Levy. Complex-curve fitting. *IRE Trans. Autom. Control*, AC-4(1):37–43, 1959.
- [55] E. -P. Li, X. -C. Wei, A. C. Cangellaris, E. -X. Liu, Y. -J. Zhang, M. D’amore, J. Kim, and T. Sudo. Progress review of electromagnetic compatibility analysis technologies for packages, printed circuit boards, and novel interconnects. *IEEE Trans. Electromagn. Compat.*, 52(2):248–265, 2010.

- [56] C. -K. Liao, C. -Y. Chang, and J. Lin. A vector-fitting formulation for parameter extraction of lossy microwave filters. *IEEE Microw. Wirel. Compon. Lett.*, 17(4):277–279, 2007.
- [57] H. Lin, M. F. Pantoja, L. D. Angulo, J. Alvarez, R. G. Martin, and S. G. Garcia. FDTD modeling of graphene devices using complex conjugate dispersion material model. *IEEE Microw. Wirel. Compon. Lett.*, 22(12):612–614, 2012.
- [58] M. Luo and K. -M. Huang. An extended delay-rational macromodel for electromagnetic interference analysis of mixed signal circuits. *Prog. Electromagn. Res.*, 127:189–210, 2012.
- [59] Y. S. Mekonnen and J. E. Schutt-Aine. Broadband macromodeling of sampled frequency data using z-domain vector-fitting method. In *2007 IEEE Workshop on Signal Propagation on Interconnects*, pages 45–48. IEEE, 2007.
- [60] K. A. Michalski. On the low-order partial-fraction fitting of dielectric functions at optical wavelengths. *IEEE Trans. Antennas Propag.*, 61(12):6128–6135, 2013.
- [61] A. Morched, B. Gustavsen, and M. Tartibi. A universal model for accurate calculation of electromagnetic transients on overhead lines and underground cables. *IEEE Trans. Power Deliv.*, 14(3):1032–1038, 1999.
- [62] T. Noda. Identification of a multiphase network equivalent for electromagnetic transient calculations using partitioned frequency response. *IEEE Trans. Power Deliv.*, 20(2):1134–1142, 2005.
- [63] S. B. Olivadese and S. Grivet-Talocia. Compressed passive macromodeling. *IEEE Trans. Compon. Packag. Manuf. Technol.*, 2(8):1378–1388, 2012.
- [64] P. Triverio, S. Grivet-Talocia, M. Bandinu, and F. Canavero. Geometrically-parameterized circuit models of printed circuit board traces inclusive of antenna coupling. *IEEE Trans. Electromagn. Compat.*, 52:471–478, 2010.
- [65] A. G. Polimeridis, T. V. Yioultsis, and T. D. Tsiboukis. A robust method for the computation of Green's functions in stratified media. *IEEE Trans. Antennas Propag.*, 55(7):1963–1969, 2007.
- [66] S. R. Robinson, C. T. Nguyen, and J. B. Allen. Characterizing the ear canal acoustic impedance and reflectance by pole-zero fitting. *Hear. Res.*, 301:168–182, 2013.
- [67] S. Roy and A. Dounavis. Transient simulation of distributed networks using delay extraction based numerical convolution. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 30(3):364–373, 2011.
- [68] A. E. Ruehli and A. C. Cangellaris. Progress in the methodologies for the electrical modeling of interconnects and electronic packages. *Proc. IEEE*, 89(5):740–771, 2001.
- [69] C. Sanathanan and J. Koerner. Transfer function synthesis as a ratio of two complex polynomials. *IEEE Trans. Autom. Control*, 8(1):56–58, 1963.
- [70] W. H. Schilders. The need for novel model order reduction techniques in the electronics industry. In *Model reduction for circuit simulation*, pages 3–23. Springer, 2011.
- [71] M. K. Sharp, G. M. Pantalos, L. Minich, L. Y. Tani, E. C. McGough, and J. A. Hawkins. Aortic input impedance in infants and children. *J. Appl. Physiol.*, 88(6):2227–2239, 2000.
- [72] G. Shi. On the nonconvergence of the vector fitting algorithm. *IEEE Trans. Circuits Syst. II, Express Briefs*, 63(8):718–722, 2016.
- [73] K. Steiglitz and L. McBride. A technique for the identification of linear systems. *IEEE Trans. Autom. Control*, 10(4):461–464, 1965.
- [74] M. Swaminathan, D. Chung, S. Grivet-Talocia, K. Bharath, V. Laddha, and J. Xie. Designing and modeling for power integrity. *IEEE Trans. Electromagn. Compat.*, 52(2):288–310, 2010.
- [75] P. Triverio. Vector Fitting Resources. <http://www.modelics.org/vf.html>. Accessed: 2019-08-23.
- [76] P. Triverio. Robust causality check for sampled scattering parameters via a filtered fourier transform. *IEEE Microw. Wirel. Compon. Lett.*, 24(2):72–74, 2014.
- [77] P. Triverio and S. Grivet-Talocia. A robust causality verification tool for tabulated frequency data. In *2006 IEEE Workshop on Signal Propagation on Interconnects*, pages 65–68. IEEE, 2006.

- [78] P. Triverio and S. Grivet-Talocia. Robust causality characterization via generalized dispersion relations. *IEEE Trans. Adv. Packaging*, 31(3):579–593, 2008.
- [79] P. Triverio, S. Grivet-Talocia, and A. Chinea. Identification of highly efficient delay-rational macromodels of long interconnects from tabulated frequency data. *IEEE Trans. Microw. Theory Tech.*, 58(3):566–577, 2010.
- [80] P. Triverio, S. Grivet-Talocia, and M. S. Nakhla. An improved fitting algorithm for parametric macromodeling from tabulated data. In *2008 12th IEEE Workshop on Signal Propagation on Interconnects*, pages 1–4. IEEE, 2008.
- [81] P. Triverio, S. Grivet-Talocia, and M. S. Nakhla. A parameterized macromodeling strategy with uniform stability test. *IEEE Trans. Adv. Packaging*, 32(1):205–215, 2009.
- [82] P. Triverio, S. Grivet-Talocia, M. S. Nakhla, F. G. Canavero, and R. Achar. Stability, causality, and passivity in electrical interconnect models. *IEEE Trans. Adv. Packaging*, 30(4):795–808, 2007.
- [83] P. Triverio, M. Nakhla, and S. Grivet-Talocia. Parametric macromodeling of multiport networks from tabulated data. In *2007 IEEE Workshop on Electrical Performance of Electronic Packaging*, pages 51–54. IEEE, 2007.
- [84] P. Triverio, M. Nakhla, and S. Grivet-Talocia. Extraction of parametric circuit models from scattering parameters of passive RF components. In *The 40th European Microwave Conference*, pages 1635–1638. IEEE, 2010.
- [85] P. Triverio, M. S. Nakhla, and S. Grivet-Talocia. Passive parametric macromodeling from sampled frequency data. In *2010 IEEE 14th Workshop on Signal Propagation on Interconnects*, pages 117–120. IEEE, 2010.
- [86] P. Verboven, P. Guillaume, and B. Cauberghe. Multivariable frequency–response curve fitting with application to modal parameter estimation. *Automatica*, 41(10):1773–1782, 2005.
- [87] R. Wang and J. -M. Jin. Incorporation of multiport lumped networks into the hybrid time-domain finite-element analysis. *IEEE Trans. Microw. Theory Tech.*, 57(8):2030–2037, 2009.
- [88] W. River. Technology. CMP-28 Channel Modeling Platform. <https://wildrivertech.com/index.php/cmp-28-cmp-32>. Accessed: 2019-05-17.
- [89] N. Wong and C. -U. Lei. IIR approximation of FIR filters via discrete-time vector fitting. *IEEE Trans. Signal Process.*, 56(3):1296–1302, 2008.
- [90] T. -L. Wu, F. Buesink, and F. Canavero. Overview of signal integrity and EMC design technologies on PCB: Fundamentals and latest progress. *IEEE Trans. Electromagn. Compat.*, 55(4):624–638, 2013.
- [91] S. Yan, P. Wang, C. -Y. Tian, and L. Li. Analysis of graphene-based devices using wave equation-based discontinuous Galerkin time domain method. *IEEE Antennas Wirel. Propag. Lett.*, 17(12):2169–2173, 2018.
- [92] A. Zanco, S. Grivet-Talocia, T. Bradde, and M. De Stefano. Enforcing passivity of parameterized LTI macromodels via Hamiltonian-driven multivariate adaptive sampling. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 39(4):225–238, 2020.
- [93] K. Zhou, J. C. Doyle, and K. Glover. *Robust and optimal control*, volume 40. Prentice Hall, 1996.

9 Kernel methods for surrogate modeling

Abstract: This chapter deals with kernel methods as a special class of techniques for surrogate modeling. Kernel methods have proven to be efficient in machine learning, pattern recognition and signal analysis due to their flexibility, excellent experimental performance and elegant functional analytic background. These data-based techniques provide so called kernel-expansions, i. e., linear combinations of kernel functions which are generated from given input–output point samples that may be arbitrarily scattered. In particular, these techniques are meshless, do not require or depend on a grid, hence are less prone to the curse of dimensionality, even for high-dimensional problems.

In contrast to projection-based model reduction, we do not necessarily assume a high-dimensional model, but a general function that models input–output behavior within some simulation context. This could be some micro-model in a multiscale simulation, some submodel in a coupled system, some initialization function for solvers, coefficient function in Partial Differential Equations (PDEs), etc.

First, kernel surrogates can be useful if the input–output function is expensive to evaluate, e. g. as a result of a finite element simulation. Here, acceleration can be obtained by sparse kernel expansions. Second, if a function is available only via measurements or a few function evaluation samples, kernel approximation techniques can provide function surrogates that allow for global evaluation.


We present some important kernel approximation techniques, which are kernel interpolation, greedy kernel approximation and support vector regression. Pseudocode is provided for ease of reproducibility. In order to illustrate the main features, commonalities and differences, we compare these techniques on a real-world application. The experiments clearly indicate the enormous acceleration potential.

Keywords: regularized kernel interpolation, support vector regression, surrogate modeling, greedy approximation, reproducing kernel Hilbert spaces

MSC 2010: 65D05, 65D15, 46C05, 68T05

Acknowledgement: The authors acknowledge the support of the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075. The authors would like to thank Florian Rieg for the careful proofreading of this manuscript.

Gabriele Santin, Bernard Haasdonk, Institute of Applied Analysis and Numerical Simulation, University of Stuttgart, Stuttgart, Germany

Open Access. © 2021 Gabriele Santin and Bernard Haasdonk, published by De Gruyter.  This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

9.1 Introduction

This chapter deals with kernel methods as tools to construct surrogate models of arbitrary functions, given a finite set of arbitrary samples.

These methods generate approximants based solely on input–output pairs of the unknown function, without geometrical constraints on the sample locations. In particular, the surrogates do not necessarily depend on the knowledge of an high-dimensional model but only on its observed input–output behavior at the sample sites, and they can be applied on arbitrarily scattered points in high dimension.

These features are particularly useful when these methods are applied within some simulation context. For example, kernel surrogates can be useful if the input–output function is expensive to evaluate, e. g. is a result of a finite element simulation. Here, acceleration can be obtained by sparse kernel expansions. Moreover, if a function is available only via measurements or a few function evaluation samples, kernel approximation techniques can provide function surrogates that allow global evaluation.

Kernel methods are used with much success in Model Order Reduction, and far beyond the scope of this chapter. For example, they have been used in the modeling of geometry transformations and mesh coupling [3, 12, 13], and in mesh repair methods [33], or in the approximation of stability factors and error indicators [14, 32, 34], where only a few samples of the exact indicators are sufficient to construct an efficient surrogate to be used in the online phase. Moreover, kernel methods have been combined with projection-based MOR methods, e. g. to obtain simulation-based classification [60], or to derive multi-fidelity Monte Carlo approximations [40]. Kernel surrogates have been employed in optimal control problems [51, 59], in the coupling of multi-scale simulations in biomechanics [25, 69], in real time prediction for parameter identification and state estimation in biomechanical systems [29], in gas transport problems [22], in the reconstruction of potential energy surfaces [30], in the forecasting of time stepping methods [6], in the reduction of nonlinear dynamical systems [67], in uncertainty quantification [28], and for nonlinear balanced truncation of dynamical systems [5].

In further generality, there exist many kernel-based algorithms and application fields that we do not address here. Mainly, we address the solution of PDEs, in which several approaches have emerged in the last years, and which particularly allow one to solve problems with unstructured grids on general geometries, including high dimensional manifolds (see e. g. [11, 17]). Moreover, several other techniques are studied within Machine Learning, such as classification, density estimation, novelty detection or feature extraction (see e. g. [53, 54]).

Furthermore, we remark that these methods are members of the larger class of machine learning and approximation techniques, which are generally suitable to construct models based on samples to make prediction on new inputs. These models are

usually referred to as surrogates when they are then used as replacements of the model that generated the data, as they are able to provide an accurate and faster response. Some examples of these techniques are classical approximation methods such as polynomial interpolation, which are used in this context especially in combination with sparse grids to deal with high-dimensional problems (see [19]), and (deep) neural network models. The latter in particular have seen a huge increase in analysis and application in the recent years. For a recent treatment of deep learning, we refer e. g. to [21].

Despite these very diverse applications and methodologies, kernel methods can be analyzed to some extent in the common framework of Reproducing Kernel Hilbert spaces and, although the focus of this chapter will be on the construction of sparse surrogate models, parts of the following discussion can be the starting point for the analysis of other techniques.

In general terms, kernel methods can be viewed as nonlinear versions of linear algorithms. As an example, assume to have some set $X_n := \{x_k\}_{k=1}^n \subset \mathbb{R}^d$ of data points and target data values $Y_n := \{y_k\}_{k=1}^n \subset \mathbb{R}$. We can construct a surrogate $s : \mathbb{R}^d \rightarrow \mathbb{R}$ that predicts new data via linear regression, i. e., find $w \in \mathbb{R}^d$ s. t. $s(x) := \langle w, x \rangle$, where $\langle \cdot, \cdot \rangle$ is the scalar product in \mathbb{R}^d . A good surrogate model s will give predictions such that $|s(x_k) - y_k|$ is small. If we can write $w \in \mathbb{R}^d$ as $w = \sum_{j=1}^n \alpha_j x_j$ for a set of coefficients $(\alpha_i)_{i=1}^n \in \mathbb{R}^n$, then s can be rewritten as

$$s(x) := \sum_{j=1}^n \alpha_j \langle x_j, x \rangle.$$

Note that this formulation includes also regression with an offset (or bias) $b \neq 0$, which can be written in this form by an extended representation as

$$s(x) := \langle w, x \rangle + b =: \langle \bar{w}, \bar{x} \rangle,$$

where $\bar{x} := (x, 1)^T \in \mathbb{R}^{d+1}$ and $\bar{w} := (w, b)^T \in \mathbb{R}^{d+1}$.

Using now the Gramian matrix $A \in \mathbb{R}^{n \times n}$ with entries $A_{ij} := \langle x_i, x_j \rangle$ and rows $A_i^T \in \mathbb{R}^n$, we look for the surrogate s which minimizes

$$\sum_{i=1}^n (s(x_i) - y_i)_2^2 = \sum_{i=1}^n (A_i^T \alpha - y_i)_2^2 = \|A\alpha - y\|_2^2.$$

Additionally, a regularization term can be added to keep the norm of α small, e. g. in terms of the value $\alpha^T A \alpha$. Thus, the surrogate can be characterized as the solution of the optimization problem

$$\min_{\alpha \in \mathbb{R}^n} \|A\alpha - y\|_2^2 + \lambda \alpha^T A \alpha,$$

i. e., $\alpha = (A + \lambda I)^{-1} y$ if $\lambda > 0$.

In many cases this (regularized) linear regression is not sufficient to obtain a good surrogate. A possible idea is to try to combine this linear, simple method with a nonlinear function which maps the data to a higher dimensional space, where the hope is that the image of the data can be processed linearly. For this we consider a so-called feature map $\Phi : \mathbb{R}^d \rightarrow H$, where H is a Hilbert space, and apply the same algorithm to the transformed data $\Phi(X_n) := \{\Phi(x_i)\}_{i=1}^n$ with the same values Y_n . Since the algorithm depends on X_n only via the Gramian A , it is sufficient to replace it with the new Gramian $A_{ij} := \langle \Phi(x_i), \Phi(x_j) \rangle_H$ to obtain a nonlinear algorithm.

We will see that $\langle \Phi(x), \Phi(y) \rangle_H$ defines in fact a positive definite kernel, and if any numerical procedure can be written in terms of inner products of the inputs, it can be transformed in the same way into a new nonlinear algorithm simply by replacing the inner products with kernel evaluations (the so-called kernel trick). We will discuss the details of this procedure in the next sections in the case of interpolation and Support Vector Regression, but this immediately gives a glance of the ample spectrum of algorithms in the class of kernel methods.

This chapter is organized as follows. Section 9.2 covers the basic notions on kernels and kernel-based spaces which are necessary for the development and understanding of the algorithms. The next Section 9.3 presents the general ideas and tools to construct kernel surrogates as characterized by the Representer Theorem, and these ideas are specialized to the case of kernel interpolation in Section 9.4 and Support Vector Regression in Section 9.5. In both cases, we provide the theoretical foundations as well as the algorithmic description of the methods, with particular attention to techniques to enforce sparsity in the model. These surrogates can be used to perform various analyses of the full model, and we give some examples in Section 9.6. Section 9.7 presents a general strategy to choose the various parameters defining the model, whose tuning can be critical for a successful application of the algorithms. Finally, we discuss in Section 9.8 the numerical results of the methods on a real application dataset, comparing training time (offline), prediction time (online), and accuracy.

9.2 Background on kernels

We start by introducing some general facts of positive definite kernels. Further details on the general analytical theory of reproducing kernels can be found e. g. in the recent monograph [45], while the books [15, 65] and [53, 55] contain a treatment of kernel theory from the point of view of pattern analysis and scattered data interpolation, respectively.

9.2.1 Positive definite kernels

Given a nonempty set Ω , which can be a subset of \mathbb{R}^d , $d \in \mathbb{N}$, but also a set of structured objects such as strings or graphs, a real- and scalar-valued kernel K on Ω is a bivariate

symmetric function $K : \Omega \times \Omega \rightarrow \mathbb{R}$, i. e., $K(x, y) = K(y, x)$ for all $x, y \in \Omega$. For our purposes, we are interested in (strictly) positive definite kernels, defined as follows.

Definition 9.1 (Positive definite kernels). Let Ω be a nonempty set. A kernel K on Ω is positive definite (PD) on Ω if for all $n \in \mathbb{N}$ and for any set of n pairwise distinct elements $X_n := \{x_i\}_{i=1}^n \subset \Omega$, the kernel matrix (or Gramian matrix) $A := A_{K, X_n} \in \mathbb{R}^{n \times n}$ defined as $A_{ij} := K(x_i, x_j)$, $1 \leq i, j \leq n$, is positive semidefinite, i. e., for all vectors $\alpha := (\alpha_i)_{i=1}^n \in \mathbb{R}^n$ we have

$$\alpha^T A \alpha = \sum_{i,j=1}^n \alpha_i \alpha_j K(x_i, x_j) \geq 0. \quad (9.1)$$

The kernel is strictly positive definite (SPD) if the kernel matrix is positive definite, i. e., (9.1) holds with strict inequality when $\alpha \neq 0$.

The further class of conditionally (strictly) positive definite kernels is also of interest in certain contexts. We refer to [65, Chapter 8] for their extensive treatment, and we just mention that they are defined as above, except that the condition (9.1) has to be satisfied only for the subset of coefficients α which match a certain orthogonality condition. When this condition is defined with respect to a space of polynomials of degree $m \in \mathbb{N}$, the resulting kernels are used e. g. to guarantee a certain polynomial exactness of the given approximation scheme, and they are often employed in certain methods for the solution of PDEs.

9.2.2 Examples and construction of kernels

Despite the abstract definition, there are several ways to construct functions $K : \Omega \times \Omega \rightarrow \mathbb{R}$ which are (strictly) positive definite kernels, and usually the proper choice of the kernel is a crucial step in the successful application of the method. We list here a general strategy to construct kernels, and some notable examples.

An often used, constructive approach to designing a new kernel is via feature maps as follows.

Proposition 9.1 (Kernels via feature maps). Let Ω be a nonempty set. A feature map Φ is any function $\Phi : \Omega \rightarrow H$, where $(H, \langle \cdot, \cdot \rangle_H)$ is any Hilbert space (the feature space). The function

$$K(x, y) := \langle \Phi(x), \Phi(y) \rangle_H \quad x, y \in \Omega,$$

is a PD kernel on Ω .

Proof. K is a PD kernel since it is symmetric and positive definite, because the inner product is bilinear, symmetric and positive definite. \square

In many cases, H is either \mathbb{R}^m with very large m or even an infinite dimensional Hilbert space. The computation of the possibly expensive m - or infinite-dimensional inner product can be avoided if a closed form for K can be obtained. This implies a significant reduction of the computational time required to evaluate the kernel and thus to execute any kind of algorithm.

We see now some examples.

Example 9.1 (Expansion kernels). The construction comprises finite dimensional linear combinations, i. e., for a set of functions $\{v_j\}_{j=1}^m : \Omega \rightarrow \mathbb{R}$, the function $K(x, y) := \sum_{k=1}^m v_k(x)v_k(y)$ is a positive definite kernel, having a feature map

$$\Phi(x) := (v_1(x), v_2(x), \dots, v_m(x))^T \in H := \mathbb{R}^m. \quad (9.2)$$

This idea can be extended to an infinite number of functions provided $\{v_j(x)\}_{j=1}^\infty \in H := \ell_2(\mathbb{N})$ uniformly in Ω , and the resulting kernels are called Hilbert–Schmidt or expansion kernels, which can be proven to be even SPD under additional conditions (see [49]). As an example in $d = 1$, we mention the Brownian Bridge kernel $K(x, y) := \max(x, y) - xy$, defined with a feature map $v_j(x) := \sqrt{2(j\pi)^{-1}} \sin(j\pi x)$ for $j \in \mathbb{N}$, which is SPD on $\Omega := (0, 1)$. We remark that the kernel can be extended to $(0, 1)^d$ with $d > 1$ using a tensor product of one-dimensional kernels.

This feature map representation proves also that $\dim(H) =: m < \infty$ means that the kernel is not SPD in general: e. g., if X_n contains n pairwise distinct points and $m < n$, then the vectors $\{\Phi(x_i)\}_{i=1}^n$ cannot be linearly independent, and thus the kernel matrix is singular.

Example 9.2 (Kernels for structured data). Feature maps are also employed to construct positive definite kernels on sets Ω of structured data, such as sets of strings, graphs, or any other object. For example, the convolution kernels introduced in [20, 26] consider a finite set of features $v_1(x), \dots, v_m(x) \in \mathbb{R}$ of an object $x \in \Omega$, and define a feature map exactly as in (9.2).

Example 9.3 (Polynomial kernels). For $a \geq 0$, $p \in \mathbb{N}$, $x, y \in \mathbb{R}^d$, the polynomial kernel

$$K(x, y) := (\langle x, y \rangle + a)^p = \left(\sum_{i=1}^d x^{(i)} y^{(i)} + a \right)^p, \quad x := (x^{(1)}, \dots, x^{(d)})^T, \quad (9.3)$$

is PD on any $\Omega \subset \mathbb{R}^d$. It is a d -variate polynomial of degree p , which contains the monomial terms of degrees $j := (j^{(1)}, \dots, j^{(d)}) \in J$, for a certain set $J \subset \mathbb{N}_0^d$. If $m := |J|$, a feature space is \mathbb{R}^m with feature map

$$\Phi(x) := (\sqrt{a_1} x^{j_1}, \dots, \sqrt{a_m} x^{j_m})^T,$$

for some positive numbers $\{a_j\}_{j=1}^m$ and monomials $x^{j_m} := \prod_{i=1}^d (x^{(i)})^{j_m^{(i)}}$.

Observe that using the closed form (9.3) of the kernel instead of the feature map is very convenient, since we work with d -dimensional instead of m -dimensional vectors, where possibly $m := |J| = \binom{d+p}{d} = \dim(\mathbb{P}_p(\mathbb{R}^d)) \gg d$.

Example 9.4 (RBF kernels). For $\Omega \subset \mathbb{R}^d$ in many applications the most used kernels are translational invariant kernels, i. e., there exists a function $\phi : \mathbb{R}^d \rightarrow \mathbb{R}$ with

$$K(x, y) := \phi(x - y), x, y \in \Omega,$$

and in particular radial kernels, i. e., there exists a univariate function $\phi : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ with

$$K(x, y) := \phi(\|x - y\|), x, y \in \Omega.$$

A radial kernel, or Radial Basis Function (RBF), is usually defined up to a shape parameter $\gamma > 0$ that controls the scale of the kernel via $K(x, y) := \phi(\gamma\|x - y\|)$.

The main example of such kernels is the Gaussian $K(x, y) := e^{-\gamma^2\|x - y\|^2}$, which is in fact strictly positive definite. An explicit feature map has been computed in [56]: If $\Omega \subset \mathbb{R}^d$ is nonempty, a feature map is the function $\Phi_\gamma : \Omega \rightarrow L_2(\mathbb{R}^d)$ defined by

$$\Phi_\gamma(x) := \frac{(2\gamma)^{\frac{d}{2}}}{\pi^{\frac{d}{4}}} \exp(-2\gamma^2\|x - \cdot\|^2), \quad x \in \Omega.$$

In this case it is even more evident how working with the closed form of K is much more efficient than working with a feature map and computing L_2 -inner products.

RBF kernels offer a significant easiness of implementation in arbitrary space dimension d . The evaluation of the kernel $K(\cdot, x)$, $x \in \mathbb{R}^d$, on a vector of n points can indeed be realized by first computing a distance vector $D \in \mathbb{R}^n$, $D_i := \|x - x_i\|$, and then applying the univariate function ϕ on D . A discussion and comparison of different algorithms (in Matlab) to efficiently compute a distance matrix can be found in [15, Chapter 4], and most scientific computing languages comprise a built-in implementation (such as `pdist2`¹ in Matlab and `distance_matrix`² in Scipy).

Translational invariant and RBF kernels can be often analyzed in terms of their Fourier transforms, which provide proofs of their strict positive definiteness via the Bochner theorem (see e. g. [65, Chapter 6]), and connections to certain Sobolev spaces, as we will briefly see in Section 9.2.3.

Among various RBF kernels, there are also compactly supported kernels, i. e., $K(x, y) = 0$ if $\|x - y\| > 1/\gamma$, which produce sparse kernel matrices if γ is large enough. The most used ones are the Wendland kernels introduced in [63], which are even radial polynomial within their support.

¹ <https://www.mathworks.com/help/stats/pdist2.html>

² https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance_matrix.html

There are, in addition, various operations to combine positive definite kernels and obtain new ones. For example, sums and products of positive definite kernels and multiplication by a positive constant $a > 0$ produce again positive definite kernels. Moreover, if K' is a positive definite kernel and K'' is symmetric with $K' \preceq K''$ (i. e., $K := K'' - K'$ is PD) then also K'' is positive definite. Furthermore, if $\Omega = \Omega' \times \Omega''$ and K', K'' are PD kernels on Ω', Ω'' , then $K(x, y) := K'(x', y')K''(x'', y'')$ and $K(x, y) := K'(x', y') + K''(x'', y'')$ are also PD kernels on Ω , i. e., kernels can be defined to respect tensor product structures of the input.

Further details and examples can be found in [45, Chapters 1–2].

9.2.3 Kernels and Hilbert spaces

Most of the analysis of kernel-based methods is possible through the connection with certain Hilbert spaces. We first give the following definition.

Definition 9.2 (Reproducing Kernel Hilbert Space). Let Ω be a nonempty set, \mathcal{H} an Hilbert space of functions $f : \Omega \rightarrow \mathbb{R}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. Then \mathcal{H} is called a Reproducing Kernel Hilbert Space (RKHS) on Ω if there exists a function $K : \Omega \times \Omega \rightarrow \mathbb{R}$ (the reproducing kernel) such that

1. $K(\cdot, x) \in \mathcal{H}$ for all $x \in \Omega$,
2. $\langle f, K(\cdot, x) \rangle_{\mathcal{H}} = f(x)$ for all $x \in \Omega, f \in \mathcal{H}$ (reproducing property).

The reproducing property is equivalent to state that, for $x \in \Omega$, the x -translate $K(\cdot, x)$ of the kernel is the Riesz representer of the evaluation functional $\delta_x : \mathcal{H} \rightarrow \mathbb{R}$, $\delta_x(f) := f(x)$ for $f \in \mathcal{H}$, which is hence a continuous functional in \mathcal{H} . Also the converse holds, and the following result gives an abstract criterion to check if a Hilbert space is a RKHS.

Theorem 9.1. *An Hilbert space of functions $\Omega \rightarrow \mathbb{R}$ is a RKHS if and only if the point evaluation functionals are continuous in \mathcal{H} for all $x \in \Omega$, i. e., $\delta_x \in \mathcal{H}'$, the dual space of \mathcal{H} . Moreover, the reproducing kernel K of \mathcal{H} is strictly positive definite if and only if the functionals $\{\delta_x : x \in \Omega\}$ are linearly independent in \mathcal{H}' .*

Proof. The first part is clear from the reproducing property, while strict positive definiteness can be checked by verifying that the quadratic form in Definition 9.1 cannot be zero for $\alpha \neq 0$ if $\{\delta_x : x \in \Omega\}$ are linearly independent. \square

We see two concrete examples.

Example 9.5 (Finite dimensional spaces). Any finite dimensional Hilbert space \mathcal{H} of functions on a nonempty set Ω is a RKHS. If $m := \dim(\mathcal{H})$ and $\{v_j\}_{j=1}^m$ is an orthonormal

basis, then a reproducing kernel is given by

$$K(x, y) := \sum_{j=1}^m v_j(x)v_j(y), \quad x, y \in \Omega.$$

Indeed, the two properties of Definition 9.2 can be easily verified by direct computation.

Example 9.6 (The Sobolev space $H_0^1(0, 1)$). The Sobolev space $H_0^1(0, 1)$ with inner product $\langle f, g \rangle_{H_0^1} := \int_0^1 f'(y)g'(y) dy$ is a RKHS with the Brownian Bridge kernel

$$K(x, y) := \min(x, y) - xy, \quad x, y \in (0, 1)$$

as reproducing kernel (see e. g. [8]). Indeed, $K(\cdot, x) \in H_0^1(0, 1)$, and the reproducing property (2) follows by explicitly computing the inner product.

The following result proves that reproducing kernels are in fact positive definite kernels in the sense of Definition 9.1. Moreover, the first two properties are useful to deal with the various type of approximants of Section 9.4 and Section 9.5, which will be exactly of this form.

Proposition 9.2. *Let \mathcal{H} be a RKHS on Ω with reproducing kernel K . Let $n, n' \in \mathbb{N}, \alpha \in \mathbb{R}^n, \alpha' \in \mathbb{R}^{n'}, X_n, X_{n'}' \subset \Omega$, and define the functions*

$$f(x) := \sum_{i=1}^n \alpha_i K(x, x_i), \quad g(x) := \sum_{j=1}^{n'} \alpha'_j K(x, x'_j), \quad x \in \Omega.$$

Then we have the following:

1. $f, g \in \mathcal{H}$,
2. $\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^n \sum_{j=1}^{n'} \alpha_i \alpha'_j K(x_i, x'_j)$.
3. K is the unique reproducing kernel of \mathcal{H} and it is a positive definite kernel.

Proof. The first two properties follow from Definition 9.2, and in particular from \mathcal{H} being a linear space and from the bilinearity of $\langle \cdot, \cdot \rangle_{\mathcal{H}}$.

For Property (3), the fact that K is symmetric and positive definite, hence a PD kernel, follows from Property (1) of Definition 9.2, and from the symmetry and positive definiteness of the inner product. Moreover, the reproducing property implies that, if K, K' are two reproducing kernels of \mathcal{H} , then for all $x, y \in \Omega$ we have

$$K(x, y) = \langle K(\cdot, y), K'(\cdot, x) \rangle_{\mathcal{H}} = K'(x, y).$$

□

It is common in applications to follow instead the opposite path, i. e., to start with a given PD kernel, and try to see if an appropriate RKHS exists. This is in fact always the case, as proven by the following fundamental theorem from [2].

Theorem 9.2 (RKHS from kernels – Moore–Aronszajn theorem). *Let Ω be a nonempty set and $K : \Omega \times \Omega \rightarrow \mathbb{R}$ a positive definite kernel. Then there exists a unique RKHS $\mathcal{H} := \mathcal{H}_K(\Omega)$ with reproducing kernel K .*

Proof. The theorem was first proven in [2], to which we refer for a detailed proof. The idea is to deduce that, by Property (1) of Proposition 3, a candidate RKHS \mathcal{H} of K needs to contain the linear space

$$\mathcal{H}_0 := \text{span} \{K(\cdot, x) : x \in \Omega\}$$

of finite linear combinations of kernel translates. Moreover, from Property (2) of Proposition 9.2, the inner product on this \mathcal{H}_0 needs to satisfy

$$\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^n \sum_{j=1}^{n'} \alpha_i \alpha'_j K(x_i, x'_j). \quad (9.4)$$

With this observation in mind, the idea of the construction of \mathcal{H} is to start by \mathcal{H}_0 , prove that (9.4) defines indeed an inner product on \mathcal{H}_0 , and that the completion of \mathcal{H}_0 w. r. t. this inner product is a RKHS having K as reproducing kernel. Uniqueness then follows from Property (3) of the same proposition. \square

As it is common in the approximation literature, we will sometimes refer to this unique \mathcal{H} as the native space of the kernel K on Ω .

Remark 9.1 (Kernel feature map). Among other consequences, this construction allows one to prove that any PD kernel is generated by at least one feature map. Indeed, the function $\Phi : \Omega \rightarrow \mathcal{H}$, $\Phi(x) := K(\cdot, x)$, is clearly a feature map for K with feature space \mathcal{H} , since the reproducing property implies that

$$\langle \Phi(x), \Phi(y) \rangle_{\mathcal{H}} = \langle K(\cdot, x), K(\cdot, y) \rangle_{\mathcal{H}} = K(x, y) \quad \text{for all } x, y \in \Omega.$$

Remark 9.2. For certain translational invariant kernels it is possible to prove that the associated native space is norm equivalent to a Sobolev spaces of the appropriate smoothness, which is related to the kernels' smoothness (see [65, Chapter 10]). This is particularly interesting since the approximation properties of the different algorithms, including certain optimality that we will see in the next sections, are in fact optimal in these Sobolev spaces (with an equivalent norm).

The various operations on positive definite kernels mentioned in Section 9.2.2 have an analogous effect on the corresponding native spaces. For example, the scaling by a positive number $a > 0$ does not change the native space, but scales the inner product correspondingly, and, if $K' \leq K''$ are positive definite kernels, then $\mathcal{H}_{K'}(\Omega) \subset \mathcal{H}_{K''}(\Omega)$. We remark that the latter property has been used for example in [71] to prove inclusion relations for the native spaces of RBF kernels with different shape parameters.

9.2.4 Kernels for vector-valued functions

So far we only dealt with scalar-valued kernels, which are suitable to treat scalar-valued functions. Nevertheless, it is clear that the interest in model reduction is typically also on vector-valued or multi-output functions, which thus require a generalization of the theory presented so far. This has been done in [35], and it is based on the following definition of matrix-valued kernels.

Definition 9.3 (Matrix-valued PD kernels). Let Ω be a nonempty set and $q \in \mathbb{N}$. A function $K : \Omega \times \Omega \rightarrow \mathbb{R}^{q \times q}$ is a matrix-valued kernel if it is symmetric, i. e., $K(x, y) = K(y, x)^T$ for all $x, y \in \Omega$. It is a PD (resp., SPD) matrix-valued kernel if the kernel matrix $A \in \mathbb{R}^{nq \times nq}$ is positive semidefinite (resp., positive definite) for all $n \in \mathbb{N}$ and for all sets $X_n \subset \Omega$ of pairwise distinct elements.

This more general class of kernels is also associated to a uniquely defined native space of vector-valued functions, where the notion of RKHS is replaced by the following.

Definition 9.4 (RKHS for matrix-valued kernels). Let Ω be a nonempty set, $q \in \mathbb{N}$, \mathcal{H} an Hilbert space of functions $f : \Omega \rightarrow \mathbb{R}^q$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. Then \mathcal{H} is called a vector-valued RKHS on Ω if there exists a function $K : \Omega \times \Omega \rightarrow \mathbb{R}^{q \times q}$ (the matrix-valued reproducing kernel) such that

1. $K(\cdot, x)v \in \mathcal{H}$ for all $x \in \Omega, v \in \mathbb{R}^q$,
2. $\langle f, K(\cdot, x)v \rangle_{\mathcal{H}} = f(x)^T v$ for all $x \in \Omega, v \in \mathbb{R}^q, f \in \mathcal{H}$ (directional reproducing property).

A particularly simple version of this construction can be realized by considering separable matrix-valued kernels (see e. g. [1]), i. e., kernels that are defined as $K(x, y) := \tilde{K}(x, y)B$, where \tilde{K} is a standard scalar-valued PD kernel, and $B \in \mathbb{R}^{q \times q}$ is a positive semidefinite matrix. In the special case $Q = I$ (the $q \times q$ identity matrix), in [70] it is shown that the native space of K is the tensor product of q copies of the native space of \tilde{K} , i. e.,

$$\mathcal{H}_K(\Omega) = \{f : \Omega \rightarrow \mathbb{R}^q : f_j \in \mathcal{H}_{\tilde{K}}(\Omega), 1 \leq j \leq q\}$$

with

$$\langle f, g \rangle_{\mathcal{H}_K} = \sum_{j=1}^q \langle f_j, g_j \rangle_{\mathcal{H}_{\tilde{K}}}.$$

This simplification will give convenient advantages when implementing some of the methods discussed in Section 9.4.

9.3 Data based surrogates

We can now introduce in general terms the two surrogate modeling techniques that we will discuss, namely (regularized) kernel interpolation and Support Vector Regression (SVR).

For both of them, the idea is to represent the expensive map to be reduced as a function $f : \Omega \rightarrow \mathbb{R}^q$ that maps an input $x \in \Omega$ to an output $y \in \mathbb{R}^q$. Here f is assumed to be only continuous, and the set Ω can be arbitrary as long as a positive definite kernel K can be defined on it. Moreover, the function does not need to be known in any particular way except than through its evaluations on a finite set $X_n := \{x_k\}_{k=1}^n \subset \Omega$ of pairwise distinct data points, resulting in data values $Y_n := \{y_k := f(x_k)\}_{k=1}^n \subset \mathbb{R}^q$.

The goal is to construct a function $s \in \mathcal{H}$ such that $s(x)$ is a good approximation of $f(x)$ for all $x \in \Omega$ (and not only for $x \in X_n$), while being significantly faster to evaluate. The process of computing s from the data (X_n, Y_n) is often referred to as training of the surrogate s , and the set (X_n, Y_n) is thus called training dataset.

The computation of the particular surrogate is realized as the solution of an infinite dimensional optimization problem. In general terms, we define a loss function

$$L : \mathcal{H} \times \Omega^n \times (\mathbb{R}^q)^n \rightarrow \mathbb{R}_{\geq 0} \cup \{+\infty\},$$

which takes as input a candidate surrogate $g \in \mathcal{H}$ and the values $X_n \in \Omega^n$, $Y_n \in (\mathbb{R}^q)^n$, and returns a measure of the data-accuracy of g . Then the surrogate s is defined as a minimizer, if it exists, of the cost function

$$J(g) := L(g, X_n, Y_n) + \lambda \|g\|_{\mathcal{H}}^2,$$

where the second part of J is a regularization term that penalizes solutions with large norm. The tradeoff between the data-accuracy term and the regularization term is controlled by the regularization parameter $\lambda \geq 0$.

For the sake of presentation, we restrict in the remaining of this section to the case of scalar-valued functions, i. e., $q = 1$. The general case follows by using matrix valued kernels as introduced in Section 9.2.4, and the corresponding definition of orthogonal projections.

The following fundamental Representer Theorem characterizes exactly some solutions of this problem, and it proves that the surrogate will be a function

$$s \in V(X_n) := \text{span} \{K(\cdot, x_i), x_i \in X_n\}$$

i. e., a finite linear combination of kernel translates on the training points. A first version of this result was proven in [27], while we refer to [52] for a more general statement.

Theorem 9.3 (Representer Theorem). *Let Ω be a nonempty set, K a PD kernel on Ω , $\lambda > 0$ a regularization parameter, and let (X_n, Y_n) be a training set. Assume that $L(s, X_n, Y_n)$ depends on s only via the values $s(x_i)$, $x_i \in X_n$.*

Then, if the optimization problem

$$\operatorname{argmin}_{g \in \mathcal{H}} J(g) := L(g, X_n, Y_n) + \lambda \|g\|_{\mathcal{H}}^2 \quad (9.5)$$

has a solution, it has in particular a solution of the form

$$s(x) := \sum_{j=1}^n \alpha_j K(x, x_j), \quad x \in \Omega, \quad (9.6)$$

for suitable coefficients $\alpha \in \mathbb{R}^n$.

Proof. We prove that for every $g \in \mathcal{H}$ there exists $s \in V(X_n)$ such that $J(s) \leq J(g)$. To see this, we decompose $g \in \mathcal{H}$ as

$$g = s + s^\perp, \quad s \in V(X_n), \quad s^\perp \in V(X_n)^\perp.$$

In particular, since $K(\cdot, x_i) \in V(X_n)$, we have by the reproducing property of the kernel

$$s^\perp(x_i) = \langle s^\perp, K(\cdot, x_i) \rangle_{\mathcal{H}} = 0, \quad 1 \leq i \leq n,$$

thus $g(x_i) = s(x_i) + s^\perp(x_i) = s(x_i)$ for $1 \leq i \leq n$, and it follows that $L(g, X_n, Y_n) = L(s, X_n, Y_n)$. Moreover, again by orthogonal projection we have $\|g\|_{\mathcal{H}}^2 = \|s\|_{\mathcal{H}}^2 + \|s^\perp\|_{\mathcal{H}}^2$. Since $\lambda \geq 0$, we obtain

$$\begin{aligned} J(s) &= L(s, X_n, Y_n) + \lambda \|s\|_{\mathcal{H}}^2 = L(g, X_n, Y_n) + \lambda \|s\|_{\mathcal{H}}^2 \\ &= L(g, X_n, Y_n) + \lambda \|g\|_{\mathcal{H}}^2 - \lambda \|s^\perp\|_{\mathcal{H}}^2 = J(g) - \lambda \|s^\perp\|_{\mathcal{H}}^2 \leq J(g). \end{aligned}$$

Thus, if $g \in \mathcal{H}$ is a solution then $s \in V(X_n)$ is also a solution. \square

The existence of a solution will be guaranteed by choosing a convex cost function J , i. e., since the regularization term is always convex, by choosing a convex loss function. Then the theorem states that solutions of the infinite dimensional optimization problem can be computed by solving a finite dimensional convex one.

This is a great result, but observe that the evaluation of $s(x)$, $x \in \Omega$, requires the evaluation of the n -terms linear combination (9.6), where n is the size of the dataset. Assuming that the kernel can be evaluated in constant time, the complexity of this operation is $\mathcal{O}(n)$. Thus, to achieve the promised speedup in evaluating the surrogate in place of the function f , we will consider in the following methods that enforce sparsity in s , i. e., which compute approximate solution where most of the coefficients α_i are zero. If the nonzero coefficients correspond to an index set $I_N := \{i_1, \dots, i_N\} \subset \{1, \dots, n\}$, the complexity is reduced to $\mathcal{O}(N)$.

Taking into account this sparsity and denoting $X_N := \{x_i \in X_n : i \in I_N\}$ and $\alpha := (\alpha_i : i \in I_N)$, we can summarize in Algorithm 9.1 the online phase for any of the following algorithms, consisting in the evaluation of s on a set of points $X_{\text{te}} \subset \Omega$. Here and in the following, we denote by $s(X) := (s(x_1), \dots, s(x_m))^T \in \mathbb{R}^m$ the vector of evaluations of s on a set of points $X := \{x_i\}_{i=1}^m \subset \Omega$.

Algorithm 9.1: Kernel surrogate – online phase.

-
- 1: Input: $X_N \in \Omega^N, \alpha \in \mathbb{R}^N$, kernel K (and kernel parameters), test points $X_{te} := \{x_i^{te}\}_{i=1}^{n_{te}} \in \Omega^{n_{te}}$
 - 2: Compute the kernel matrix $A_{te} \in \mathbb{R}^{n_{te} \times N}$, $(A_{te})_{ij} := K(x_i^{te}, x_j)$.
 - 3: Evaluate the surrogate $s(X^{te}) = A_{te}\alpha$.
 - 4: Output: evaluation of the surrogate $s(X^{te}) \in \mathbb{R}^{n_{te}}$.
-

Remark 9.3 (Normalization of the cost function). It is sometimes convenient to weight the loss term in the cost function (9.5) by a factor $1/n$, which normalizes its value with respect to the number of data. We do not use this convention here, and we only remark that this is equivalent to the use of a regularization parameter $\lambda = n\lambda'$ for a given $\lambda' > 0$.

9.4 Kernel interpolation

The first method that we discuss is (regularized) kernel interpolation. In this case, we consider the square loss function

$$L(s, X_n, Y_n) := \sum_{i=1}^n (s(x_i) - y_i)^2,$$

which measures the pointwise distance between the surrogate and the target data. We have then the following special case of the Representer Theorem. We denote by $y \in \mathbb{R}^n$ the vector of output data, assuming again for now that $q = 1$.

Corollary 9.1 (Regularized interpolant). *Let Ω be a nonempty set, K a PD kernel on Ω , $\lambda \geq 0$ a regularization parameter. For any training set (X_n, Y_n) there exists an approximant of the form*

$$s(x) = \sum_{j=1}^n \alpha_j K(x, x_j), \quad x \in \Omega, \quad (9.7)$$

where the vector of coefficients $\alpha \in \mathbb{R}^n$ is a solution of the linear system

$$(A + \lambda I)\alpha = y, \quad (9.8)$$

where $A \in \mathbb{R}^{n \times n}$, $A_{ij} := K(x_i, x_j)$, is the kernel matrix on X_n . Moreover, if K is SPD this is the unique solution of the minimization problem (9.5).

Proof. The loss L is clearly convex, so there exists a solution of the optimization problem, and by Theorem 9.3 we know that we can restrict to solutions in $V(X_n)$.

We then consider functions $s := \sum_{j=1}^n \alpha_j K(\cdot, x_j)$ for some unknown $\alpha \in \mathbb{R}^n$. Computing the inner product as in Proposition 9.2, we obtain

$$s(x_i) = \sum_{j=1}^n \alpha_j K(x_i, x_j) = (A\alpha)_i, \quad \|s\|_{\mathcal{H}}^2 = \sum_{i,j=1}^n \alpha_i \alpha_j K(x_i, x_j) = \alpha^T A \alpha.$$

The functional J restricted to $V(X_n)$ can be parametrized by $\alpha \in \mathbb{R}^n$, and thus it can be rewritten as $\tilde{J} : \mathbb{R}^n \rightarrow \mathbb{R}$ with

$$\begin{aligned} \tilde{J}(\alpha) &= \|A\alpha - y\|_2^2 + \lambda \alpha^T A \alpha = (A\alpha - y)^T (A\alpha - y) + \lambda \alpha^T A \alpha \\ &= \alpha^T A^T A \alpha - 2\alpha^T A^T y + y^T y + \lambda \alpha^T A \alpha, \end{aligned}$$

which is convex in α since A is positive semidefinite. Since A is symmetric, its gradient is

$$\nabla_{\alpha} \tilde{J}(\alpha) = 2A^T A \alpha - 2A^T y + 2\lambda A \alpha = 2A(A\alpha - y + \lambda \alpha),$$

i. e., $\nabla_{\alpha} \tilde{J}(\alpha) = 0$ if and only if $A(A + \lambda I)\alpha = Ay$, which is satisfied by α such that $(A + \lambda I)\alpha = y$. If K is SPD then both A and $A + \lambda I$ are invertible, so this is the only solution. \square

The extension to vector-valued functions, i. e. $q > 1$, is straightforward using the separable matrix-valued kernels with $B = I$ of Section 9.2.4. Indeed, in this case the data values are vectors $y_i := f(x_i) \in \mathbb{R}^q$, and thus in the interpolant (9.7) also the coefficients are vectors $\alpha_j \in \mathbb{R}^q$. The linear system (9.8) has the same matrix, but instead $\alpha, y \in \mathbb{R}^{n \times q}$ are defined as

$$\alpha := (\alpha_1, \dots, \alpha_n)^T, \quad y := (y_1, \dots, y_n)^T. \quad (9.9)$$

We remark that in the following the values $x_i, y_i, s(x)$, and α_k have always to be understood as row vectors when $q > 1$. This notation is very convenient when representing the coefficients as the solution of a linear system. Furthermore, the representation of the dataset samples (x, y) is quite natural when dealing with tabular data, where each column represents a feature and each row a sample vector.

For K SPD and pairwise distinct sample locations X_n we can also set $\lambda := 0$ and obtain pure interpolation, i. e., the solution satisfies $L(s, X_n, Y_n) = 0$, or

$$s(x_i) = y_i, \quad 1 \leq i \leq n.$$

Observe that this means that with this method we can exactly interpolate arbitrary continuous functions on arbitrary pairwise distinct scattered data in any dimension, as opposite to many other techniques which require complicated conditions on the interpolation points or a grid structure. Moreover, this approximation process has several optimality properties in \mathcal{H} , which remind one of similar properties of spline interpolation.

Proposition 9.3 (Optimality of kernel interpolation). *Let K be SPD, $f \in \mathcal{H}$, and $\lambda = 0$. Then s is the orthogonal projection of f in $V(X_n)$, and in particular*

$$\|f - s\|_{\mathcal{H}} = \min_{g \in V(X_n)} \|f - g\|_{\mathcal{H}}.$$

Moreover, if $S := \{g \in \mathcal{H} : g(x_i) = f(x_i), 1 \leq i \leq n\}$, then

$$\|s\|_{\mathcal{H}} = \min_{g \in S} \|g\|_{\mathcal{H}},$$

i. e., s is the minimal norm interpolant of f on X_n .

Proof. The proof is analogous to the proof of the Representer Theorem, using a decomposition $f = g + g^\perp$, and proving that $s = g$. \square

We will see in Section 9.7 a general technique to tune λ using the data, which should return $\lambda = 0$ (or very small) when this is the best option. Nevertheless, also for an SPD kernel there are at least two reasons to still consider regularized interpolation. First, the data can be affected by noise, and thus an exact pointwise recovery does not make much sense. Second, a positive parameter $\lambda > 0$ improves the condition number of the linear system, and thus the stability of the solution. Indeed, the 2-condition number of $A + \lambda I$ is

$$\kappa(\lambda) := \frac{\lambda_{\max}(A + \lambda I)}{\lambda_{\min}(A + \lambda I)} = \frac{\lambda_{\max}(A) + \lambda}{\lambda_{\min}(A) + \lambda},$$

which is a strictly decreasing function of λ , with $\kappa(0) = \kappa(A)$ and $\lim_{\lambda \rightarrow \infty} \kappa(\lambda) = 1$. Moreover (see [66]) this increased stability can be achieved by still controlling the pointwise accuracy. Namely, if $f \in \mathcal{H}$, we have

$$\|y_i - s(x_i)\|_2 \leq \sqrt{\lambda} \|f\|_{\mathcal{H}} \quad 1 \leq i \leq n.$$

We can then summarize the offline phase for regularized kernel interpolation in Algorithm 9.2.

Algorithm 9.2: Regularized Kernel interpolation – offline phase.

- 1: Input: training set $X_n \in \Omega^n$, $Y_n \in (\mathbb{R}^q)^n$, kernel K (and kernel parameters), regularization parameter $\lambda \geq 0$.
 - 2: Compute the kernel matrix $A \in \mathbb{R}^{n \times n}$, $A_{ij} := K(x_i, x_j)$.
 - 3: Solve the linear system $(A + \lambda I)\alpha = y$.
 - 4: Output: coefficients $\alpha \in \mathbb{R}^{n \times q}$.
-

Remark 9.4 (Flat limit). The matrix A can be seriously ill-conditioned for certain kernels, and this constitutes a problem at least in the case of pure interpolation. It can also be proven that kernels which guarantee a faster error convergence result in a worse conditioned matrix [48].

For RBF kernels, this happens especially for $\gamma \rightarrow 0$ (the so called flat limit), and it is usually not a good idea to directly solve the linear system. In the last years there has been very active research to compute s via different formulations, which rely on different representations of the kernel. We mention here mainly the RBF-QR algorithm³ [18, 31] and the Hilbert–Schmidt SVD⁴ [16]. Both methods are limited so far to only some kernels, but they manage to achieve a great accuracy, which is usually impossible to obtain with the direct solution of the linear system.

Remark 9.5 (Error estimation). For SPD translational invariant kernels there is a very detailed error analysis of the interpolation process ($\lambda = 0$), for which we refer to [65, Chapter 11]. We only mention that these error bounds assume that $f \in \mathcal{H}$, and are of the form

$$\|f - s\|_{L_\infty(\Omega)} \leq Ch_n^p \|f\|_{\mathcal{H}},$$

where $C > 0$ is a constant independent of f , and h_n is the fill distance of X_n in Ω , i. e.,

$$h_n := h_{X_n, \Omega} := \sup_{x \in \Omega} \min_{x_j \in X_n} \|x - x_j\|,$$

which is the analogue of the mesh width for scattered data. Moreover, the order of convergence $p > 0$ is dependent on the smoothness of the kernel. In particular, these error bounds can be proven to be optimal when the native space of K is a Sobolev space.

Moreover, these results have been recently extended to the case of regularized interpolation ($\lambda > 0$) in [43, 66].

9.4.1 Kernel greedy approximation

The surrogate constructed via Corollary 9.1 involves a linear combination of n terms, where n is the size of the dataset. In general, there is no reason to assume that the result has any sparsity, i. e., in general all the α_j will be nonzero, and it is thus necessary to introduce some technique to enforce this sparsity.

A very effective way to achieve this result is via greedy algorithms. The idea is to select a small subset $X_N \subset X_n$, $N \ll n$, given by indices $I_N \subset \{1, \dots, n\}$, and to solve the

³ http://www.it.uu.se/research/scientific_computing/software/rbf_qr

⁴ <http://math.iit.edu/~mccomic/gaussqr/>

corresponding restricted problem with the dataset (X_N, Y_N) to compute a surrogate

$$s_N(x) := \sum_{k \in I_N} \alpha_k K(x, x_k), \quad (9.10)$$

where the coefficient vectors are computed based on (9.8), and are in general different from the ones of the full surrogate. If we manage to select I_N in a proper way, we will obtain $s_N(x) \approx f(x)$ for all $x \in \Omega$, while the evaluation of $s_N(x)$ is now only of order $\mathcal{O}(N)$.

An optimal selection of X_N is a combinatorial problem and thus is very expensive and in practice computationally intractable. The idea of greedy algorithms is instead to perform this selection incrementally, i. e., adding at each iteration only the most promising new point, based on some error indicator.

The general structure of the algorithm is described in Algorithm 9.3. For the moment, we consider a generic selection rule $\eta : X_n \times \mathbb{N} \times \Omega^n \times (\mathbb{R}^q)^n \rightarrow \mathbb{R}_{\geq 0}$ that selects points based on the value $\eta(x, N, X_n, Y_n)$. This is a compact notation to denote that the selection rule assigns a score to a point $x \in \Omega$, and it is computed using various quantities that depend on the dataset (X_n, Y_n) and on the iteration number N , including in particular the surrogate computed at the previous iteration. The algorithm is terminated by means of a given tolerance $\tau > 0$.

Algorithm 9.3: Kernel greedy approximation – offline phase.

- 1: Input: training set $X_n \in \Omega^n$, $Y_n \in (\mathbb{R}^q)^n$, kernel K (and kernel parameters), regularization parameter $\lambda \geq 0$, selection rule η , tolerance τ .
 - 2: Set $N := 0$, $X_0 := \emptyset$, $V(X_0) := \{0\}$, $s_0 := 0$.
 - 3: **repeat**
 - 4: Set $N := N + 1$
 - 5: Select $x_N := \operatorname{argmax}_{x \in X_n \setminus X_{N-1}} \eta(x, N, X_n, Y_n)$.
 - 6: Define $X_N := X_{N-1} \cup \{x_N\}$ and $V(X_N) := \operatorname{span} \{K(\cdot, x_i), x_i \in X_N\}$
 - 7: Compute the surrogate s_N with dataset (X_N, Y_N) with (9.8).
 - 8: **until** $\eta(x_N, N, X_n, Y_n) \leq \tau$
 - 9: Output: surrogate s_N (i. e. coefficients $\alpha \in \mathbb{R}^{N \times q}$).
-

Remark 9.6. In the case that the maximizer of η the line 5 of Algorithm 9.3 is not unique, only one of the multiple points is selected and included in X_N .

In line 7 of the algorithm, we need to compute the surrogate s_N with dataset (X_N, Y_N) . This step can be highly simplified by reusing s_{N-1} as much as possible, thus improving the efficiency of the algorithm. As a side effect, with this incremental procedure it is easy to update the surrogate if the accuracy has to be improved.

This can be achieved using the Newton basis, which is defined in analogy to the Newton basis for polynomial interpolation. It has been introduced in [37, 39] for K SPD, and extended to the case of K PD and $\lambda > 0$ in [47], and we refer to these papers for the proof of the following result.

Proposition 9.4 (Newton basis). *Let Ω be non empty, $\lambda \geq 0$, K be PD on Ω or SPD when $\lambda = 0$. Let $X_n \subset \Omega$ be pairwise distinct, and let $I_N \subset \{1, \dots, n\}$. Let moreover $K_\lambda(x, y) := K(x, y) + \lambda \delta_{xy}$ for all $x, y \in \Omega$, and denote its RKHS as \mathcal{H}_λ .*

The Newton basis $\{v_j\}_{j=1}^N$ is defined as the Gram–Schmidt orthonormalization of $\{K_\lambda(\cdot, x_i)\}_{i \in I_N}$ in \mathcal{H} , i. e.,

$$\begin{aligned} v_1(x) &:= \frac{K_\lambda(x, x_{i_1})}{\|K_\lambda(\cdot, x_{i_1})\|_{\mathcal{H}_\lambda}} = \frac{K_\lambda(x, x_{i_1})}{\sqrt{K_\lambda(x_{i_1}, x_{i_1})}}, \\ \tilde{v}_k(x) &:= K_\lambda(x, x_{i_k}) - \sum_{j=1}^{k-1} v_j(x_{i_k}) v_j(x), \\ v_k(x) &:= \frac{\tilde{v}_k(x)}{\|\tilde{v}_k\|_{\mathcal{H}_\lambda}} = \frac{\tilde{v}_k(x)}{\sqrt{\tilde{v}_k(x_{i_k})}}, \quad 1 < k \leq N. \end{aligned}$$

Moreover, for all $1 \leq k \leq N$, we have

$$v_k(x) = \sum_{j=1}^N \beta_{jk} K_\lambda(x, x_{i_j}),$$

and, if $B \in \mathbb{R}^{N \times N}$, $B_{jk} := \beta_{jk}$, and $V \in \mathbb{R}^{N \times N}$, $V_{jk} := v_k(x_{i_j})$, then B, V are triangular, $B = V^{-T}$, and

$$A_N + \lambda I = VV^T$$

is the Cholesky decomposition of the regularized kernel matrix $A_N + \lambda I \in \mathbb{R}^{N \times N}$, $A_{jk} := K(x_{i_j}, x_{i_k})$, with pivoting given by I_N .

Observe that this basis is nested, i. e., we can incrementally add a new element without recomputing the previous ones. Even more, with this basis the surrogate can be computed as follows.

Proposition 9.5 (Incremental regularized interpolation). *Let Ω be non empty, $\lambda \geq 0$, K be PD on Ω or SPD when $\lambda = 0$. Let (X_N, Y_N) be the subset of (X_n, Y_n) corresponding to indices I_N , for all $N \leq n$.*

Let $\tilde{s}_0 := 0$, and, for $N \geq 1$, compute the following incremental function

$$\tilde{s}_N(x) = \sum_{k=1}^N c_k v_k(x) = c_N v_N(x) + s_{N-1}(x), \quad c_N := \frac{y_{i_N} - \tilde{s}_N(x_{i_N})}{v_N(x_{i_N})}. \quad (9.11)$$

Then, for all N , the regularized interpolant can be computed as

$$s_N(x) = \sum_{j=1}^N \alpha_j K(x, x_{i_j}) \quad \text{where } \alpha := V^{-T}c.$$

Remark 9.7. In the case $\lambda = 0$ and K SPD, the function \tilde{s}_N coincides with the interpolant s_N . We refer to [39, 47] for the details.

We are now left to define the selection rules, represented by η , to select the new point at each iteration.

For this, we first need to define the power function, which gives an upper bound on the interpolation error, and it can be defined using the Newton basis as

$$P_N(x)^2 := K_\lambda(x, x) - \sum_{j=1}^N v_j(x)^2. \quad (9.12)$$

Its relevance is due to the fact that it provides an upper bound on the pointwise (regularized) interpolation error, i. e., if $x \notin X_n$, and K is PD, or SPD when $\lambda = 0$, we have for all $f \in \mathcal{H}$ that

$$|f(x) - s_N(x)| \leq P_n(x) \|f\|_{\mathcal{H}}. \quad (9.13)$$

This function is well known and has been studied in the case of pure interpolation (see e. g. [65, Chapter 11]), for which the upper bound holds for all $x \in \Omega$, and it can be easily extended to the case of regularized interpolation (see [47]). In both cases, it can be proven that $P_n(x) = 0$ if and only if $x \in X_n$, and its maximum is strictly decreasing with N .

Remark 9.8. This interpolation technique is strictly related to the kriging method and to Gaussian Process Regression (see e. g. [38, 42]). In this case the kernel represents the covariance kernel of the prior distribution, and the power function is the Kriging variance, or variance of the posterior distribution (see [50]).

We can then define the following selection rules. We assume to have a dataset (X_n, Y_n) , and to have already selected N points corresponding to indices I_{N-1} . We use the notation $[1, n] := \{1, \dots, n\}$, and we have

- P -greedy: $i_N := \operatorname{argmax}_{i \in [1, n] \setminus I_{N-1}} P_{N-1}(x_i)$;
- f -greedy: $i_N := \operatorname{argmax}_{i \in [1, n] \setminus I_{N-1}} |y_i - s_{N-1}(x_i)|$;
- f/P -greedy: $i_N := \operatorname{argmax}_{i \in [1, n] \setminus I_{N-1}} \frac{|y_i - s_{N-1}(x_i)|}{P_{N-1}(x_i)}$.

Observe that all the selections are well defined, since $P_{N-1}(x_i) \neq 0$ for all $i \notin I_{N-1}$ if X_N are pairwise distinct, and they can be efficiently implemented by using the update rules (9.11) for s_N and (9.12) for P_N . Moreover, they are motivated by different ideas: The P -greedy selection tries to minimize the Power function, thus providing a uniform upper bound on the error for any function $f \in \mathcal{H}$ via (9.13); the f - and f/P -greedy (which

reads “ f -over- P -greedy”), on the other hand, use also the output data, and produce points which are suitable to approximate a single function and thus are expected to result in a better approximation. In the case of f -greedy this is done by including in the set of points the location where the current largest error is achieved, thus reducing the maximum error. The f/P -greedy selection, instead, reduces the error in the \mathcal{H} -norm, and indeed it can be proven to be locally optimal, i. e., it guarantees the maximal possible reduction of the error, in the \mathcal{H} -norm, at each iteration.

We can now describe the full computation of the greedy regularized interpolant in Algorithm 9.4. It realizes the computation of the sparse surrogate s_N by selecting the points X_N via the index set I_N , and computing only the nonzero coefficients α . Moreover, using the nested structure of the Newton basis and the incremental computation of Proposition 9.5, the algorithm needs only to compute the columns of the full kernel matrix corresponding to the index set I_N , and thus there is no need to compute nor store the full $n \times n$ matrix, i. e., the implementation is matrix-free. In addition, again using Proposition 9.5 most of the operations are done in-place, i. e., some vectors are used to store and update the values of the Power Function and of y . In the algorithm, we use a Matlab-like notation, i. e., $A(I_N, :)$ denotes the submatrix of A consisting of rows I_N and of all the columns. Moreover, the notation v^2 denotes the pointwise squaring of the entries of the vector v .

Algorithm 9.4: Kernel greedy approximation – offline phase.

- 1: Input: training set $X_n \in \Omega^n$, $Y_n \in (\mathbb{R}^q)^n$, kernel K (and kernel parameters), regularization parameter $\lambda \geq 0$, selection rule η , tolerance τ .
 - 2: Set $N := 0$, $I_0 := \emptyset$, $V := [\cdot] \in \mathbb{R}^{n \times 0}$, $p := \text{diag}(K_\lambda(X_n, X_n)) \in \mathbb{R}^n$
 - 3: **repeat**
 - 4: Set $N = N + 1$
 - 5: Select $i_N := \text{argmax}_{i \in [1, n] \setminus I_{N-1}} \eta(x_i, N, X_n, Y_n)$.
 - 6: Generate column $v := K_\lambda(X_n, x_{i_N})$
 - 7: Project $v := v - VV(i_N, :)^T$
 - 8: Normalize $v = v / \sqrt{v(i_N)}$
 - 9: Compute $c_N := y(i_N) / v(i_N)$
 - 10: Update the power function $p := p - v^2$
 - 11: Update the residual $y := y - c_N v$
 - 12: Update $I_N := I_{N-1} \cup \{i_N\}$
 - 13: Add the column $V = [V, v_N]$
 - 14: Update the inverse $C^T = V(I_N, :)^{-1}$
 - 15: Add the coefficient $c = [c^T, c_N]^T$
 - 16: **until** $\eta(x_N, N, X_n, Y_n) \leq \tau$
 - 17: Set $\alpha = Cc$
 - 18: Output: $\alpha \in \mathbb{R}^{N \times q}$, I_N .
-

The set of points X_N defined by I_N , and the coefficients α , can then be used in the online phase of Algorithm 9.1.

Remark 9.9 (Vector-valued functions and implementation details). Algorithm 9.4 and the overall procedure are well defined for arbitrary $q \geq 1$. Indeed, using the separable matrix-valued kernel of Section 9.2.4, the Newton basis only depends on the scalar-valued kernel K , while the computation of the coefficients is valid by considering that now c, α are matrices instead of vectors. In particular, the computation of c_N (line 14) and the update of y (line 11) has to be done via column-wise multiplications.

Moreover, observe that in line 12 we employ a standard technique to update the inverse of a lower triangular matrix, i. e., given $V_N \in \mathbb{R}^{N \times N}$ lower triangular with inverse V_N^{-1} , we define

$$V_{N+1} = \begin{bmatrix} V_N & 0 \\ v^T & w \end{bmatrix}$$

for $v \in \mathbb{R}^N$, $w \in \mathbb{R}$, and compute V_{N+1}^{-1} by a simple row-update as

$$V_{N+1}^{-1} = \begin{bmatrix} V_N^{-1} & 0 \\ -v^T V_N^{-1}/w & 1/w \end{bmatrix}.$$

The present version of the algorithm for vector-valued functions has been introduced in [68] and named Vectorial Kernel Orthogonal Greedy Algorithm (VKOGA). We keep the same abbreviation also for the regularized version, which has been studied in [47].

Remark 9.10 (Convergence rates). When the greedy algorithm is run by selecting points over Ω instead of X_N , there are also convergence rates for the resulting approximation processes. For pure interpolation (i. e., K SPD, $\lambda = 0$) convergence of f -greedy has been proven in [36], of P -greedy in [46], and of f/P -greedy in [68], while in [47] the convergence rate of P -greedy has been extended to regularized interpolation. All the results make additional assumptions on the kernels, for which we refer to the cited literature. Nevertheless, we remark that the convergence rates for interpolation with P -greedy are quasi-optimal for translational invariant kernels, while the results for the other algorithms guarantee only a possibly significantly slower convergence rate. These results are believed to be significantly sub-optimal, since extensive experiments indicate that f - and f/P -greedy cases behave much better. This seems to suggest that there is space for a large improvement in the theoretical understanding of the methods.

Remark 9.11 (Other techniques). There are other techniques that can be applied to reduce the complexity of the evaluation of the surrogate s , which do not use greedy algorithms but instead different approaches. First, there is a domain decomposition technique, known as Partition of Unity Method, which partitions Ω into subdomains,

solves the (regularized) interpolation problem restricted to each patch, and then combines the results by a weighted sum of the local interpolants to obtain a global approximant. This method has the advantage that this offline phase can be completely parallelized. Moreover, when evaluating the surrogate only the few local interpolant having a support containing the test points have to be evaluated, thus requiring the evaluation of a few, small kernel expansions, thus providing a significant speedup. The efficiency of this technique relies on an efficient search procedure to determine the local patches including the given points, which is the only limitation in the application to high dimensional problems. Both theoretical results and efficient implementations are available [7, 64].

Moreover, other sparsity-inducing techniques have been proposed, namely, the use of an ℓ_1 -regularization term [10], and the method of the Least Absolute Shrinkage and Selection Operator (LASSO) [61].

9.5 Support vector regression

The second method that we present is Support Vector Regression (SVR) [53], which is based on different premises, but it still fits in the general framework of Section 9.3. In this case, we consider the ε -insensitive loss function

$$L(s, X_n, Y_n) := \sum_{i=1}^n L_\varepsilon(s(x_i), y_i), \quad L_\varepsilon(s(x_i), y_i) := \max(0, |s(x_i) - y_i| - \varepsilon),$$

which is designed to linearly penalize functions s which have values outside of an ε -tube around the data, while no distinction is made between function values that are inside this tube.

In this setting it is common to use the regularization parameter to scale the cost by a factor $1/\lambda$, and not the regularization term by a factor λ . The two choices are clearly equivalent, but we adopt here this different normalization to facilitate the comparison with the existing literature, and because this offers additional insights in the structure of the surrogate.

Since the problem is not quadratic (and not smooth), we first derive an equivalent formulation of the optimization problem (9.5). Assuming again that the output is scalar, i. e., $q = 1$, the idea is to introduce non-negative slack variables $\xi^+, \xi^- \in \mathbb{R}^n$ which represent upper bounds on L via

$$\begin{aligned} \xi_i^+ &\geq \max(0, s(x_i) - y_i - \varepsilon), \quad 1 \leq i \leq n, \\ \xi_i^- &\geq \max(0, y_i - s(x_i) - \varepsilon), \quad 1 \leq i \leq n, \end{aligned} \tag{9.14}$$

and to minimize them in place of the original loss. With these new variables we can rewrite the optimization problem in the following equivalent way.

Definition 9.5 (SVR – primal form). Let Ω be a nonempty set, K a PD kernel on Ω , $\lambda > 0$ a regularization parameter. For a training set (X_n, Y_n) the SVR approximant $(s, \xi^+, \xi^-) \in \mathcal{H} \times \mathbb{R}^{2n}$ is a solution of the quadratic optimization problem

$$\begin{aligned} \min_{s \in \mathcal{H}, \xi^+, \xi^- \in \mathbb{R}^n} & \frac{1}{\lambda} \mathbb{1}_n^T (\xi^+ + \xi^-) + \|s\|_{\mathcal{H}}^2 \\ \text{s. t. } & s(x_i) - y_i - \varepsilon \leq \xi_i^+, \quad 1 \leq i \leq n \\ & -s(x_i) + y_i - \varepsilon \leq \xi_i^-, \quad 1 \leq i \leq n \\ & \xi_i^+, \xi_i^- \geq 0, \quad 1 \leq i \leq n, \end{aligned} \quad (9.15)$$

where $\mathbb{1}_n := (1, \dots, 1)^T \in \mathbb{R}^n$.

For this rewriting of the optimization problem, we can now specialize the Representer Theorem as follows.

Corollary 9.2 (SVR – alternative primal form). Let Ω be a nonempty set, K a PD kernel on Ω , $\lambda > 0$ a regularization parameter. For any training set (X_n, Y_n) there exists an SVR approximant of the form

$$s(x) = \sum_{j=1}^n \alpha_j K(x, x_j), \quad x \in \Omega, \quad (9.16)$$

where $(\alpha, \xi^+, \xi^-) \in \mathbb{R}^{3n}$ is a solution of the quadratic optimization problem

$$\begin{aligned} \min_{\alpha, \xi^+, \xi^- \in \mathbb{R}^n} & \frac{1}{\lambda} \mathbb{1}_n^T (\xi^+ + \xi^-) + \alpha^T A \alpha \\ \text{s. t. } & (A\alpha)_i - y_i - \varepsilon \leq \xi_i^+, \quad 1 \leq i \leq n \\ & -(A\alpha)_i + y_i - \varepsilon \leq \xi_i^-, \quad 1 \leq i \leq n \\ & \xi_i^+, \xi_i^- \geq 0, \quad 1 \leq i \leq n, \end{aligned} \quad (9.17)$$

with $\mathbb{1}_n := (1, \dots, 1)^T \in \mathbb{R}^n$, and $A \in \mathbb{R}^{n \times n}$, $A_{ij} := K(x_i, x_j)$, the kernel matrix on X_n . Moreover, if K is SPD this is the unique solution of the minimization problem (9.5).

Proof. The result is an immediate consequence of Proposition 9.5, where we use the form (9.16) for s and compute its squared norm via Proposition 9.2. \square

The slack variables (9.14) have a nice geometric interpretation. Indeed, the optimization process clearly tries to reduce their value as much as possible, while respecting the constraints. We state a more precise result in the following proposition, and give a schematic illustration in Figure 9.1.

Proposition 9.6 (Slack variables). Let $\alpha, \xi^+, \xi^- \in \mathbb{R}^n$ be a solution of (9.17), and let s be the corresponding surrogate (9.16). Then, for each index $i \in \{1, \dots, n\}$, the values ξ_i^+, ξ_i^- represent the distance of $s(x_i)$ from the ε -tube around y_i , and in particular

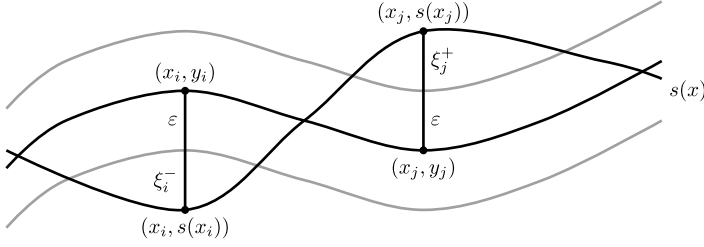


Figure 9.1: Illustration of the role of the slack variables in (9.17).

1. If $s(x_i) > y_i + \varepsilon$, then $\xi_i^+ > 0$ and $\xi_i^- = 0$.
2. If $s(x_i) < y_i - \varepsilon$, then $\xi_i^+ = 0$ and $\xi_i^- > 0$.
3. If $y_i - \varepsilon \leq s(x_i) \leq y_i + \varepsilon$, then $\xi_i^+ = 0$ and $\xi_i^- = 0$.

In particular, only one of ξ_i^+ and ξ_i^- can be nonzero.

Instead of solving the primal problem of Corollary 9.2, it is more common to derive and solve the following dual problem. Here again we denote by $y \in \mathbb{R}^n$ the vector of all scalar training target values.

Proposition 9.7 (SVR – dual form). *Let Ω be a nonempty set, K a PD kernel on Ω , $\lambda > 0$ a regularization parameter. For any training set (X_n, Y_n) there exists a solution $(\alpha^+, \alpha^-) \in \mathbb{R}^{2n}$ of the problem*

$$\begin{aligned} \min_{\alpha^+, \alpha^- \in \mathbb{R}^{2n}} & \frac{1}{4}(\alpha^- - \alpha^+)^T A(\alpha^- - \alpha^+) + \varepsilon \mathbb{1}_n^T(\alpha^+ + \alpha^-) + y^T(\alpha^+ - \alpha^-) \\ \text{s. t. } & \alpha^+, \alpha^- \in [0, 1/\lambda]^n, \end{aligned} \quad (9.18)$$

which is unique if K is SPD. Moreover, a solution of (9.17) is given by

$$s(x) := \sum_{j=1}^n \frac{\alpha_j^- - \alpha_j^+}{2} K(x, x_j), \quad x \in \Omega, \quad (9.19)$$

with $\xi_i^+ := \max(0, s(x_i) - y_i - \varepsilon)$, $\xi_i^- := \max(0, y_i - s(x_i) - \varepsilon)$.

Proof. We give a sketch of the proof, although a formal derivation requires more care, and we refer to [53, Chapter 9] for the details. The idea is to first derive the Lagrangian $\mathcal{L} := \mathcal{L}(\alpha, \xi^+, \xi^-, \alpha^+, \alpha^-, \mu^+, \mu^-)$ for the primal problem (9.17) using non-negative Lagrange multipliers $\alpha^+, \alpha^-, \mu^+, \mu^- \in \mathbb{R}^n$ for the inequality constraints, and then derive the dual problem by imposing the Karush–Kuhn–Tucker (KKT) conditions (see e.g. Chapter 6 in [53]).

The Lagrangian is defined as

$$\begin{aligned} \mathcal{L} = & \frac{1}{\lambda} \mathbb{1}_n^T(\xi^+ + \xi^-) + \alpha^T A \alpha + (\mu^+)^T(-\xi^+) + (\mu^-)^T(-\xi^-) \\ & + (A \alpha - y - \varepsilon \mathbb{1}_n - \xi^+)^T \alpha^+ + (y - A \alpha - \varepsilon \mathbb{1}_n - \xi^-)^T \alpha^- \end{aligned} \quad (9.20)$$

$$\begin{aligned}
&= (\alpha + \alpha^+ - \alpha^-)^T A \alpha + \left(\frac{1}{\lambda} \mathbb{1}_n - \alpha^+ - \mu^+ \right)^T \xi^+ + \left(\frac{1}{\lambda} \mathbb{1}_n - \alpha^- - \mu^- \right)^T \xi^- \\
&\quad - \varepsilon \mathbb{1}_n^T (\alpha^+ + \alpha^-) - y^T (\alpha^+ - \alpha^-).
\end{aligned}$$

Using the symmetry of A , the partial derivatives of \mathcal{L} with respect to the primal variables can be computed as

$$\nabla_{\alpha} \mathcal{L} = 2A\alpha + A(\alpha^+ - \alpha^-), \quad \nabla_{\xi^+} \mathcal{L} = \frac{1}{\lambda} \mathbb{1}_n - \alpha^+ - \mu^+, \quad \nabla_{\xi^-} \mathcal{L} = \frac{1}{\lambda} \mathbb{1}_n - \alpha^- - \mu^-, \quad (9.21)$$

and setting these three equalities to zero we obtain equations for α, μ^+, μ^- , where in particular $\alpha = \frac{1}{2}(\alpha^- - \alpha^+)$ (which is the unique solution if A is invertible). Substituting these values in the Lagrangian we get

$$\begin{aligned}
\mathcal{L} &= (\alpha + \alpha^+ - \alpha^-)^T A \alpha - \varepsilon \mathbb{1}_n^T (\alpha^+ + \alpha^-) - y^T (\alpha^+ - \alpha^-) \\
&= -\frac{1}{4}(\alpha^- - \alpha^+)^T A (\alpha^- - \alpha^+) - \varepsilon \mathbb{1}_n^T (\alpha^+ + \alpha^-) - y^T (\alpha^+ - \alpha^-).
\end{aligned}$$

The remaining conditions in (9.18) stem from the requirements that the Lagrange multipliers are non-negative, and in particular $0 \leq \mu_i^+ = 1/\lambda - \alpha_i^+$, i. e., $\alpha_i^+ \leq 1/\lambda$, and similarly for α_i^- . \square

This dual formulation is particularly convenient to explain that the SVR surrogate has a built-in sparsity, i. e., the optimization process provides a solution where possibly many of the entries of $\alpha = \frac{1}{2}(\alpha^- - \alpha^+)$ are zero. This behavior is in strong contrast with the case of interpolation of Section 9.4 where we needed to adopt special techniques to enforce this property. The points $x_i \in X_n$ with $\alpha_i \neq 0$ are called support vectors, which gives the name to the method.

In particular, as for the slack variables there is a clean geometric description of this sparsity pattern, this gives additional insights into the solution. To see this we remark that, in addition to the stationarity KKT conditions (9.21), an optimal solution satisfies also the complementarity KKT conditions

$$\alpha_i^+ (s(x_i) - y_i - \varepsilon - \xi_i^+) = 0, \quad \alpha_i^- (y_i - s(x_i) - \varepsilon - \xi_i^-) = 0, \quad (9.22)$$

$$\xi_i^+ (1/\lambda - \alpha_i^+) = 0, \quad \xi_i^- (1/\lambda - \alpha_i^-) = 0. \quad (9.23)$$

We then have the following:

1. Equation (9.22) states that $\alpha_i^+ \neq 0$ only if $s(x_i) - y_i - \varepsilon - \xi_i^+ = 0$, and similarly for α_i^- . Since $\xi_i^+ \geq 0$, this happens only when $s(x_i) - y_i \geq \varepsilon$, i. e., only for points $(x_i, s(x_i))$ which are outside or on the boundary of the ε -tube.
2. In particular, if $\alpha_i^+ \neq 0$ it follows that $s(x_i) - y_i \geq \varepsilon$, and thus $y_i - s(x_i) - \varepsilon - \xi_i^- \neq 0$, and then necessarily $\alpha_i^- = 0$. Thus, at most one of α_i^+ and α_i^- can be nonzero.
3. Equation (9.23) implies that $\alpha_i^+, \alpha_i^- = 1/\lambda$ whenever ξ_i^+, ξ_i^- is nonzero, i. e., whenever $s(x_i)$ is strictly outside of the ε -tube. The corresponding x_i are called bounded

support vectors, and the value of the corresponding coefficients is indeed kept bounded by the value of the regularization parameter. Reducing λ , i. e., using less regularization, allows solutions with coefficients of larger magnitude.

In summary, we can then expect that, if s is a good approximation of the data, it will be also a sparse approximation.

We summarize the offline phase for SVR in Algorithm 9.5. We remark that in this case the extension to vector-valued functions is not as straightforward as for kernel interpolation, and it is thus common to train a separate SVR for each output component.

Algorithm 9.5: SVR – offline phase.

- 1: Input: training set $X_n \in \Omega^n$, $Y_n \in \mathbb{R}^n$, kernel K (and kernel parameters), regularization parameter $\lambda \geq 0$, tube width $\varepsilon > 0$.
 - 2: Compute the kernel matrix $A \in \mathbb{R}^{n \times n}$, $A_{ij} := K(x_i, x_j)$.
 - 3: Solve the quadratic problem (9.18).
 - 4: Set $I_N := \{i : \alpha_i^- \neq 0 \text{ or } \alpha_i^+ \neq 0\}$.
 - 5: Set $\alpha_i := (\alpha_i^- - \alpha_i^+)/2$ for $i \in I_N$.
 - 6: Output: $\alpha \in \mathbb{R}^N$, I_N .
-

Remark 9.12 (General Support Vector Machines). SVR is indeed one member of a vast collection of algorithms related to Support Vector Machines (SVMs). Standard SVMs solve classification problems, i. e., $Y_n \subset \{0, 1\}$. The original algorithm has been introduced as a linear algorithm (or, in the present understanding, as limited to the linear kernel, i. e., the polynomial kernel with $a = 0$, $p = 1$), and it has later been extended via the kernel trick to its general kernel version in [4]. The SVR algorithms have instead been introduced in [53].

Moreover, the version presented here is usually called ε -SVR. There exists also another non equivalent version called ν -SVR, which adds another term in the cost function multiplied by a factor $\nu \in [0, 1]$. This plays the role of giving an upper bound on the number of support vectors and on the fraction of training data which are outside of the ε -tube (see Chapter 9 in [53]).

We also remark that it is sometimes common to include in any SVM-based algorithm also an offset or bias term $b \in \mathbb{R}$, i. e., to obtain a surrogate $s(x) = \sum_{j=1}^n \alpha_j K(x, x_j) + b$. This changes in an obvious way the primal problem (9.17), while the dual contains also the constraint $\sum_{i=1}^n (\alpha_i^+ + \alpha_i^-) = 0$. However, we stick here to this formulation and refer to [57] for a discussion of statistical and numerical benefits of not using this offset term, at least in the case of SPD kernels.

Remark 9.13 (Error estimation). Also for SVR there is a detailed error theory, usually formulated in the framework of statistical learning theory (see [62]). Results are ob-

tained by assuming that the dataset (X_n, Y_n) is drawn from a certain unknown probability distribution, and then quantifying the approximation power of the surrogate. For a detailed treatment of this theory, we refer to [53, 55]. Moreover, recently also deterministic error bounds for translational invariant kernels have been proven in [43, 44].

9.5.1 Sequential minimal optimization

Although the optimization problem (9.18) can in principle be solved with any quadratic optimization method, there exists a special algorithm, called Sequential Minimal Optimization (SMO) that is designed for SVMs and that performs possibly much better.

SMO is an iterative method which improves an initial feasible guess for $\alpha^+, \alpha^- \in \mathbb{R}^n$ until convergence, and the update is made such that the minimal possible number of entries of α are affected. In this way, very large problems can be efficiently solved. The original version of the algorithm has been introduced in [41] for SVM, and it has later been adapted to more general methods such as SVR, which we use here to illustrate the structure of its implementation.

The idea is to find at each iteration $\ell \in \mathbb{N}$ a minimal set of indices $I^\ell \subset \{1, \dots, n\}$ and optimize only the variables with indices in I^ℓ . The procedure is then iterated until the optimum is reached. If the SVR includes an offset term, as explained in the previous section we have constraints

$$\begin{aligned} \alpha_i^+, \alpha_i^- &\in [0, 1/\lambda], \quad 1 \leq i \leq n, \\ \sum_{i=1}^n (\alpha_i^+ + \alpha_i^-) &= 0. \end{aligned} \tag{9.24}$$

Given a feasible solution $(\alpha_i^+, \alpha_i^-)^{(\ell)}$ at iteration $\ell \in \mathbb{N}$, it is thus not possible to update a single entry of α_i^+ or α_i^- without violating the KKT conditions (since at most one between α_i^+ and α_i^- need to be nonzero) or violating the second constraint. It is instead possible to select two indices $I^\ell := \{i, j\}$ and in this case we have variables $\alpha_i^+, \alpha_i^-, \alpha_j^+, \alpha_j^-$ and we can solve the restricted quadratic optimization problem under the constraints

$$\alpha_i^+, \alpha_i^- \in [0, 1/\lambda], i \in I^\ell, \quad \sum_{i \in I^\ell} (\alpha_i^+ + \alpha_i^-) = R^\ell := - \sum_{i \notin I^\ell} (\alpha_i^+ + \alpha_i^-),$$

which can be solved analytically.

The crucial step is to select I^ℓ , and this is done by finding a first index that does not satisfy the KKT conditions and a second one with some heuristic. It can be proven that, if at least one of the two violates the KKT conditions, then the objective is strictly decreased and convergence is obtained. Moreover, the vectors $\alpha^+ = \alpha^- = 0 \in \mathbb{R}^n$ are always feasible and can thus be used as a first guess. In practice, the iteration is stopped when a sufficiently small value of the cost function is reached.

In the case of SVR without offset discussed in the previous section the situation is even simpler, since the second constraint in (9.24) is not present and it is thus possible to update a single pair (α_i^+, α_i^-) at each iteration. Nevertheless, it has been proven in [57] that using also in this case two indices improves significantly the speed of convergence. Moreover, the same paper introduces several additional details to select the pair, to optimize the restricted cost function, and to establish termination conditions.

A general version of SMO for SVR is summarized in Algorithm 9.6, where we assume that the function $\eta : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ implements the selection rule of I^ℓ .

Algorithm 9.6: SMO.

- 1: Input: training set $X_n \in \Omega^n$, $Y_n \in \mathbb{R}^n$, kernel K (and kernel parameters), regularization parameter $\lambda \geq 0$, tube width $\varepsilon > 0$, selection rule η , tolerance τ .
 - 2: Set $\ell := 0$ and $(\alpha^+, \alpha^-)^{(0)} := (0, 0)$.
 - 3: **while** $(\alpha^+, \alpha^-)^{(\ell)}$ does not satisfy KKT conditions within tolerance τ . **do**
 - 4: Set $\ell = \ell + 1$.
 - 5: Set $I^\ell := \{i, j\} := \eta(\{1, \dots, n\})$.
 - 6: Set $(\alpha_k^+, \alpha_k^-)^{(\ell)} := (\alpha_k^+, \alpha_k^-)^{(\ell-1)}$ for $k \notin I^\ell$.
 - 7: Solve the optimization problem restricted to I^ℓ .
 - 8: **end while**
 - 9: Set $I_N := \{i : \alpha_i^- \neq 0 \text{ or } \alpha_i^+ \neq 0\}$.
 - 10: Set $\alpha_i := (\alpha_i^- - \alpha_i^+)/2$ for $i \in I_N$.
 - 11: Output: $\alpha \in \mathbb{R}^N$, I_N .
-

Remark 9.14 (Reference implementations). We remark that there exist commonly used implementations of SVR (and other SVM-related algorithms), which are available in several programming languages and implement also some version of this algorithm. We mention especially LIBSVM⁵ [9] and liquidSVM⁶ [58].

9.6 Model analysis using the surrogate

Apart from predicting new inputs with good accuracy and a significant speedup, the surrogate model can be used to perform a variety of different tasks related to meta-modeling, such as uncertainty quantification and state estimation. This can be done in a non-intrusive way, meaning that the full model is employed as a black-box that provides input–output pairs to train the surrogate, but is not required to be modified.

⁵ <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>

⁶ <https://github.com/liquidSVM/liquidSVM>

In principle, any kind of analysis that requires multiple evaluations can be significantly accelerated by the use of a surrogate, including the ones that are not computationally feasible due to the high computational cost of the full model. An example is uncertainty quantification, where the expected value of f can be approximated by a Monte Carlo integration of s using a set $X_m \subset \Omega$ of integration points, i. e.,

$$\int_{\Omega} f(x) dx \approx \frac{1}{m} \sum_{i=1}^m s(x_i).$$

Once the surrogate is computed using a training set (X_n, Y_n) , this approximate integral can be evaluated also for $m \gg n$ with a possibly very small cost, since the evaluation of s is significantly cheaper than the one of f .

Another example, which we describe in detail in the following, is the solution of an inverse problem to estimate the input parameter which generated a given output, i. e., from a given vector $\bar{y} \in \mathbb{R}^q$ we want to estimate $x \in \Omega$ such that $f(\bar{x}) = \bar{y}$. This can be done by considering a state-estimation cost function $C : \Omega \rightarrow \mathbb{R}$ defined by

$$C(x) := \frac{1}{2\|\bar{y}\|_2^2} \|s(x) - \bar{y}\|_2^2, \quad (9.25)$$

and estimating \bar{x} by the value x^* defined as

$$x^* := \min_{x \in \Omega} C(x).$$

In principle, we could perform the same optimization also using f instead of s in (9.25), but the surrogate allows a rapid evaluation of C . Moreover, if K is at least differentiable, then also s is differentiable, and thus we can use gradient-based methods to minimize C .

To detail this approach, we assume $f : \Omega \rightarrow \mathbb{R}^q$ and to have a surrogate obtained as in Section 9.4.1 with the separable matrix-valued kernel of Section 9.2.4, i. e., from (9.10) we have

$$s_N(x) = \sum_{k \in I_N} \alpha_k K(x, x_k).$$

As explained in (9.9), in the vector-valued case $q > 1$ we always assume that the output $s_N(x)$ and the coefficients α_k are row vectors, and in particular $\alpha \in \mathbb{R}^{N \times q}$ and $s_N(x) \in \mathbb{R}^{1 \times q}$. In this case we have the following.

Proposition 9.8 (Gradient of the state-estimation cost). *For $x \in \Omega \subset \mathbb{R}^d$ and $\bar{y} \in \mathbb{R}^q$, the gradient of the cost (9.25) can be computed in $x \in \Omega$ as*

$$\nabla C(x) = \frac{1}{\|\bar{y}\|_2^2} (D\alpha) E^T,$$

where $D \in \mathbb{R}^{d \times N}$ with columns $D_j := \nabla_x K(x, x_j)$, and $E := s_N(x) - \bar{y} \in \mathbb{R}^{1 \times q}$.

Proof. By linearity, the gradient of s_N in x can be computed as

$$\nabla s_N(x) = \sum_{j=1}^n \alpha_j \nabla_x K(x, x_j) = D\alpha \in \mathbb{R}^{d \times q},$$

and thus

$$\begin{aligned} \nabla C(x) &= \frac{1}{\|\bar{y}\|_2^2} (s_N(x) - \bar{y}) \nabla_x (s_N(x) - \bar{y}) = \frac{1}{\|\bar{y}\|_2^2} (s_N(x) - \bar{y}) \nabla s(x) \\ &= \frac{1}{\|\bar{y}\|_2^2} (D\alpha) E^T. \end{aligned} \quad \square$$

Observe in particular that whenever K is known in closed form the matrix D can be explicitly computed, and thus the gradient can be assembled using only matrix-vector multiplications of matrices of dimensions N, d, q , but independent of n . The solution x^* can then be computed by any gradient-based optimization method, and each iteration can be performed in an efficient way.

9.7 Parameter and model selection

For all the methods that we have seen the approximation quality of the surrogate depends on several parameters, which need to be carefully chosen to obtain good results. There are both parameters defining the kernel, such as the shape parameter $\gamma > 0$ in a RBF kernel, and model parameters such as the regularization parameter $\lambda \geq 0$. To some extent, also the selection of the kernel itself can be considered as a parametric dependence of the model. Moreover, it is essential to test the quality of the surrogate on an independent test set of data, since tuning it on the training set alone can very likely lead to overfitting, i. e., to obtain a model that is excessively accurate on the training set, while failing to generalize its prediction capabilities to unseen data.

In practical applications the target function f is unknown, so it cannot be used to check if the approximation is good, and all we know is the training set (X_n, Y_n) . In this case the most common approach is to split the sets into train, validation and test sets in the following sense. We permute (X_n, Y_n) , fix numbers $n_{\text{tr}}, n_{\text{val}}, n_{\text{te}}$ such that $n = n_{\text{tr}} + n_{\text{val}} + n_{\text{te}}$, and define a partition of the dataset as

$$\begin{aligned} X_{\text{tr}} &:= \{x_i, 1 \leq i \leq n_{\text{tr}}\}, \\ X_{\text{val}} &:= \{x_i, n_{\text{tr}} + 1 \leq i \leq n_{\text{tr}} + n_{\text{val}}\}, \\ X_{\text{te}} &:= \{x_i, n_{\text{tr}} + n_{\text{val}} + 1 \leq i \leq n\}, \end{aligned}$$

and similarly for $Y_{\text{tr}}, Y_{\text{val}}, Y_{\text{te}}$.

The idea is then to use the validation set $(X_{\text{val}}, Y_{\text{val}})$ to validate (i. e., choose) the parameters, and the test set $(X_{\text{te}}, Y_{\text{te}})$ to evaluate the error. Having disjoint sets allows one to have a fair way to test the algorithm.

For the process we also need an error function that returns the error of the surrogate s evaluated on a generic set of points $X := \{x_i\}_i \subset \Omega$ w. r. t. the exact values $Y := \{y_i\}_i$. We denote by $|X|$ the number of elements of X . Common examples are the maximal error and the Root Mean Square Error (RMSE) defined as

$$E(s, X, Y) := \max_{1 \leq i \leq |X|} \|s(x_i) - y_i\|_2 \quad \text{or} \quad E(s, X, Y) := \sqrt{\frac{1}{|X|} \sum_{i=1}^{|X|} \|s(x_i) - y_i\|_2^2}. \quad (9.26)$$

Then one chooses a set of possible parameter instantiations $\{p_1, \dots, p_{n_p}\}$, $n_p \in \mathbb{N}$ that has to be checked. A common choice for positive numerical parameters is to take them logarithmically equally spaced, since the correct scale is not known in advance, in general.

The training and validation process is described in Algorithm 9.7, where we denote by $s(p_i)$ the surrogate obtained with parameter p_i . It works as an outer loop with respect to the training of any of the surrogates that we have considered, and it has thus to be understood as part of the offline phase.

Algorithm 9.7: Model selection by validation.

- 1: Input: $X_{\text{tr}}, X_{\text{val}}, X_{\text{te}}, Y_{\text{tr}}, Y_{\text{val}}, Y_{\text{te}}, \{p_1, \dots, p_{n_p}\}$
 - 2: **for** $i = 1, \dots, n_p$ **do**
 - 3: Train surrogate $s(p_i)$ with data $(X_{\text{tr}}, Y_{\text{tr}})$
 - 4: Compute validation error $e_i := E(s(p_i), X_{\text{val}}, Y_{\text{val}})$
 - 5: **end for**
 - 6: Choose parameter $\bar{p} := p_i$ with $i := \text{argmin } e_i$
 - 7: Train surrogate $s(\bar{p})$ with data $(X_{\text{tr}} \cup X_{\text{val}}, Y_{\text{tr}} \cup Y_{\text{val}})$
 - 8: Compute test error $\bar{E} = E(s(\bar{p}), X_{\text{te}}, Y_{\text{te}})$
 - 9: Output: surrogate $s(\bar{p})$, optimal parameter \bar{p} , test error \bar{E}
-

A more advanced way to realize the same idea is via k -fold cross validation. To have an even better selection of the parameters, one can repeat the validation step (lines 2–6 in the previous algorithm) by changing the validation set at each step. To do so we do not select a validation set (so $n = n_{\text{tr}} + n_{\text{te}}$), and instead consider a partition of $X_{\text{tr}}, Y_{\text{tr}}$ into a fixed number $k \in \{1, \dots, n_{\text{tr}}\}$ of disjoint subsets, all approximately of the same size, i. e.,

$$X_{\text{tr}} := \{x_i, 1 \leq i \leq n_{\text{tr}}\} := \bigcup_{i=1}^k X_i$$

$$X_{\text{te}} := \{x_i, n_{\text{tr}} + 1 \leq i \leq n\},$$

and similarly for $Y_{\text{tr}} := \cup_{i=1}^k Y_i$ and for Y_{te} . In the validation step each of the X_i is used as a validation set, and the validation is repeated for all $i = 1, \dots, k$. In this case the error e_i for the parameter p_i is defined as the average error over all these permutations, as described in Algorithm 9.8.

Algorithm 9.8: Model selection by k -fold cross validation.

```

1: Input:  $X_{\text{tr}} = \cup_{i=1}^k X_i, X_{\text{te}}, Y_{\text{tr}} = \cup_{i=1}^k Y_i, Y_{\text{te}}, \{p_1, \dots, p_{n_p}\}$ 
2: for  $i = 1, \dots, n_p$  do
3:   for  $j = 1, \dots, k$  do
4:     Train surrogate  $s(p_i)$  with data  $(\cup_{\ell \neq j} X_\ell, \cup_{\ell \neq j} Y_\ell)$ 
5:     Compute error  $e^{(j)} := E(s(p_i), X_j, Y_j)$ 
6:   end for
7:    $e_i := \text{mean}\{e^{(j)}, 1 \leq j \leq k\}$ 
8: end for
9: Choose parameter  $\bar{p} := p_i$  with  $i := \text{argmin } e_i$ 
10: Train surrogate  $s(\bar{p})$  with data  $(X_{\text{tr}}, Y_{\text{tr}})$ 
11: Compute test error  $\bar{E} = E(s(\bar{p}), X_{\text{te}}, Y_{\text{te}})$ 
12: Output: surrogate  $s(\bar{p})$ , optimal parameter  $\bar{p}$ , test error  $\bar{E}$ 

```

We remark that, in the extreme case $k = N$, this k -fold cross validation is usually called Leave One Out Cross Validation (LOOCV).

9.8 Numerical examples

For the testing and illustration of the two methods of Section 9.4 and Section 9.5, we consider a real-world application dataset describing the biomechanical modeling of the human spine introduced and studied in [69]. We refer to that paper for further details and we just give a brief description in the following.

The input–output function $f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ represents the coupling between a global multibody system (MBS) and a Finite Elements (FEM) submodel. The human spine is represented as a MBS consisting of the vertebra, which are coupled by the interaction through intervertebral disks (IVDs). The PDE representing the behavior of each IVD is approximated by a FEM discretization, and it has the input geometry parameters as boundary conditions, and computes the output mechanical response as a result of the simulation. In particular, the three inputs are two spatial displacements and an angular inclination of a vertebra, and the three outputs are the corresponding two force components and the momentum which are transferred to the next vertebra. The dataset is generated by running the full model for $n := 1370$ different input parameters X_n and generating the corresponding set of outputs Y_n .

The dataset, as described in Section 9.7, is first randomly permuted and then divided in training and test datasets $(X_{n_{tr}}, Y_{n_{tr}})$, $(X_{n_{te}}, Y_{n_{te}})$ with $n_{tr} := 1238$ and $n_{te} = 132$, corresponding to roughly 90 % and 10 % of the data. We remark that the full model predicts a value $(0, 0, 0)^T$ for the input $(0, 0, 0)^T$ and this sample pair is present in the dataset. We thus manually include it in the training set independently of the permutation. The training and test sets can be seen in Figure 9.2.

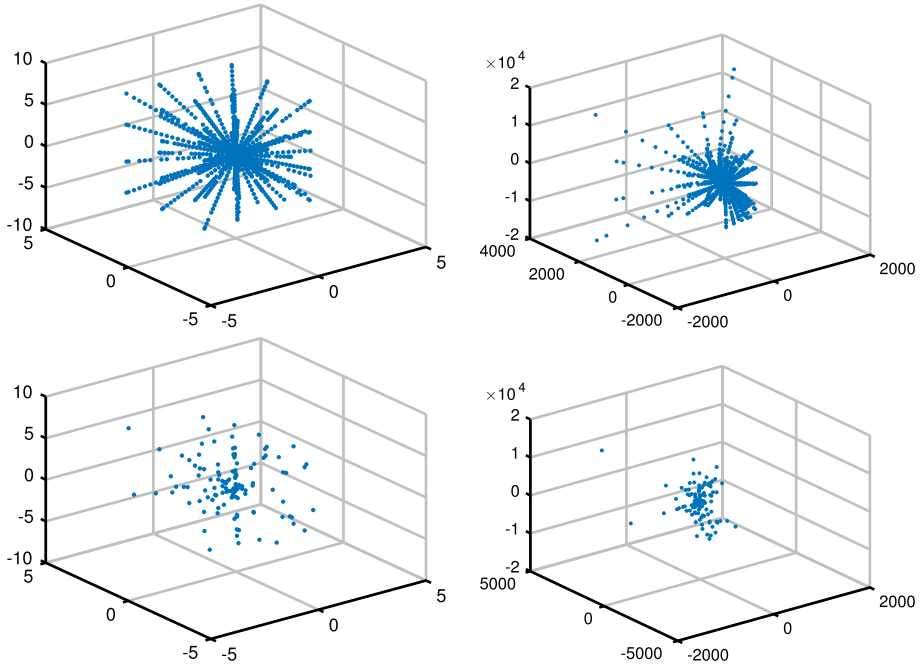


Figure 9.2: Input parameters (left) and corresponding outputs (right) for the training (top row) and test set (bottom row).

The models are trained using a Matlab implementation of the algorithms. For VKOGA we use an own implementation,⁷ while for SVR we employ the KerMor package,⁸ which provides an implementation of the 2-index SMO for the SVR without offset that is discussed in Section 9.5.1. We remark that this implementation requires the output data to be scaled in $[-1, 1]$, and thus we perform this scaling for the training and validation, while the testing is executed by scaling back the predictions to the original range. To have a fair comparison, we use the same data normalization also for the VKOGA models.

⁷ <https://gitlab.mathematik.uni-stuttgart.de/pub/ians-anm/vkoga>

⁸ <https://www.morepas.org/software/kermor/index.html>

The regularized VKOGA (with f -, P -, and f/P -greedy selection rules) and the SVR models are trained with the Gaussian kernel. Both algorithms depend on the shape parameter γ of the kernel and on the regularization parameter λ , while SVR additionally depends on the width ε of the tube. These parameters are selected by k -fold cross validation as described in Section 9.7. The values of k and of the parameter samples used for validation are reported in Table 9.1, where each parameter set is obtained by generating logarithmically equally spaced samples in the given interval, i. e., 400 parameter pairs are tested for VKOGA and 4000 triples for SVR. As an error measure we use the max error in (9.26). We remark that the SVR surrogate is obtained by training a separate model for each output, as described in Section 9.5, but only one cross validation is used. This means that for each parameter triple three models are trained, and then the parameter is evaluated in the prediction of the three-dimensional output.

Table 9.1: Parameters ranges and sample numbers used in the k -fold cross validation.

k	γ_{\min}	γ_{\max}	n_γ	λ_{\min}	λ_{\max}	n_λ	ε_{\min}	ε_{\max}	n_ε
5	10^{-2}	10^1	20	10^{-16}	10^3	20	10^{-10}	10^{-3}	10

Moreover, the training of the VKOGA surrogates is terminated when the square of the power function is below the tolerance $\tau_P := 10^{-12}$, or when the training error is below the tolerance $\tau_f := 10^{-6}$. Additionally, it would be possible to use a maximal number of selected points as stopping criterion, and this offers the significant advantage of directly controlling the expansion size, which could be reduced to any given number (of course at the price of a reduced accuracy). In the case of SVR, instead, the number N is a result of the tuning of the remaining parameters.

In Table 9.2 we report the values of the parameters selected by the validation procedure for the four models, as well as the number N of nonzero coefficients in the trained kernel expansions. Observe that for SVR the three values of N refer to the number of support vectors for the three scalar-valued models. Moreover, the number of support vectors or kernel centers is only slightly larger for SVR than for the VKOGA models, but, as discussed in the following, the VKOGA models give prediction errors which are up to two orders of magnitude smaller than the ones of the SVR model.

We can now test the four models in the prediction on the test set. Table 9.3 contains various error measures between the prediction of the surrogates and the exact data. We report the values of the maximum error E_{\max} and the RMSE E_{RMSE} defined in (9.26), and the relative maximum error $E_{\max, \text{rel}}$ obtained by scaling each error by the norm of the exact output.

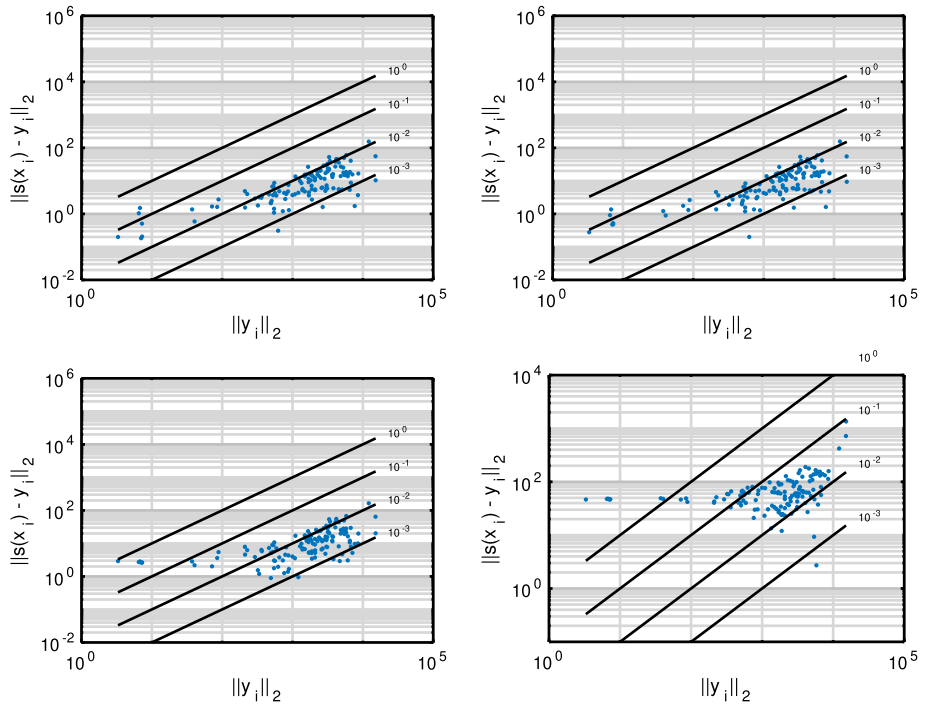
To provide a better insight in the approximation quality of the methods, we show in Figure 9.3 the distribution of the error over the test set. The plots show, for each sam-

Table 9.2: Selected parameters and number of nonzero coefficients in the kernel expansions.

Method	N	$\bar{\gamma}$	$\bar{\lambda}$	$\bar{\epsilon}$
VKOGA P -greedy	1000	$4.9 \cdot 10^{-2}$	10^{-11}	—
VKOGA f -greedy	879	$4.3 \cdot 10^{-2}$	10^{-11}	—
VKOGA f/P -greedy	967	$6.2 \cdot 10^{-2}$	10^{-9}	—
SVR, output 1	359	$1.8 \cdot 10^{-1}$	10^2	$7.7 \cdot 10^{-7}$
output 2	378			
output 3	405			

Table 9.3: Test errors: maximum error E_{\max} , RMSE error E_{RMSE} , maximum relative error $E_{\max, \text{rel}}$.

Method	E_{\max}	E_{RMSE}	$E_{\max, \text{rel}}$
VKOGA P -greedy	$1.6 \cdot 10^2$	22.3	$2.2 \cdot 10^{-1}$
VKOGA f -greedy	$1.6 \cdot 10^2$	22.4	$2.0 \cdot 10^{-1}$
VKOGA f/P -greedy	$1.6 \cdot 10^2$	23.2	$8.8 \cdot 10^{-1}$
SVR	$1.3 \cdot 10^3$	$1.6 \cdot 10^2$	$1.4 \cdot 10^1$

**Figure 9.3:** Absolute errors as functions of the magnitude of the output, and relative error levels from 10^0 to 10^{-3} for the surrogates obtained with P -greedy VKOGA (top left), f -greedy VKOGA (top right), f/P -greedy VKOGA (bottom left) and SVR (bottom right).

ple (x_i, y_i) in the test set, the absolute error $\|y_i - s(x_i)\|_2$ as a function of the magnitude $\|y_i\|_2$ of the output. Moreover, the black lines represent a relative error from 10^0 to 10^{-3} . It is clear that in all cases the maximum and RMS errors of Table 9.3 are dominated by the values obtained for outputs of large norm, where the VKOGA models obtain a much better accuracy than SVR. The relative errors, on the other hand, are not evenly distributed for SVR, where most of the test set is approximated with a relative error between 10^1 and 10^{-2} except for the samples with small magnitude of the output. For these data, the model gives increasingly bad predictions as the magnitude is smaller, reaching a relative error much larger than 1. The VKOGA models, instead, obtain a relative error smaller than 10^{-2} on the full test set except for the entries of small magnitude. For these samples, the f - and P -greedy versions of the algorithm perform almost the same and better than the f/P -greedy variant, thus giving an overall smaller relative error in Table 9.3. Moreover, these results are obtained with a significantly smaller expansion size for f -greedy than for P -greedy. Indeed, even if the SVR surrogates for the individual output components are smaller than the VKOGA ones, the overall number of nonzero coefficients is $359 + 378 + 405 = 1142$, i. e., more than the one of each of the three VKOGA models, thus leading to a less accurate and more expensive surrogate.

Regarding the runtime requirements, we can now estimate both the offline (training) and the online (prediction) times. The offline time required for the validation and training of the models is essentially determined by the number of parameters tested in the k -fold cross validation, while the training time of a single model is almost negligible. As a comparison, we report in Table 9.4 the average runtime $\tilde{T}_{\text{offline}}$ for 10 runs of the training of the models for the fixed set of parameters of Table 9.2. All the reported times are in the ranges of seconds (for VKOGA) and below one minute (for SVR). We remark that this timing is only a very rough indication and not a precise comparison, since the times highly depends on the number of selected points (for VKOGA) and the number of support vectors for SVR, and both are dependent on the used parameters. For example, we repeated the experiment for SVR with the same parameter set but with $\varepsilon = 10^{-1}$. In this case this value of ε is overly large (if compared to the one selected by cross validation) and it likely produces a useless model, but nevertheless we obtain an average training time of 0.03 sec.

Table 9.4: Average offline time (training only), online time, and projected speedup factor for the four different models.

Method	N	$\tilde{T}_{\text{offline}}$	$\tilde{T}_{\text{online}}$	$\tilde{T}_{\text{full}}/\tilde{T}_{\text{online}}$
VKOGA P -greedy	1000	1.67 sec	$9.97 \cdot 10^{-6}$ sec	$3.01 \cdot 10^5$
VKOGA f -greedy	879	1.41 sec	$9.44 \cdot 10^{-6}$ sec	$3.18 \cdot 10^5$
VKOGA f/P -greedy	967	1.66 sec	$9.92 \cdot 10^{-6}$ sec	$3.02 \cdot 10^5$
SVR (3 models)	1142	52.0 sec	$2.28 \cdot 10^{-5}$ sec	$1.32 \cdot 10^5$

A more interesting comparison is the online time, which directly determines the efficiency of the surrogate models in the replacement of the full simulation. In this case, we evaluate the models 5000 times on the full test set consisting of $n_{te} = 132$ samples, and we report the average online time \tilde{T}_{online} per single test sample in Table 9.4. The table contains also again the number N of elements of the corresponding kernel expansions, and it is evident that a smaller value leads to a faster evaluation of the model.

In the original paper [69], it has been estimated that a 30 sec full simulation with 24 IVDs with a timestep $\Delta t = 10^{-3}$ sec requires $7.2 \cdot 10^5$ evaluations of the coupling function f , and these were estimated to require 600 h. This corresponds to an average of $\tilde{T}_{full} = 3$ sec per evaluation of f , giving a speedup $\tilde{T}_{full}/\tilde{T}_{online}$ as reported in Table 9.4.

These surrogates can now be employed to solve different tasks that require multiple evaluations of f . As an example, we employ the f -greedy model (as the most accurate and most efficient) to solve a parameter estimation problem as described in Section 9.6. We consider the output values $Y_{n_{te}}$ in the test set as a set of measures that have not been used in the training of the model, and we try to estimate the values of $X_{n_{te}}$. For each output vector $y_i \in \mathbb{R}^3$ we define a target value $\bar{y} := y_i + \eta \|y_i\|_2 \nu$ to define the cost (9.25), where $\nu \in \mathbb{R}^3$ is a uniform random vector representing some noise, and $\eta \in [0, 1]$ is a noise level. We then use a built-in Matlab optimizer with the gradient of Proposition 9.8, with initial guess $x_0 := 0 \in \mathbb{R}^3$, to obtain an estimate x_i^* of x_i . The results of the estimate for each output value in the test set are depicted in Figure 9.4 for $\eta = 0, 0.1$, where we report also the final value of the cost function $C(x_i^*)$. In all cases, the optimizer seems to converge, since the value of the cost function is in all cases smaller than 10^{-4} , which represents a relative value smaller than 10^{-3} with respect to the magnitude of the input values. The maximum absolute error in the estimations is quite uniform for all the samples in the test set, and this results in a good relative error of about 10^{-1} for large inputs, while for inputs of very small magnitude the relative error is larger than 1, and a larger noise level leads to less accurate predictions. This behavior is coherent with the analysis of the test error discussed above, since the approximant is less accurate on inputs of small magnitude, and thus it provides a less reliable surrogate in the cost function.

9.9 Conclusions and outlook

In this chapter we discussed the use of kernel methods to construct surrogate models based on scattered data samples. These methods can be applied to data with general structure, and they scale well with the dimension of the input and output values. In particular, we analyzed issues and methods to obtain sparse solutions, which are then extremely fast to evaluate, while still being very accurate. These properties have been further demonstrated on numerical tests on a real application dataset. These

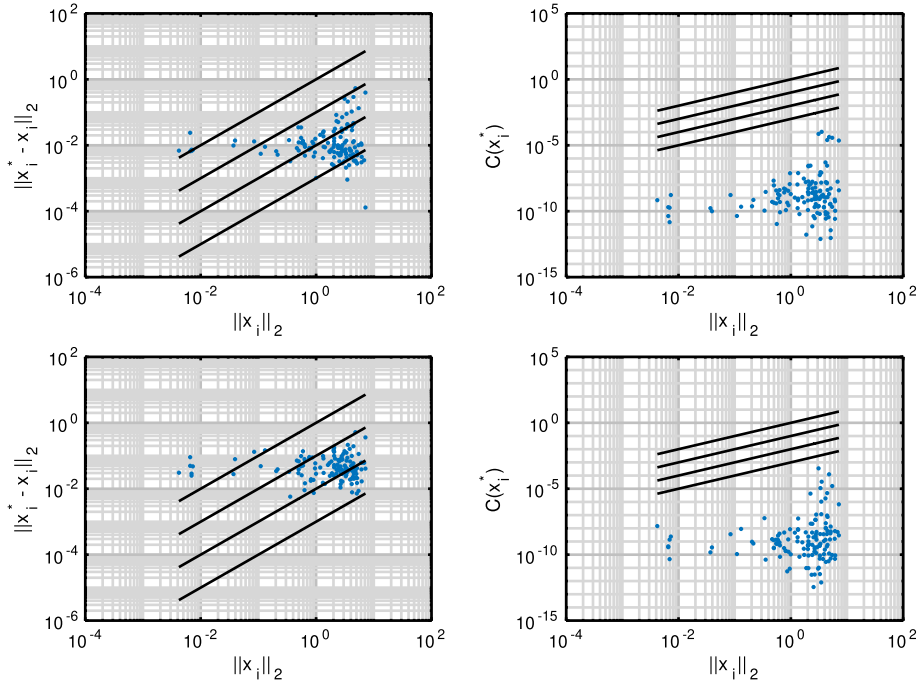


Figure 9.4: Absolute errors of the input estimation as functions of the magnitude of the output (left), and value of the cost function at the estimated input (right) for a noise level $\eta = 0$ (top row) and $\eta = 0.1$ (bottom row) using the f -greedy VKOGA model. The dotted lines represent relative error levels from 10^0 to 10^{-3} .

methods can be analyzed in the common framework of Reproducing Kernel Hilbert Spaces, which provides solid theoretical foundations and a high flexibility to derive new algorithms.

The integration of machine learning and model reduction is promising and many interesting aspects have still to be investigated. For example, surrogate models have been used in [23, 24] to learn a representation with respect to projection-based methods, and generally a more extensive application of machine learning to dynamical systems requires additional understanding and the derivation of new techniques. Moreover, the field of data-based numerics is very promising, where classical numerical methods are integrated or accelerated with data-based models.

Bibliography

- [1] M. Alvarez, L. Rosasco, and N. D. Lawrence. Kernels for vector-valued functions: a review. *Found. Trends Mach. Learn.*, 4(3):195–266, 2012.
- [2] N. Aronszajn. Theory of reproducing kernels. *Trans. Am. Math. Soc.*, 68:337–404, 1950.

- [3] A. Beckert and H. Wendland. Multivariate interpolation for fluid-structure-interaction problems using radial basis functions. *Aerosp. Sci. Technol.*, 5(2):125–134, 2001.
- [4] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92*, pages 144–152. ACM, New York, NY, USA, 1992.
- [5] J. Bouvrie and B. Hamzi. Kernel methods for the approximation of nonlinear systems. *SIAM J. Control Optim.*, 55(4):2460–2492, 2017.
- [6] T. Brünnette, G. Santin, and B. Haasdonk. Greedy kernel methods for accelerating implicit integrators for parametric ODEs. In F. A. Radu, K. Kumar, I. Berre, J. M. Nordbotten and I. S. Pop, editors, *Numerical Mathematics and Advanced Applications - ENUMATH 2017*, pages 889–896. Springer, Cham, 2019.
- [7] R. Cavoretto, A. De Rossi, and E. Perracchione. Efficient computation of partition of unity interpolants through a block-based searching technique. *Comput. Math. Appl.*, 71(12):2568–2584, 2016.
- [8] R. Cavoretto, G. Fasshauer, and M. McCourt. An introduction to the Hilbert-Schmidt SVD using iterated Brownian bridge kernels. *Numer. Algorithms*, 68(2):1–30, 2014.
- [9] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [10] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput.*, 20(1):33–61, 1998.
- [11] W. Chen, Z.-J. Fu, and C.-S. Chen. *Recent advances in radial basis function collocation methods*. Springer, 2014.
- [12] S. Deparis, D. Forti, and A. Quarteroni. A rescaled localized Radial Basis Function interpolation on non-Cartesian and nonconforming grids. *SIAM J. Sci. Comput.*, 36(6):A2745–A2762, 2014.
- [13] S. Deparis, D. Forti, and A. Quarteroni. *A Fluid–Structure Interaction Algorithm Using Radial Basis Function Interpolation Between Non-Conforming Interfaces*, pages 439–450. Springer, Cham, 2016.
- [14] M. Drohmann and K. Carlberg. The ROMES method for statistical modeling of Reduced-Order-Model error. *SIAM/ASA J. Uncertain. Quantificat.*, 3(1):116–145, 2015.
- [15] G. E. Fasshauer and M. McCourt. *Kernel-Based Approximation Methods Using MATLAB. Interdisciplinary Mathematical Sciences*, volume 19. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2015.
- [16] G. E. Fasshauer and M. J. McCourt. Stable evaluation of Gaussian radial basis function interpolants. *SIAM J. Sci. Comput.*, 34(2):A737–A762, 2012.
- [17] B. Fornberg and N. Flyer. *A primer on radial basis functions with applications to the geosciences*. SIAM, 2015.
- [18] B. Fornberg, E. Larsson, and N. Flyer. Stable computations with Gaussian radial basis functions. *SIAM J. Sci. Comput.*, 33(2):869–892, 2011.
- [19] J. Garcke and M. Griebel. *Sparse grids and applications*, volume 88. Springer, 2012.
- [20] T. Gärtner, J. W. Lloyd, and P. A. Flach. Kernels for structured data. In S. Matwin and C. Sammut, editors, *Inductive Logic Programming*, pages 66–83. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.
- [21] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [22] S. Grundel, N. Hornung, B. Klaassen, P. Benner, and T. Clees. *Computing Surrogates for Gas Network Simulation Using Model Order Reduction*, pages 189–212. Springer New York, New York, NY, 2013.
- [23] M. Guo and J. S. Hesthaven. Reduced order modeling for nonlinear structural analysis using Gaussian process regression. *Comput. Methods Appl. Mech. Eng.*, 341:807–826, 2018.

- [24] M. Guo and J. S. Hesthaven. Data-driven reduced order modeling for time-dependent problems. *Comput. Methods Appl. Mech. Eng.*, 345:75–99, 2019.
- [25] B. Haasdonk and G. Santin. Greedy kernel approximation for sparse surrogate modeling. In W. Keiper, A. Milde and S. Volkwein, editors, *Reduced-Order Modeling (ROM) for Simulation and Optimization: Powerful Algorithms as Key Enablers for Scientific Computing*, pages 21–45. Springer, Cham, 2018.
- [26] D. Haussler. Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, UC Santa Cruz, 1999.
- [27] G. S. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *Ann. Math. Stat.*, 41(2):495–502, 1970.
- [28] M. Köppel, F. Franzelin, I. Kröker, S. Oladyskin, G. Santin, D. Wittwar, A. Barth, B. Haasdonk, W. Nowak, D. Pflüger, and C. Rohde. Comparison of data-driven uncertainty quantification methods for a carbon dioxide storage benchmark scenario. *Comput. Geosci.*, 23(2):339–354, 2019.
- [29] T. Köppl, G. Santin, B. Haasdonk, and R. Helmig. Numerical modelling of a peripheral arterial stenosis using dimensionally reduced models and kernel methods. *Int. J. Numer. Methods Biomed. Eng.*, 34(8):e3095, 2018. cnm.3095.
- [30] M. Kowalewski, E. Larsson, and A. Heryudono. An adaptive interpolation scheme for molecular potential energy surfaces. *J. Chem. Phys.*, 145(8):084104, 2016.
- [31] E. Larsson, E. Lehto, A. Heryudono, and B. Fornberg. Stable computation of differentiation matrices and scattered node stencils based on Gaussian radial basis functions. *SIAM J. Sci. Comput.*, 35(4):A2096–A2119, 2013.
- [32] A. Manzoni and F. Negri. Heuristic strategies for the approximation of stability factors in quadratically nonlinear parametrized PDEs. *Adv. Comput. Math.*, 41(5):1255–1288, 2015.
- [33] E. Marchandise, C. Piret, and J.-F. Remacle. CAD and mesh repair with Radial Basis Functions. *J. Comput. Phys.*, 231(5):2376–2387, 2012.
- [34] I. Martini. Reduced Basis Approximation for Heterogeneous Domain Decomposition Problems. PhD thesis, IANS, University of Stuttgart 2017.
- [35] C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Comput.*, 17(1):177–204, 2005.
- [36] S. Müller. Komplexität und Stabilität von kernbasierten Rekonstruktionsmethoden (Complexity and Stability of Kernel-based Reconstructions). PhD thesis, Fakultät für Mathematik und Informatik, Georg-August-Universität Göttingen 2009.
- [37] S. Müller and R. Schaback. A Newton basis for kernel spaces. *J. Approx. Theory*, 161(2):645–655, 2009.
- [38] R. A. Olea. *Geostatistics for engineers and earth scientists*. Springer, 2012.
- [39] M. Pazouki and R. Schaback. Bases for kernel-based spaces. *J. Comput. Appl. Math.*, 236(4):575–588, 2011.
- [40] B. Peherstorfer and Y. Marzouk. A transport-based multifidelity preconditioner for Markov chain Monte Carlo. *Adv. Comput. Math.*, 45(5):2321–2348, 2019.
- [41] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, April 1998.
- [42] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
- [43] C. Rieger and B. Zwicknagl. Sampling inequalities for infinitely smooth functions, with applications to interpolation and machine learning. *Adv. Comput. Math.*, 32(1):103–129, 2008.
- [44] C. Rieger and B. Zwicknagl. Deterministic error analysis of support vector regression and related regularized kernel methods. *J. Mach. Learn. Res.*, 10:2115–2132, 2009.
- [45] S. Saitoh and Y. Sawano. *Theory of Reproducing Kernels and Applications. Developments in Mathematics*, volume 44. Springer, Singapore, 2016.

- [46] G. Santin and B. Haasdonk. Convergence rate of the data-independent P-greedy algorithm in kernel-based approximation. *Dolomites Res. Notes Approx.*, 10:68–78, 2017.
- [47] G. Santin, D. Wittwar, and B. Haasdonk. Greedy regularized kernel interpolation/ University of Stuttgart, 2018. ArXiv preprint 1807.09575.
- [48] R. Schaback. Error estimates and condition numbers for radial basis function interpolation. *Adv. Comput. Math.*, 3(3):251–264, 1995.
- [49] R. Schaback and H. Wendland. Approximation by positive definite kernels. In M. Buhmann and D. Mache, editors, *Advanced Problems in Constructive Approximation. International Series in Numerical Mathematics*, volume 142, pages 203–221. 2002.
- [50] M. Scheuerer, R. Schaback, and M. Schlather. Interpolation of spatial data – a stochastic or a deterministic problem? *Eur. J. Appl. Math.*, 24(4):601–629, 2013.
- [51] A. Schmidt and B. Haasdonk. Data-driven surrogates of value functions and applications to feedback control for dynamical systems. *IFAC-PapersOnLine*, 51(2):307–312, 2018. 9th Vienna International Conference on Mathematical Modelling.
- [52] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In D. Helmbold and B. Williamson, editors, *Computational Learning Theory*, pages 416–426. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.
- [53] B. Schölkopf and A. Smola. *Learning with Kernels*. The MIT Press, 2002.
- [54] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
- [55] I. Steinwart and A. Christmann. *Support Vector Machines*. Springer, 2008.
- [56] I. Steinwart, D. Hush, and C. Scovel. An explicit description of the Reproducing Kernel Hilbert Spaces of Gaussian RBF kernels. *IEEE Trans. Inf. Theory*, 52(10):4635–4643, 2006.
- [57] I. Steinwart, D. Hush, and C. Scovel. Training SVMs Without Offset. *J. Mach. Learn. Res.*, 12:141–202, 2011.
- [58] I. Steinwart and P. Thomann. liquidSVM: A fast and versatile SVM package, 2017. arXiv:1702.06899.
- [59] J. Suykens, J. Vanderwalle, and B. D. Moor. Optimal control by least squares support vector machines. *Neural Netw.*, 14:23–35, 2001.
- [60] T. Taddei, J. D. Penn, M. Yano, and A. T. Patera. Simulation-based classification; a model-order-reduction approach for structural health monitoring. *Arch. Comput. Methods Eng.*, 1–23, 2016.
- [61] R. Tibshirani. Regression shrinkage and selection via the LASSO. *J. R. Stat. Soc. B*, 58(1):267–288, 1996.
- [62] V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.
- [63] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. Comput. Math.*, 4(1):389–396, 1995.
- [64] H. Wendland. Fast evaluation of radial basis functions: methods based on partition of unity. In *Approximation theory, X (St. Louis, MO, 2001)*. *Innov. Appl. Math.*, pages 473–483 Vanderbilt Univ. Press, Nashville, TN, 2002.
- [65] H. Wendland. *Scattered Data Approximation. Cambridge Monographs on Applied and Computational Mathematics*, volume 17. Cambridge University Press, Cambridge, 2005.
- [66] H. Wendland and C. Rieger. Approximate interpolation with applications to selecting smoothing parameters. *Numer. Math.*, 101(4):729–748, 2005.
- [67] D. Wirtz and B. Haasdonk. A-posteriori error estimation for parameterized kernel-based systems. In *Proc. MATHMOD 2012 - 7th Vienna International Conference on Mathematical Modelling*, 2012.
- [68] D. Wirtz and B. Haasdonk. A vectorial kernel orthogonal greedy algorithm. *Dolomites Res. Notes Approx.*, 6:83–100, 2013.

- [69] D. Wirtz, N. Karajan, and B. Haasdonk. Surrogate modelling of multiscale models using kernel methods. *Int. J. Numer. Methods Eng.*, 101(1):1–28, 2015.
- [70] D. Wittwar, G. Santin, and B. Haasdonk. Interpolation with uncoupled separable matrix-valued kernels. *Dolomites Res. Notes Approx.*, 11:23–29, 2018.
- [71] H. Zhang C and L. Zhao. On the inclusion relation of reproducing kernel Hilbert spaces. *Anal. Appl.*, 11, 2013.

Jack P. C. Kleijnen

10 Kriging: methods and applications

Abstract: In this chapter we present Kriging—also known as a Gaussian process (GP) model—which is a relatively simple metamodel—or emulator or surrogate—of the corresponding complex simulation model. To select the input combinations to be simulated, we use Latin hypercube sampling (LHS); these combinations may have uniform and non-uniform distributions. Besides deterministic simulation we discuss random—or stochastic—simulation, which requires adjusting the design and analysis. We discuss sensitivity analysis of simulation models, using “functional analysis of variance” (FANOVA)—also known as Sobol sensitivity indices. Finally, we discuss optimization of the simulated system, including “robust” optimization.

Keywords: Gaussian process, metamodel, emulator, surrogate, optimization

10.1 Introduction

Kriging is the mathematical interpolation method that is named after the South African mining-engineer Krige (who lived from 1919 through 2013). He solved the problem of interpolating the outputs (or responses) that were obtained at a limited number of locations for gold mining; see the details on Krige’s life in [32].

Next, Krige’s method was formalized by the French mathematician *Matheron* (1930–2000), who developed a novel type of mathematical statistics—called *geo-statistics* or *spatial statistics*. He based this formalization on the stationary *Gaussian process* (GP). This stationarity implies that the GP has a constant mean (expected value), a constant variance, and covariances that depend only on the distances between “points” in a (say) k -dimensional space; obviously, in spatial statistics $k \leq 3$ (length, width, height). This GP defines a multivariate normal (or Gaussian) distribution. Spatial statistics is detailed in [14], which is a popular textbook with 900 pages; other books that reflect the French tradition are [12] and [40]. Recent survey articles are [4] and [13]. The connection between Krige and Matheron is also discussed in [32].

Later on, these GPs were applied in *machine learning*, which is a “hot” subdiscipline within computer science. The best-known textbook on GPs in machine learning is [35].

However, in this chapter we focus on the development and application of GPs in experiments with computerized simulation models; this field is known as *design and*

Acknowledgement: I thank Dick den Hertog (Universiteit van Amsterdam, Netherlands) and Ihsan Yanikoğlu (Özyeğin University, Turkey) for their comments on a previous version of this chapter.

Jack P. C. Kleijnen, Tilburg University, Postbox 90153, 5000 LE Tilburg, Netherlands

Open Access. © 2021 Jack P. C. Kleijnen, published by De Gruyter.  This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

<https://doi.org/10.1515/9783110498967-010>

analysis of computer experiments (DACE). Obviously, these models may have many inputs, which implies $k \gg 1$. The pioneering article on DACE is [36]; a recent textbook is [38]. DACE publications focus on *deterministic* simulation, whereas we shall also discuss *random* or *stochastic* simulation (e. g., queueing simulation models). A random simulation uses *pseudo-random numbers* (PRNs), which by definition are uniformly distributed on $[0, 1]$ and mutually independent. The design and analysis of experiments with random simulation models require the development of novel statistical methods, such as methods for *sample-size determination* and *stochastic Kriging* (SK). Indeed, we need to determine these sample sizes because we should select the number of replications per simulated point (input combination) in order to control the noise created by the PRNs (in Section 10.4 we shall see that this problem is not yet solved satisfactorily). We need SK because we should account for the so-called “intrinsic” noise created by the PRNs. The pioneering article on SK is [1].

We call a GP a *metamodel* of the underlying simulation model; i. e., this metamodel is a simpler and explicit mathematical function that approximates the complex and implicit function defined by the simulation model (this model is either deterministic or random). The DACE literature often calls the metamodel a *surrogate* or an *emulator*. We shall see (in the next section) that Kriging also quantifies the uncertainty of its predictor (say) \hat{y} ; i. e., Kriging also gives $\text{Var}(\hat{y})$, whereas many other metamodels (e. g., neural nets, splines) do not.

Kriging may have different *goals*; see [22, p. 9]. We shall discuss prediction, sensitivity analysis, and optimization. Besides these goals, [2] also discusses uncertainty quantification and uncertainty propagation.

Note: Simulation is applied in many scientific disciplines, which have their own terminologies and mathematical symbols. We use the terminology and symbols in [22]; e. g., we write Gaussian and Kriging with capitals (because these words refer to the proper names Gauss and Krige), and we use the symbol k (instead of d , which is used in many other publications on GP). The hasty reader may skip paragraphs that start with “Note:”.

Note: [30, 29] consider so-called *intrinsic Kriging* (IK)—which originated in geostatistics—and derive several IK types for deterministic simulations and random simulations, respectively.

We base this chapter on the Chapters 5 and 6 in [22], but we update these chapters, adding novel methods and applications for Kriging. Furthermore, corrections and additions for [22] are available on <https://sites.google.com/site/kleijnenjackpc/home/publications/corrections-additions-of-2015-springer-book>.

Besides [22] we also use [24, 25]. We do not use a Bayesian approach, which is used in many publications on GP.

We organize the rest of this chapter as follows. In Section 10.2 we present so-called “Ordinary Kriging” (OK), comparing OK with popular linear-regression metamodels. In Section 10.3 we present Latin hypercube sampling (LHS) for selecting the input combinations to be simulated, which results in the input/output (I/O) data analyzed by

OK. In Section 10.4 we consider the adjustments in the design and analysis of random simulation when using Kriging. In Section 10.5 we discuss sensitivity analysis (SA) of simulation models that are analyzed through Kriging; this SA may be global instead of local, and use “functional analysis of variance” (FANOVA)—also known as Sobol sensitivity indices. In Section 10.6 we discuss optimization of the simulated system including “robust” optimization, using Kriging. In Section 10.7 we present our conclusions, including topics that require further research.

10.2 Ordinary Kriging

To explain OK, we start with the following *linear-regression* model (we assume that the readers are familiar with this model; otherwise, they can read Chapters 2 and 3 in [22]); we add the subscript “reg”, to distinguish between regression and OK models:

$$\mathbf{y}_{\text{reg}} = \mathbf{X}_{\text{reg}}\boldsymbol{\beta}_{\text{reg}} + \mathbf{e}_{\text{reg}} \quad (10.1)$$

where \mathbf{y}_{reg} denotes the n -dimensional vector with the observations on the dependent (explained) regression variable where n denotes the number of observed (simulated) input combinations or “points”, \mathbf{X}_{reg} is the $n \times q$ matrix of independent (explanatory) regression variables, $\boldsymbol{\beta}_{\text{reg}}$ is the q -dimensional vector with regression parameters (coefficients), and \mathbf{e}_{reg} is the n -dimensional vector with the residuals $E(\mathbf{y}_{\text{reg}}) - E(\mathbf{w})$ where \mathbf{w} is the n -dimensional vector with simulated outputs. \mathbf{X}_{reg} consists of the n rows with the q -dimensional vectors $\mathbf{x}'_{\text{reg};i}$ with $i = 1, \dots, n$. We assume a univariate (scalar) output. The simulation model has k inputs x_j ($j = 1, \dots, k$); an independent regression variable may be identical to a simulation input or it may be a function of one or more simulation inputs; e. g., $x_{\text{reg};2} = x_1^2$ or $x_{\text{reg};3} = x_1x_2$. Classic regression analysis assumes that \mathbf{e}_{reg} is *white noise*; i. e., \mathbf{e}_{reg} is normally (Gaussian) distributed with zero means, constant variances (say) σ^2 , and zero correlations so the covariance matrix of \mathbf{e}_{reg} is $\boldsymbol{\Sigma}_{\text{reg}} = \sigma^2 \mathbf{I}_{n \times n}$ where $\mathbf{I}_{n \times n}$ denotes the $n \times n$ identity matrix.

Because white noise implies independent residuals, we cannot learn from the residuals. Kriging, however, assumes that the residuals at two points (say) \mathbf{x} and \mathbf{x}' have values that are more similar, as \mathbf{x} and \mathbf{x}' are closer; i. e., Kriging assumes that the residuals are positively correlated.

The simplest Kriging model is the *OK* model

$$y(\mathbf{x}) = \mu + M(\mathbf{x}) \quad (10.2)$$

where μ is the constant mean $E[y(\mathbf{x})]$ and $M(\mathbf{x})$ is a zero-mean *stationary GP* (a more complicated type of Kriging may replace this μ by a low-order polynomial; see universal Kriging or UK, discussed in the Note at the end of this section). $M(\mathbf{x})$ is called the *extrinsic noise*, because the term “intrinsic noise” is used for random simulation analyzed through SK.

The OK model leads to the *linear predictor* (say) $\hat{y}(\mathbf{x}_0)$ for the *new point* $\mathbf{x}_0 = (x_{0j})$ that combines the *n old* outputs collected in \mathbf{w}_n —or briefly \mathbf{w} —that are observed at the *n old* points with *k* inputs, so the $n \times k$ matrix \mathbf{X} has the rows $\mathbf{x}_i = (x_{i1}, \dots, x_{ik})$ with $i = 1, \dots, n$ and $\hat{y}(\mathbf{x}_0)$ uses the *n* weights λ_i collected in the vector $\boldsymbol{\lambda}$:

$$\hat{y}(\mathbf{x}_0) = \sum_{i=1}^n \lambda_i w(\mathbf{x}_i) = \boldsymbol{\lambda}' \mathbf{w}. \quad (10.3)$$

To derive the optimal $\boldsymbol{\lambda}$, we use the *best linear unbiased predictor* (BLUP) criterion. By definition, the predictor $\hat{y}(\mathbf{x})$ is unbiased if $E[\hat{y}(\mathbf{x})] = E[y(\mathbf{x})]$. This implies that $\mathbf{x} = \mathbf{x}_i$ gives $\hat{y}(\mathbf{x}_i) = w(\mathbf{x}_i)$; i. e., $\hat{y}(\mathbf{x})$ is an *exact interpolator*. Such interpolation makes perfect sense in deterministic simulation.

Note: A regression model determines the optimal $\boldsymbol{\beta}_{\text{reg}}$ in (10.1) through the criterion of the *best linear unbiased estimator* (BLUE) of $\boldsymbol{\beta}_{\text{reg}}$ where “best” means “minimum variance”; this BLUE is $\hat{\boldsymbol{\beta}}_{\text{reg}} = (\mathbf{X}'_{\text{reg}} \mathbf{X}_{\text{reg}})^{-1} \mathbf{X}'_{\text{reg}} \mathbf{w}$. The BLUE $\hat{\boldsymbol{\beta}}_{\text{reg}}$ is identical to the “maximum likelihood” (ML) estimator and the “least squares” (LS) estimator. LS is a mathematical instead of a statistical criterion; instead of this L_2 -norm, some mathematical models use either the L_1 -norm or the L_∞ -norm. The BLUE is not always equal to the MLE; e. g., the BLUE of σ^2 has the denominator $n - 1$, whereas the MLE has n . Obviously, $\hat{y}_{\text{reg}}(\mathbf{x}) = \mathbf{x}'_{\text{reg}} \hat{\boldsymbol{\beta}}_{\text{reg}}$ is not an exact interpolator—unless $n = q$.

Furthermore, we can prove that the optimal $\boldsymbol{\lambda}'$ in (10.3) is

$$\boldsymbol{\lambda}'_o = \left[\boldsymbol{\sigma}_M(\mathbf{x}_0) + \mathbf{1} \frac{1 - \mathbf{1}' \boldsymbol{\Sigma}_M^{-1} \boldsymbol{\sigma}_M(\mathbf{x}_0)}{\mathbf{1}' \boldsymbol{\Sigma}_M^{-1} \mathbf{1}} \right]' \boldsymbol{\Sigma}_M^{-1} \quad (10.4)$$

where $\boldsymbol{\Sigma}_M$ denotes the $n \times n$ matrix with the covariances between the metamodel’s “old” outputs y_i (so, $\boldsymbol{\Sigma}_M = (\sigma_{i:i'}) = (\text{Cov}(y_i, y_{i'}))$ with $i, i' = 1, \dots, n$), and $\boldsymbol{\sigma}_M(\mathbf{x}_0)$ denotes the n -dimensional vector with the covariances between the metamodel’s new output y_0 and the *n* old outputs y_i (so, $\boldsymbol{\sigma}_M(\mathbf{x}_0) = (\sigma_{0:i}) = (\text{Cov}(y_0, y_i))$). Obviously, $\boldsymbol{\Sigma}_M$ is determined by the old I/O simulation data (\mathbf{X}, \mathbf{w}) , whereas $\boldsymbol{\sigma}_M(\mathbf{x}_0)$ varies with \mathbf{x}_0 (so we might write $\boldsymbol{\lambda}_o(\mathbf{x}_0)$ if we would want to point out that $\boldsymbol{\lambda}_o$ varies with \mathbf{x}_0 , whereas $\hat{\boldsymbol{\beta}}_{\text{reg}}$ remains constant). Furthermore, a stationary process implies that λ_i decreases with the *distance* between \mathbf{x}_0 and \mathbf{x}_i . Substituting $\boldsymbol{\lambda}_o$ —defined in (10.4)—into (10.3), and using $\mathbf{1}_n$ —to denote the n -dimensional vector with all elements equal to 1—gives

$$\hat{y}(\mathbf{x}_0) = \mu + \boldsymbol{\sigma}_M(\mathbf{x}_0)' \boldsymbol{\Sigma}_M^{-1} (\mathbf{w} - \mu \mathbf{1}_n). \quad (10.5)$$

To denote $\text{Var}(y_i) (= \sigma_{i:i} = \sigma_i^2 = \sigma^2)$, we use the symbol τ^2 (τ^2 is the more usual symbol in the Kriging literature). Then the *mean squared residual* (MSE) of $\hat{y}(\mathbf{x}_0)$ is

$$\text{MSE}[\hat{y}(\mathbf{x}_0)] = \tau^2 - \boldsymbol{\sigma}_M(\mathbf{x}_0)' \boldsymbol{\Sigma}_M^{-1} \boldsymbol{\sigma}_M(\mathbf{x}_0) + \frac{[1 - \mathbf{1}'_n \boldsymbol{\Sigma}_M^{-1} \boldsymbol{\sigma}_M(\mathbf{x}_0)]^2}{\mathbf{1}'_n \boldsymbol{\Sigma}_M^{-1} \mathbf{1}_n}. \quad (10.6)$$

Because $\hat{y}(\mathbf{x}_0)$ is unbiased, $\text{MSE}[\hat{y}(\mathbf{x}_0)]$ reduces to $\text{Var}[\hat{y}(\mathbf{x}_0)]$. If $\mathbf{x}_0 = \mathbf{x}_i$, then $\hat{y}(\mathbf{x}_0) = w(\mathbf{x}_0)$ and $\text{Var}[\hat{y}(\mathbf{x}_0)] = 0$.

It is often convenient to switch from covariances to correlations. The correlation matrix $\mathbf{R} = (\rho_{ii'})$ equals $\tau^{-2}\boldsymbol{\Sigma}_M$, and the correlation vector $\boldsymbol{\rho}(\mathbf{x}_0)$ equals $\tau^{-2}\boldsymbol{\sigma}_M(\mathbf{x}_0)$. There are several types of correlation functions; see (e. g.) [35, pp. 80–104]. In simulation, the most popular function is the *Gaussian correlation function*:

$$\rho(\mathbf{h}, \boldsymbol{\theta}) = \prod_{j=1}^k \exp(-\theta_j h_j^2) = \exp\left(-\sum_{j=1}^k \theta_j h_j^2\right) \quad \text{with } \theta_j \geq 0 \quad (10.7)$$

with distance vector $\mathbf{h} = (h_j)$ where $h_j = |x_{g,j} - x_{g',j}|$ and $g, g' = 0, 1, \dots, n$, and with $\boldsymbol{\theta} = (\theta_j)$ so $\mathbf{R} = \mathbf{R}(\boldsymbol{\theta})$. We collect the $2 + k$ *Kriging (hyper)parameters* in $\boldsymbol{\psi} = (\mu, \tau^2, \boldsymbol{\theta}')'$. We estimate $\boldsymbol{\psi}$ through the *maximum likelihood* (ML) criterion, which gives the *ML estimator* (MLE) $\hat{\boldsymbol{\psi}}$. To compute $\hat{\boldsymbol{\psi}}$, it is convenient to switch to the log-likelihood function:

$$\min_{\boldsymbol{\psi}} \ln[|\tau^2 \mathbf{R}(\boldsymbol{\theta})|] + (\mathbf{w} - \mu \mathbf{1}_n)' [\tau^2 \mathbf{R}(\boldsymbol{\theta})]^{-1} (\mathbf{w} - \mu \mathbf{1}_n) \quad \text{with } \boldsymbol{\theta} \geq \mathbf{0} \quad (10.8)$$

where $|\mathbf{R}|$ denotes the determinant of \mathbf{R} . Solving (10.8) is a mathematical challenge; e. g., different solutions $\hat{\boldsymbol{\psi}}$ may result from different software packages or from initializing the same package with different starting values; see [18]. The various software packages for Kriging *standardize* the inputs such that the k inputs are limited to a k -dimensional hypercube $[0, 1]^k$.

In practice, $\hat{\boldsymbol{\psi}}$ is *plugged* into (10.5) and (10.6), which gives $\hat{y}(\mathbf{x}_0, \hat{\boldsymbol{\psi}})$ and $\hat{\text{Var}}[\hat{y}(\mathbf{x}_0, \hat{\boldsymbol{\psi}})]$. Most publications ignore the fact that $\hat{y}(\mathbf{x}_0, \hat{\boldsymbol{\psi}})$ becomes nonlinear, and $\hat{\text{Var}}[\hat{y}(\mathbf{x}_0, \hat{\boldsymbol{\psi}})]$ underestimates the “true” Kriging variance. However, the true Kriging variance is estimated in [22, pp. 191–197]—and also in [11]—applying the bootstrap method and the related method called “conditional simulation”. Nevertheless, in this chapter we shall simply plug-in $\hat{\boldsymbol{\psi}}$, and we shall not explicitly display the dependence of \hat{y} and $\hat{\text{Var}}(\hat{y})$ on $\hat{\boldsymbol{\psi}}$. This gives

$$s^2[\hat{y}(\mathbf{x}_0)] = \hat{\tau}^2 - \hat{\boldsymbol{\sigma}}_M(\mathbf{x}_0)' \hat{\boldsymbol{\Sigma}}_M^{-1} \hat{\boldsymbol{\sigma}}_M(\mathbf{x}_0) + \frac{[1 - \mathbf{1}' \hat{\boldsymbol{\Sigma}}_M^{-1} \hat{\boldsymbol{\sigma}}_M(\mathbf{x}_0)]^2}{\mathbf{1}' \hat{\boldsymbol{\Sigma}}_M^{-1} \mathbf{1}}. \quad (10.9)$$

We combine $\hat{y}(\mathbf{x}_0)$ and $s^2[\hat{y}(\mathbf{x}_0)]$ in the following two-sided *confidence interval* (CI) with nominal coverage $1 - \alpha$ where $z_{\alpha/2}$ denotes $\alpha/2$ -quantile of the standard normal $N(0, 1)$:

$$\hat{y}(\mathbf{x}_0) \pm z_{\alpha/2} s[\hat{y}(\mathbf{x}_0)]. \quad (10.10)$$

The actual (true) coverage of this CI may be lower than $1 - \alpha$, because of the following three factors: (i) the plug-in predictor $\hat{y}(\mathbf{x}_0)$ is biased; (ii) the plug-in variance estimator $s^2[\hat{y}(\mathbf{x}_0)]$ underestimates $\text{Var}[\hat{y}(\mathbf{x}_0)]$; and (iii) the absolute value of the Gaussian quantile $|z_{\alpha/2}|$ is lower than the absolute value of the Student quantile with (say) f degrees of freedom $|t_{f;\alpha/2}|$ if $f < \infty$, where $t_{f;\alpha/2}$ with the proper (but unknown) f seems to be the correct factor for a CI that uses an estimated variance.

Note: If we replace the constant $\mu = E(y)$ by a trend (e. g., $E(y) = \boldsymbol{\beta}'\mathbf{x}$), then we get UK; details on UK are found in [22, pp. 197–198] and also in [9] and [33].

10.3 Latin hypercube sampling

LHS is a popular type of *space-filling design*. Other types are orthogonal array, uniform, maximum entropy, minimax, maximin, integrated mean squared prediction error, and “optimal” designs; see [22, p. 198]. Furthermore, [3] discusses so-called bridge designs, which are space-filling and may satisfy various criteria and input constraints such that the input space is not a k -dimensional cube. Finally, [9] shows that “there is substantial variation in prediction accuracy over equivalent designs”.

Originally, LHS was invented as an alternative for Monte Carlo sampling in *risk analysis or uncertainty analysis* through deterministic simulation models that have (random) uncertain inputs; risk analysis estimates the probability of the output exceeding a given threshold as a function of an uncertain input x_j ; for details on risk analysis we refer to [22, pp. 218–222] and [2]. LHS assumes that an adequate meta-model is more complicated than a low-order polynomial, but LHS does not assume a specific metamodel (e. g., a Kriging model). LHS usually assumes that the k inputs are *independently* distributed (so their joint distribution is the product of the k individual marginal distributions); in this chapter we also use this assumption. Often LHS assumes that these distributions are *uniform* (symbol U) in the interval $[0, 1]$, so $x_j \sim U(0, 1)$. In risk analysis, however, LHS often assumes a specific *non-uniform* distribution for x_j with its mode at $x_{0,j}$ where $x_{0,j}$ denotes the j th coordinate of \mathbf{x}_0 and $j = 1, \dots, k$ (standardization implies $0 \leq x \leq 1$). This mode may be the value of the input that the experts think is most likely. There are many non-uniform distributions; see [26, pp. 286–305]. For example, we may use *beta distributions*, provided we select the correct values for the two parameters of the beta distribution; moreover, a different combination of these parameters gives a different variation around the mode; see [26, pp. 295–297]. We discuss a special case of the beta distributions; namely, a *triangular* distribution with its mode at $x_{m,j}$; we denote this distribution by $T(x_{m,j})$. We shall detail LHS for $U(0, 1)$ and for $T(x_{m,j})$ later on in this section. More details on LHS can be found in [22, pp. 198–203]; recent algorithms are detailed in [16, 27].

Whatever the marginal distributions are, LHS with a sample size (number of input combinations) n defines n mutually exclusive and exhaustive subintervals (or classes) with equal probability (namely, $1/n$) for x_j with $j = 1, \dots, k$. We denote these subintervals by $[l_{g,j}, h_{g,j}]$ with $g = 1, \dots, n$; the standardization $0 \leq x_j \leq 1$ implies $l_{1,j} = 0$ and $h_{n,j} = 1$. Altogether, if F_j denotes the *cumulative distribution function* (CDF) of x_j , then

$$P(l_{g,j} \leq x_j \leq h_{g,j}) = F_j(h_{g,j}) - F_j(l_{g,j}) = F_j\left(\frac{g}{n}\right) - F_j\left(\frac{g-1}{n}\right) = \frac{1}{n} \quad (10.11)$$

where $\min(g-1)/n = (1-1)/n = 0$, so $F_j((g-1)/n) = F_j(0) = 0$ because $\min x_j = 0$; likewise, $\max(g)/n = n/n = 1$, so $F_j(g/n) = F_j(1) = 1$ because $\max_j x = 1$. Obviously, (10.11) implies that near the mode of $T(x_{m,j})$ the subintervals $[l_{g,j}, h_{g,j}]$ are relatively short, compared with the other subintervals; however, if $x_j \sim U(0, 1)$, then each interval has the same length $1/n$.

LHS offers the following two *options*. Option (i) fixes x_j to the n *midpoints* (symbol m_j) of its n subintervals, so $x_{g,j} = m_{g,j} = (l_{g,j} + h_{g,j})/2$; e. g., if $x \sim U(0, 1)$, then these midpoints are equispaced with distance $1/n$ over the interval $[0, 1]$ so these midpoints are $1/(2n), 3/(2n), \dots, 1-1/(2n) = (2n-1)/(2n)$. Option (ii) samples x_j within its subinterval, accounting for F_j . Option (i) implies that x_j has a *discrete* PDF, so (10.11) becomes

$$P(x_{g,j} = m_{g,j}) = F_j(h_{g,j}) - F_j(l_{g,j}) = F_j\left(\frac{g}{n}\right) - F_j\left(\frac{g-1}{n}\right) = \frac{1}{n}. \quad (10.12)$$

Furthermore, LHS *samples without replacement*, so the midpoint $m_{j,g}$ is sampled only once in the sample of size n . We denote the *inverse CDF* by F_j^{-1} , so $y = F_j(x)$ with $0 \leq y \leq 1$ implies $x = F_j^{-1}(y)$; we observe that $U(0, 1)$ and $T(x_m)$ imply that F_j is continuous. Altogether, \mathbf{x}_j is a permutation of the n values $F_j^{-1}(0.5/n), F_j^{-1}(1.5/n), \dots, F_j^{-1}(1-0.5/n)$. Option (ii) first samples $r \sim U(a, b)$ with $a = (g-1)/n$ and $b = g/n$ where $g = 1, \dots, n$, and then computes

$$x = F_j^{-1}(r) \quad \text{with } r \sim U\left(\frac{g-1}{n}, \frac{g}{n}\right). \quad (10.13)$$

Algorithm 10.1 is a (pseudo)algorithm for LHS for option (i), which gives the $n \times k$ design matrix \mathbf{X}_L (the subscript L stands for LHS); also see [22, pp. 200].

Algorithm 10.1:

1. Read n, k, F_j ($j = 1, \dots, k$).
 2. Initialize: $j = 1$.
 3. Use F_j to divide the range of x_j into n mutually exclusive and exhaustive intervals of equal probability with midpoint $m_{j,g}$ ($g = 1, \dots, n$), and find $\mathbf{x}_j = (m_{j,1}, m_{j,2}, \dots, m_{j,n})'$.
 4. Randomly permute the n elements of \mathbf{x}_j , and save the result as column j of \mathbf{X}_L .
 5. If $j < k$ then $j = j + 1$ and go to Step 3; else stop.
-

For option (ii), Step 3 becomes: Use F_j to divide the range of x_j into n mutually exclusive and exhaustive intervals of equal probability, and apply (10.13) to find $\mathbf{x}_j = (x_{j,1}, x_{j,2}, \dots, x_{j,n})'$.

For both options, however, the *random* permutations in Step 4 may give a “bad” \mathbf{X}_L ; to decide on a “good” \mathbf{X}_L , our algorithm needs a criterion. We use the *maximin* criterion, which maximizes the minimum Euclidean distance between the n (k -dimensional) points in $[0, 1]^k$. We perform these random permutations (say) M times, and select the “best” design; e. g., MATLAB’s default is $M = 5$.

LHS does not impose a strict mathematical relationship between n and k . We observe that in Kriging $n \geq 10$ implies that we can estimate the correlation parameters θ_j with $j = 1, \dots, k$ (see (10.7)) reasonably accurate, because LHS implies that projection

of the n (k -dimensional) points onto the k individual axes gives n non-collapsing values per axis. If LHS uses a “small” n and a “large” k , then LHS covers $[0, 1]^k$ sparsely (so there are only a few old points close to the new point) and the Kriging predictor may be inaccurate. We note that [28] gives the rule-of-thumb $n = 10k$ for LHS if Kriging has SA as its goal; so, $n \geq 10$ if $k \geq 1$; [28] is revisited in [19].

There is much *software* for LHS. For example, Microsoft’s Excel spreadsheet software has add-ins that include LHS; LHS is also included in Oracle’s Crystal Ball, Palisade’s @Risk, and Frontline Systems’ Risk Solver. LHS is also available in the MATLAB Statistics toolbox, the R package, and Sandia’s DAKOTA software. An interesting website is <http://www.spacefillingdesigns.nl/>. However, some software (e. g., MATLAB) does not allow a non-uniform distribution such as $T(x_m)$, so we now present LHS for $U(0, 1)$ and $T(x_m)$; for details we refer to [24].

The CDF of $U(0, 1)$ is $F_{U;j}(x) = x$ if $0 \leq x \leq 1$. This CDF and (10.12) imply that option (i) samples $U(0, 1)$ through

$$x_{j;g} = m_{g;j} \quad \text{if } l_{g;j} < x_{g;j} < h_{g;j} \quad (g = 1, \dots, n).$$

This CDF and (10.13) imply that option (ii) samples $U(0, 1)$ through the PRN $r \sim U(0, 1)$ and computes

$$x_{j;g} = l_{g;j} + r(h_{g;j} - l_{g;j}) \quad \text{if } l_{g;j} < x_{g;j} < h_{g;j} \quad (g = 1, \dots, n).$$

The CDF of $T(x_m)$ is (see [26, pp. 304–305]):

$$\begin{aligned} F_{T;j}(x) &= \frac{x^2}{x_{m;j}} \quad \text{if } 0 \leq x \leq x_{m;j}, \\ F_{T;j}(x) &= 1 - \frac{(1-x)^2}{1-x_{m;j}} \quad \text{if } x_{m;j} \leq x \leq 1, \end{aligned} \quad (10.14)$$

so this CDF has a *kink* at $x_{m;j}$. Combining this equation with (10.12), option (i) samples

$$\begin{aligned} x_{j;g} &= \sqrt{\frac{x_{m;j}(2g-1)}{2n}} \quad \text{with } x_g \leq x_{m;j} \quad (g = 1, \dots, n), \\ x_{j;g} &= 1 - \sqrt{\frac{(1-x_{m;j})(2n-(2g-1))}{2n}} \quad \text{with } x_{j;g} \geq x_{m;j}. \end{aligned} \quad (10.15)$$

Option (ii) samples x_j (within the specific subinterval) via the first line of (10.14) if $h_{g;j} < x_{m;j}$. If $l_{g;j} > x_{m;j}$, then it samples x_j via the second line of (10.14). If $l_{g;j} < x_{m;j} < h_{g;j}$, then it first samples the PRN $r \sim U(0, 1)$; if $r < x_{m;j}$, then it samples x_j via the first line of (10.14); if $r > x_{m;j}$, then it samples x_j via the second line of (10.14). Altogether, option (ii) samples

$$\begin{aligned} x_{j;g} &= \sqrt{x_{m;j}r} \quad \text{if either } x_{m;j} > h_{g;j} \text{ or } l_{g;j} < x_{m;j} < h_{g;j} \text{ and } r < x_{m;j}, \\ x_{j;g} &= 1 - \sqrt{(1-x_{m;j})(1-r)} \quad \text{if either } x_{m;j} < l_{g;j} \text{ or } l_{g;j} < x_{m;j} < h_{g;j} \text{ and } r > x_{m;j}. \end{aligned} \quad (10.16)$$

Both options are compared in [24].

We observe that [23] develops a method (or algorithm) that uses a sequential design for Kriging in deterministic simulation aimed at SA. That method considers—but does not yet simulate—a set of candidate combinations selected through LHS, and finds the candidate with the highest estimated predictor variance. That method is also summarized in [22, pp. 204–206]. For constrained optimization in deterministic simulation, [25] also uses Kriging and a sequential design based on LHS for selecting candidate combinations, and finds the candidate with the highest criterion value; see Section 10.6.1.

10.4 Random simulation: Kriging analysis and experimental design

In random simulation we wish to control the noise of the simulation output at point \mathbf{x}_i ($i = 1, \dots, n$), so we obtain (say) $m_i \geq 1$ replications; we shall return to the choice of a value for m_i . To analyze the I/O data of a random simulation, [1] develops *stochastic Kriging* (SK). This SK adds the *intrinsic noise* term $\varepsilon_r(\mathbf{x}_i) \sim N(0, \text{Var}[\varepsilon(\mathbf{x}_i)])$ for replication r ($r = 1, \dots, m_i$). After averaging over the m_i replications, SK uses the formulas for the OK predictor \hat{y} in (10.5) and $\text{Var}(\hat{y})$ in (10.6), but replaces \mathbf{w} by $\bar{\mathbf{w}}$ and $M(\mathbf{x}_i)$ by $M(\mathbf{x}_i) + \bar{\varepsilon}(\mathbf{x}_i)$ where $\bar{\varepsilon}(\mathbf{x}_i) \sim N(0, \text{Var}[\varepsilon(\mathbf{x}_i)]/m_i)$ and $\bar{\varepsilon}(\mathbf{x}_i)$ is assumed to be independent of $M(\mathbf{x})$. We use the symbol $\boldsymbol{\psi}_{+\varepsilon}$ to denote $\boldsymbol{\psi}$ augmented with $\text{Var}[\varepsilon(\mathbf{x}_i)]$. The SK predictor $\hat{y}(\boldsymbol{\psi}_{+\varepsilon})$ is not an exact interpolator anymore—which makes sense in random simulation, which gives only estimates of the true simulation outputs. Obviously, $\Sigma_{\bar{\varepsilon}}$ is diagonal if no *common random numbers* (CRN) are used (if we applied CRN and used $m_i = m$, then $\Sigma_{\bar{\varepsilon}} = \Sigma_{\varepsilon}/m$; however, we assume no CRN in this section). To estimate $\text{Var}[\varepsilon(\mathbf{x}_i)]$, SK may use the classic unbiased estimator

$$s^2(w_i) = \frac{\sum_{r=1}^{m_i} (w_{i;r} - \bar{w}_i)^2}{m_i - 1} \quad (i = 1, \dots, n). \quad (10.17)$$

However, these $s^2(w_i)$ are rather noisy estimators, so SK may use a second Kriging metamodel for $\text{Var}[\varepsilon(\mathbf{x}_i)]$ —besides the Kriging metamodel for the mean $E[y(\mathbf{x}_i)]$. This second metamodel is only a rough approximation, because $s^2(w_i)$ is not normally distributed. The transformation $\log[s^2(w_i)]$ may give a normal distribution. For more details we refer to [22, p. 208].

An alternative for SK is *hetGP* developed in [5]. This alternative assumes $m_i \geq 1$, whereas SK assumes $m_i \gg 1$. Whereas SK gives a biased $\hat{\boldsymbol{\psi}}_{+\varepsilon}$ because SK fits Kriging models for the mean and the intrinsic variances independently, *hetGP* couples these models through a joint likelihood for $\boldsymbol{\psi}_{+\varepsilon}$ that is optimized in one shot. This alternative requires computational time of the same order as SK does. A recent alternative for SK is developed in [45].

The output of a random simulation may be a *quantile* (not an average); e. g., a quantile may be relevant in *chance-constrained optimization*. References are given in [22, p. 208].

In our preceding discussion of Kriging, we assumed $m_i \geq 1$ replications at the point \mathbf{x}_i with $i = 1, \dots, n$. In practice, we must decide *how many points* and *which points* to simulate, and *how many replications* to obtain for these points. Unfortunately, there are no simple solutions for this problem. Actually, [6] and [41] present sequential designs using the criterion called the *integrated mean squared prediction error* (IMSPE), which integrates $\text{Var}[\hat{y}(\mathbf{x})]$ for \mathbf{x} in $[0, 1]^k$. Obviously, this criterion assumes that the goal of Kriging is SA. Furthermore, [42] extends [5], using advanced mathematical analysis. If the goal of Kriging is optimization, then other criteria may be used (see Section 10.6). The main problem is that m_i with $i = 1, \dots, n$ should depend on both M (external noise) and Σ_{ε} (internal noise); also see [30].

Note: We observe that [10] allows some inputs to be qualitative. Furthermore, [47] uses SK, accounting for the discrepancy between real observations and simulated observations, possibly using CRN. Finally, [37] uses “generalized integrated Brownian fields” as simulation metamodels.

10.5 Sensitivity analysis

SA may be one of the goals of simulation modeling and Kriging. SA may be either global or local. SA is also closely related to “what if analysis”, “gaining insight”, and “prediction”. So, in practice, goals may be known under different names, and may be ambiguous. For further discussion of various goals we refer to [22, p. 9].

In this chapter we give the following *definition*: SA quantifies how the simulation output changes, as one or more simulation inputs change. To start our discussion of SA, we assume that the simulation is deterministic and that we change only one of the k inputs and that this single input (say) x_1 is continuous. Then we can quantify this sensitivity at a given point \mathbf{x} —so the SA is *local*—through $\partial w / \partial x_1|_{\mathbf{x}}$. This local SA is simplest if we can adequately approximate the I/O behavior of the simulation model by a first-order polynomial in \mathbf{x} so $\partial w / \partial x_1|_{\mathbf{x}} = \beta_1$. If we use a second-order polynomial, then interactions between x_1 and the other $k - 1$ inputs play a role and the marginal effect of x_1 is not constant.

Instead of local SA we may perform *global* SA: how does the simulation output change, as one or more simulation inputs change over the whole area of interest $[0, 1]^k$? Moreover, “the” output may have a distribution (instead of a single value) if the deterministic simulation has one or more inputs that are uncertain so we assume a prespecified distribution for these uncertain inputs (as in LHS; see Section 10.3). Now we discuss *global sensitivity analysis* (GSA) or *functional analysis of variance*

(FANOVA), which uses variance-based indices originally proposed by the Russian mathematician Sobol.

FANOVA decomposes σ_w^2 —the variance of the random simulation output w —into fractions that refer to sets of inputs:

$$\sigma_w^2 = \sum_{j=1}^k \sigma_j^2 + \sum_{j < j'}^k \sigma_{jj'}^2 + \cdots + \sigma_{1,\dots,k}^2 \quad (10.18)$$

with the main-effect variance $\sigma_j^2 = \text{Var}[E(w|x_j)]$, the two-factor interaction variance $\sigma_{jj'}^2 = \text{Var}[E(w|x_j, x_{j'})] - \text{Var}[E(w|x_j)] - \text{Var}[E(w|x_{j'})]$, etc. This σ_j^2 gives the *first-order sensitivity index* or the *main-effect index* $\zeta_j = \sigma_j^2 / \sigma_w^2$, which quantifies the effect of varying x_j alone—averaged over the variations in all the other $k - 1$ inputs—where the denominator σ_w^2 standardizes ζ_j to provide a fractional contribution. Altogether the sum of the 2^{k-1} indices is 1. In practice, we assume that only the ζ_j —and possibly the $\zeta_{jj'}$ —are important, and that they sum up to a fraction “close enough” to 1.

To *estimate* these measures, we may use LHS and replace the simulation model by a Kriging metamodel; see [22, pp. 216–218]. In practice, FANOVA may show that (say) 70 % of σ_w^2 is caused by x_1 , 20 % by x_2 , and 10 % by the interaction between x_1 and x_2 . Software for FANOVA is included in Open TURNS, discussed in [2]; a MATLAB toolbox is presented in [34]. SA is the topic of many recent publications; see the website with corrections and additions for [22] that was mentioned in Section 10.1.

10.6 Optimization

There are many methods for the optimization of a simulated system; see the references in [22, p. 242]. Some methods require relatively many input combinations; e. g., evolutionary algorithms (EAs) and particle swarm optimization (PSO) do. We may therefore apply these methods to a computationally cheap metamodel of the underlying computationally expensive simulation model; e. g., PSO and Kriging are combined in [20]. In this chapter, however, we focus on *efficient global optimization* (EGO) in Section 10.6.1 and *robust optimization* (RO) in Section 10.6.2.

10.6.1 Efficient global optimization

Originally, [21] developed EGO, which uses Kriging. EGO is a *sequential* method that balances *local* and *global* search; i. e., EGO balances *exploitation* and *exploration*. We detail only the *basic* EGO-variant for *deterministic* simulation. The goal of this variant is to estimate the input combination \mathbf{x}_0 that minimizes the simulation output $w(\mathbf{x})$.

We start with an initial or pilot sample of input combinations \mathbf{x}_i with $i = 1, \dots, n$, selected through LHS. We use these \mathbf{x}_i as input for the simulation model, which

gives (\mathbf{x}_i, w_i) or $(\mathbf{X}_n, \mathbf{w}_n)$. We then find the best simulated output so far: $w_{\min} = \min_{1 \leq i \leq n} w(\mathbf{x}_i)$.

Next we select a new input combination, considering both \hat{y} and $s(\hat{y})$; e. g., if the two combinations \mathbf{x} and \mathbf{x}' have $\hat{y}(\mathbf{x}) \approx \hat{y}(\mathbf{x}')$ and $s[\hat{y}(\mathbf{x})] > s[\hat{y}(\mathbf{x}')]$, then we prefer \mathbf{x} because \mathbf{x} has a higher *probability of improvement* (or PI) so it may give a smaller w . We know that $s[\hat{y}(\mathbf{x}_{n+1})]$ increases as \mathbf{x}_{n+1} lies farther away from \mathbf{x}_i ($i = 1, \dots, n$); see (10.6). Actually, we estimate the maximum of the *expected improvement* (EI), which is reached if either $\hat{y}(\mathbf{x}_{n+1})$ is much smaller than w_{\min} or $s[\hat{y}(\mathbf{x}_{n+1})]$ is relatively large so $\hat{y}(\mathbf{x}_{n+1})$ is relatively uncertain.

To obtain \hat{y} and $s(\hat{y})$, we fit a Kriging metamodel. This gives $EI(\mathbf{x}) = E[\max(w_{\min} - \hat{y}(\mathbf{x}), 0)]$. Let Φ and ϕ denote the cumulative distribution function (CDF) and the PDF of the standard normal variable z . Then [21] derives that the estimated EI for the input combination \mathbf{x} is

$$\widehat{EI}(\mathbf{x}) = (w_{\min} - \hat{y}(\mathbf{x}))\Phi\left(\frac{w_{\min} - \hat{y}(\mathbf{x})}{s[\hat{y}(\mathbf{x})]}\right) + s[\hat{y}(\mathbf{x})]\phi\left(\frac{w_{\min} - \hat{y}(\mathbf{x})}{s[\hat{y}(\mathbf{x})]}\right). \quad (10.19)$$

Using (10.19), we find the \mathbf{x} that maximizes $\widehat{EI}(\mathbf{x})$; we denote this optimal combination by $\hat{\mathbf{x}}_o$ (with the subscript “o” for “optimal”). To find this $\hat{\mathbf{x}}_o$, we may use a relatively large set of candidate input combinations that are selected through LHS (say) \mathbf{X}_{cand} ; we do not simulate these candidates, but we find the candidate with the highest $\widehat{EI}(\mathbf{x})$ with $\mathbf{x} \in \mathbf{X}_{\text{cand}}$.

We use this candidate $\hat{\mathbf{x}}_o$ as the input combination that is simulated next, and obtain $w(\hat{\mathbf{x}}_o)$. Then we re-estimate the Kriging metamodel from the I/O data $(\mathbf{X}_n, \mathbf{w}_n)$ augmented with $(\hat{\mathbf{x}}_o, w(\hat{\mathbf{x}}_o))$. We update n , and return to (10.19)—until we satisfy a stopping criterion; e. g., $\widehat{EI}(\hat{\mathbf{x}}_{\text{opt}})$ is “close” to 0 or the computer budget is exhausted.

There are many more EGO-variants, for deterministic simulation and random simulation, constrained optimization, multi-objective optimization including Pareto frontiers, RO, the “excursion set” or “admissible set”, estimation of a quantile, and Bayesian approaches; see [22, pp. 267–269]. For example, a variant for constrained optimization in deterministic simulation—with one goal output and several constrained outputs—is developed in [25].

10.6.2 Robust optimization

In [15] Krige’s (meta)model and Taguchi’s world view are combined. Taguchi designed *robust* engineering products such as cars (at Toyota), emphasizing that in practice some inputs are under complete control of the engineers (e. g., the car’s design), whereas other inputs are not (the car’s driver, and the roads). He therefore distinguished between (i) *controllable* or decision variables, and (ii) *noncontrollable* or environmental noisy (or random) factors. So, the estimated optimum (see the pre-

ceding section) may turn out to be inferior because this optimum ignores *uncertainties* in some of the simulation inputs. Therefore we may proceed as follows.

To simplify our discussion, we sort the k simulation inputs such that the first k_C inputs are controllable, and the next k_{NC} inputs are noncontrollable. We let \mathbf{z}_C and \mathbf{z}_{NC} denote the vector with the k_C controllable and the k_{NC} noncontrollable original (nonstandardized) inputs \mathbf{z} . Taguchi assumes a single output (say) w , and focuses on its mean $E(w)$ and its variance; obviously, this variance is caused by \mathbf{z}_{NC} , so $\text{Var}(w|\mathbf{z}_C) > 0$. Taguchi combines $E(w)$ and $\text{Var}(w|\mathbf{z}_C)$ into a scalar loss function such as the *signal-to-noise* or *mean-to-variance* ratio $E(w)/\text{Var}(w|\mathbf{z}_C)$; see the detailed discussion of Taguchi's approach in [31, pp. 486–488], which is the classic textbook on so-called *response surface methodology* (RSM).

Taguchi's approach is successful in production engineering, but statisticians criticize its statistical methods. We add that—compared with real-life experiments (discussed in [31])—simulation experiments have more inputs, more input values, and more input combinations. Actually, [31, pp. 502–506] combines Taguchi's worldview with the statisticians' RSM.

However, [15] uses $E(w)$ and $\sigma(w|\mathbf{z}_C)$ *separately*; obviously, $\sigma(w|\mathbf{z}_C)$ has the same scale as $E(w)$ has. *Mathematical optimization* (MO) can then be used to solve the *constrained optimization*: $\min_{\mathbf{z}_C} E(w|\mathbf{z}_C)$ such that $\sigma(w|\mathbf{z}_C) \leq c_\sigma$ where c_σ is a prespecified upper threshold for σ . Constrained optimization is also discussed in [31, p. 492]. Whereas [31] superimposes contour plots for $E(w|\mathbf{z}_C)$ and $\sigma(w|\mathbf{z}_C)$ to estimate the optimal \mathbf{z}_C , [15] uses MO.

This MO, however, requires specification of c_σ (threshold for σ). In practice, managers may find it hard to select a specific value for c_σ . Therefore we try different c_σ values, and estimate the corresponding *Pareto-optimal* efficiency frontier. This frontier, is uncertain because it depends on the estimators of $E(w|\mathbf{z}_C)$ and $\sigma(w|\mathbf{z}_C)$. To estimate the variability of the frontier, we may apply *bootstrapping*. For details on this type of RO we refer to Dellino et al. [15], which is summarized in [22, pp. 280–284].

An application of RO using Kriging to estimate the Pareto frontier is [46]. Kriging for RO is also used in [8], comparing this approach with several alternative metamodel types (e. g., neural networks).

Note that [44] presents an approach that follows RO as developed in MO (instead of Taguchian publications). RO in MO was originally developed by the Israeli mathematician Ben-Tal and uses concepts such as “uncertainty sets”, “robust counterparts”, and “adjustable decision rules”. The approach in [44] does not need a known distribution for the environmental inputs; i. e., this approach uses only experimental data combined with so-called “phi-divergence” uncertainty sets. This approach is applied to low-order polynomial metamodels, but may be extended to Kriging metamodels (without introducing additional complexity to the problem formulation). RO for both linear-regression and Kriging metamodels is detailed in [39].

10.7 Conclusions and outlook

We provided an overview of a specific type of metamodel; namely, Kriging or Gaussian process (GP). Kriging is popular in geostatistics and machine learning, and is also gaining popularity in the analysis of simulation experiments. However, many issues remain in Kriging; e. g., should we use a simple constant mean or a low-order polynomial trend? Should we select a Gaussian or a Matérn correlation function? How should we estimate the intrinsic variance of the simulation output for new input combinations of the simulation model? Kriging in random simulation needs further research on sequential designs for the selection of additional input combinations and the number of replications for old and new combinations. Kriging is used in classic “efficient global optimization” (EGO); currently, many researchers also investigate random simulation and constrained optimization. Robust optimization (RO) has just started in simulation, whereas it is a hot topic in Mathematical Optimization (MO). The application of various types of Kriging for sensitivity analysis and optimization remains challenging.

Future research may try to solve the *curse of dimensionality* in Kriging; currently, the number of inputs is usually limited to (say) 20. To solve this problem, we may precede Kriging by *factor screening*; several screening methods are presented in Chapter 4 of [22]. Other solutions are discussed in [7, 43].

Another topic of future research is *big data*, in which the number of input/output combinations is so high that the computation of the inverse of the estimated covariance matrix is impossible. Various solutions are discussed in [17, 24].

Bibliography

- [1] B. Ankenman, B. Nelson, and J. Staum. Stochastic Kriging for simulation metamodeling. *Oper. Res.*, 58(2):371382, 2010.
- [2] M. Baudin, A. Dutfoy, B. Iooss, and A-L. Popelin. Open TURNS: an industrial software for uncertainty quantification in simulation. In R. Ghanem, D. Higdon and H. Owhadi, editors, *Handbook of uncertainty quantification*, pages 2001–2038. Springer, 2016.
- [3] E. Benková, R. Harman, and W. G. Müller. Privacy sets for constrained space-filling. *J. Stat. Plan. Inference*, 171:1–9, 2016.
- [4] A. Bhosekar and M. Ierapetritou. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Comput. Chem. Eng.*, 108:250–267, 2018.
- [5] M. Binois, R. B. Gramacy, and M. Ludkovski. Practical heteroskedastic Gaussian process modeling for large simulation experiments. *Journal. J. Comput. Graph. Stat.*, 27(4):808–821, 2018.
- [6] M. Binois, J. Huang, R. B. Gramacy, and M. Ludkovskiz. Replication or exploration? Sequential design for stochastic simulation experiments. *Technometrics*, 61(1):7–23, 2019.
- [7] M. A. Bouhlef and J. R. R. A. Martins. Gradient-enhanced Kriging for high-dimensional problems. *Eng. Comput.*, 35:157173, 2019.

- [8] T. Chatterjee, S. Chakraborty, and R. Chowdhury. A critical review of surrogate assisted robust design optimization. *Arch. Comput. Methods Eng.*, 26:245–274, 2019.
- [9] H. Chen, J. L. Loepky, J. Sacks, and W. J. Welch. Analysis methods for computer experiments: how to assess and what counts? *Stat. Sci.*, 31(1):40–60, 2016.
- [10] X. Chen, K. Wang, and F. Yang. Stochastic Kriging with qualitative factors. In R. Pasupathy, S-H. Kim, A. Tolk, R. Hill and M. E. Kuhl, editors, *Proceedings of the 2013 winter simulation conference*, pages 790–801, 2013.
- [11] C. Chevalier, X. Emery, and D. Ginsbourger. Fast update of conditional simulation ensembles, 2014. hal-00984515.
- [12] J-P. Chilès and P. Delfiner. *Geostatistics: modeling spatial uncertainty*. Wiley, New York, 1999.
- [13] J.-P. Chilès and N. Desassis. Fifty years of Kriging. In B. S. Daya Sagar, Q. Cheng and F. Agterberg, editors, *Handbook of Mathematical Geosciences; fifty years of IAMG*, pages 589–612, Springer, 2018.
- [14] N. A. C. Cressie. *Statistics for spatial data; revised edition*. Wiley, New York, 1993.
- [15] G. Dellino, J. P. C. Kleijnen, and C. Meloni. Robust optimization in simulation: Taguchi and Krige combined. *INFORMS J. Comput.*, 24(3):471–484, 2012.
- [16] H. Dong and M. K. Nakayama. Quantile estimation with Latin hypercube sampling. *Oper. Res.*, 65(6):1678–1695, 2017.
- [17] A. M. Edwards and R. B. Gramacy. Precision aggregated local models, 2020. arXiv preprint arXiv:2005.13375.
- [18] C. B. Erickson, B. E. Ankenman, and S. M. Sanchez. Comparison of Gaussian process modeling software. *Eur. J. Oper. Res.*, 266:179–192, 2018.
- [19] O. Harari, D. Bingham, A. Dean, and D. Higdon. Computer experiments: prediction accuracy, sample size and model complexity revisited. *Stat. Sin.*, 28(2):899–919, 2018.
- [20] F. Jilu, S. Zhili, and S. Hongzhe. Optimization of structure parameters for angular contact ball bearings based on Kriging model and particle swarm optimization algorithm. *Proc. Inst. Mech. Eng., Part C, J. Mech. Eng. Sci.*, 231(23):4298–4308, 2017.
- [21] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *J. Glob. Optim.*, 13:455–492, 1998.
- [22] J. P. C. Kleijnen. *Design and analysis of simulation experiments*, second edition. Springer, 2015.
- [23] J. P. C. Kleijnen and W. C. M. van Beers. Application-driven sequential designs for simulation experiments: Kriging metamodeling. *J. Oper. Res. Soc.*, 55(9):876–883, 2004.
- [24] J. P. C. Kleijnen and W. C. M. van Beers. Prediction for big data through Kriging. *J. Math. Manag. Sci.*, 2020. <https://doi.org/10.1080/01966324.2020.1716281>.
- [25] J. P. C. Kleijnen, I. van Nieuwenhuyse, and W. C. M. van Beers. Constrained optimization in simulation. 2021, in preparation.
- [26] A. M. Law. *Simulation modeling and analysis*, fifth edition. McGraw-Hill, Boston, 2015.
- [27] K. Le Guiban, A. Rimmel, M-A. Weisser, and J. Tomasik. The first approximation algorithm for the maximin Latin hypercube design problem. *Oper. Res.*, 66(1):253–266, 2018.
- [28] J. L. Loepky, J. Sacks, and W. J. Welch. Choosing the sample size of a computer experiment: A practical guide. *Technometrics*, 51:366–376, 2009.
- [29] E. Mehdad and J. P. C. Kleijnen. Efficient global optimization for black-box simulation via sequential intrinsic Kriging. *J. Oper. Res. Soc.*, 69(11):1725–1737, 2018.
- [30] E. Mehdad and J. P. C. Kleijnen. Stochastic intrinsic Kriging for simulation metamodeling. *Appl. Stoch. Models Bus. Ind.*, 34:322–337, 2018.
- [31] R. H. Myers, D. C. Montgomery, and C. M. Anderson-Cook. *Response surface methodology: process and product optimization using designed experiments*, third edition. Wiley, New York, 2009.

- [32] R. C. A. Minnitt and W. Assibey-Bonsu. Professor D.G. Krige FRSSAf. *Trans. R. Soc. S. Afr.*, 68(3):199–202, 2013.
- [33] T. Mukhopadhyay, S. Chakraborty, S. Dey, S. Adhikari, and R. Chowdhury. A critical assessment of Kriging model variants for high-fidelity uncertainty quantification in dynamics of composite shells. *Arch. Comput. Methods Eng.*, 1–24, 2016.
- [34] F. Pianosi, F. Sarrazin, and T. Wagener. A Matlab toolbox for global sensitivity analysis. *Environ. Model. Softw.*, 70:80–85, 2015.
- [35] C. Rasmussen and C. Williams. *Gaussian processes for machine learning*. MIT Press, Cambridge, Massachusetts, 2006.
- [36] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments (includes Comments and Rejoinder). *Stat. Sci.*, 4(4):409–435, 1989.
- [37] P. Salemi, J. Staum, and B. L. Nelson. Generalized integrated Brownian fields for simulation metamodeling. *Oper. Res.*, 67(3):874–891, 2019.
- [38] T. J. Santner, B. J. Williams, and W. I. Notz. *The design and analysis of computer experiments*, second revised edition. Springer, New York, 2018.
- [39] E. Stinstra and D. den Hertog. Robust optimization using computer experiments. *Eur. J. Oper. Res.*, 191:816–837, 2008.
- [40] H. Wackernagel. *Multivariate geostatistics: an introduction with applications*, third edition. Springer, Berlin, 2003.
- [41] B. Wang and J. Hu. Some monotonicity results for stochastic Kriging metamodels in sequential settings. *INFORMS J. Comput.*, 30(2):278–294, 2018.
- [42] W. Wang and B. Haaland. Controlling sources of inaccuracy in stochastic Kriging. *Technometrics*, 61(3):309–321, 2019.
- [43] M. A. Winkel, J. W. Stallrich, C. B. Storlie, and B. J. Reich. Sequential optimization in locally important dimensions. *Technometrics*, 63(2):236–248, 2021.
- [44] I. Yanikoğlu, D. den Hertog, and J. P. C. Kleijnen. Robust dual-response optimization. *Ind. Eng. Res. Dev.*, 48(3):298–312, 2016.
- [45] Q. Zhang and Y. Ni. Improved Most Likely Heteroscedastic Gaussian Process Regression via Bayesian Residual Moment Estimator. *IEEE Trans. Signal Process.*, 68:3450–3460, 2020. <https://doi.org/10.1109/TSP.2020.2997940>.
- [46] W. Zhang and W. Xu. Simulation-based robust optimization for the schedule of single-direction bus transit route: the design of experiment. *Transp. Res., Part E*, 106:203–230, 2017.
- [47] L. Zou and X. Zhang. Stochastic Kriging for inadequate simulation models, 2018. arXiv:1802.00677v2 [stat.ME].

Index

- a-posteriori error bound 69
- a-posteriori error estimation 69, 75
- AAA algorithm 182, 216, 219, 220
- abstract Lie theory 256
- adaptive sampling 167, 168
- admissible set 366
- affine-invariant metric 258
- alternating directions implicit (ADI) based method 26
- applications
 - acoustic and vibration 12
 - biomechanical modeling 312, 343
 - cardiovascular system 12, 275
 - aortic input impedance 286
 - chemical process optimization 12
 - computational aerodynamics 12
 - electromagnetics 12, 163
 - fluid dynamics 12
 - gas transport problems 312
 - high-frequency electronics 163
 - high-speed interconnect in a mobile device 168
 - industrial production settings 13
 - integrated circuit design 60
 - interconnect link on a high-performance PCB 172
 - mechanical engineering 12
 - microelectromechanical systems (MEMS) 63
 - microelectronics 12
 - multiport interconnect on PCB 291
 - network systems 13
 - neurosciences 13
 - radio-frequency amplifier/antenna design 275
 - structural dynamics 97, 118
 - geometrically nonlinear 104
- applied force method 122
- approximation theory 215
- Arnoldi method 166
- asymptotic waveform evaluation (AWE) 59, 60
-
- Baker–Campbell–Hausdorff series 268
- balancing-based MOR 8, 15, 17, 257
 - bounded real balanced truncation 16, 30
 - cross-Gramian balanced truncation 35
 - frequency-limited balanced truncation 16
 - frequency-weighted balanced truncation 16
 - \mathcal{H}_∞ balanced truncation 16
 - passivity-preserving balanced truncation 29
 - positive real balanced truncation 16, 29, 30
 - square root balanced truncation 24, 30, 38
 - stochastic balanced truncation 16, 33
 - left spectral factor balanced truncation 33
 - right spectral factor balanced truncation 33
- band-stop filter 199
- bandlimited Gramian 150
- Bartels–Stewart algorithm 26
- Bayesian approach 356, 366
- Ben-Tal 367
- benchmark CD player 209
- Bessel function 214
- best linear unbiased estimator (BLUE) 358
- best linear unbiased predictor (BLUP) 358
- beta distribution 360
- Bézier curves 249
- bi-invariant metric 255
- BICOMB method 81
- bifurcation 154
- big data 368
- bijective differentiable function 235
- bilinear control systems 41
- Bochner theorem 317
- Bode plot 46
- bootstrap method 359, 367
- bounded real (BR) conditions 164
- bounded real Gramian 31
- Bounded Real Lemma (BRL) 164
- bounded realness 163
- Brownian Bridge kernel 316, 319
- Burgers’ equation 183
-
- Carleman linearization 41
- Carleman linearization process 79
- chance-constrained optimization 364
- Chebyshev norm 205
- Cholesky decomposition 329
- Cholesky factorization 38, 45, 151
- classic unbiased estimator 363
- combinatorial problem 328
- common random numbers (CRN) 363, 364
- compact Stiefel manifold 244
- component mode synthesis 3, 103
- conditional simulation method 359
- confidence interval 359
- congruence transformations 257

- constrained optimization 153, 363, 367
- contractivity 30
- controllability Gramian 150
- convex optimization 153
- coordinate chart 234
- Coset spaces 244
- covariance kernel 330
- covariant derivative 240
- cross approximation algorithm 205
- cross-Gramian 35
- cumulative distribution function (CDF) 360, 362, 366
- CUR factorization 181, 182, 203, 204
- curse of dimensionality 368

- DAE control system 36, 39
- data splitting 208
 - alternate splitting 208
 - disjoint splitting 208
 - magnitude alternate splitting 208
 - magnitude splitting 208
- data-based cost functions 152
- De Casteljau algorithm 249
- deep learning 313
- descriptor realization 191
- descriptor systems 36, 162, 163
- design and analysis of computer experiments (DACE) 355
- deterministic error bounds 338
- deterministic simulation 356, 363
- differentiable manifold 234
- differential submanifold 236
- differential geometry 233, 234
- differential-algebraic equations (DAEs) 5, 16, 36
 - higher index DAEs 162
 - linear DAEs 36
 - semi-explicit DAEs 39
 - non-linear DAEs 36
 - quadratic–bilinear DAEs (QBDAEs) 87, 89
 - quadratic–linear DAEs (QLDAEs) 87
 - Stokes-like DAEs of index 2 39
 - index-one DAEs 162, 163
 - index-two DAEs 162
- digital filter 303
- discontinuous Galerkin method 306
- dissipation inequality 142
- distributed systems 303
- domain decomposition technique 332
 - partition of unity method 332

- dual modes (DMs) 114
- dynamic mode decomposition (DMD) 4, 12
- dynamical systems 349
 - bilinear systems 78, 183
 - discrete-time systems 43, 278
 - finite-dimensional systems 141
 - geometrically nonlinear systems 110
 - infinite-dimensional systems 16
 - linear mechanical systems 97
 - linear parametric systems 73, 78, 92
 - linear switched systems 183
 - linear time varying systems 16, 183
 - mildly nonlinear systems 78
 - multi-port linear systems 183
 - non-parametric systems 71
 - nonlinear mechanical systems 97
 - nonlinear systems 16
 - parametric bilinear systems 92
 - parametric nonlinear systems 90, 92
 - parametrized linear systems 183
 - parametrized systems 166
 - periodic discrete-time systems 16
 - polynomial systems 183
 - quadratic bilinear systems 87, 183
 - quadratic parametric systems 91
 - quadratic systems 183
 - second-order systems 44, 75
 - symmetric second-order systems 45
 - singular/rectangular systems 183
 - weakly nonlinear systems 87

- efficient global optimization (EGO) 365, 368
- eigenvalue decomposition (EVD) 161, 191, 233, 256, 268
 - generalized EVD 191
 - symmetric EVD problems 255
- eigenvalue perturbation 154
 - first-order eigenvalue perturbation 154
- eigenvalue problem 99, 112
- embedded Riemannian submanifolds 239
- embedded submanifolds 235
 - n -dimensional embedded submanifold 235
- emulator 355
- enforced displacement (ED) method 118, 121, 123, 125, 126, 133
- enhanced enforced displacement (EED) method 121–123, 125, 126, 133
- equivalent electric circuits 299
- Euclidean distance 361

- Euler–Bernoulli beam 111, 181, 209, 217
- evolutionary algorithms (EAs) 365
- excursion set 366
- expansion kernels 316
- expected improvement (EI) 366
- experimental design 363
- external noise 364
- extrinsic noise 357

- factor screening 368
- feature maps 314–316, 320
- finite difference time domain (FDTD) method 306
- finite dimensional spaces 318
- finite element (FE) formulation 97
- finite element time domain method 306
- finite-time controllability Gramian 19
- finite-time observability Gramian 19
- first-order perturbation analysis 167
- first-order sensitivity index 365
- flat manifold 258
- flat metric 253
- Fokker–Planck equation 41
- Fourier transform 287
- frequency-domain analyses 300
- frequency-domain inequalities 139, 144, 165
- Frobenius norm 152, 289
- functional analysis of variance (FANOVA) 355, 357, 364, 365

- Galerkin orthogonality 10
- Galerkin projection 4, 10, 58, 63, 71, 75, 87, 91, 105, 116
- Gaussian correlation function 359, 368
- Gaussian kernel 345
- Gaussian process regression 330
- Gaussian processes (GPs) 9, 355
 - stationary GP 357
- general linear group $GL(n)$ 252
- generalized controllability matrix 185
- generalized eigenvalue problem 160, 163
- (generalized) Hamiltonian eigenvalue problem 163
- generalized inverse approach 198
- generalized observability matrix 187
- generalized Sanathanan–Koerner form 167
- generalized Smith method 38
- geodesic averaging 249
- geodesic endpoint problem 241, 259, 265
- geodesic extrapolation 250
- geodesic interpolation 247
- geodesics 238, 239
- geometric first-order Taylor approximation 250
- geostatistics 355
- Gilbert realization 298, 303
- global sensitivity analysis 364
- gradient descent method 248
- gradient-based optimization method 341
- Gram–Schmidt orthonormalization 329
- Gramian-based cost functions 150, 163
- Grassmann exponential 264, 265
- Grassmann logarithm 265
 - modified version 266
- Grassmann manifold 234, 243, 263
- greedy algorithm 10, 11, 69, 110

- \mathcal{H}_∞ norm condition 164
- half-size passivity tests 148
- Hamilton–Jacobi equations 16
- Hamiltonian matrices 139, 145–147, 154, 164, 165
 - perturbed Hamiltonian matrices 154
- Hamiltonian perturbation 157
- Hamiltonian-based passivity check 149, 163, 168
- Hamiltonian-based passivity constraints 154
- Hammarling’s method 26
- Hankel matrices 205
- Hankel operator 23
- Hankel singular values 23, 25, 36, 38
 - improper Hankel singular values 38
 - proper Hankel singular values 38
- Hardy space 19
- heat equation 46, 209, 218
- Helmholtz (wave) equation 303
- Hermite interpolation 224
- Hermitian manifold interpolation 249
- Hessian 114
- high fidelity model (HFM) 98
- Hilbert space 315
- Hilbert–Schmidt kernels 316
- Hopf–Rinow theorem 256
- Hurty–Craig–Bampton method 104
- hyperreduction techniques 11, 107, 129, 134
 - discrete empirical interpolation method (DEIM) 108–110, 205, 206, 231
 - empirical interpolation method (EIM) 3, 11, 231

- energy conserving sampling and weighting (ECSW) method 108, 110, 129
- missing point estimation (MPE) 231
- implicit condensation method 122, 125, 126
- implicit function theorem 244
- improper controllability Gramian 38
- improper observability Gramian 38
- inclusion relations 320
- index-preserving methods 4
- infinite dimensional optimization problem 323
- infinite-time controllability Gramian 20, 24
- infinite-time observability Gramian 20, 24
- integrated mean squared prediction error (IMSPE) 364
- internal noise 364
- interpolatory projection framework 4
- interpolatory projections 183
- interpolatory projectors 189
- intrinsic noise 356, 357, 363
- inverse Laplace transform 297
- inverse problem 154
- iterative rational Krylov algorithm (IRKA) 7, 67, 68, 71, 182, 219
 - bilinear IRKA (BIRKA) 78, 89
 - \mathcal{H}_2 -optimal IRKA 67
- Jacobi identity 242
- k -fold cross validation 342, 343, 345
- Kalman–Yakubovich–Popov (KYP) lemma 28, 144, 164, 165
- Karcher mean 249
- Karhunen–Loeve decomposition 116
- Karush–Kuhn–Tucker (KKT) conditions 335, 338
 - complementarity KKT conditions 336
 - stationarity KKT conditions 336
- kernel methods 9, 311, 312
 - greedy kernel approximation 311
 - incremental regularized interpolation 329
 - kernel greedy approximation 327, 328, 331
 - kernel interpolation 311, 314, 322, 324
 - pattern analysis 314
 - regularized kernel interpolation 326
 - scattered data interpolation 314
 - support vector regression 311, 314, 322, 333, 337, 344–347
 - ν -SVR 337
 - ε -SVR 337
 - alternative primal form 334
 - dual form 335
 - primal form 334
- vectorial kernel orthogonal greedy algorithm (VKOGA) 332, 344, 345
 - f -greedy VKOGA 346, 347, 349
 - f/P -greedy VKOGA 346, 347
 - P -greedy VKOGA 346, 347
 - regularized VKOGA 345
- Kirchhoff’s laws 36
- Kriging (hyper)parameters 359
- Kriging method 9, 330, 355
 - hetGP 363
 - intrinsic Kriging 356
 - ordinary Kriging 357
 - stochastic Kriging 363
 - stochastic Kriging (SK) method 356
 - universal Kriging 357, 359
- Kriging variance 330, 359
- Krylov subspace method 26, 34, 38, 62, 166, 263
- Krylov subspaces 62, 81
 - block Krylov subspace 82
- Lagrange multiplier 104, 336
- Lagrangian framework 106
- Lanczos method 166
- Laplace transform 59, 184
- Latin hypercube sampling (LHS) 355, 356, 360, 365, 366
- Laurent series 185
- least absolute shrinkage and selection operator (LASSO) method 333
- least squares estimator 358
- least squares problem 116
 - linear 278
 - nonlinear 278, 289
- leave one out cross validation (LOOCV) 343
- left spectral factor 33
- left-invariant metric 253
- Levi-Civita connection 240
- Levy functional 278
- Lie algebra 242
- Lie bracket 242
- Lie groups 243
- Lie subgroups 243
- linear connection 240
- linear matrix inequalities (LMIs) 28, 139, 143, 144, 164

- linear quadratic Gaussian (LQG) Gramian 32
- linear quadratic optimal regulator problem 31
- linear regression 313, 356, 357
- linear time invariant (LTI) systems 6, 17, 57, 63, 139, 141, 182, 233
 - finite-dimensional systems 141
 - non-parametrized LTI systems 70
 - parametrized LTI systems 8
- linearly constrained minimum norm problem 156
- LMI-based passivity check 147
- local (adaptive) sampling 148
- local passivity violations 154
- local perturbation 158
- local sensitivity analysis 364
- Loewner CUR algorithm 205
- Loewner framework 4, 166, 167, 181, 182, 188, 291
 - LoewCUR 220
 - LoewCUR-cross 216, 219
 - LoewCUR-DEIM 219
 - Loewner-MIMO 182, 194
 - Loewner-SISO 182, 193
 - LoewSVD 216, 220
 - parametrized Loewner framework 167
- Loewner pencil 189, 193, 196
- Loewner quadruple 189, 202
- log-Euclidean metric 258
- log-likelihood function 359
- low-rank generalized ADI method 38
- LTI control systems 27
- Lur’e matrix equation 28, 29, 32, 34
 - bounded real Lur’e equation 30
 - dual Lur’e equation 29, 32
 - positive real Lur’e equation 30, 33
- Lyapunov condition for (simple) stability 147
- Lyapunov equations 3, 20, 25, 33, 34, 150, 151
 - continuous-time Lyapunov equations 43
 - discrete-time Lyapunov equations 43
 - periodic discrete-time Lyapunov equations 44
 - projected continuous-time Lyapunov equations 37, 38
 - projected discrete-time Lyapunov equations 38
- machine learning 312, 349, 355
- main-effect index 365
- main-effect variance 365
- manifold interpolation 230, 249
- Matérn correlation function 368
- Matheron 355
- matrix commutator 242
- matrix Lie group 242
- matrix manifolds 229, 233, 246, 250, 252
- matrix rational approximation 193
- maximin criterion 361
- maximum likelihood estimator 358, 359
- Maxwell’s equations 2, 36, 168, 276
- McMillan degree 189, 191, 303
- mean squared residual 358
- mean-to-variance ratio 367
- metamodel 355, 356
- modal coordinates 99
- modal decomposition 97
- modal decomposition method 101
- modal derivatives (MDs) 112, 114, 129, 130
 - static modal derivatives (SMDs) 113
- modal displacement method (MDM) 100, 115, 132
- modal methods 97
- modal truncation 3
- modal truncation augmentation (MTA) technique 102
- modal truncation augmentation vectors 103
- mode acceleration correction (MAC) method 101, 103
- model order reduction (MOR) 1, 2, 15, 140, 312, 349
 - balancing-based MOR 3
 - data-driven MOR 4, 131, 152, 275, 276
 - Gramian-based bilinear MOR 80
 - interpolatory MOR 188
 - localized MOR 12
 - non-intrusive MOR 4, 117
 - parametric MOR (PMOR) 8, 58, 71, 229, 230
 - structure-preserving PMOR 75
 - variational analysis-based PMOR 92
 - bilinear PMOR 92
 - quadratic PMOR 91
 - projection-based MOR 5, 130, 311
 - stability-preserving MOR 166
 - structure-preserving MOR 75
- model realization 297
- model selection 341–343
- modified Gram–Schmidt process 64, 65, 67, 70, 72

- moment-matching methods 7, 58, 59, 63, 78, 131, 182
 - bilinearization method 78, 79
 - quadratic method 78
- Monte Carlo integration 340
- Monte Carlo sampling 360
- Moore–Aronszajn theorem 320
- multi-fidelity Monte Carlo approximations 312
- multi-input multi-output (MIMO) systems 58, 59, 69, 76, 188, 199, 289, 303
 - symmetric MIMO systems 36
- multi-moment-matching methods 58, 71, 80
 - multi-moment matching PMOR methods 71, 75, 91, 92
- multi-scale simulations 312
- multivariable Maclaurin series 81
- natural metric 258
- Navier–Stokes equations 36, 230
- negative imaginary systems 158
- neural networks 356, 367
- Newton basis 329, 332
- Newton–Raphson iteration 128
- Newton’s method 34
- non-flat manifolds 244
- nonlinear manifolds
 - quadratic manifold 129, 130, 134
 - spectral submanifold (SSM) 129, 134
- nonlinear normal modes (NNMs) 127
- nonsymmetric Lanczos process 60
- normal coordinates 241
- normal (or Gaussian) distribution 355
- observability Gramian 37
- offline phase 326, 328, 331, 337
- offline–online decomposition 11
- online phase 232, 324, 332
- optimal control problems 312
- orbit space 243
- order estimation 301
- ordinary differential equations (ODEs) 5, 233
 - second-order ODEs 98
- orthogonal group 235, 237, 255
- Oseen equations 183
- Padé approximation 60
- Padé via Lanczos (PVL) algorithm 60
- Padua points 215
- para-Hermitian pencils 163
- parameter identification 312
- parameter selection 341
- parameter-space sampling procedures 131
 - Latin Hyper-cube 131
 - Smolyak sparse grid 131
- Pareto-optimal efficiency frontier 367
- partial differential equations (PDEs) 26, 57, 233, 312
 - parabolic second-order PDEs 26
 - parametric linear PDE 10
 - parametric PDEs 3
- particle swarm optimization (PSO) 365
- passive and reduced-order interconnect
 - macromodeling algorithm (PRIMA) 60, 63, 141, 166
- passivity enforcement 139, 153, 166, 173
 - via Hamiltonian perturbation 154, 157
 - via LMI constraints 153
 - via local perturbation 158, 160
- passivity-preserving interpolation scheme 167
- perturbation approach 149
- Petrov–Galerkin method 10
- Petrov–Galerkin projection 18, 58, 72, 75, 78, 105, 183, 185, 231
- “phi-divergence” uncertainty sets 367
- Plancherel theorem 19
- plug-in predictor 359
- plug-in variance estimator 359
- POD-greedy algorithm 10
- polar decomposition 268
- polynomial approximation 82
- polynomial interpolation method 313, 329
- polynomial kernels 316
- poor man’s truncated balanced realization (PR-TBR) algorithm 141
- position controllability Gramian 45
- position observability Gramian 45
- positive definite kernels 314, 315
 - matrix-valued 321
- positive real Gramian 30
- Positive Real Lemma (PRL) 144, 146, 153, 163
- positive real (PR) conditions 144, 158
- principal angles 267
- principal component analysis (PCA) 16, 116
- principal matrix logarithm 234, 257, 267
- printed circuit board (PCB) 172, 291
- Procrustes problem 266, 269
- proper controllability Gramian 37
- proper generalized decomposition (PGD) 3, 11

- proper orthogonal decomposition (POD) 3, 11, 59, 109, 116, 117, 129, 131, 230, 233
- pseudo-random numbers (PRNs) 356
- QR decomposition 278, 283, 291, 296
 - incremental QR factorization 206
- quadratic optimization problem 334
- quadratic–bilinear control systems 42
- quadratic–bilinearization method 87
- quasi-geodesics 260
- quasi-linear extrapolation 250
- queuing simulation models 356
- quotient manifold theorem 243
- quotient manifolds 243
- quotient map 243
- quotients 243
- radial basis function (RBF) 317
- radial basis function (RBF) kernels 317, 327
- radial basis function (RBF)–QR algorithm 327
- random simulation 356, 357, 363
- rank-revealing factorization 184, 198
- rational functions 277
- rational interpolation method 61, 63, 185
- Rayleigh damping 99
- realness condition 294
- reciprocal system 35
- reduced basis method (RBM) 11, 59, 70
- reduced-order controller
 - linear quadratic Gaussian (LQG) balanced truncation 16, 31
- reduced-order models (ROMs) 5, 16, 173, 250
 - nonlinear reduced-order model (NLROM) 97, 105, 107
 - parametric ROM (PROMs)
 - nonlinear PROMs 130
- reduced-order variational formulation 11
- regularized interpolant 324
- representer theorem 322, 326, 334
- reproducing kernel Hilbert space 313, 318, 319, 349
 - matrix-valued kernels 321
- response surface methodology (RSM). 367
- Riccati equation 28
- Riemannian barycenter 245
- Riemannian center of mass 245, 247
- Riemannian connection 240
- Riemannian curvature tensor 240
- Riemannian distance function 238
- Riemannian exponential 234, 240, 250, 253
 - SPD(n)-exponential 259
- Riemannian exponential maps 256
- Riemannian logarithm 234, 241, 245, 253
 - SPD(n) logarithm 259
- Riemannian logarithm maps 256
- Riemannian metric 234, 238, 241, 244, 253
- Riemannian objective function 248
- Riemannian optimization problem 248
- Riemannian shooting method 254
- Riemannian Stiefel logarithm 260
- Riesz representer 318
- right spectral factor 33
- risk analysis 360
- Ritz vectors 102
- Ritz–Galerkin method 10
- robust optimization (RO) 365, 366, 368
- Runge phenomenon 208, 215
- Sanathanan–Koerner algorithm 276–279, 305
- scalar rational approximation 193
- scattering representation 163, 164, 292
- Schmidt–Eckart–Young–Mirsky theorem 203
- Schur decomposition 166
- sensitivity analysis 355–357, 364
- separable matrix valued kernel 321, 332, 340
- sequential minimal optimization (SMO)
 - algorithm 338, 339
 - 2-index SMO 344
- signal-to-noise ratio 367
- simulation-based classification 312
- single-input single-output (SISO) systems 58, 59, 64, 67, 76, 185, 192, 195, 205, 289
 - non-parametric SISO systems 66
 - SISO bilinear systems 81
- singular perturbation approximation 16, 34
- singular systems 36
- singular value decomposition (SVD) 12, 25, 45, 109, 115, 117, 181, 182, 203, 215, 230, 233, 251, 291, 299
 - Hilbert–Schmidt SVD 327
 - randomized SVD 181, 182
 - rank-revealing SVD 193, 194, 202
 - truncated SVD 298
- singular value functions 40
- singular values 164
- slack variables 334
- Sobol 365
- Sobol sensitivity indices 355, 357

- Sobolev space 317
- software package
 - and Frontline Systems' Risk 362
 - ARPACK 205
 - MATLAB 365
 - Systems and Control Toolbox 26
 - MATLAB Statistics toolbox 362
 - Microsoft's Excel spreadsheet software 362
 - Open TURNS 365
 - Oracle's Crystal Ball 362
 - Risk 362
 - R package 362
 - Sandia's DAKOTA software 362
- space-filling design 360
- sparse kernel expansions 311
- sparsity inducing techniques 333
- spatial statistics 355
- special orthogonal group 255
- spectral perturbation 154
- spring–mass–damper system 195
- standard matrix exponential 267
- state estimation 312, 339
- static condensation method 110, 128
- statistical learning theory 337
- Stiefel exponential 261, 262
- Stiefel logarithm 262
- Stiefel manifold 234, 243, 251, 260
- stiffness evaluation procedure (STEP) 118
- stochastic Gramian 34
- stochastic simulation 356
- structure-preserving interpolation 229
- substructuring 3, 103
- support vector machines (SVMs) 337
 - LIBSVM 339
 - liquidSVM 339
- surrogate modeling 311
- Sylvester equation 36, 187, 189
- system identification 107
 - data-driven 197
- Taguchi 366
- tangent spaces 236
- tangential interpolation problem 188
- Taylor extrapolation 251
- Tellegen's theorem 143
- trace metric 258
- trace parameterization 154
- trajectory piece-wise linear method 89
- trajectory piece-wise linear (TPWL) method 250
- translational invariant kernels 317, 320, 338
- transport equation 48
- triangular distribution 360
- two-factor interaction variance 365
- uncertainty analysis 360
- uncertainty quantification (UQ) 13, 339, 356
- validation set 342
- Vandermonde blocks 279
- Vandermonde matrices 278
- variational analysis method 82, 86
- vector fitting method 8, 142, 150, 166–168, 172, 182, 216, 219, 220, 275, 276, 279, 281, 285, 291, 299, 300, 305
 - fast vector fitting algorithm 290
 - real-valued version 294, 297
 - orthonormal vector fitting algorithm 303
 - parametric vector fitting algorithms 304
 - quadrature vector fitting algorithm (QuadVF) 303
 - relaxed vector fitting algorithm 301
 - time domain vector fitting algorithm 302
- velocity controllability Gramian 45
- velocity observability Gramian 45
- Volterra series 81, 86
- von Karman kinematic model 106
- Weierstrass canonical form 37
- weighted residual method 10
- weighted Riemannian center 249
- Wendland kernels 317
- what if analysis 364
- white noise 357
- Zolotarev's fourth problem 221