CHEMNITZ UNIVERSITY OF TECHNOLOGY

Efficient Numerical Solution of Large Scale Algebraic Matrix Equations in PDE Control and Model Order Reduction

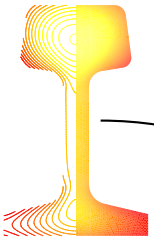
Dissertation

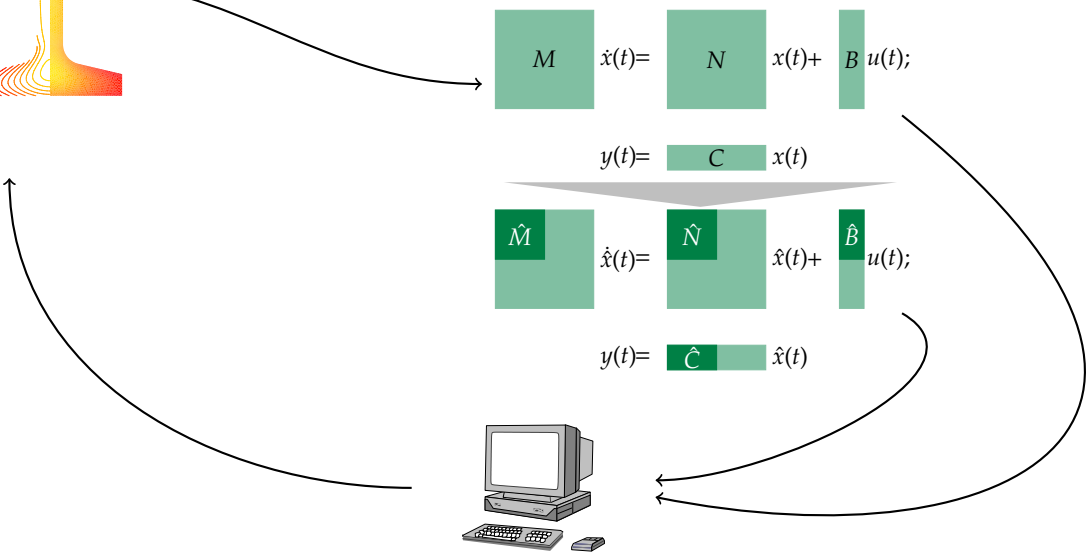
submitted to **Faculty of Mathematics**

at **Chemnitz University of Technology**

in accordance with the requirements for the degree

Dr. rer. nat.





presented by: Dipl. Math. Jens Saak

Advisor: Prof. Dr. Peter Benner

Reviewers: Prof. Dr. Enrique S. Quintana-Ortí

Prof. Dr. Ekkehard W. Sachs

Chemnitz July 6, 2009

to Şenay

ACKNOWLEDGEMENTS

Financial Support. Large parts of this research have been refined in the projects *Parallele numerische Lösung von Optimalsteuerungsproblemen für instationäre Diffusions-Konvektions-Reaktionsgleichungen* (project A15 in SFB393 *Parallele Numerische Simulation für Physik und Kontinuumsmechanik*), *Numerische Lösung von Optimalsteuerungsproblemen für instationäre Diffusions-Konvektions- und Diffusions-Reaktionsgleichungen* and *Integrierte Simulation des Systems "Werkzeugmaschine-Antrieb-Zerspanprozess" auf der Grundlage ordnungsreduzierter FEM-Strukturmodelle* supported by the German Research Foundation (DFG) over the past years. Besides these, the integration and exchange project *Parallele Algorithmen für hochdimensionale, dünnbesetzte algebraische Riccatigleichungen und Anwendungen in der Regelungstheorie* inside the *Acciones Integradas Hispano-Alemanas* program of the German Academic Exchange Service (DAAD) has enabled me to undertake some very helpful and inspiring trips to *Universitat Jaume I* in Castellón (Spain).

Personal Thanks. My primary thanks go to my advisor and teacher Peter Benner for the introduction to and the guidance in this highly fascinating field of research with all its interesting types of applications. On the other hand the best mentor still depends on the many people working in the background. Therefore my thanks go to the colleagues and friends that I had the pleasure to work with at TU Chemnitz during the past almost 6 years. I cannot mention single persons without forgetting other important ones. So I will only pick the three most important ones, Ulrike Baur, Sabine Hein and Hermann Mena. As for Hermann I can only cite himself. *I am particularly grateful to my dear friend* Hermann Mena *together with whom (among the most interesting topics in life) large parts of this work were discussed in our mid-afternoon coffee breaks.* The countless inspiring discussions with Sabine have given me an increasingly deeper insight to many aspects of LQR and LQG design for parabolic PDEs. Also, I would know hardly half as much as I do today about model order reduction, if there had not been Ulrike providing me with endless advice and numerous suggestions regarding the field.

I also want to thank Enrique S. Quintana-Ortí and his workgroup at *Universitat Jaume I* for the warm welcomes in Spain and for providing me a quiet place to work out

the foundation of this document undisturbedly. Especially Alfredo Remón and Sergio Barrachina have always been more than helpful in organizing my journeys and made the visits to Spain some of the most enjoyable time of the past years.

My special thanks are expressed to the student assistant Martin Köhler for the massive work in implementing the upcoming C.M.E.S.S. library and performing the extensive testing that made large parts of Section 4.4.3 and the corresponding numerical results in Chapter 8 possible.

My family and many friends have supported me in hundreds of ways over the years and I hope that I have outweighed their help adequately every now and then.

Finally I want to thank the most important people in the current period of my life and the best friends that I can think of. Although both of them may have been hundreds to thousands of kilometers away almost all the time, they have been closer than anyone. Lars Fischer, whom I cannot thank enough for constantly pushing me a few steps further than I actually wanted to go – it has always been a source of personal progress. Last and definitely most importantly I thank Şenay Yavuz, who opened the door to all of this. I owe her more than I can ever tell or repay.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ACRONYMS

ADI	alternating directions implicit (iterative parametric solver, see, e.g. [112, 145])
AMD	approximate minimum degree (reordering)
(C)ARE	(continuous time) algebraic Riccati equation
ATLAS	**A**utomatically **T**uned **L**inear **A**lgebra **S**oftware
BLAS	**B**asic **L**inear **A**lgebra **S**ubprograms
BT	balanced truncation
C.M.E.S.S.	C version of M.E.S.S.
DRE	differential Riccati equation
DSPMR	dominant subspace projection model reduction
EVP	eigenvalue problem
FDM	finite difference method
FEM	finite element method
KPIK	**K**rylov **P**lus **I**nverse **K**rylov (a rational Krylov subspace based solver for Lyapunov equations)
LAPACK	**L**inear **A**lgebra **PACK**age
LQG	linear-quadratic Gaussian
LQR	linear-quadratic regulator
LRCF	low-rank Cholesky factor. A factorization $A \approx LL^H$, where $L \in \mathbb{R}^{n\times m}$ with $m \leq \operatorname{rank}(A) < n$. If $\operatorname{rank}(L) = k = \operatorname{rank}(A)$ also called full rank factor.
LRCF-ADI	versions of ADI computing LRCFs of the solution rather than the solution itself.
G-LRCF-ADI	version of LRCF-ADI for generalized Lyapunov equations.
LRCF-ADI-S	LRCF-ADI starting with an initial guess $Z_0 \neq 0$
LRCF-NM	the LRCF based Newton method for solving large scale sparse AREs
LRCFP	dyadic product LL^H of LRCF L
LR-SRM	low-rank square-root method
LR-SRBT	low-rank square-root balanced truncation method
LTI	linear time invariant (system)
LTV	linear time varying (system)

LyaPack	**LYA**punov **PACK**age; A MATLAB toolbox for the solution of certain large scale problems in control theory, which are closely related to Lyapunov equations.
M.E.S.S.	**M**atrix **E**quation **S**parse **S**olver; the upcoming successor of LyaPack.
MIMO	multiple-input multiple-output (system)
ODE	ordinary differential equation
PDE	partial differential equation
RCM	reverse Cuthill-McKee reordering
RRQR	rank-revealing QR factorization
QADI	"Quadratic ADI" for AREs
SAMDP	subspace accelerated MIMO dominant pole algorithm
SISO	single-input single-output (system)
spd	symmetric positive definite
splr	sparse plus low-rank
SRBT	square-root balanced truncation
SVD	singular value decomposition
TFM	transfer function matrix

Sets and Spaces

$\mathbb{C}$	field of complex numbers
$\mathbb{C}_{>0}$, $\mathbb{C}_{<0}$	open right/open left complex half plane
$\mathbb{R}$	field of real numbers
$\mathbb{R}^n$	vector space of real n-tuples
$\mathbb{C}^n$	vector space of complex n-tuples
$\mathbb{R}^{m\times n}$	real $m \times n$ matrices
$\mathbb{C}^{m\times n}$	complex $m \times n$ matrices
Ω	computational domain; subset of $\mathbb{R}^k$, for $k \in \{1, 2, 3\}$
$\mathcal{L}^p(\Omega)$	Banach space of Lebesgue measurable functions defined on Ω and bounded with respect to the norm $\|u\|_p$ (see below)
$\mathcal{H}^{m,p}(\Omega)$	Sobolev space over Ω with differentiation index m and integration index p
$\mathcal{H}^k(\Omega)$	$:= \mathcal{H}^{k,2}(\Omega)$
$\mathcal{H}^1([t_0, T_f); \mathcal{H}^1(\Omega))$	Bochner space of $\mathcal{H}^1$-regular functions on $[t_0, T_f) \times \Omega$
$\mathcal{L}(\mathcal{X}, \mathcal{Y})$	set of linear, continuous operators mapping from $\mathcal{X}$ to $\mathcal{Y}$, for $\mathcal{X}$,$\mathcal{Y}$ normed vector spaces.
$\mathcal{K}(\mathcal{X}, \mathcal{Y})$	subset of compact operators in $\mathcal{L}(\mathcal{X}, \mathcal{Y})$
$\mathcal{X}'$	$:= \mathcal{L}(\mathcal{X}, \mathbb{F})$ dual space a.k.a. space of linear functional maps from $\mathcal{X}$ to the underlying field $\mathbb{F} = \mathbb{C}, \mathbb{R}$
(u, v)	inner product of $u, v \in \mathcal{X}$
$u \cdot v$	short version for the inner product of $u, v \in \mathcal{X}$ (mainly used in the formulation of PDEs)
$< f, u >$	$:= f(u(\xi))$ dual pairing $u \in \mathcal{X}$, $f \in \mathcal{X}'$
$\overset{\circ}{\mathcal{M}}$	the interior of $\mathcal{M}$
$\bar{\mathcal{M}}$	the closure of $\mathcal{M}$
$\mathcal{B}_\varepsilon(\xi)$	the ball of radius $\varepsilon > 0$ around ξ.

$\mathcal{H}_\infty(\mathbb{C}_{>0} \to \mathbb{C}^{m\times n})$	the Hardy space of bounded analytic functions from the positive complex half-plane to complex $m \times n$ matrices (boundedness is taken with respect to the Hardy-norm).

Matrices

a_{ij}	the i, j-th entry of A
A^T	the transpose of A
A^H	$:= (\bar{a}_{ij})^T$, the conjugate transpose
A^*	either of the above depending on the context; also the Hilbert space adjoint for operators A.
$\Lambda(A)$	spectrum of matrix A
$\lambda_j(A)$	j-th eigenvalue of A
$\rho(A)$	spectral radius of A
$\sigma_{\max}(A)$	largest singular value of A
$\mathrm{tr}\,(A)$	$:= \sum_{i=1}^n a_{ii}$ trace of A
$\mathrm{cond}_p\,(A)$	$:= \|A\|_p \|A^{-1}\|_p$ the p-norm condition number for A
$\mathrm{cond}\,(A)$	the 2-norm condition number for A
$A > 0; A \geq 0$	short form for A is selfadjoint positive definite or positive semi-definite respectively
$A > B$	$:\Leftrightarrow A - B > 0$
$A \geq B$	$:\Leftrightarrow A - B \geq 0$
$A \otimes B$	the Kronecker product of A and B.

Norms

$\|u\|_p$	$:= \sqrt[p]{\int_\Omega \|u\|^p}$ for functions $u(\xi)$ and $1 \leq p < \infty$
	$:= \sqrt[p]{\sum_{i=1}^n \|u_i\|}$ for n-tuples u and $1 \leq p < \infty$
$\|u\|_\infty$	the maximum norm, i.e., the maximum absolute value of components (u an n-tuple) or function values (u a continuous function)
$\|A\|_p$	$:= \sup\left\{\|Au\|_p : \|u\|_p = 1\right\}$ for operators A (including matrices) and $1 \leq p \leq \infty$
$\|A\|_F$	$:= \sqrt{\sum_{i,j} a_{ij}^2} = \sqrt{\mathrm{tr}\,(A^*A)}$ the Frobenius-norm of matrix $A \in \mathbb{R}^{m\times n}$

$\|u\|_{m,p}$	Sobolev Norm for space $\mathcal{H}^{m,p}(\Omega)$ (see Section 2.1 for detailed definition)
$\|.\|_{\mathcal{H}_\infty}$	Hardy-norm

Operators

$\partial_t f := \frac{\partial}{\partial t} f$	the derivative with respect to time of f, often abbreviated as $\dot{f} = \frac{\partial}{\partial t} f$.
$\partial_j f := \frac{\partial}{\partial x_j} f$	j–th partial derivative of f.
$\partial_j^k f := \underbrace{\partial_j \dots \partial_j f}_{k\text{–times}}$	k–fold j–th partial derivative,
$\partial^\alpha f := \partial_1^{\alpha_1} \dots \partial_n^{\alpha_n} f$	is the α–th partial derivative of f, for the multi-index α
$\nabla f := (\partial_1 f, \dots, \partial_n f)^T$	the gradient of f.
$\partial_\nu f := \nu \cdot \nabla f$	the derivative of f in direction of ν. In case of the outer normal simply the normal derivative of f.
$\Delta f := \sum_{i=1}^{n} \partial_i^2 f$	the *Laplacian operator* applied to f.
$\mathrm{tr}\,(.)$	the trace operator $\mathcal{H}^1(\Omega) \to \mathcal{L}^2(\partial\Omega)$.
$e^{t\mathbf{A}}$	(analytic) operator semigroup generated by $\mathbf{A}$.

The whole is more than the sum of the parts.

Metaphysica
ARISTOTLE

CHAPTER

ONE

INTRODUCTION

Motivation. Matrix equations play an important role in many applications. Two of the important fields are the balancing based model order reduction of large linear dynamical systems and the linear-quadratic optimal control of parabolic partial differential equations. The thesis at hand picks these two applications as references to demonstrate that efficient methods to solve continuous time algebraic matrix equations do exist even for large scale sparse applications. We will discuss the solution of large sparse standard continuous time algebraic Lyapunov equations

$$FX + XF^T = -GG^T, \tag{1.1}$$

as well as generalized Lyapunov equations

$$FXE^T + EXF^T = -GG^T, \tag{1.2}$$

and large sparse standard continuous time algebraic Riccati equations

$$C^TC + A^TX + XA - XBB^TX = 0, \tag{1.3}$$

as well as generalized Riccati equations

$$C^TC + A^TXE + E^TXA - E^TXBB^TXE = 0. \tag{1.4}$$

The notion "sparse" in this case refers to the sparsity of the quadratic matrices building these equations, i.e., the matrices A, F, and E. Although sparsity of the matrices B, C^T and G can help to reduce the computational effort, it is not crucial for the efficiency of the methods. The more important requirement of these matrices is that they are thin upright matrices, i.e., consist of a lot less columns than rows. However, note that even if the coefficient matrices are sparse, the solutions to equations (1.1)–(1.4) will in general be dense quadratic matrices. For very large matrices A and F, these can obviously not

be stored element by element, due to the quadratic memory demands. Motivated by the observation that the solutions X often have low numerical rank, one therefore computes thin rectangular matrices Z such that $X = ZZ^T$ rather than the solution itself.

One of the classes of methods that can be written in a way such that Z is computed rather than X, is the class of alternating directions implicit (ADI) based algorithms. We will concentrate on this class of methods throughout this thesis. Besides this class also some very fast Krylov subspace projection based methods for solving large sparse Lyapunov [131] and Riccati [69] equations have been presented in the literature. Banks and Ito [11] introduce a hybrid method combining the Chandrasekhar algorithm [39] with the Kleinman iteration [83] to solve the Riccati equation, which is further refined in [110]. Also the implicit low-rank Cholesky factor Newton method [18] reviewed in Chapter 4 can be seen as a modification of [11], although it was derived in a different context.

The projection based methods generally have the requirement that $A+A^T < 0, F + F^T < 0$, respectively, such that stability of the projected equations can be ensured and may fail when this is not the case. Note that this is not necessary in general for ADI based methods, such that these can stay applicable where the projection based methods fail. On the other hand, in examples where the spectrum of A or F is dominated by eigenvalues with large imaginary parts close to the imaginary axis, ADI normally shows very bad convergence properties and the Krylov subspace based methods should be favorable.

All the concepts presented in the context of the ADI methods for large sparse matrices can also be generalized to the case of data sparse matrices, i.e., matrices that allow the treatment as hierarchical matrices [60, 59].

Chapter Outline. The thesis is structured as follows. The following chapter introduces the basic notations and properties from the different fields of mathematical research applied in the subsequent chapters. **Chapter 3** then introduces the test examples and model problems used to illustrate the theoretical results and ideas. Most of the models are only sketched and more detailed descriptions are referenced. The modeling of the optimal cooling of rail profiles and the derivation of the system matrices for this model is treated with some more detail to give a better idea on the global process of solving these kinds of problems.

The largest part of the thesis and the main focus of this research – besides the extensive numerical testing in Chapter 8 – is taken by **Chapter 4** on the efficient solution of large scale matrix equations. Chapter 4 gives a review and short derivation of the basic low-rank ADI method and introduces some convergence-accelerating extensions and modifications to the existing algorithms. New shift parameter strategies are introduced and the projection idea of the aforementioned fast Krylov subspace methods is picked up to increase the quality of the iterates during the ADI process. Also column compression techniques optimizing the memory requirements and computational effort are discussed.

Chapter 5 then provides another of the main contributions of this thesis. The application of matrix pencil techniques in the ADI context avoids the decomposition of the mass matrix that was necessary in earlier approaches to the implicit transformation of the system to standard state space form.

The next two chapters shed some light on the fields of application of the matrix equation approaches: The linear quadratic regulator control of parabolic partial differential equations (Chapter 6) on the one hand and balancing based model order reduction on the other hand (Chapter 7). In the **Chapter 6**, on the LQR optimal control of PDEs, extensions of the existing linear stabilization theory to tracking type control systems and systems governed by quasilinear equations are discussed. Furthermore a new suboptimality result for the usage of numerically computed controls in the real world process is proven.

Chapter 7 on model reduction applications can be split into two parts. The first part has rather summarizing character providing a commented collection of facts regarding the application of low-rank techniques in balanced truncation of first order systems. The second part shows the new contribution of this thesis. It extends the application of low-rank balancing based model order reduction to second order systems in an efficient way that is capable of optimally exploiting the sparsity and structure of the original system matrices.

As mentioned above, **Chapter 8** then collects all the numerical experiments undertaken with the different methods introduced in the prior chapters. Appendix A contains the results on the approximation of the abstract Cauchy problem for the steel example by finite dimensional semi-discrete LQR Systems. Also, some implementation details on the underlying solver for the simulation task used to generate the system matrices are given. The appendix has rather repetitive character but gives some additional details for the rail model in Section 3.3.

The graph in Figure 1.1 visualizes the dependencies of the single chapters in relation to each other. Note that the dependencies on the basic concepts chapter are neglected, since every chapter depends on Chapter 2 in one way or another.

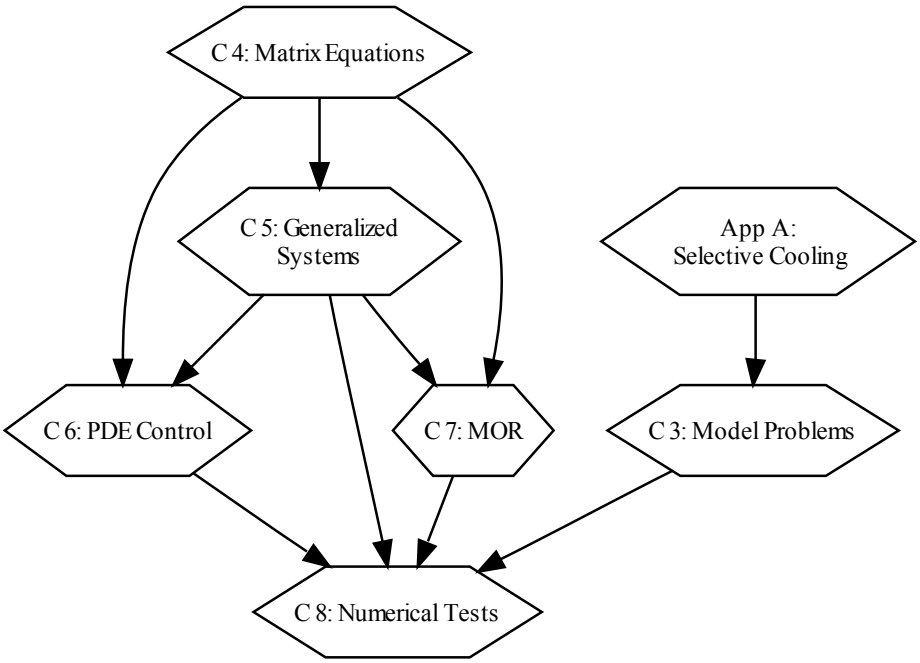


Figure 1.1.: Chapter dependencies

> Science is facts; just as houses are made of stones, so is science made of facts; but a pile of stones is not a house and a collection of facts is not necessarily science.
>
> HENRI POINCARE

CHAPTER

TWO

BASIC CONCEPTS

Contents

This chapter is intended to introduce the basic notation and present the most common concepts and results from the fields of research touched by the subsequent chapters of this thesis. It does not claim to be complete in any sense and proofs will only be given where it is absolutely necessary or where they might provide deeper insight to the interrelations of interest.

The chapter is organized as follows. In the first section the most basic notations and symbols will be introduced. Section 2.2 then provides the required concepts and results from systems and control theory needed to solve approximating systems in numerical applications, or simply support the understanding of concepts in infinite (operator based) theory by the well known finite dimensional (matrix based) analogues. In Section 2.3 the theoretic background for the model problems in Chapter 3 is presented. Further it contains the approximation results allowing us to apply the matrix equation solvers introduced in Chapter 4 to the model problems listed in Chapter 3. The chapter ends with a section introducing all the results and tools from the field of model order

reduction and especially balanced truncation based methods that are required as the basis for Chapter 7.

2.1. Notation

A table of the notations and symbols used can be found in the List of Symbols in the front matter. Although everything is listed there we will give an introduction to the notations again here, to be able to describe the one or the other symbol a bit more elaborately – especially when it comes to the description of function spaces and distinction between Sobolev spaces and Hardy spaces and their symbolic representations.

Throughout this thesis we will denote by $\mathbb{R}^{m\times n}$, $\mathbb{C}^{m\times n}$, the spaces of $m \times n$ real/complex matrices. The *complex plane* is denoted by $\mathbb{C}$ and the *open left half-plane* by $\mathbb{C}^-$. For a matrix A, A^T stands for the *transpose*, if $A \in \mathbb{C}^{m\times n}$, A^H denotes the *conjugate transpose*. The *identity matrix* of order n is denoted by I_n or just I if dimensions are evident. In case of an operator A the *Hilbert space adjoint* is denoted by A^*. We also write A^* for the *adjoint matrix*, falling back to the transpose in the real and conjugate transpose in the complex case. In all the above cases we write range (A) for the *range*, $\ker(A)$ for the *null space*, and dom (A) for the *domain* of A.

By $\mathcal{H}^{m,p}(\Omega)$ we denote the *Sobolev space* $\mathcal{H}^{m,p}(\Omega) = \{u \in \mathcal{L}^p(\Omega) : \partial^\alpha u \in \mathcal{L}^p(\Omega),\ |\alpha| \le m\}$ of m-times weakly differentiable functions in $\mathcal{L}^p(\Omega)$ with its *norm*

$$\|u\|_{m,p} := \left(\sum_{|\alpha|\le m} \int_\Omega |\partial^\alpha u(x)|^p dx \right)^{\frac{1}{p}}.$$

In general we will restrict ourself to the case of Hilbert spaces where $p = 2$ and the *scalar/inner product* is

$$(u,v)_{m,2} := \left(\sum_{|\alpha|\le m} \int_\Omega \partial^\alpha u(x) \partial^\alpha v(x) dx \right)^{\frac{1}{2}}.$$

We then write $\mathcal{H}^m(\Omega) := \mathcal{H}^{m,2}(\Omega)$. These spaces are appropriate to describe solutions of the elliptic equation associated to the parabolic problem. Let $[t_0, T) \subset \mathbb{R}$ be the time interval of interest. We will then write $\mathcal{H}^1([t_0, T); \mathcal{H}^1(\Omega))$ for the H^1-space defined with respect to the Bochner-integral analogously to the one above which uses Lebesgue-integrals, see, e.g., [146].

We denote the space of *linear, bounded operators* from a Banach space $\mathcal{X}$ to a Banach space $\mathcal{Y}$ by $\mathcal{L}(\mathcal{X}, \mathcal{Y})$ and its *subspace of compact operators* by $\mathcal{K}(\mathcal{X}, \mathcal{Y})$. In case $\mathcal{Y} = \mathcal{X}$ we simply write $\mathcal{L}(\mathcal{X})$ and $\mathcal{K}(\mathcal{X})$. Following the notation in [111] we call a linear operator A *dissipative* if for every $x \in$ dom $(A) \subset \mathcal{X}$ there exists $x^* \in \mathcal{X}^*$ with $< x^*, x >= \|x\|^2 = \|x^*\|^2$

and $\mathrm{Re}\,(< x^*, Ax >) \leq 0$, where the *duality product* $< \,.\,,\,. >$ is defined via $< x^*, x >:= x^*(x)$. Note that we distinguish between the duality product $< \,.\,,\,. >$ and the outer product $(\,.\,,\,.\,)$ here, although we are primarily working in Hilbert space settings where they can be identified via Riesz representation.

In contrast to the Sobolev spaces $\mathcal{H}^k$ we denote the important class of *Hardy spaces* by $\mathcal{H}_k$ with a lower index allowing us to distinguish them more easily.

When modelling a technical process by a partial differential equation (PDE) and formulating the corresponding control problem, we will need as much as three different representations of the system. The first step will be the reformulation of the PDE as an abstract Cauchy problem in an adequate Hilbert space setting. That one will be an infinite dimensional first order operator ordinary differential equation (ODE). For numerical considerations we then need to approximate this by abstract finite dimensional operator ODEs and finally discretize these to obtain matrix representations of the finite dimensional operators for use on the computer. Therefore we formulate the abstract infinite dimensional setting using bold letters ($\Sigma(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$), whereas the finite dimensional approximate systems are described in regular letters with an upper index N ($\Sigma(A^N, B^N, C^N, D^N)$) representing the approximating dimension. Finally the matrix representations of these finite dimensional operators will be given by regular letters with an upper index h ($\Sigma(A^h, B^h, C^h, D^h)$) reflecting the discretizations mesh width. Spaces and sets are generally written in calligraphic or math-black-bold letters to distinguish them from the operators easily.

2.2. Finite Dimensional Systems and Control Theory Basics

Here we only give a very brief introduction on the most important properties and results for theory of linear time invariant (LTI) finite dimensional (i.e., ODE related) control systems. An overview giving the required basics also for linear time varying systems with ODE constraints can be found in [13]. A nice introduction that is easily readable even at undergraduate level is given in [101]. [6] gives a more model reduction oriented introduction from a linear algebraic point of view. An in depth presentation of the topic can be found in textbooks like [70, 99, 102, 40].

2.2.1. LTI Systems in State Space Representation

A *linear time invariant* (LTI) system is a set of equations of the form:

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

Here $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$ and $D \in \mathbb{R}^{p\times m}$ are called the *system matrices* and the system is shortly referred to as $\Sigma(A, B, C, D)$. Further, A is called the *state space matrix*,

B/C are called the *input/output map* respectively and D is the *direct transmission map*. The vectors $x \in \mathbb{R}^n$, $y \in \mathbb{R}^p$ and $u \in \mathbb{R}^m$ are called the *state*, *output* and *input* (or *control*) of the system. The first equation is also referred to as *state equation* whereas the second is called the *output equation*. If $m = p = 1$ the system is called single input single output (SISO) otherwise it is called multiple input multiple output (MIMO).

The linear time invariant system (2.1) classically directly appears from the modelling of an applications process, or as a linearization of a nonlinear model. In simulations of control systems where partial differential equations are involved it also arises from the spatial semi-discretizations. The latter is one field of application for the methods presented in the remainder of this thesis.

A *linear time varying system* (LTV-system) consequently is a system where the system matrices may depend on time as well. If the system matrices are depending on the state x or the control u as well the system is said to be nonlinear. We will concentrate on the LTI case here. Moreover we have $D = 0$ in most of our applications.

Next we will introduce the important properties stability and detectability that we will need to guarantee the existence and uniqueness of the optimal control in Section 2.3. We also give the stronger properties controllability and observability and present the notion of stabilizability.

A matrix is said to be *Hurwitz-stable* if all its eigenvalues are located in the open left half of the complex plain, i.e., $\lambda \in \Lambda(A) \Rightarrow \lambda \in \mathbb{C}_{<0}$. An LTI system (2.1) is called *asymptotically stable* if A is Hurwitz stable, i.e. all solutions for $u \equiv 0$ tend to 0 asymptotically as t goes to infinity. Often a Hurwitz-stable matrix is simply referred to as *Hurwitz* or *stable*, whereas asymptotically stable systems are abbreviately called *stable*. The system (2.1) is called *stabilizable*, if there exists a matrix $F \in \mathbb{R}^{m\times n}$ such that $A - BF$ is (Hurwitz-) stable. A more generally applicable definition of stabilizability is that of demanding for an input function u such that the solution of (2.1) tends to 0 asymptotically as t tends to infinity under the application of u. In the context of Section 2.3 the existence of F is the more suitable requirement, though. We also abbreviatingly say (A, B) is stabilizable. Stabilizability is equivalent to rank $([A - \lambda I, B]) = n$ for all $\lambda \in \mathbb{C}_{>0}$. If the later condition holds for all $\lambda \in \mathbb{C}$ the system is called *controllable* , i.e., for every state x_1 we find a time $t_1 > 0$ and an admissible control u, such that for the corresponding solution trajectory we have $x_u(t_1) = x_1$. As above we also call the matrix pair (A, B) controllable. Note that controllability is the stronger concept.

The system

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

is called the *adjoint system* for (2.1). Employing the adjoint system one can easily define the notions of detectability and observability of a system. A system (2.1) (or the matrix pair (C, A)) is called *detectable* if the adjoint system (or the pair (A^T, C^T)) is stabilizable. Similarly the system (or pair (C, A)) is called *observable* if the adjoint system (or pair (A^T, C^T) is controllable. A less compressed collection of the most important properties

and test for these expressions can be found in [13], an in depth study and presentation is available in many textbooks as, e.g., [133]. Note that stabilizability is equivalently applicable in infinite dimensions as well, whereas controllability as defined here is limited to finite dimensional systems. The same obviously holds true for the dual properties detectability and observability.

A valuable tool in the analysis of (2.1) (preferably in the SISO case) is the transfer function matrix. It arises when the Laplace transformation (see, e.g., [2]) is applied to the state equation and the result is inserted into the output equation. The transfer function matrix $H(s)$ for (2.1) is

$$H(s) := D - C(A - sI)^{-1}B. \tag{2.3}$$

Note that the derivation assumes $x(t_0 = 0) = 0$, which is no restriction in the case of linear systems, but may require a transformation first.

Since the Laplace transform maps the system into frequency domain representation, the transfer function matrix relates inputs to outputs via $Y(s) = H(s)U(s)$ in the frequency domain. Here $Y(s)$ and $U(s)$ are the Laplace transformations of the outputs $y(t)$ and inputs $u(t)$ respectively. Applying the state space transformation $x \mapsto Tx$ for a non-singular transformation matrix $T \in \mathbb{R}^{n\times n}$ and computing the transfer function matrix for the transformed system $(TAT^{-1}, TB, CT^{-1}, D)$, we immediately see that it is invariant under state space transformations. All representations of the same system (that can be transformed into each other) are called *realizations* of the system. There exist also realizations of order $\tilde{n} \neq n$. Where those with $\tilde{n} > n$ are generally not of interest in contrast to those with $\tilde{n} < n$. The lower limit $\hat{n}$ for the order of the system is called the *McMillan degree* of the system and a realization of order $\hat{n}$ is called a *minimal realization*.

2.2.2. Generalized State Space Form and Descriptor Systems

In many applications the system arises in generalized form. If, e.g., one applies the finite element method for the spatial semi-discretization of a parabolic partial differential equation constraint control problem, the resulting system takes the *generalized state space form*:

$$\begin{aligned} M\dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{2.4}$$

Here $M \in \mathbb{R}^{n\times n}$ is called the *mass matrix* and is in general symmetric and positive definite, i.e., especially M is invertible. If on the other hand (2.4) appears in the process of modelling electrical circuits in chip design, M is in general not invertible. This is often indicated by writing E instead of M. The system is then a *differential algebraic equation* and also called *descriptor system*.

In Chapter 5 two ways of extending the methods presented in Chapter 4 to generalized systems are given. The case of differential algebraic equations and their derivation is

discussed, e.g., in [87] and [64]. [106] gives an introduction to handling large scale versions of (2.4) in a model order reduction context.

We will concentrate on the case of invertible mass matrices here. In that case all concepts, properties and result from the previous section can be extended to the generalized system by applying them to the equivalent standard state space system $\Sigma(M^{-1}A, M^{-1}B, C, D)$.

The transfer function $H(s)$ of (2.4) is given by

$$H(s) = D - C(A - sM)^{-1}B. \tag{2.5}$$

Note that we only have to shift with the mass matrix M instead of the identity in the inner inverse. Analogously properties can be expressed in terms of the *matrix pencil* $(A - sM)$ instead of the state space matrix $M^{-1}A$ of the equivalent standard state space form. E.g., for the eigenvalue problem it is obvious, that we can replace $(M^{-1}A - sI)x = 0$ by $(A - sM)x = 0$. This property will be exploited in Section 5.2 for efficient handling of M in the large scale contexts.

2.2.3. Second Order Systems

Whenever oscillations play an important role in modelling processes, accelerations are a non-negligible ingredient of the resulting system. This leads to an additional second order (with respect to time derivatives) term. An example is the vibration analysis for large constructions as buildings. See, e.g., the model BUILD_I in the SLICOT[1] benchmark collection [42]. This example comes from modeling vibrations of a building at *Los Angeles University Hospital*. Note that there the resulting second order differential equation is transformed into a standard LTI system. Similar models arise in chip design where resonant circuits are involved. The general representation of a *time invariant second order system* takes the form

$$\begin{aligned} M\ddot{x}(t) + G\dot{x} + Kx &= Bu(t), \\ y(t) &= C_p x(t) + C_v \dot{x}(t) + Du(t), \end{aligned} \tag{2.6}$$

where M, G, $K \in \mathbb{R}^{n\times n}$ are called *mass matrix* , *damping matrix* and *stiffness matrix*, $B \in \mathbb{R}^{n\times m}$ is the *input map* as in the first order case and C_p, $C_v \in \mathbb{R}^{p\times n}$ are the counterpart for the output map which here is split into the *proportional output map* C_p and the *velocity output map* C_v.

Under the assumption that M is invertible, we can easily transform (2.6) into a system of the form (2.4) and hence to standard state space representation (2.1). We will perform the transformation to phase space representation as an example here. Alternative approaches can be found, e.g., in [129, Chapter 3], or [139]. First define $\tilde{x}(t) := (x(t), \dot{x}(t))^T$

[1] http://www.slicot.org

then $\tilde{x} \in \mathbb{R}^{2n}$ and defining

$$\tilde{M} := \begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix}, \qquad \tilde{A} := \begin{bmatrix} 0 & I \\ -K & -G \end{bmatrix}, \qquad \tilde{B} := \begin{bmatrix} 0 \\ B \end{bmatrix}, \qquad \tilde{C} := \begin{bmatrix} C_p & C_v \end{bmatrix} \tag{2.7}$$

we obtain the generalized state space system

$$\begin{aligned} \tilde{M}\dot{\tilde{x}}(t) &= \tilde{A}\tilde{x}(t) + \tilde{B}u(t) \\ y(t) &= \tilde{C}\tilde{x}(t) + Du(t) \end{aligned} \tag{2.8}$$

Now multiplying from the left with $\tilde{M}^{-1}$ produces,

$$\hat{A} := \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}G \end{bmatrix}, \qquad \hat{B} := \begin{bmatrix} 0 \\ M^{-1}B \end{bmatrix},$$

which leads to the equivalent first order standard state space system

$$\begin{aligned} \dot{\tilde{x}}(t) &= \hat{A}\tilde{x}(t) + \hat{B}u(t) \\ y(t) &= \tilde{C}\tilde{x}(t) + Du(t). \end{aligned} \tag{2.9}$$

Obviously these system matrices should never be assembled for numerical computations in large scale contexts. On the one hand, the inversion of M will destroy the sparsity. On the other hand, it is desirable to use sparse direct solvers, which cannot exploit the structure very well even when applied to $\Sigma(\tilde{M}, \tilde{A}, \tilde{B}, \tilde{C}, D)$. In contrast to this the original matrices, which often arise in finite element contexts are (especially in 2d problems) much better suited for these solvers. Note that in cases where M, G, K are symmetric, we can preserve the symmetry in the first order representation by rewriting, e.g, in the form

$$\begin{bmatrix} -K & 0 \\ 0 & M \end{bmatrix} \dot{z}(t) = \begin{bmatrix} 0 & -K \\ -K & -G \end{bmatrix} z(t) + \begin{bmatrix} 0 \\ B \end{bmatrix} u(t), \qquad y(t) = \begin{bmatrix} C_p & C_v \end{bmatrix} z(t). \tag{2.10}$$

Here again $z(t) = (x(t), \dot{x}(t))^T$, since I can be replaced by an arbitrary non-singular matrix in (2.7). However, note that reestablishing symmetry we sacrifice definiteness of the mass matrix here.

We also provide the transfer function matrix representation in terms of the original system matrices here

$$H(s) := D + \left(C_p + sC_v\right)\left(-|s|^2 M + sG + K\right)^{-1} B. \tag{2.11}$$

The exploitation of the block structure from (2.7) in sparse model reduction algorithms is the subject of Section 7.2. Note that in many coupled structure mechanical models the mass matrix suffers a rank deficiency introduced by the rigid body modes from the underlying mechanical system. Then we obviously get a system in descriptor form, since with M also $\tilde{M}$ in (2.7) is singular. Therfore we call second order system of this form descriptor systems as well. As in the generalized first order case these systems can not be treated with the methods presented in this thesis. We have investigated an example of this kind and approaches for the reduction of such systems will soon be available in [37, 38].

2.2.4. Linear-Quadratic Optimal Control in Finite Dimensions

One of the goals of this thesis is to compute a *closed loop controller* for systems of the form (2.1). The idea in closed loop control (as opposed to open loop control) is to create a control, that processes the current measured state of the system (i.e., its output), or the complete state itself to compute the control. We will thus build a system loop that feeds the output/state back into the system as the input. In that sense the control loop is closed in contrast to *open loop control* where the control is computed entirely in advance and can not react on current unpredicted deviations of the state from the precomputed/desired trajectory. Depending on whether the state or the output is used to determine the input we distinguish the more precise terms *state feedback* and *output feedback*. Throughout this thesis we will focus on the state feedback case.

In the case of linear-quadratic optimal control the open and closed loop approaches coincide. That means, at least in theoretic considerations, they compute the same input function. On the other hand in numerical computations, as well as technical applications the feedback approach can compensate system perturbations due to round off errors, modeling errors and process disturbances, which may lead to differing system behaviour for the two approaches.

Consider the quadratic *cost functional*:

$$\mathfrak{J}(u) = \mathfrak{J}(x, u, x_0) := \int_{t_0}^{T_f} (x, Qx) + (u, Ru)\, dt, \tag{2.12}$$

with $x_0 := x(t_0)$ the initial state at initial time t_0 (we will without loss of generality consider $t_0 = 0$ here, since (2.1) is linear) and $T_f \in \mathbb{R}_{>t_0} \cup \infty$ the final time. The matrices $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{p\times p}$ are assumed to be symmetric and $Q = C^T \hat{Q} C$ for a symmetric matrix $\hat{Q} \in \mathbb{R}^{p\times p}$. Further Q is considered to be positive semi-definite and R needs to be positive definite and thus invertible. We will consider $T_f < \infty$ in the derivation of the required equations and concepts in the following section and concentrate on the asymptotic case in a separate section thereafter.

Note that (2.12) can easily be generalized to abstract settings as soon as we have an inner product available (as, e.g., in the Hilbert settings we will formulate the abstract Cauchy problems for the PDE case, that is used in Chapter 3 and 6, and will be introduced in Section 2.3). The short representation $\mathfrak{J}(u)$ is validated by the fact that we will have to assume regularity such that solution trajectory and control are uniquely dependent on each other anyway.

LQR Problems on Finite Time Horizons

In this section we will consider the finite final time case, i.e., $\mathbb{R} \ni T_f < \infty$. We will need this restriction for now since we want to formulate the LQR problem as an augmented

boundary value problem in contrast to the given initial value problem. From the boundary value problem we will then be able to derive a representation of the feedback control incorporating the solution of the matrix equations in the focus of this thesis. In Section 2.3.1 we will then see how this can help computing the feedback control for a PDE constraint optimal control problem.

We can now formulate the *linear-quadratic regulator* (LQR) problem a.k.a. linear quadratic optimal control problem as

Definition 2.1 (LQR problem):
Minimize the cost function (2.12) over all admissible controls, with respect to the state space system (2.1). ◊

The LQR problem has been extensively discussed in the open literature. Trying to give a complete list therefore is utopian. An undergraduate introduction to the existence theory can be found in [101]. An introduction with a strong focus on the needs of numerical solvers has been given in [13]. An in depth introduction, partially also relating the open and closed loop systems, can be found in textbooks like [40, 99, 5, 133].

Note that this is only the most simple case of a quadratic cost functional. Many authors also include a mixed term (x, Su) under the integral and have a penalty term for the final output as additional factor. Since the mixed term has not been discussed to any extend in the literature concerned with PDE constrained problems we will not take it into account here to keep the presentation as simple as possible. The penalty term for the final output on the other hand will not play a role on the infinite time horizon, since the state has to go to zero when time tends to ∞ anyway in order to have the integral exist. That means we can omit it for the sake of simplicity as well. A summarized derivation incorporating the final time penalty term can be found, e.g., in [107] and references therein.

Proposition 2.2 (existence of the co-state):
Let u_* be the piecewise continuous optimal control for (2.1), (2.12) and x_* the according optimal trajectory generated by (2.1). Then there exists a co-state function $\mu_* \in \mathbb{R}^n$ such that x_*, u_*, μ_* solve the *boundary value problem*

$$\begin{bmatrix} I_n & 0 & 0 \\ 0 & -I_n & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{\mu} \\ \dot{u} \end{bmatrix} = \begin{bmatrix} A & 0 & B \\ Q & A^T & 0 \\ 0 & B^T & R \end{bmatrix} \begin{bmatrix} x \\ \mu \\ u \end{bmatrix}, \tag{2.13}$$

$$x(t_0) = x_0, \qquad \mu(T_f) = 0. \tag{2.14}$$

◊

Note that if optimization is considered rather than optimal control, the co-state is normally referred to as the adjoint state, especially in PDE constrained optimization problems. The second row equation in (2.13) corresponds to the adjoint equation (2.2) arising

from the variational inequalities in the optimization approach. Note further, that equation (2.13) does not require additional regularity of u since the time derivative $\dot{u}$ of u only appears formally because the last column of the left hand side matrix consists of all zero entries. Since R is assumed to be regular we can use the last row equation to eliminate u from (2.13) yielding an ordinary boundary value problem. This is also the key feature in the derivation of the matrix equations of interest, as we will see after stating the optimality result for the above solution triple.

Proposition 2.3 (optimality of the solution):
Let x_*, u_*, μ_* solve (2.13), (2.14) and Q, R have the form stated above. Then

$$\mathfrak{J}(x_*, u_*, x_0) \leq \mathfrak{J}(x, u, x_0),$$

for every triple (x, u, x_0) solving (2.1). ◊

Propositions 2.2 and 2.3 are often formulated together as, e.g., in [40], where the proof is available as well.

The explicit representation of u from (2.13) is

$$u(t) = R^{-1}B^T\mu(t).$$

Inserting this into the state equation we get

$$\dot{x}(t) = Ax(t) + BR^{-1}B^T\mu(t),$$

in turn of which we can rewrite (2.13) as

$$\begin{bmatrix}\dot{x}(t)\\ \dot{\mu}(t)\end{bmatrix} = \begin{bmatrix}A & BR^{-1}B^T\\ Q & -A^T\end{bmatrix}\begin{bmatrix}x(t)\\ \mu(t)\end{bmatrix}, \qquad \begin{pmatrix}x(t_0) = x_0,\\ \mu(T_f) = 0.\end{pmatrix} \tag{2.15}$$

Now making the ansatz $\mu(t) := -X(t)x(t)$, the terminal condition for the co-state yields $\mu(T_f) = X(T_f)x(T_f)$ and thus $X(T_f) = 0$ since $x(T_f)$ is not specified a priory. Inserting $\mu(t)$ and $\dot{\mu}(t) = -\dot{X}(t)x(t) - X(t)\dot{x}(t)$ in (2.15) we obtain

$$\dot{x}(t) = Ax(t) - BR^{-1}B^TX(t)x(t), \tag{2.16}$$

$$\left(\dot{X}(t) + X(t)A + A^TX(t) - X(t)BR^{-1}B^TX(t) + Q\right)x(t) = 0. \tag{2.17}$$

Again by variation of $x(t)$ we end up with the *differential Riccati equation* (DRE)

$$-\dot{X}(t) = X(t)A + A^TX(t) - X(t)BR^{-1}B^TX(t) + Q. \tag{2.18}$$

This autonomous nonlinear matrix-valued differential equation yields an initial value problem for $X(t)$ in reverse time together with the terminal condition $X(T_f) = 0$.

Remark 2.4:

- It can be shown that the solution X_* of (2.18) is unique under the given assumptions [1, Thm. 4.1.6].

- From transposition of (2.18) we immediately see, that the solution X is symmetric.
- In cases where the penalty term for the final output/state in the cost functional is nonzero, that will also specify the terminal condition for X.
- Exploiting the uniqueness of X_* we can proof that the solution in Proposition 2.2 is also unique. ◊

Summarizing the above we obtain the following

Theorem 2.5 (existence and uniqueness of the optimal feedback control):
If $Q \geq 0$, $R > 0$ are symmetric and $T_f < \infty$, then there exists a unique solution of the LQR problem (2.1), (2.12). The optimal control is given in feedback form by

$$u_*(t) = -R^{-1}B^T X_*(t)x(t).$$

Here $X_*(t)$ is the unique symmetric solution satisfying the DRE

$$-\dot{X}(t) = X(t)A + A^T X(t) - X(t)BR^{-1}B^T X(t) + Q,$$

and the terminal condition $X(T_f) = 0$. Moreover, for any initial value x_0 of (2.1) the optimal cost is given by

$$\mathfrak{J}(u_*) = \frac{1}{2}x_0^T X_*(t_0)x_0.$$ ◊

We have thus achieved the goal of creating a closed loop control on the finite time horizon. The feedback map $K_*(t) := -R^{-1}B^T X_*(t)$ is also called the *optimal gain matrix*. The next task will be to lift this result to the infinite time horizon.

LQR Problems on the Infinite Time Horizon

In the previous section we showed how the optimal gain matrix and the optimal control employing it for the LQR problem can be computed for finite final time. We will now extend the results achieved there to the infinite final time case. Doing so we will see that this is in fact the more easy case, since things simplify drastically from the viewpoint of numerical computations for the cost of some additional work in the theoretical part.

Now let $T_f = \infty$. We consider $Q \geq 0$ and $R > 0$ as above. Note that the cost functional (2.12) turns into an improper integral and thus we can only expect it to exist if the two inner products converge to zero as time approaches infinity. Since $R > 0$ that means we need

$$\begin{aligned} \lim_{t\to\infty} u(t) &= 0, \\ \lim_{t\to\infty} (x(t), Qx(t)) &= 0. \end{aligned}$$

By the terminal condition on μ we have $\lim_{t\to\infty} \mu(t) = 0$ and if we would have $X(t) \equiv X$ independent of time t, then we would immediately have $\lim_{t\to\infty} x(t) = 0$ and thus

$\lim_{t\to\infty} u(t) = 0$. We can easily comprehend that $X(t) \equiv X$ must hold. Due to the uniqueness of the solution of (2.18) the solutions on two time intervals $[t_0, t_1]$ and $[t_0, t_2]$ must emerge from each other by scaling the time variable. That means $X_2(t) = X_1(ct)$ (with the index relating the solutions to the intervals) for $c = \frac{t_1}{t_2}$. Obviously we then have $\dot{X}_2(t) = c\dot{X}_1(ct)$ by the chain rule. Now taking the limit for $t_2 \to \infty$ we realize

$$\lim_{t_2\to\infty} \dot{X}_2(\tilde{t}) = \lim_{t_2\to\infty} \frac{t_1}{t_2}\dot{X}_1(t) = 0$$

independent of $\tilde{t}$ and t. Thus $X(t)$ is constant and taking the limit in (2.18) we obtain the *algebraic Riccati equation* (ARE)

$$0 = \Re(X_\infty) = Q + X_\infty A + A^T X_\infty - X_\infty B R^{-1} B^T X_\infty. \tag{2.19}$$

Thus for the infinite final time case we have found an algebraic equation doing the job of the differential equation in the finite time context. Since many solvers for the differential equation require solving the algebraic equation in every time-step (see [107] and references therein), this simplifies numerical considerations enormously. Unfortunately, in contrast to the DRE the ARE does *not* have a unique solution. Even though the solution should be symmetric as it is the limit of symmetric solutions, yet this does not unify it. We need further inspection to derive conditions under which we can consider it unique. The following theorem [89, 105, 83] shows, that we can guarantee the uniqueness under certain, not too strong conditions on the underlying state space system. A detailed discussion of algebraic Riccati equations can be found in many books and monographs as [89, 105, 83, 154] to mention only a few important ones.

Theorem 2.6 (Uniqueness of the ARE solution):
If $F \geq 0$, $G \geq 0$, (A, G) stabilizable and (F, A) detectable, then the ARE

$$F + AX + XA^T - XGX = 0,$$

has a unique, symmetric, stabilizing solution X_*, i.e., $\Lambda(A - GX_*) \subset \mathbb{C}_{<0}$. ◊

For the proof see, e.g., [89] or many references given therein. Note that we assume rather strict properties in the above theorem. [89] especially discusses which prerequisites can be weakened to still obtain uniqueness of the solution. If in addition we demand for (F, A) to be observable this will guarantee the positive definiteness of the solution.

For the LQR problem considered here we can now formulate a corollary as a direct consequence of Theorem 2.6.

Corollary 2.7:
If $\hat{Q} \geq 0$, $R > 0$, (A, B) is stabilizable and $(C^T\hat{Q}C, A)$ is detectable, then the LQR problem (2.1), (2.12) with $T_f = \infty$ has a unique solution given in feedback form

$$u_*(t) = -R^{-1}B^T X_* x(t),$$

where X_* is the unique stabilizing solution of the ARE (2.19) with $Q = C^T\hat{Q}C$. ◊

2.3. Linear-Quadratic Optimal Control of Parabolic Partial Differential Equations

We consider nonlinear parabolic convection-diffusion and diffusion-reaction systems of the form

$$\frac{\partial \mathbf{x}}{\partial t} + \nabla \cdot (\mathbf{c}(\mathbf{x}) - \mathbf{k}(\nabla \mathbf{x})) + \mathbf{q}(\mathbf{x}) = \mathbf{B}(\xi)\mathbf{u}(t), \qquad t \in [0, T_f], \tag{2.20}$$

in $\Omega \in \mathbb{R}^d, d = 1, 2, 3$, with appropriate initial and boundary conditions. Here, $\mathbf{c}$ is the convective part, $\mathbf{k}$ the diffusive part and $\mathbf{q}$ is an uncontrolled source term. The state of the system depends on $\xi \in \Omega$ and the time $t \in [0, T_f]$ and is denoted by $\mathbf{x}(\xi, t)$. The control is called $\mathbf{u}(t)$ and is assumed to depend only on the time $t \in [0, T_f]$.

A general control problem for the above PDE is defined as

Definition 2.8 (PDE constraint optimal control system):

$$\min_{\mathbf{u}} \mathfrak{J}(\mathbf{x}, \mathbf{u}, \mathbf{x}_0) \text{ subject to (2.20)}, \tag{2.21}$$

where $\mathfrak{J}(\mathbf{x}, \mathbf{u})$ is a performance index which will be specified later. ◊

We will see in the following, that in cases where (2.20) is linear and $\mathfrak{J}$ is quadratic (compare (2.25), (2.30)) this is exactly the extension of Definition 2.1 to the PDE case.

There are two possibilities for the appearance of the control. If the control occurs in the boundary condition, we call this problem a *boundary control problem*. It is called *distributed control problem* if the control acts in Ω or a sub-domain $\Omega_u \subset \Omega$. The control problem as in (2.20) is well-suited to describe a distributed control problem while boundary control will require the specification of the boundary conditions as, for instance, given below.

The major part of this thesis deals with the linear version of (2.20),

$$\frac{\partial \mathbf{x}}{\partial t} - \nabla . (a(\xi)\nabla \mathbf{x}) + d(\xi)\nabla \mathbf{x} + r(\xi)\mathbf{x} = \mathbf{B}_V(\xi)u(t), \quad \xi \in \Omega, \ t > 0, \tag{2.22}$$

with initial and boundary conditions

$$\begin{aligned} \alpha(\xi)\frac{\partial \mathbf{x}(\xi, t)}{\partial n} + \gamma(\xi)\mathbf{x}(\xi, t) &= \mathbf{B}_R u(t), \qquad \xi \in \partial\Omega, \\ \mathbf{x}(\xi, 0) &= \mathbf{x}_0(\xi), \quad \xi \in \Omega, \end{aligned}$$

for sufficiently smooth parameters $a, d, r, \alpha, \gamma, \mathbf{x}_0$. We assume that either $\mathbf{B}_V = 0$ (boundary control system) or $\mathbf{B}_R = 0$ (distributed control system). In addition, we include in our problem an output equation of the form

$$\mathbf{y} = \mathbf{C}\mathbf{x}, \qquad t \geq 0,$$

taking into account that in practice, often not the whole state $\mathbf{x}$ is available for measurements. Here, $\mathbf{C}$ is a linear operator which often is a restriction operator.

To solve optimal control problems (2.21) with a linear system (2.22) we interpret it as a linear quadratic regulator (LQR) problem. The theory behind the LQR ansatz has already been studied in detail, e.g., in [90, 91, 92, 98, 34, 9], to name only a few.

Nonlinear control problems are still undergoing extensive research. We will apply model predictive control (MPC) here, i.e., we solve linearized problems on small time frames. This idea is similar to the one presented by Ito and Kunisch in [79]. We will briefly sketch the main ideas of this approach and the differences to the idea in [79] in Section 6.3. An in depth analysis of the Ito/Kunisch approach can be found in the PhD thesis by Sabine Hein [68].

There exists a rich variety of other approaches to solve linear and nonlinear optimal control problems for partial differential equations. We can only refer to a selection of ideas, see e.g. [142, 33, 71, 98, 73, 74].

In the remainder of this section we will formulate the LQR problem. We assume that $\mathcal{X}, \mathcal{Y}, \mathcal{U}$ are separable Hilbert spaces where $\mathcal{X}$ is called the state space, $\mathcal{Y}$ the observation space and $\mathcal{U}$ the control space.

Furthermore the linear operators

$$\begin{aligned} \mathbf{A}: &\quad \operatorname{dom}(\mathbf{A}) \subset \mathcal{X} \to \mathcal{X}, \\ \mathbf{B}: &\quad \mathcal{U} \to \mathcal{X}, \\ \mathbf{C}: &\quad \mathcal{X} \to \mathcal{Y} \end{aligned}$$

are given. Such an abstract system can now be understood as a Cauchy problem for a linear evolution equation of the form

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \qquad \mathbf{x}(.,0) = \mathbf{x}_0 \in \mathcal{X}. \tag{2.23}$$

Since in many applications the state $\mathbf{x}$ of a system can not be observed completely we consider the observation equation

$$\mathbf{y} = \mathbf{C}\mathbf{x}, \tag{2.24}$$

which describes the map between the states and the outputs of the system.

The abstract LQR problem is now given as the minimization problem

$$\min_{\mathbf{u} \in \mathcal{L}^2(0,T_f;\mathcal{U})} \frac{1}{2} \int_0^{T_f} \langle \mathbf{y}, \mathbf{Q}\mathbf{y} \rangle_{\mathcal{Y}} + \langle \mathbf{u}, \mathbf{R}\mathbf{u} \rangle_{\mathcal{U}} \, dt \tag{2.25}$$

with self-adjoint, positive definite, linear, bounded operators $\mathbf{Q}$ and $\mathbf{R}$ on $\mathcal{Y}$ and $\mathcal{U}$, respectively. Recall that if (2.23) is an ordinary differential equation with $\mathcal{X} = \mathbb{R}^n$, $\mathcal{Y} = \mathbb{R}^p$ and $\mathcal{U} = \mathbb{R}^m$, equipped with the standard scalar product, then we obtain an

LQR problem for a *finite-dimensional system* (see Section 2.2.4). For partial differential equations we have to choose the function spaces $\mathcal{X}, \mathcal{Y}, \mathcal{U}$ appropriately and we get an LQR system for an *infinite-dimensional system* [44, 45].

Consider the heat equation

$$\begin{aligned} \partial_t \mathbf{x} - \Delta \mathbf{x} &= \mathbf{f}(\xi, t) && \text{on } \Omega := [0,1]^2, \\ \mathbf{x} &= 0 && \text{for } \xi_1 \in \{0,1\}, \text{ or } \xi_2 = 1, \\ \mathbf{x}(\xi, t) &= \mathbf{v}(\xi, t) && \text{for } \xi_2 = 0, \end{aligned}$$

where as in (2.20) and (2.22)

$$\mathbf{v}(\xi, t) := \mathbf{B}(\xi) u(t) := \left(1 + \frac{1}{2} \sin\left(-\frac{\pi}{2} + \xi_1 2\pi\right)\right) \cdot u(t).$$

Then at every instant of time the corresponding elliptic equation is known to have a solution in $\mathcal{H}^2(\Omega)$ and thus $\mathcal{X} = \mathcal{H}^2(\Omega)$. We have a scalar input u such that $\mathcal{U} = \mathbb{R}$. Note that then $\mathbf{v}(\xi, t) \in \mathcal{L}^2([0,\infty), \mathbb{R})$ and $\mathbf{B}$ is obviously bounded. If we further consider only the temperature at $\xi = (\frac{1}{2}, \frac{1}{2})^T$ as an output, then also $\mathcal{Y} = \mathbb{R}$, which completes our Hilbert space setting.

Many optimal control problems for instationary linear partial differential equations can be described using the abstract LQR problem above. Additionally, many control, stabilization and parameter identification problems can be reduced to the LQR problem, see [10, 45, 90, 91, 92].

Semigroups and Mild Solutions

The concept of solutions to finite dimensional systems we are following in the context of LQR problems is that of matrix exponential based representations. In the operator case found for PDE control problems we will follow the principle of mild solutions and operator semigroups, that is closely related to the above concept. It is the direct extension to the operator setting in Banach and Hilbert spaces $\mathcal{X}$. Clearly for a bounded operator $\mathbf{A} \in \mathcal{L}(\mathcal{X})$ and $t \geq 0$ we can define the operator valued exponential

$$\mathbf{T}(t) := e^{t\mathbf{A}} := \sum_{k=0}^{\infty} \frac{t^k \mathbf{A}^k}{k!}.$$

Then as in the matrix exponential and scalar cases $\mathbf{T}$ for all $t, s \geq 0$ has the properties

$$\begin{aligned} \mathbf{T}(t+s) &= \mathbf{T}(t)\mathbf{T}(s), \\ \mathbf{T}(0) &= \mathbf{I}, \end{aligned} \tag{2.26}$$

and

$$\frac{d\mathbf{T}(t)}{dt} = \mathbf{A}\,\mathbf{T}(t). \tag{2.27}$$

These properties motivate the name *operator semigroup*. If the above properties even hold for all $t, s \in \mathbb{R}$ one also speaks of *operator groups*.

$\mathbf{T}$ as defined above is also called *uniform* or *analytic* semigroup. Unfortunately the concept of uniform semigroups is much to strong to be applicable in many cases. Therefore the weaker concept of *strongly continuous semigroups* or $\mathcal{C}^0$-semigroups is introduced, which only demands for (2.26) to be fulfilled. The symbol $e^{t\mathbf{A}}$ is often kept, even though only fully valid in the analytic case, reflecting the close relationship to the matrix exponential.

The operator $\mathbf{A}$ is called the *infinitesimal generator* of the semigroup. In the case of analytic semigroups the defining quality of the infinitesimal generator is obvious. For strongly continuous semigroups it can be proven [47, Section 1 Theorem 1.4] to be a closed, densely defined operator determining the semigroup uniquely.

The *mild solution* to the closed loop system

$$\begin{aligned} \dot{\mathbf{x}}(t) &= (\mathbf{A} - \mathbf{BK})\mathbf{x}(t) + \mathbf{f}(t) \\ \mathbf{x}(0) &= \mathbf{x}_0 \end{aligned} \tag{2.28}$$

is then given as

$$\mathbf{x}(t) := \mathbf{T}(t)\mathbf{x}_0 + \int_0^t \mathbf{T}(t-s)\mathbf{f}(s)\, ds \quad \text{for} \quad t \geq 0, \tag{2.29}$$

where the semigroup $\mathbf{T}(t)$ is generated by the closed loop operator $\mathbf{A}-\mathbf{BK}$ for the optimal feedback operator $\mathbf{K}$. A more detail introduction to the concept of mild solutions can be found in the PhD thesis by Sabine Hein [68, Section 8.1], or textbooks as [45, 47, 111]. All important properties of operator semigroups needed in the context of LQR systems for parabolic PDEs have been summarized by Hermann Mena [107, Section 3.2]. An in depth discussion of one parameter semigroups is available in [47, 111]. Note that the concept of one parameter semigroups is only valid for linear time invariant systems. In the context of linear time varying systems the more general concept of two parameter semigroups needs to be applied to reflect the dependence of $\mathbf{A}$ on time in that case. If a solution to (2.28) is continuously differentiable it is also called *classical solution*. Tanabe [138] reflects the close relationship of the representation to the finite dimensional systems case by calling (2.29) the *fundamental solution*.

The Infinite Time Case

In the infinite time case we assume that $T_f = \infty$. Then the minimization problem subject to (2.23) is given by

$$\min_{\mathbf{u} \in \mathcal{L}^2(0,\infty;\mathcal{U})} \frac{1}{2} \int_0^\infty \langle \mathbf{y}, \mathbf{Qy} \rangle_{\mathcal{Y}} + \langle \mathbf{u}, \mathbf{Ru} \rangle_{\mathcal{U}}\, dt. \tag{2.30}$$

If the standard assumptions that

- $\mathbf{A}$ is the infinitesimal generator of a $\mathcal{C}^0$-semigroup $\mathbf{T}(t)$,
- $\mathbf{B}, \mathbf{C}$ are linear bounded operators and
- for every initial value there exists an admissible control $\mathbf{u} \in \mathcal{L}^2(0, \infty; \mathcal{U})$

hold, then the solution of the abstract LQR problem can be obtained analogously to the finite-dimensional case (see, e.g., [154, 44, 45, 54, 33, 34, 32, 98, 120]) discussed earlier in this chapter. We then have to consider the algebraic operator Riccati equation

$$0 = \Re(\mathbf{X}) = \mathbf{C}^*\mathbf{Q}\mathbf{C} + \mathbf{A}^*\mathbf{X} + \mathbf{X}\mathbf{A} - \mathbf{X}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^*\mathbf{X}, \tag{2.31}$$

where the linear operator $\mathbf{X}$ will be the solution of (2.31) if $\mathbf{X} : \operatorname{dom}\mathbf{A} \to \operatorname{dom}\mathbf{A}^*$ and $\langle \hat{\mathbf{x}}, \Re(\mathbf{X})\mathbf{x} \rangle = 0$ for all $\mathbf{x}, \hat{\mathbf{x}} \in \operatorname{dom}(\mathbf{A})$. The optimal control is then given as the *feedback control*

$$\mathbf{u}_*(t) = -\mathbf{R}^{-1}\mathbf{B}^*\mathbf{X}_\infty\mathbf{x}_*(t), \tag{2.32}$$

which has the form of a regulator or closed-loop control. Here, $\mathbf{X}_\infty$ is the unique stabilizing nonnegative self-adjoint solution of (2.31), $\mathbf{x}_*(t) = \mathbf{S}(t)\mathbf{x}_0(t)$, and $\mathbf{S}(t)$ is the $\mathcal{C}^0$-semigroup generated by $\mathbf{A} - \mathbf{B}\mathbf{R}^{-1}\mathbf{B}^*\mathbf{X}_\infty$. Using further standard assumptions it can be shown, see e.g. [34], that $\mathbf{X}_\infty$ is the unique nonnegative stabilizing solution of (2.31). Most of the required conditions, particularly the restrictive assumption that $\mathbf{B}$ is bounded, can be weakened [90, 91].

The Finite Time Case

The finite time case arises if $T_f < \infty$ in (2.25). Then the numerical solution is more complicated since we have to solve the operator differential Riccati equation (analogously to Theorem 2.5)

$$\dot{\mathbf{X}}(t) = -(\mathbf{C}^*\mathbf{Q}\mathbf{C} + \mathbf{A}^*\mathbf{X}(t) + \mathbf{X}(t)\mathbf{A} - \mathbf{X}(t)\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^*\mathbf{X}(t)). \tag{2.33}$$

The optimal control is obtained as

$$\mathbf{u}_*(t) = -\mathbf{R}^{-1}\mathbf{B}^*\mathbf{X}_*(t)\mathbf{x}_*(t),$$

where $\mathbf{X}_*(t)$ is the unique solution of (2.33) in complete analogy to the infinite time case in (2.32). The special challenges in this case have been discussed in [107].

2.3.1. Approximation Theory

The theoretical fundament for our approach was set by Gibson [54]. The ideas and proofs used for the boundary control problem considered here closely follow the extension of Gibson's method proposed by Banks and Kunisch [12] for distributed control systems

arising from parabolic equations. Similar approaches can be found in [27, 91]. Common to all those approaches is to formulate the control system for a parabolic system as an abstract Cauchy problem in an appropriate Hilbert space setting. For numerical approaches this Hilbert space $\mathcal{X}$ is approximated by a sequence of finite-dimensional spaces $(\mathcal{X}^N)_{N\in\mathbb{N}}$, e.g., by spatial finite element approximations, leading to large sparse systems of ordinary differential equations in $\mathbb{R}^n$. Following the theory in [12] those approximations do not even have to be subspaces of the Hilbert space of solutions.

Before stating the main theoretical result we will first collect some approximation prerequisites we will need for the theorem. We call them (BK1) and (BK2) for they were already formulated in [12] (and called H1 and H2 there). In the following P^N is the canonical projection operator mapping from the infinite-dimensional space $\mathcal{X}$ to its finite-dimensional approximation $\mathcal{X}^N$. The first and natural prerequisite is:

For each N and $x_0 \in \mathcal{X}^N$ there exists an admissible control $u^N \in \mathcal{L}^2(0,\infty;\mathcal{U})$ and any admissible control drives the states to 0 asymptotically. (BK1)

Additionally one needs the following properties for the approximation as $N \to \infty$. Assume that for each N, A^N is the infinitesimal generator of a $\mathcal{C}^0$-semigroup $T^N(t)$, then we require:

(i) For all $\varphi \in \mathcal{X}$ we have uniform convergence $T^N(t)P^N\varphi \to \mathbf{T}(t)\varphi$ on any bounded subinterval of $[0,\infty)$.

(ii) For all $\phi \in \mathcal{X}$ we have uniform convergence $T^N(t)^*P^N\phi \to \mathbf{T}(t)^*\phi$ on any bounded subinterval of $[0,\infty)$.

(iii) For all $v \in \mathcal{U}$ we have $B^N v \to \mathbf{B}v$ and for all $\varphi \in \mathcal{X}$ we have $B^{N^*}\varphi \to \mathbf{B}^*\varphi$.

(iv) For all $\varphi \in \mathcal{X}$ we have $Q^N P^N \varphi \to \mathbf{Q}\varphi$.

(BK2)

With these we can now formulate the main result.

Theorem 2.9 (Convergence of the finite-dimensional approximations):
Let *(BK1)* and *(BK2)* hold. Moreover, assume $\mathbf{R} > 0$, $\mathbf{Q} \geq 0$ and $Q^N \geq 0$. Further, let P^N be the solutions of the AREs for the finite-dimensional systems and let the minimal nonnegative self-adjoint solution $\mathbf{X}$ of *(2.31)* for (2.23), (2.24) and (2.30) exist. Moreover, let $\mathbf{S}(t)$ and $S^N(t)$ be the operator semigroups generated by $\mathbf{A} - \mathbf{B}\mathbf{R}^{-1}\mathbf{B}^*\mathbf{X}$ on $\mathcal{X}$ and $A^N - B^N\mathbf{R}^{-1}B^{N^*}P^N$ on $\mathcal{X}^N$, respectively, with $\|\mathbf{S}(t)\varphi\| \to 0$ as $t \to \infty$ for all $\varphi \in \mathcal{X}$.

If there exist positive constants M_1, M_2 and ω independent of N and t, such that

$$\begin{aligned} \|S^N(t)\|_{\mathcal{X}^N} &\leq M_1 e^{-\omega t}, \\ \|X^N\|_{\mathcal{X}^N} &\leq M_2, \end{aligned} \tag{2.34}$$

then

$$\begin{aligned} X^N P^N \varphi &\to \mathbf{X}\varphi \quad \text{for all } \varphi \in \mathcal{X}, \\ S^N(t) P^N \varphi &\to \mathbf{S}(t)\varphi \quad \text{for all } \varphi \in \mathcal{X}, \end{aligned} \tag{2.35}$$

converge uniformly in t on bounded subintervals of $[0, \infty)$ as $N \to \infty$ and

$$\|\mathbf{S}(t)\| \leq M_1 e^{-\omega t} \text{ for } t \geq 0. \tag{2.36}$$

◊

Theorem 2.9 gives the theoretical justification for the numerical method used for the linear problems described in this paper. It shows that the finite-dimensional closed-loop system obtained from optimizing the semidiscretized control problem indeed converges to the infinite-dimensional closed-loop system. Equivalent results for the finite time horizon case have been proven in [86, 107]. Deriving a similar result for the nonlinear case is an open problem.

The proof of Theorem 2.9 is given in [27]. It very closely follows that of [12, Theorem 2.2]. The only difference is the definition of the sesquilinear form on which the mechanism of the proof is based. It has an additional term in the boundary control case discussed here, but one can check that this term does not destroy the required properties of the sesquilinear form. (See Appendix A as well)

2.4. Balanced Truncation Model Order Reduction

We are considering balancing based model order reduction [109] throughout this thesis. There the main task is to solve the controllability and observability Lyapunov equations

$$AP + PA^T = -BB^T, \qquad A^T Q + QA = -C^T C. \tag{2.37}$$

From their solutions we compute projection matrices T_l and T_r such that the ROM

$$\dot{\hat{x}}(t) = \hat{A}\hat{x}(t) + \hat{B}u(t), \qquad \hat{y}(t) = \hat{C}\hat{x}(t), \tag{2.38}$$

is derived as

$$\hat{A} := T_l A T_r, \qquad \hat{B} := T_l B \quad \text{and} \quad \hat{C} := C T_r. \tag{2.39}$$

As A is assumed to be stable and thus P and Q are positive semi-definite, there exist Cholesky factorizations $P = S^T S$ and $Q = R^T R$. In the so-called *square-root* balanced truncation (SRBT) algorithms [141, 93] these are used to define the above projection matrices

$$T_l := \Sigma_1^{\frac{1}{2}} V_1^T R \quad \text{and} \quad T_r := S^T U_1 \Sigma_1^{\frac{1}{2}} \tag{2.40}$$

determining the reduced order model. Here $\Sigma_1^{\frac{1}{2}}$, U_1 and V_1 are determined via the singular value decomposition

$$SR^T = U\Sigma V^T = \begin{bmatrix} U_1 U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}, \tag{2.41}$$

where $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$ is assumed to be ordered such that $\sigma_j \geq \sigma_{j+1} \geq 0$ for all j and $\Sigma_1 = \operatorname{diag}(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r \times r}$. If $\sigma_r > \sigma_{r+1} = 0$ then r is the McMillan degree of the system and the resulting ROM is a minimal realization.

For the error in the transfer function, the global bound

$$\|H - \hat{H}\|_{\mathcal{H}_\infty} \leq 2 \sum_{j=r+1}^{n} \sigma_j \tag{2.42}$$

can be proven [55]. The ∞-norm here is the operator norm

$$\|H\|_{\mathcal{H}_\infty} := \sup_{\omega \in \mathbb{R}} \sigma_{max}(G(i\omega)), \tag{2.43}$$

induced by the 2-norm in the frequency domain (see also [55] for details), and σ_{max} denotes the largest singular value.

For large scale sparse systems it is infeasible to compute either P, Q, or their Cholesky factors, since they are generally full matrices requiring $O(n^2)$ memory for storage. In Chapter 4 we repeat and extend the low-rank ADI framework which exploits that both P and Q usually have very low (numerical) rank compared to n. Therefore the Cholesky factors are replaced by low-rank Cholesky factors (LRCFs) in the above defining the low-rank square root balanced truncation method (LR-SRBT) [118, 119]. Section 7.1 gives more details on this method and its modification for generalized state space systems. The low-rank factors can be computed directly by the low-rank Cholesky factor ADI iteration (LRCF-ADI) (See Section 4.1 and [114, 116, 117, 118, 97, 18]).

Example is the school of mankind, and they will learn at no other.

Letters on a Regicide Peace
Edmund Burke

CHAPTER

THREE

MODEL PROBLEMS AND TEST EXAMPLES

Contents

Next we introduce the different model problems and test examples used in the upcoming chapters. Some of them are only briefly sketched. Others will be presented in more detail. The main focus lies on the detailed introduction of the model of an optimal control problem arising in a rolling mill during production of steel profiles. There, the main goal is to influence the material properties by controlling the temperature distribution on single cross-sections of the profile. Thus the process to be controlled is described by a heat equation. The manufacturing process only allows for very few inputs to the system and can take only a handful of measurements into account. These features make it a

perfect model for the low-rank matrix equation solvers we are discussing throughout this thesis.

The subsections are organized such that the problem structure is increasingly more complicated. We start with two models in standard state space form. Afterwards we present some systems in generalized state space representation and we end the chapter with three second order system models used in Chapter 7.

3.1. An Academic Model Example: FDM Semi-Discretized Heat Equation

The finite differences semi-discretized heat equation on the unit square $(0,1)\times(0,1)$ will serve as the most basic test example here. The advantages of this model are obvious:

- it is fairly easy to understand,
- the discretization using the finite difference method (FDM) is easy to implement
- it allows for simple generation of test problems of almost arbitrary size.

Another important feature of the FDM is that, in contrast to the finite element method (FEM) considered in the upcoming sections of this chapter, it does not generate a mass matrix in front of the time derivative, i.e., it naturally leads to a large scale sparse system in standard state space representation.

We essentially consider two variants of this problem. The sole heat equation (diffusion only) [117, demo_r1.m]

$$\frac{\partial \mathbf{x}}{\partial t} - \Delta \mathbf{x} = \mathbf{f}(\xi)\mathbf{u}(t) \tag{3.1}$$

and a model of the heat equation with convection and reaction (see equation (8.1) and [42, Section 2.7])

$$\frac{\partial \mathbf{x}}{\partial t} - \mathbf{v}.\nabla \mathbf{x} - \Delta \mathbf{x} - \mathbf{q}\mathbf{x} = \mathbf{f}(\xi)\mathbf{u}(t). \tag{3.2}$$

The models are generated using centered finite differences, leading to the usual multi-diagonal structure (see Figures 8.1a and 8.2a) for appropriate numbering of the unknowns.

Simoncini [131] proposes a 3-d variant with convection for which two discretization levels of sizes (5832,10648) are available from the author.[1]

A third version of this problem that is used in the LyaPack and M.E.S.S. demonstration scripts will also frequently be used in Chapter 8. There the $\mathbf{q}\mathbf{x}$ term in (3.2) is dropped such that it reduces to a convection-diffusion equation. The vector $\mathbf{v}$ in (3.2) is then chosen as $\mathbf{v} = [10\ \ 100]^T$.

[1] The author wishes to thank V. Simoncini for providing her example matrices.

3.2. An Artificial Test Case with Prescribed Spectrum

This is an artificial model that has been introduced in [117]. It prescribes a certain spectrum such that the Bode plot shows "spires". The system matrix A is constructed as block diagonal from four 2×2 blocks and a diagonal with entries -1 to -400 such that the matrix has dimension 408. The four leading block are

$$A_1 = \begin{bmatrix} -0.01 & -200 \\ 200 & 0.001 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} -0.2 & -300 \\ 300 & -0.1 \end{bmatrix}, \tag{3.3}$$

$$A_3 = \begin{bmatrix} -0.02 & -500 \\ 500 & 0 \end{bmatrix}, \qquad A_4 = \begin{bmatrix} -0.01 & -520 \\ 520 & -0.01 \end{bmatrix}, \tag{3.4}$$

such that

$$A = \begin{bmatrix} A_1 & & & & & & \\ & A_2 & & & & & \\ & & A_3 & & & & \\ & & & A_4 & & & \\ & & & & -1 & & \\ & & & & & \ddots & \\ & & & & & & -400 \end{bmatrix},$$

and the eigenvalues are $-0.0045 \pm \imath 200$, $-0.15 \pm \imath 300$, $-0.01 \pm \imath 500$, $-0.01 \pm \imath 520$ and $-1, \ldots, -400$. The matrix B is chosen as the all ones vector in $\mathbb{R}^{408\times 1}$ and $C = B^T$.

3.3. Selective Cooling of Steel Profiles: Cooling a Rail in a Rolling Mill

In contrast to the very flexible but also very academic test examples of the previous sections, we will now consider an application of the proposed method in modern industrial tasks. The problem of optimal cooling of steel profiles in a rolling mill will serve as an example here. This section is in substance a reprint of the modelling section in [27] and has also been discussed and tested in [127, 26, 17]. The test matrices have also been published in the Oberwolfach Model Reduction Benchmark Collection[2] [28]. They are available for download[3] in four problem sizes (1357, 5177, 20209, 79841), reflecting different levels of global FEM refinement. The model was discretized using the ALBERTA finite element toolbox. More details are available in Section A.3.

[2] http://www.imtek.de/simulation/benchmark/
[3] http://www.imtek.de/simulation/benchmark/wb/38881/

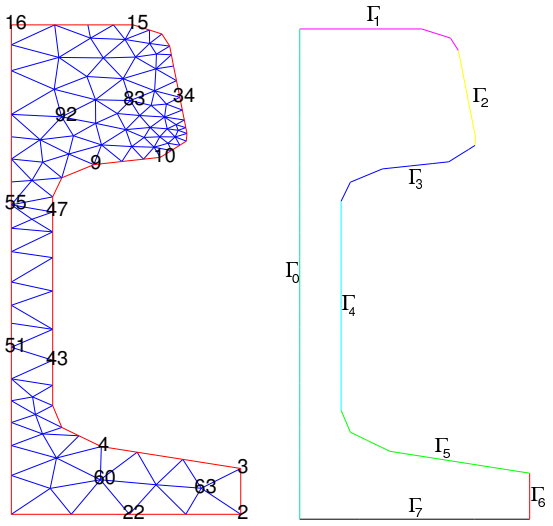


Figure 3.1.: Domain Ω for Selective Cooling of Steel Profiles: initial mesh with points of minimization and comparison (left) and partition of the boundary (right)

3.3.1. Model Background

This problem arises in a rolling mill when different steps in the production process require different temperatures of the raw material. To achieve a high production rate, economical interests suggest to reduce the temperature as fast as possible to the required level before entering the next production phase.

At the same time, the cooling process, which is executed by spraying cooling fluids onto the surface, has to be controlled so that material properties, such as durability or porosity, achieve given quality standards. Large gradients in the temperature distributions within the steel profile may lead to unwanted deformations, brittleness, loss of rigidity, and other undesirable material properties. It is therefore the engineer's goal to have a preferably even temperature distribution.

3.3.2. Model Equation

As in [143, 48, 85, 49] the steel profile is assumed to stretch infinitely into the z-direction which is justified by comparing the actual length of steel profiles like rails ($\mathcal{O}(10m)$) to their width and height ($\mathcal{O}(10cm)$). This admits the assumption of a stationary heat distribution in z-direction, or in other words, we can restrict ourselves to a 2-dimensional heat diffusion process. Therefore, we can consider the 2-dimensional cross-sections of the profile $\Omega \subset \mathbb{R}^2$ shown in Figure 3.1 as computational domain. Measurements for defining the geometry of the cross-section are taken from [143]. As one can see (e.g. Figure 3.1) the domain exploits the symmetry of the profile introducing an artificial

boundary Γ_0 on the symmetry axis. The state equation introduced in [143, 48, 85] for the temperature $x(\xi, t)$ at time t in point ξ can be summarized as follows:

$$\begin{aligned} c(\mathbf{x})\rho(\mathbf{x})\partial_t \mathbf{x}(\xi,t) &= \nabla.(\lambda(\mathbf{x})\nabla\mathbf{x}(\xi,t)) && \text{in } \Omega\times(0,T), \\ -\lambda(\mathbf{x})\partial_\nu \mathbf{x}(\xi,t) &= \mathbf{g}_i(\xi,t,\mathbf{x},u) && \text{on } \Gamma_i\times(0,T), \\ \mathbf{x}(\xi,0) &= \mathbf{x}_0(\xi) && \text{in } \Omega, \end{aligned} \tag{3.5}$$

where g_i includes temperature differences between cooling fluid and profile surface, intensity parameters for the cooling nozzles and heat transfer coefficients modeling the heat transfer to the cooling fluid, as well as radiation portions. Some variants of this boundary condition will be presented in Section 3.3.3.

We will mostly use the linearized version of the above state equation given in (3.6). The linearization is derived from (3.5) by taking means of the material parameters ρ, λ and c. This is admissible as long as we work in temperature regimes above 700°C where changes of ρ, λ and c are small and we do not have to deal with multiple phases and phase transitions in the material. Furthermore we partition the boundary into 8 parts, where one of these is the artificial boundary on the left hand side of Ω. The others are located between two neighboring corners of the domain and are enumerated clockwise (see Figure 3.1 for details). Another simplification taken here is the assumption that the cooling nozzles spray constantly onto one part of the surface. This means, u is constant with respect to the spatial variable ξ on each part Γ_i of the boundary. Hence $\mathcal{U} = \mathbb{R}^7$ in our case and we obtain the following model:

$$\begin{aligned} c\rho\partial_t \mathbf{x}(\xi,t) &= \nabla.(\lambda\nabla\mathbf{x}(\xi,t)) && \text{in } \Omega\times(0,T) \\ -\lambda\partial_\nu \mathbf{x}(\xi,t) &= \mathbf{g}_i(t,\mathbf{x},u_i) && \text{on } \Gamma_i \text{ where } i=0,\dots,7 \\ \mathbf{x}(\xi,0) &= \mathbf{x}_0(\xi) && \text{in } \Omega. \end{aligned} \tag{3.6}$$

Throughout this section we will consider the following cost functional:

$$\mathfrak{J}(\mathbf{u}) := \int_0^\infty (\mathbf{x}, \mathbf{Q}\mathbf{x})_{\mathcal{H}} + (\mathbf{u}, \mathbf{R}\mathbf{u})_{\mathcal{U}} dt. \tag{3.7}$$

in which $\mathbf{Q}$ and $\mathbf{R}$ can be chosen to weight temperature differences and the cost of spraying the cooling fluid, i.e., (2.12) with $t_0 = 0$, $T_f = \infty$ and the proper inner products.

The control problem of interest can thus be summarized as

> Minimize (3.7) with respect to (3.6), where $\mathbf{Q}$ is given as $\mathbf{Q} := \mathbf{C}^*\mathbf{C}$ or $\mathbf{Q} := \mathbf{C}^*\hat{\mathbf{Q}}\mathbf{C}$ with $\hat{\mathbf{Q}} \geq 0$. (ℛ)

We will specify $\mathbf{C}$ and $\hat{\mathbf{Q}}$ in more detail later (Section 3.3.4).

3.3.3. Boundary Conditions and Boundary Control

We now have to describe the heat transfer across the surface of the material, i.e. the boundary conditions. The most general way to model the heat flux across the boundary

is a combination of heat conduction and radiation. The heat conduction is modeled proportional to the difference between surface temperature and exterior temperature, where the proportionality coefficient is the heat transfer coefficient κ depending on the material and shape of the profile, the type of cooling fluid that is used, the temperature of the fluid and also on the spraying intensity. The radiation of heat is given by the Stefan-Boltzmann law. So we end up with

$$-\lambda\partial_\nu \mathbf{x}(\xi) = \mathbf{g}_i(t,\mathbf{u}) := \kappa_k(\mathbf{x}-\mathbf{x}_{ext,k}) + \varepsilon\sigma(\mathbf{x}^4 - \mathbf{x}_{ext,k}^4), \quad k = 1,\dots,7, \tag{3.8}$$

where $\sigma = 5.660 \cdot 10^{-8} W/m^2K^4$ is the Stefan-Boltzmann constant and $\varepsilon \in [0,1]$ the emissivity ($\varepsilon = 1$ for an ideal black body, so we should expect $\varepsilon < 1$ here.). For a physical derivation of this boundary condition we refer the reader to textbooks in physics, e.g. [66].

The boundary condition (3.8) is nonlinear and thus must be linearized if we want to use LQR design for linear systems. Therefore we will simplify it by dropping the Stefan-Boltzmann part. This is not too much of an error because that term is much smaller than the conduction part, at least in case of active cooling and results in

$$-\lambda\partial_\nu \mathbf{x}(\xi) = \kappa_k(\mathbf{x}-\mathbf{x}_{ext,k}). \tag{3.9}$$

This simplified version of the boundary condition has already been applied successfully in [48, 49, 85, 143, 127, 27]. We now have two choices for selecting the control variables. The most intuitive choice concerning the presented model problem is to regulate the intensity of the spraying nozzles. This idea results in taking the heat transfer coefficient κ as the control. As we will see later in detail this leads to a problem with the formulation of the linear feedback control system. Instead of a linear system we have to consider a bilinear system with this choice which results from the multiplication of the control and the state on the right hand side of (3.9).

Another possibility is to take the external temperature as the control. In our example this means we regulate the temperature of the cooling fluid which is sprayed onto the steel profile. This might lead to complications with the technical realization of the model because in this application it will possibly be difficult to achieve the reaction times calculated by the model. On the other hand we can think of applications of the method at hand to the modeling and control of air conditioning systems. There, reaction times are much longer and this choice would most likely be the best one in that case. The mathematical advantage of this choice is that the multiplication of control and state which lead to the difficulties in the above case are bypassed here.

A third possibility would be to write (3.9) as $-\lambda\partial_\nu \mathbf{x}(\xi) = \kappa_1\mathbf{x} - \kappa_2\mathbf{x}_{ext,k}$ and replace κ_1 by an appropriate constant, e.g., some kind of mean. We could then use κ_2 as the control, but this is equivalent to controlling by means of the exterior temperature in the formulation above. Having κ_1 fixed we can rewrite $\kappa_2 = \kappa_1 * \hat{\kappa}_2$ and define $\hat{\mathbf{x}}_{ext,k} := \hat{\kappa}_2\mathbf{x}_{ext,k}$, i.e. we can use $\hat{\mathbf{x}}_{ext,k}$ as the control like in (3.9).

Boundedness of the input operator B. The boundary condition following (3.9) and choosing $u_k = \mathbf{x}_{ext,k}$ takes the form

$$-\lambda \partial_\nu \mathbf{x}(\xi, t) - \kappa_k \mathbf{x}(\xi, t) = \mathbf{B}(\xi)|_{\Gamma_k} u(t) \text{ on } \Gamma_k.$$

Now observing, that the rail profile Ω is a regular domain, we have that the trace operator maps from $\mathcal{H}^1(\Omega)$ to $\mathcal{L}^2(\Gamma_k)$ for each k continuously [146]. Thus $\mathbf{B}$ following (A.6) is well defined and continuous. Linearity of $\mathbf{B}$ is obvious and hence it is bounded.

3.3.4. Choice of State Weighting Operator Q and Output Operator C

In ($\mathcal{R}$) we already mentioned that the control weighting operator/matrix $\mathbf{Q}$ should be chosen as $\mathbf{Q} := \mathbf{C}^*\mathbf{C}$ or $\mathbf{Q} := \mathbf{C}^*\hat{\mathbf{Q}}\mathbf{C}$. We will now show in more detail how we choose $\mathbf{C}$ and $\hat{\mathbf{Q}}$. As it was mentioned in Section 3.3.1, an even temperature distribution on cross-sections of the profile during the cooling process is desired. We want to take this fact into account by introducing certain temperature differences in the cost functional. We approximate gradients by simple differences because this turns out to be sufficient to accomplish the given task. Additionally, temperature difference calculations are cheaper to compute and easier to implement. So concerning implementation they are the primary choice here.

On the other hand this leads to slight complications in the theoretical part. We would like to evaluate the state function in single nodes of the coarsest grid, to know the temperature in a specific point. We use nodes of the coarsest grid here, because those are present on every refinement level created by our finite element method. Unfortunately the regularity of the solution is not sufficient to allow those evaluations. We do not have $\mathcal{H}^2$-regularity of the solution, since the boundary is not smooth enough (because of the two sharp corners at the ends of Γ_6 and the non-convexity of Ω) and boundary conditions may jump in the interconnection points between two parts of the boundary. Therefore we do not have continuity of the solution and thus can not evaluate the state function in single points, but we can evaluate integrals of the state over small regions in Ω. This problem is solved by defining $\mathbf{C}$ according to differences between integral means on small ε-balls around the desired grid nodes. That means if we are interested in the temperature at the i-th grid node with coordinates $\xi_i \in \Omega$ we consider

$$\eta_i := \begin{cases} \frac{1}{|\mathcal{B}_\varepsilon(\xi_i)|} \int\limits_{\mathcal{B}_\varepsilon(\xi_i)} \mathbf{x}(\xi) d\xi & \text{falls } \xi_i \in \Omega \setminus \Gamma, \\ \frac{1}{|\mathcal{B}_\varepsilon(\xi_i) \cap \Gamma|} \int\limits_{\mathcal{B}_\varepsilon(\xi_i) \cap \Gamma} \operatorname{tr}(x(\xi))\, d\sigma & \text{falls } \xi_i \in \Gamma. \end{cases} \tag{3.10}$$

With this notation we define

$$\begin{aligned} \mathbf{C}: \quad \mathcal{H} &\to \mathbb{R}^6 \\ \mathbf{x} &\mapsto \mathbf{C}(\mathbf{x}) := \begin{bmatrix} 3\eta_{60} - \eta_{22} - \eta_4 \\ 2\eta_{63} - \eta_3 - \eta_2 \\ \eta_{51} - \eta_{43} \\ 2\eta_{92} - \eta_9 - \eta_{16} \\ 3\eta_{83} - \eta_{34} - \eta_{10} - \eta_{15} \end{bmatrix} \end{aligned} \tag{3.11}$$

The grid nodes referred to in the above definition can be found in Figure 3.1. The lines of $\mathbf{C}(\mathbf{x})$ in (3.11) have to be read as: take the difference of the temperature integrals (3.10) for nodes 63 and 3 as well as the difference for nodes 63 and 2 (if we look at line 2 as an example) and add them. Note that we placed an additional weight on the temperature around node 60 in line 1 of $\mathbf{C}(\mathbf{x})$. This turned out to be important to get the profile's foot appropriately cooled down with the given cost functional, see the plots in the results section of [26] for details. Note that the operator $\mathbf{C}$ defined as in equation (3.11) is bounded since the trace operator is linear and continuous and Ω is assumed to be bounded.

Concerning $\hat{\mathbf{Q}}$ we think of choosing $\hat{\mathbf{Q}} := \beta\mathbf{I}$ for some positive real constant β. β can then be used as a weighting factor to priorize states over controls or the contrary in the cost functional. Alternative choices for $\hat{\mathbf{Q}}$ might be diagonal matrices where the diagonal entries are then weighting factors for the temperature differences in relation to each other. For example one might want to devaluate the differences on the central bar against those in the head of the profile, because temperature differences normally tend to be much smaller on the central bar than in the head.

Boundedness of the Output Operator C. We have seen that the resulting inputs are in $\mathcal{L}^2(\Gamma)$. Hence for fixed t the solution operator at least gives us $\mathbf{x}(.,t) \in \mathcal{H}^1(\Omega)$. Therefore $\mathbf{x}(.,t) \in \mathcal{L}^2(\Omega) \hookrightarrow \mathcal{L}^1(\Omega)$ and the integrals in (3.10) exist for $\xi_i \in \Omega\backslash\Gamma$. By the continuity of the trace (which we already exploited for the boundedness of $\mathbf{B}$) we also have that the integrals exist for $\xi_i \in \Gamma$. Linearity of $\mathbf{C}$ follows directly from the linearity of the integrals with respect to the integrand. Thus also $\mathbf{C}$ is linear and continuous, i.e., bounded.

3.3.5. Units of Measurement and Scaling

The final paragraph of this model introduction section will now concern units of measurement. We have introduced the model without too much concern about measurements until here. For the implementation we want to rescale the temperature regime from the interval $[0, 1000]$ °C to the unit interval $[0, 1]$ and the lengths from meters to decimeters. We are especially interested in the effect this rescaling has on the time scale. To answer this question we first list the parameters again with their units of measurement:

- specific density ϱ in $\frac{kg}{m^3}$,
- specific heat capacity c in $\frac{m^2}{s^2 °C}$,
- heat conductivity λ in $\frac{kg\ m}{s^3 °C}$,
- heat transfer coefficient κ in $\frac{kg}{s^3 °C}$.

For $\alpha = \frac{\lambda}{c\varrho}$ this leads to

$$\frac{\frac{kg\ m}{s^3\ °C}}{\frac{m^2}{s^2\ °C}\ \frac{kg}{m^3}} = \frac{m^2}{s}.$$

So the rescaling of temperature has no effect on the other units, for it cancels out in the above computation. The rescaling of lengths on the other hand has to be taken into account even squared. If we do this we can take the original values of λ, c, ϱ and κ for the numerical tests. Dividing by $-\lambda$, (3.9) becomes $\partial_\nu x = \frac{\kappa}{\lambda}(x_{ext} - x)$. So we have to take a closer look at the coefficient $\frac{\kappa}{\lambda}$. This has the unit of measurement

$$\frac{\frac{kg}{s^3\ °C}}{\frac{kg\ m}{s^3\ °C}} = \frac{1}{m}.$$

Hence we do not have to take the rescaling into account, because the normal ν on the left and the coefficient $\frac{\kappa}{\lambda}$ scale with the same factor.

3.4. Chemical Reactors: Controling the Temperature of an Inflowing Reagent

The next example is a system appearing in the optimal heating/cooling of a fluid flow in a tube. An application could be the temperature regulation of certain reagent inflows in chemical reactors. The model equations are:

$$\begin{aligned} \frac{\partial \mathbf{x}}{\partial t} - \kappa\Delta\mathbf{x} + \mathbf{v}\cdot\nabla\mathbf{x} &= 0 && \text{in } \Omega \\ \mathbf{x} &= \mathbf{x}_0 && \text{on } \Gamma_{in} \\ \frac{\partial \mathbf{x}}{\partial n} &= \sigma(\mathbf{u} - \mathbf{x}) && \text{on } \Gamma_{heat1} \cup \Gamma_{heat2} \\ \frac{\partial \mathbf{x}}{\partial n} &= 0 && \text{on } \Gamma_{out}. \end{aligned} \tag{3.12}$$

Here Ω is the rectangular domain shown in Figure 3.2. The inflow Γ_{in} is at the left part of the boundary and the outflow Γ_{out} the right one. The control is applied via the upper and lower boundaries. We can restrict ourselves to this 2d-domain assuming rotational symmetry, i.e., non-turbulent diffusion dominated flows. The test matrices have been created using the COMSOL[4] Multiphysics software and have dimension 1090. The

[4] http://www.comsol.de

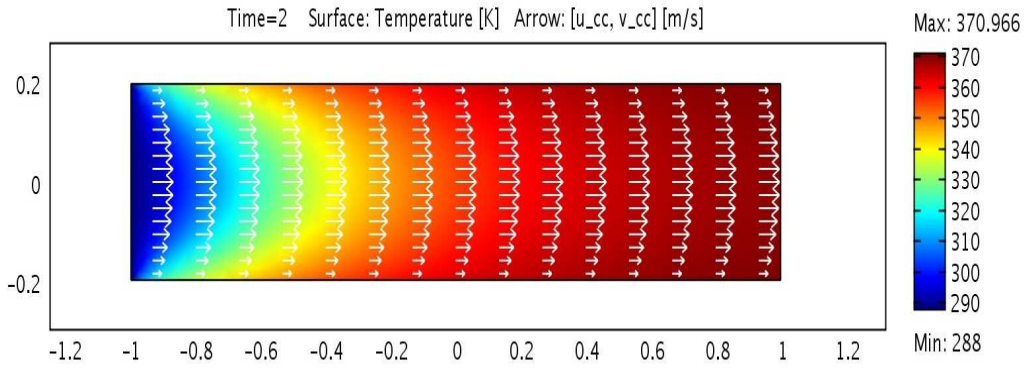


Figure 3.2.: Domain Ω for the Inflow Example: A 2d cross-section of the liquid flow in a round tube

system has a single input applied at both upper and lower boundary, due to rotational symmetry and the three outputs correspond to three values of the temperature at the outflow. Note that in this case we have a convex domain such that we can use point evaluations as the outputs. $\kappa = 0.06$ and the system results in the eigenvalue and shift distributions shown in Figure 8.4 a.

Since a finite element discretization in space has been applied here, the semi-discrete model is of the form

$$\begin{aligned} M\dot{x} &= \tilde{A}x + \tilde{B}u \\ y &= \tilde{C}x. \end{aligned} \tag{3.13}$$

This is transformed into a standard system (8.2) by decomposing M into $M = M_L M_U$ where $M_L = M_U^T$ since M is symmetric. Then M_U acts as a state space transformation and only M_L has to be inverted. Note, that the inversion should never be performed explicitly. See Chapter 5 for a more detailed presentation of this transformation.

3.5. The SLICOT CD-Player

This example is only mentioned for completeness in this chapter. It is the well known CD-Player[5] example from the SLICOT collection that has been frequently used as a test example in the literature and therefore does not need an extensive introduction. We restrict ourselves to a SISO variant of the model by extracting only the first row of C and second column of B as the new output and input operators. This model is considerably tough since it is relatively small ($n = 120$), but the Bode plot shows many peaks and oscillations, that are hard to catch especially with a small reduced order model.

[5] http://www.slicot.org/index.php?site=examples

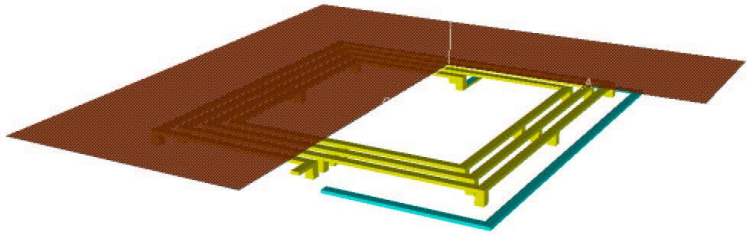


Figure 3.3.: Basic Configuration of the Spiral Inductor

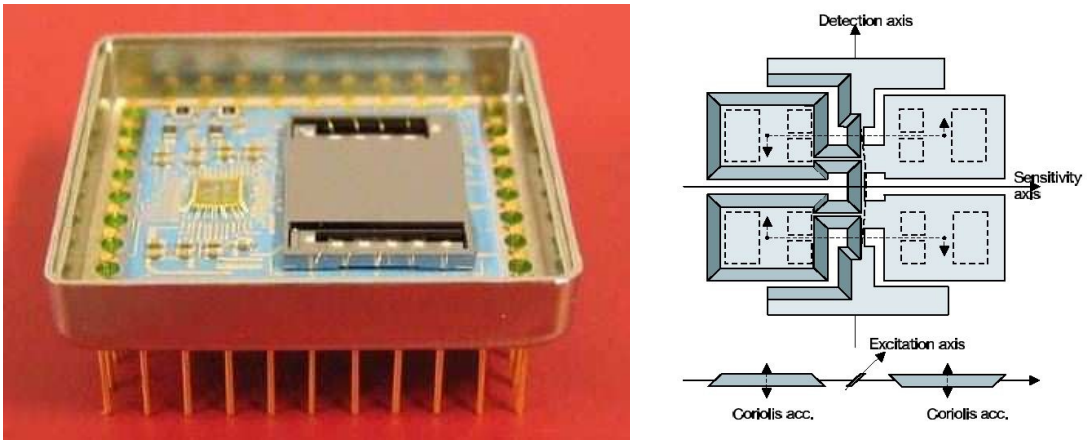
3.6. The Spiral Inductor

The spiral inductor is a model of an integrated radio frequency passive inductor. It has been equipped with an additional plane of copper (see Figure 3.3) to make it act as a proximity sensor, as well. The overall real world extensions of the spiral turns is less than 2mm in square. The matrices have been assembled according to a Partial Element Equivalent Circuit method (PEEC) modeling with 2117 filaments, resulting in an order 1434 SISO model in generalized state space form. A detailed description of the model can be found in [96]. The system matrices are part of the Oberwolfach Model Reduction Benchmark Collection and can be downloaded form the description page at IMTEK[6].

3.7. A Scalable Oscillator Example

Single oscillator chains consisting of various coupled (by spring elements) masses with, e.g., proportional damping are a basic example for second order oscillator models in textbooks and lecture notes. It is an easy exercise to see that we can express them with diagonal mass and tridiagonal stiffness and damping matrices. Here we discuss a slightly more complicated model, that has recently been used by Truhar and Veselic [144]. It consists of three such chains. Each of them coupled to a fixed mounting by an additional damper on the one end and fixed rigidly to a large mass coupling the three of them. The large mass is bound to a fixed mount by a single spring element. Each of the chains consists of n_1 equal masses and stiffnesses. Thus the model parameters are the masses m_1, m_2, m_3 and corresponding stiffnesses k_1, k_2, k_3 in the three oscillator chains,

[6] http://www.imtek.de/simulation/benchmark/wb/38891/



(a) The *Butterfly Gyro* (b) Schematic view to the *Butterfly Gyro*

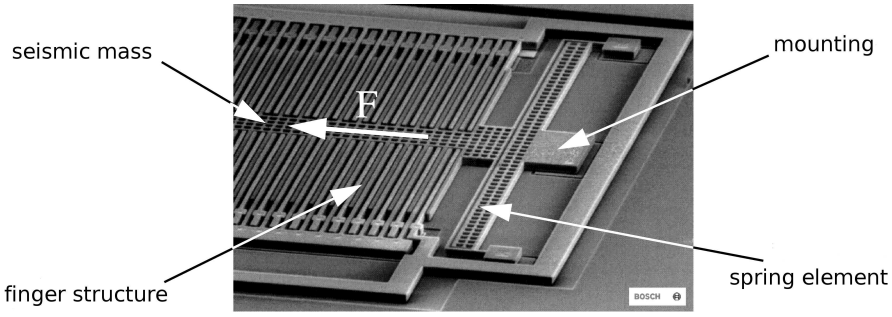
Figure 3.4.: The actual device and model scheme for the *Butterfly Gyro*

the coupling mass m_0 with its spring stiffness k_0, the viscosities v of the additional wall-mount-dampers and the length n_1 of the single oscillator chains. The resulting system then is of order $n = 3n_1 + 1$. The mass matrix M stays diagonal. The stiffness K and damping E now consist of a leading block diagonal matrix (consisting of the three stiffness matrices for the three oscillator chains) and coupling terms in the last row and column at positions $n_1, 2n_1, 3n_1$ and in the diagonal element. If it is not stated otherwise we are using $m_1 = 1$, $m_2 = 2$, $m_3 = 3$, $m_0 = 10$, $k_1 = 10$, $k_2 = 20$, $k_3 = 1$ and $k_0 = 50$ for the model. The viscosity v is taken as 5 and the default problem size is 1501, i.e., $n_1 = 500$.

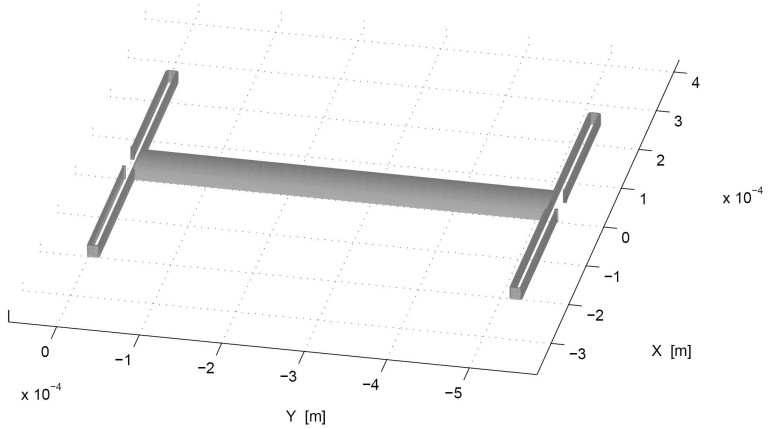
3.8. The Butterfly Gyro

The *Butterfly Gyro* is a vibrating micro mechanical gyro for application in inertial navigation. The gyroscope is a three layered silicon wafer stack of which the actual sensor element is the middle layer. The name of the device is derived from the fact that the sensor is set up as a pair of double wings connected to a common beam (See Figure 3.4b). The input matrices have been obtained by an ANSYS[7] model. The original model consists of 17,361 degrees of freedom resulting in an order 17,361 second order original model. Thus, the equivalent first order original model following (2.7) is of dimension 34,722. Both systems have a single input and 12 outputs and the number of nonzero entries in the system matrices is of order 10^7 (see also Figure 3.6a). This model is taken from the Oberwolfach Model Reduction Benchmark Collection[8] as well.

[7] http://www.ansys.com
[8] http://www.imtek.de/simulation/benchmark/wb/35889/



(a) Microscopic view to the Fraunhofer/Bosch acceleration sensor



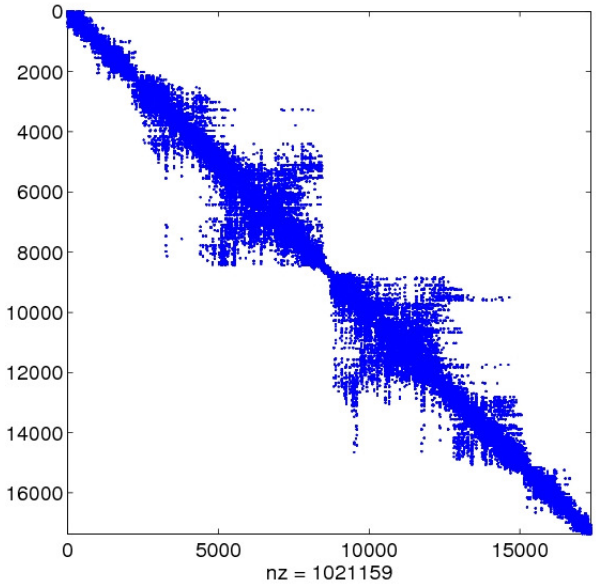
(b) Base configuration of an acceleration sensor.

Figure 3.5.: Microscopic view and model scheme for the acceleration sensor

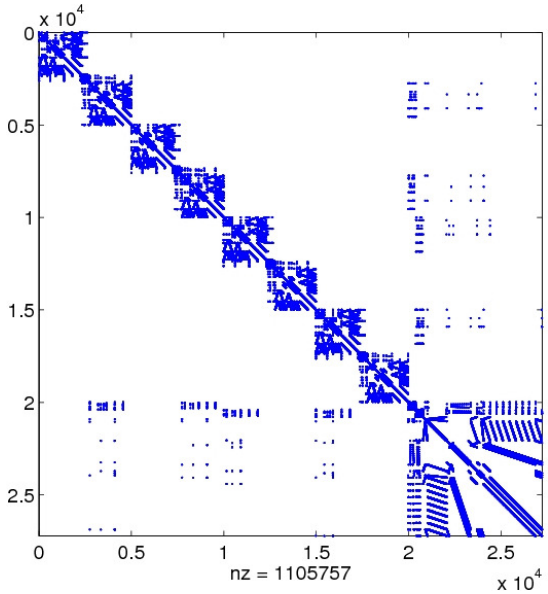
3.9. Fraunhofer/Bosch Acceleration Sensor

The basic structure of the micro mechanic acceleration sensor consists of a seismic mass coupled to two beam configurations at both its ends (see Figure 3.5b). The beam configurations as well as the seismic mass have been modeled by beam elements and connected by coupling elements. The original simulations [65] have been performed using SABER[9] and ANSYS. The model has 4 inputs an 3 outputs. The order of the second order system is 27,225 resulting in an equivalent first order system of order 54,450. Although the system is considerably larger, the number of nonzero elements is about the same as in the gyro example case (see Figure 3.6b).

[9] http://www.synopsys.com/Tools/SLD/Mechatronics/Saber/Pages/default.aspx



(a) Stiffness matrix for the *Butterfly Gyro*



(b) Stiffness matrix for the acceleration sensor

Figure 3.6.: Sparsity patterns for the Butterfly Gyro and Fraunhofer/Bosch acceleration sensor

> Good ideas are not adopted automatically. They must be driven into practice with courageous patience.
>
> HYMAN RICKOVER

CHAPTER

FOUR

EFFICIENT SOLUTION OF LARGE SCALE MATRIX EQUATIONS

Contents

We now get to the most integral part of this thesis. The numerical treatment of large scale sparse algebraic matrix equations is one of the key ingredients to many linear-quadratic optimal control problems for parabolic PDEs, as well as the balancing based model order reduction of large linear systems. The basic idea on which all methods for such

kinds of problems are based,is that often the numerical rank of the solution is very small compared to its actual dimension (see, [115, 7, 59]) and therefore it allows for a good approximation via low-rank solution factors. In this chapter we concentrate on the low-rank solution of those matrix equations based on alternating directions implicit (ADI) related methods. We start the discussion by an introduction of the basic ADI iteration and its application to Lyapunov equations. After that we treat one of the crucial issues in ADI methods – the choice and computation of good iteration parameters. Some of the major contributions of this thesis are then discussed in the fourth section. There we present acceleration techniques for the low-rank ADI iteration, that can drastically reduce the number of iterations steps and the memory requirements of the LRCF-ADI.

In the fifth section we will then discuss the combination of the LRCF-ADI with Newton type methods for the solution of algebraic Riccati equations. We especially review how these methods can be rewritten to solve the LQR problem for the feedback operator directly and relate the Newton-Kleinman-ADI approach to the recently proposed quadratic ADI (QADI) method that runs an ADI-type iteration on the quadratic Riccati equation without employing an additional Newton's method.

In the final section of this chapter we then present a handful of stopping criteria applicable for the iterations presented in the other sections and discuss their efficient evaluation.

4.1. The ADI Iteration

The ADI iteration was originally introduced in [112] as a method for solving elliptic and parabolic difference equations.

Let $A \in \mathbb{R}^{n\times n}$ be a real symmetric positive definite (spd) matrix and let $s \in \mathbb{R}^n$ be known. We can apply the ADI iteration to solve

$$Au = s,$$

when A can be expressed as the sum of matrices H and V for which the linear systems

$$\begin{aligned} (H + pI)v &= r, \\ (V + pI)w &= t \end{aligned}$$

admit an efficient solution. Here p is a suitably chosen parameter and r, t are known.

If H and V are spd, then there exist positive parameters p_j for which the two-sweep iteration defined by

$$\begin{aligned} (H + p_jI)u_{j-\frac{1}{2}} &= (p_jI - V)u_{j-1} + s, \\ (V + p_jI)u_j &= (p_jI - H)u_{j-\frac{1}{2}} + s \end{aligned} \tag{4.1}$$

for $j = 1, 2, \ldots$ converges. If the shift parameters p_j are chosen appropriately, then the convergence rate is superlinear, but convergence rates can be ensured only when

the matrices H and V commute. In the non-commutative case the ADI iteration is not competitive with other methods. This section and the following two section are essentially taken from [20]. In the following section we will show why solving Lyapunov equations is considered an ADI model problem and motivate the basic low-rank ADI iteration.

4.2. Lyapunov Equations: An ADI Model Problem

We consider a Lyapunov equation of the form

$$FY + YF^T = -GG^T \tag{4.2}$$

with stable F; (4.2) is a model ADI problem [145]. The model condition that the component matrices commute is retained. It can be seen when one recognizes that this is equivalent to a linear operator $M : Y \mapsto F^TY + YF = -GG^T$ where M is the sum of commuting operators: $M_L : Y \mapsto F^TY$ and $M_R : Y \mapsto YF$. In fact, unrolling matrices into vectors in (4.2) one observes that the corresponding Kronecker products $I \otimes F^T$ and $F^T \otimes I$ commute.

Applying the ADI iteration (4.1) to (4.2) yields

$$\begin{aligned} (F + p_jI)Y_{j-\frac{1}{2}} &= -GG^T - Y_{j-1}(F^T - p_jI), \\ (F + p_jI)Y_j^T &= -GG^T - Y_{j-\frac{1}{2}}^T(F^T - p_jI). \end{aligned} \tag{4.3}$$

Note that the matrix $Y_{j-\frac{1}{2}}$ is in general not symmetric after the first sweep of each iteration, but the result of the double sweep is symmetric.

The Basic Idea of Low-Rank ADI. The key observation [116] towards a low-rank version of this iteration is, that after rewriting it into a one step iteration

$$Y_j = -2p_j(F + p_jI)^{-1}GG^T(F + p_jI)^{-T} + (F + p_jI)^{-1}(F - p_jI)Y_{j-1}(F - p_jI)^T(F + p_jI)^{-T}, \tag{4.4}$$

we find that (4.4) is symmetric. Now assuming $Y_j = Z_jZ_j^T$ and $Y_0 = 0$ we can write the iteration in terms of the factors Z_j, as

$$\begin{aligned} Z_1 &= \sqrt{-2p_1}(F + p_1I)^{-1}G, \\ Z_j &= \left[\sqrt{-2p_1}(F + p_jI)^{-1}G,\ (F + p_jI)^{-1}(F - p_jI)Z_{j-1}\right]. \end{aligned}$$

Hence we can write (4.3) such that it forms the factors by successively adding a fixed number of columns in every step. In the current formulation however all columns have to be processed in every step, which makes the iteration increasingly expensive. Now let $J \in \mathbb{N}$ be the number of shift parameters we have at hand. Then defining the matrices

$T_j := (F - p_j I)$ and inverse matrices $S_j := (F + p_j I)^{-1}$ as in [97] we can express the J-th iterate as

$$Z_J = \left[S_J \sqrt{-2p_J} G,\ S_J(T_J S_{J-1}) \sqrt{-2p_{J-1}} G,\ \ldots,\ S_J T_J \cdots S_2(T_2 S_1) \sqrt{-2p_1} G \right].$$

Now observing that the S_j and T_j commute we can reorder these matrices such that we note that every block of the dimension of G essentially contains its left neighbor, i.e., predecessor in the iteration. Thus we find that we can rewrite the factor in the form

$$Z_J = \left[z_J,\ P_{J-1} z_J,\ P_{J-2}(P_{J-1} z_J),\ \ldots,\ P_1(P_2 \cdots P_{J-1} z_J) \right], \tag{4.5}$$

and only need to apply a *step operator*

$$\begin{aligned} P_i &:= \frac{\sqrt{-2\,\mathrm{Re}\,(p_i)}}{\sqrt{-2\,\mathrm{Re}\,(p_{i+1})}} (F + p_i I)^{-1}(F - p_{i-1} I) \\ &= \frac{\sqrt{-2\,\mathrm{Re}\,(p_i)}}{\sqrt{-2\,\mathrm{Re}\,(p_{i+1})}} \left[I - (p_i + \overline{p_{i-1}})(F + p_i I)^{-1} \right] \end{aligned} \tag{4.6}$$

to compute the new columns in every step. Especially note that only the new column need to be processed this way. In summary this gives the presentation in Algorithm 4.1.

Convergence and Shift Parameters. If the shift parameters p_j in (4.3) are chosen appropriately then $\lim_{j\to\infty} Y_j = Y$ with super-linear convergence rate. The error in iterate j is given by $e_j = R_j e_{j-1}$, where

$$R_j := (F + p_j I)^{-1}(F^T - p_j I)(F^T + p_j I)^{-1}(F - p_j I).$$

and $e_0 := Y_0 - Y$. Thus the error after J iterations satisfies

$$e_J = W_J e_0, \quad W_J := \prod_{j=1}^{J} R_j.$$

Writing R_j in terms of the Kronecker products $I \otimes F^T$ and $F^T \otimes I$ one observes that the factors in R_j commute and $\|W_j\|_2 = \rho(W_j)$. Therefore

$$\|e_J\|_2 \le \rho(W_J)\|e_0\|_2, \quad \rho(W_J) = k(\mathbf{p})^2,$$

where $\mathbf{p} = \{p_1, p_2, \ldots, p_J\}$ and

$$k(\mathbf{p}) = \max_{\lambda \in \Lambda(F)} \left| \prod_{j=1}^{J} \frac{(p_j - \lambda)}{(p_j + \lambda)} \right|. \tag{4.7}$$

Obviously, this suggests to choose the ADI parameters such that we minimize $\rho(W_J)$ which leads to the rational min-max problem

$$\min_{\{p_j \in \mathbb{R}: j=1,\ldots,J\}} k(\mathbf{p}) \tag{4.8}$$

Algorithm 4.1 Low-rank Cholesky factor ADI iteration (LRCF-ADI)

Input: F,G defining $FX + XF^T = -GG^T$ and shift parameters $\{p_1, \ldots, p_{i_{max}}\}$
Output: $Z = Z_{i_{max}} \in \mathbb{C}^{n \times t_{imax}}$, such that $ZZ^H \approx X$

1: $V_1 = \sqrt{-2\,\mathrm{Re}\,(p_1)}(F + p_1 I)^{-1} G$
2: $Z_1 = V_1$
3: **for** $i = 2, 3, \ldots, i_{max}$ **do**
4: $\quad V_i = \sqrt{\mathrm{Re}\,(p_i)/\,\mathrm{Re}\,(p_{i-1})}(V_{i-1} - (p_i + \overline{p_{i-1}})(F + p_i I)^{-1} V_{i-1})$
5: $\quad Z_i = [Z_{i-1} \;\; V_i]$
6: **end for**

for the shift parameters p_j, see e.g. [145]. This minimization problem is also known as the rational Zolotarev problem since, in the real case (i.e. $\Lambda(F) \subset \mathbb{R}$) it is equivalent to the third of four approximation problems solved by Zolotarev in the 19th century, see [94]. For a complete historical overview see [140].

4.3. ADI Shift Parameter Selection

4.3.1. Review of Existing Parameter Selection Methods

Many procedures for constructing optimal or suboptimal shift parameters have been proposed in the literature [76, 116, 135, 145]. Most of the approaches cover the spectrum of F by a domain $\Omega \subset \mathbb{C}_{<0}$ and solve (4.8) with respect to Ω instead of $\Lambda(F)$. In general one must choose among the various approaches to find effective ADI iteration parameters for specific problems. One could even consider sophisticated algorithms like the one proposed by Istace and Thiran [76] in which the authors use numerical techniques for nonlinear optimization problems to determine optimal parameters. However, it is important to take care that the time spent in computing parameters does not outweigh the convergence improvement derived therefrom.

Wachspress et al. [145] compute the optimum parameters when the spectrum of the matrix F is real or, in the complex case, if the spectrum of F can be embedded in an elliptic functions region

$$D(p) = \{\, p = \mathrm{dn}(zK, k) \quad | \quad z = x + \imath y, \quad 0 \le x \le 1, \text{ and } |y| \le r \,\},$$

which often occurs in practice. If $1 - k \ll 1$ then these are egg-shaped regions in the complex plane, whose normalized logarithms are elliptic regions and symmetric with respect to both the real and imaginary axes. For the definitions of k and K see the corresponding paragraph on the optimal parameters below. More details and broad discussion of elliptic functions regions of special structure can be found in [145, Chapter IV]. The optimum parameters may be chosen real even if the spectrum is complex as long as the imaginary parts of the eigenvalues are *small* compared to their real parts

(see [100, 145] for details). The method applied by Wachspress in the complex case is similar to the technique of embedding the spectrum into an ellipse and then using Chebyshev polynomials. In case that the spectrum is not well represented by the elliptic functions region a more general development by Starke [135] describes how generalized Leja points yield asymptotically optimal iteration parameters. Finally, an inexpensive heuristic procedure for determining ADI shift parameters, which often works well in practice, was proposed by Penzl [116]. We will summarize these approaches here.

Leja Points. Gonchar [58] characterizes the general min-max problem and shows how asymptotically optimal parameters can be obtained with generalized Leja or Fejér points. Starke [136] applies this theory to the ADI min-max problem (4.8). The generalized Leja points are defined as follows. Given $\mathcal{E}, \mathcal{F}$ subsets of $\mathbb{C}$ containing the spectra of $I \otimes F^T$ and $F^T \otimes I$, as well as arbitrarily points $\varphi_j \in \mathcal{E}$ and $\psi_j \in \mathcal{F}$, then for $j = 1, 2, \ldots$, the new points $\varphi_{j+1} \in \mathcal{E}$ and $\psi_{j+1} \in \mathcal{F}$ are chosen recursively in such a way that, with

$$r_j(z) = \prod_{i=1}^{j} \frac{z - \varphi_j}{z - \psi_j} \tag{4.9}$$

the two conditions

$$\max_{x \in \mathcal{E}} |r_j(z)| = |r_j(\varphi_{j+1})|, \qquad \max_{x \in \mathcal{F}} |r_j(z)| = |r_j(\psi_{j+1})|, \tag{4.10}$$

are fullfilled. Bagby [8] shows that the rational functions r_j obtained by this procedure are asymptotically minimal for the rational Zolotarev problem.

The generalized Leja points can be determined numerically for a large class of boundary curves $\partial\mathcal{E}$. When relatively few iterations are needed to attain the prescribed accuracy, the Leja points may be poor. Moreover their computation can be quite time consuming when the number of Leja points generated is large, since the computation gets more and more expensive the more prior Leja points have already been calculated. A potential theory based computation framework for the Leja point based shifts was introduced by Sabino in [128].

Optimal Parameters. In this section we will briefly summarize the parameter selection procedure given in [145].

Define the spectral bounds a, b and a sector angle α for the matrix F as

$$a = \min_i(Re\{\lambda_i\}), \quad b = \max_i(Re\{\lambda_i\}), \quad \alpha = \tan^{-1} \max_i \left| \frac{Im\{\lambda_i\}}{Re\{\lambda_i\}} \right|, \tag{4.11}$$

where $\lambda_1, \ldots, \lambda_n$ are eigenvalues of $-F$. It is assumed that the spectrum of $-F$ lies inside the elliptic functions region determined by a, b, α, as defined in [145]. Let

$$\cos^2 \beta = \frac{2}{1 + \frac{1}{2}\left(\frac{a}{b} + \frac{b}{a}\right)}, \quad m = \frac{2\cos^2 \alpha}{\cos^2 \beta} - 1. \tag{4.12}$$

If $\alpha < \beta$, then $m \geq 1$ and the parameters are real. We define

$$k_1 = \frac{1}{m + \sqrt{m^2 - 1}}, \quad k = \sqrt{1 - {k_1}^2}. \tag{4.13}$$

Define the elliptic integrals K and v via

$$F[\psi, k] = \int_0^{\psi} \frac{dx}{\sqrt{1 - k^2 \sin^2 x}}, \tag{4.14}$$

as

$$K = K(k) = F\left[\frac{\pi}{2}, k\right], \quad v = F\left[\sin^{-1}\sqrt{\frac{a}{bk_1}}, k_1\right], \tag{4.15}$$

where F is the incomplete elliptic integral of the first kind, k is its modulus and ψ is its amplitude.

The number of the ADI iterations required to achieve $k(\mathbf{p})^2 \leq \epsilon$ is $J = \lceil \frac{K}{2v\pi} \log \frac{4}{\epsilon} \rceil$, and the ADI parameters are given by

$$p_j = -\sqrt{\frac{ab}{k_1}} \operatorname{dn}\left[\frac{(2j-1)K}{2J}, k\right], \quad j = 1, 2, \ldots, J, \tag{4.16}$$

where $\operatorname{dn}(u, k)$ is the elliptic function (see [2]).

If $m < 1$, the parameters are complex. In this case we define the dual elliptic spectrum,

$$a' = \tan\left(\frac{\pi}{4} - \frac{\alpha}{2}\right), \quad b' = \frac{1}{a'}, \quad \alpha' = \beta.$$

Substituting a' in (4.12), we find that

$$\beta' = \alpha, \quad m' = \frac{2\cos^2\beta}{\cos^2\alpha} - 1.$$

By construction, m' must now be greater than 1. Therefore we may compute the optimum real parameters p'_j for the dual problem. The corresponding complex parameters for the actual spectrum can then be computed from:

$$\cos\alpha_j = \frac{2}{p'_j + \frac{1}{p'_j}}, \tag{4.17}$$

and for $j = 1, 2, \ldots, \lceil \frac{1+J}{2} \rceil$

$$p_{2j-1} = \sqrt{ab}\exp[\imath\alpha_j], \quad p_{2j} = \sqrt{ab}\exp[-\imath\alpha_j]. \tag{4.18}$$

Heuristic Parameters. The bounds needed to compute optimal parameters are too expensive to be computed exactly in case of large scale systems because they need the knowledge about the shape of the whole spectrum of F. In fact, this computation would be more expensive than the application of the ADI method itself.

An alternative was proposed by Penzl in [116]. He presents a heuristic procedure which determines suboptimal parameters based on the idea of replacing $\Lambda(F)$ by an approximation $\mathcal{R}$ of the spectrum in (4.8). Specifically, $\Lambda(F)$ is approximated using the Ritz values computed by the Arnoldi process (or any other large scale eigen-solver). Due to the fact that the Ritz values tend to be located near the largest magnitude eigenvalues, the inverses of the Ritz values related to F^{-1} are also computed to get an approximation of the smallest magnitude eigenvalues of F yielding a better approximation of $\Lambda(F)$. Note however that for symmetric matrices F the Arnoldi method reduces to the Lanczos method which converges to largest and smallest magnitude eigenvalues at the same time. Thus the additional computation with F^{-1} is not necessary there. The suboptimal parameters $\mathcal{P} = \{p_1, \ldots, p_k\}$ are then chosen among the elements of the approximation because the function

$$s_{\mathcal{P}}(t) = \frac{|(t-p_1)\ldots(t-p_k)|}{|(t+p_1)\ldots(t+p_k)|},$$

becomes small over $\Lambda(F)$ if there is one of the shifts p_j in the neighborhood of each eigenvalue. The procedure determines the parameters as follows. First, the element $p_j \in \mathcal{R}$ which minimizes the function $s_{\{p_j\}}$ over $\mathcal{R}$ is chosen. The set $\mathcal{P}$ is initialized by either $\{p_j\}$ or the pair of complex conjugates $\{p_j, \bar{p}_j\}$. Now $\mathcal{P}$ is successively enlarged by the elements or pairs of elements of $\mathcal{R}$, for which the maximum of the current $s_{\mathcal{P}}$ is attained. Doing this the elements of $\mathcal{R}$ giving the largest contributions to the value of $s_{\mathcal{P}}$ are successively canceled out. Therefore the resulting $s_{\mathcal{P}}$ is non-zero only in the elements of $\mathcal{R}$ where its value is comparably small anyway. In this sense (4.8) is solved heuristicly.

Discussion. We are looking for alternative strategies for non-real spectra with non-dominating imaginary parts. These are, e.g., spectra of the convection-diffusion-models in Sections 3.1 and 3.4 where the diffusive part is dominating the reaction or convection terms, respectively. Thus the resulting operator has a spectrum with only moderately large imaginary components compared to the real parts. In these problems the Wachspress approach should always be applicable and lead to real shift parameters in many cases. In problems, where the reactive and convective terms are absent, i.e., we are considering a plain heat equation and therefore the spectrum is part of the real axis, the Wachspress parameters are proven to be optimal. The heuristics proposed by Penzl is more expensive to compute there and Starke notes in [136], that the generalized Leja approach will not be competitive here since it is only asymptotically optimal. For the complex spectra case common strategies to determine the generalized Leja points generalize the idea of enclosing the spectrum by a polygonal domain, where the starting roots are placed in the corners. So one needs quite exact information about the shape of

Algorithm 4.2 Approximate optimal ADI parameter computation

Input: $F \in \mathbb{R}^{n\times n}$ Hurwitz stable

if $\Lambda(F) \subset \mathbb{R}$ **then**
  Compute the spectral bounds and set $a = \min \Lambda(-F)$ and $b = \max \Lambda(-F)$,
  $k_1 = \frac{a}{b}$, $k = \sqrt{1-k_1^2}$,
  $K = F(\frac{\pi}{2}, k)$, $v = F(\frac{\pi}{2}, k_1)$.
  Compute J and the parameters according to (4.16).
else
  Compute $\tilde{a} = \min \mathrm{Re}\,(\Lambda(-F))$, $\tilde{b} = \max \mathrm{Re}\,(\Lambda(-F))$ and $c = \frac{\tilde{a}+\tilde{b}}{2}$.
  Compute l largest magnitude eigenvalues $\hat{\lambda}_i$ for the shifted matrix $-F + cI$ by an Arnoldi process or alike.
  Shift these Eigenvalues back, i.e. $\tilde{\lambda}_i = \hat{\lambda}_i + c$.
  Compute a, b and α from the $\tilde{\lambda}_i$ like in (4.11).
  if $m \geq 1$ in (4.12) **then**
    Compute the parameters by (4.12)–(4.16).
  else {The ADI parameters are complex in this case}
    Compute the dual variables.
    Compute the parameters for the dual variables by (4.12)–(4.16).
    Use (4.17) and (4.18) to get the complex shifts.
  end if
end if

the spectrum there. In practice this would require the computation of the eigenvalues with largest imaginary parts already for a simple rectangular enclosure of the spectrum. Since this still does not work reliably, we decided to avoid the comparison with that approach here, although it is an option in cases where the Wachspress parameters are no longer applicable or one knows some a-priori information on the spectrum.

4.3.2. Suboptimal Parameter Computation

In this section we discuss our new contribution to the parameter selection problem. The idea is to avoid the problems of the methods reviewed in the previous section and on the other hand combine their advantages.

Since the important information that we need to know for the Wachspress approach is the outer shape of the spectrum of the matrix F, we will describe an algorithm approximating the outer spectrum. With this approximation the input parameters a, b and α for the Wachspress method are determined and the optimal parameters for the approximated spectrum are computed. Obviously, these parameters have to be considered suboptimal for the original problem, but if we can approximate the outer spectrum at a similar cost to that of the heuristic parameter choice we end up with a method giving nearly optimal parameters at a drastically reduced computational cost

compared to the optimal parameters.

In the following we discuss the main computational steps in Algorithm 4.2, that computes the parameters described above. The basic idea of the algorithm is to center the spectrum around the origin via a real shift c, such that all important eigenvalues determining the outer shape of the spectrum of F tend to be largest magnitude eigenvalues of the shifted matrix $F - cI$. Then the main workload for the determination of the shape of the spectrum lies on the Arnoldi process with respect to $F - cI$ and we avoid the application of F^{-1} as far as possible. Also eigenvalues with relatively large imaginary parts are fixed easier in this representation.

Real Spectra. In the case where the spectrum is real we can simply compute the upper and lower bounds of the spectrum by an Arnoldi or Lanczos process and enter the Wachspress computation with these values for a and b, and set $\alpha = 0$, i.e., we only have to compute two complete elliptic integrals by an arithmetic geometric mean process. This is very cheap since it is a quadratically converging scalar computation (see below).

Complex Spectra. For complex spectra we introduce an additional shifting step to be able to apply the Arnoldi process more efficiently. Since we are dealing with stable systems , we compute the largest magnitude and smallest magnitude eigenvalues and use the arithmetic mean of their real parts as a horizontal shift, such that the spectrum is centered around the origin. Now Arnoldi's method is applied to the shifted spectrum, to compute a number of largest magnitude eigenvalues. These will now automatically include the smallest magnitude eigenvalues of the original system after shifting back. So we can avoid extensive application of the Arnoldi method to the inverse of F. We only need it to get a rough approximation of the smallest magnitude eigenvalue to determine $\tilde{a}$ and $\tilde{b}$ for the shifting step.

The number of eigenvalues we compute can be seen as a tuning parameter here. The more eigenvalues we compute, the better the approximation of the shape of the spectrum is and the closer we get to the exact a, b and α, but obviously the computation becomes more and more expensive. Especially the dimension of the Krylov subspaces is rising with the number of parameters requested and with it the memory consumption in the Arnoldi process. But in cases where the spectrum is filling a rectangle or an egg-like shape, a few eigenvalues are sufficient (compare Section 8.1.1).

A drawback of this method can be that in case of small (compared to the real parts) imaginary parts of the eigenvalues, one may need a large number of eigenvalue approximations to find the ones with largest imaginary parts, which are crucial to determine α accurately. On the other hand in that case the spectrum is *almost* real and therefore it will be sufficient to compute the parameters for the approximate real spectrum in most applications.

Computation of the Elliptic Integrals. The new as well as the Wachspress parameter algorithms require the computation of certain elliptic integrals presented in (4.14). These are equivalent to the integral

$$F[\psi, k] = \int_0^{\psi} \frac{dx}{\sqrt{(1-k^2)\sin^2 x + \cos^2 x}} = \int_0^{\psi} \frac{dx}{\sqrt{(k_1^2)\sin^2 x + \cos^2 x}}. \tag{4.19}$$

In the case of real spectra, $\psi = \frac{\pi}{2}$ and $F[\frac{\pi}{2}, k]$ is a complete elliptic integral of the form

$$I(a,b) = \int_0^{\frac{\pi}{2}} \frac{dx}{\sqrt{a^2\cos^2 x + b^2 \sin^2 x}}$$

and $I(a,b) = \frac{\pi}{2M(a,b)}$, where $M(a,b)$ is the arithmetic geometric mean of a and b. The proof for the quadratic convergence of the arithmetic geometric mean process is given in many textbooks (e.g., [137]).

For incomplete elliptic integrals, i.e., the case $\psi < \frac{\pi}{2}$, an additional Landen's transformation has to be performed. Here, first the arithmetic geometric mean is computed as above, then a descending Landen's transformation is applied (see [2, Chapter 17]), which comes in at the cost of a number of scalar tangent computations equal to the number of iteration steps taken in the arithmetic geometric mean process above.

The value of the elliptic function dn from equation (4.16) is also computed by an arithmetic geometric mean process (see [2, Chapter 16]).

To summarize the advantages of the proposed method we can say:

- We compute real shift parameters even in case of many complex spectra, where the heuristic method would compute complex ones. This results in a significantly cheaper ADI iteration considering memory consumption and computational effort, since complex computations are avoided.
- We have to compute less Ritz values compared to the heuristic method, reducing the time spent in the computational overhead for the acceleration of the ADI method.
- We compute a good approximation of the Wachspress parameters at a drastically reduced computational cost compared to their exact computation.

Test computations utilizing the proposed methods in comparison to the legacy heuristic method by Penzl can be found in Section 8.1.

4.3.3. Dominant Pole Based Shifts for Balancing Based MOR

All the shift parameter selection methods discussed in the above aim at minimizing the spectral radius $\rho(W_J)$ of the global iteration matrix in the ADI process. In balancing

based MOR there might on the other hand be other choices that can contribute more to the actual goal of the computation. In contrast to solely solving the Lyapunov equation it is the generation of a reduced order model that preserves the input output behavior of the original system as good as possible. A commonly used measure for this approximation is the frequency response of the system. For general linear time invariant systems, the dominant eigenvalues are the poles of the transfer function that contribute significantly to the frequency response. The key observation for the definition of the dominant poles is that the transfer function of a MIMO system can be expressed as the sum of residues over the first order poles (eigenvalues) λ_i

$$H(s) = \sum_{i=1}^{\tilde{n}} \frac{R_i}{s - \lambda_i},$$

where the residue matrices R_j are

$$R_j = (C^* x_j)(y_j^* B).$$

Here x_j and y_j are the right and left eigenvectors corresponding to λ_j, which are scaled such that $y_j^* M x_j = 1$ and $y_i^* M x_j = 0$ if $j \neq i$. Then the *dominant poles* are those poles λ_j corresponding to relatively large residuals $\|R_j\|_2 / |\operatorname{Re}(\lambda_j)|$ in the above sum. Further details on the definition of dominant poles especially for the descriptor system case ($M = E$ singular) can be found in [125]. There the author also presents methods to compute these dominant poles in large scale sparse applications.

We further note that fast convergence of the ADI iteration is not always desirable in MOR applications, since the number of iterations taken limits the rank of the Gramian factors computed. The smaller of the ranks of the two Gramian factors entering the LR-SRBT (Section 2.4) via the low-rank square-root method (LR-SRM, see Algorithm 7.1) limits the order of the ROM and therefore the accuracy of the reduction. We will return to this topic in more detail in Chapter 7.

Thus it can be desirable to decrease the convergence speed of the ADI process as long as new subspace information enters the factors with every step taken, and with it the rank of the factor is increased. Now returning to the idea of the dominant poles above, we would expect that the essential information to cover all peaks of the transfer function is to catch the dominant poles and their corresponding eigenspaces. Therefore using the dominant poles as ADI shifts in MOR contexts suggests itself on a philosophic level. A mathematically rigorous analysis of this interrelation is an open problem. Still the numerical tests taken so far (see Section 8.1.4) show interesting phenomena that suggest further study.

4.4. Acceleration of the LRCF-ADI Method for Lyapunov Equations

The most criticized property of the ADI iteration for solving Lyapunov equations is its demand for a set of good shift parameters to ensure fast convergence. Although we have investigated several parameter computation techniques above that are cheaply computable, most of these are suboptimal in many cases and thus fast convergence can indeed not be guaranteed. Additionally, if the convergence is slow, the LRCFs may grow without adding essential information in subsequent iteration steps. If so the number of columns in the factors may easily exceed the rank of the factor leading to undesirable memory requirements.

Here we discuss several techniques trying to overcome these problems. In the next section we will present a column compression technique minimizing the number of columns of the LRCF and thus decreasing the "per step" computational cost. The remaining sections will then show how we can decrease the number of steps taken in the outer iteration.

4.4.1. Column Compression for the LRCFs

If we are in the situation that we cannot afford to compute good ADI shifts or all of the above suboptimal techniques are leading to slow convergence in many steps, then we will only slowly increase the dimension of the subspace spanned by the columns of the LRCFs but we will still add a fixed number of columns every step in finite arithmetic, i.e., in calculations carried out on a computer. These new columns do not only fill memory where it is not required, but also increase the computational cost of the iteration, since residual computations required in the stopping criteria will incorporate these columns as well. It is therefore highly desirable to keep the factors as small as possible. That means we need a method to reduce the number of columns of the LRCFs to their current rank. In [63] the authors propose to employ a sequential Karhunen-Loeve algorithm for this task. Since this involves a full QR decomposition and an SVD we suggest a cheaper method based on the rank revealing QR decomposition (RRQR) [35] here.

Consider $X = ZZ^T$ where $Z \in \mathbb{R}^{n \times r_c}$ and the numerical rank $\operatorname{rank}(Z, \tau) = r < r_c$. We compute the RRQR of $Z^T = QR\Pi$, where

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} \quad \text{and } R_{11} \in \mathbb{R}^{r \times r}. \tag{4.20}$$

This enables us to set $\tilde{Z}^T = [R_{11}\ R_{12}]\,\Pi^T$ and find that then $\tilde{Z}\tilde{Z}^T =: \tilde{X} \approx X$. We especially emphasize that we do not even have to accumulate Q during the computation and use it in the definition of $\tilde{Z}$, since it would cancel out in the product $\tilde{Z}\tilde{Z}^T$ anyway, due to its orthogonality. Note that for $\tau = 0$ we have $\operatorname{rank}(Z) = r$. Hence $R_{22} = 0$ and we find $\tilde{X} = X$.

Algorithm 4.3 Galerkin Projection accelerated LRCF-ADI (LRCF-ADI-GP)

Input: F,G defining $FX + XF^T = -GG^T$ and shift parameters $\{p_1, \ldots, p_{i_{max}}\}$
Output: $Z = Z_{i_{max}} \in \mathbb{C}^{n \times t_{i_{max}}}$, such that $ZZ^H \approx X$

1: $V_1 = \sqrt{-2\operatorname{Re}(p_1)}(F + p_1 I)^{-1} G$
2: $Z_1 = V_1$
3: **for** $i = 2, 3, \ldots, i_{max}$ **do**
4: $\quad V_i = \sqrt{\operatorname{Re}(p_i)/\operatorname{Re}(p_{i-1})}(V_{i-1} - (p_i + \overline{p_{i-1}})(F + p_i I)^{-1} V_{i-1})$
5: $\quad Z_i = [Z_{i-1} \;\; V_i]$
6: $\quad$ Orthogonalize the columns of Z_i, e.g., U_i =`mgs`(Z_i), or $[U_i, R_i, \Pi_i]$ =`qr`$(Z, 0)$
7: $\quad F_i = U_i^T F U_i$, $G_i = U_i^T G$
8: $\quad$ Solve $F_i Y_i + Y_i F_i^T = -G_i G_i^T$ exactly for R_i with $Y_i = R_i^T R_i$.
9: $\quad Z_i = U_i R_i$
10: $\quad$ Update V_i from the last column block of Z_i.
11: **end for**

In practical implementations the rank decision has to be performed on the basis of the truncation tolerance τ in the RRQR. Benner and Quitana-Ortí [23, equation (1.25)] noted that a truncation tolerance of $\sqrt{u}$, where u is the machine precision, is sufficient to achieve an error of the magnitude of the machine precision for the solution $X = ZZ^T$ of the corresponding Lyapunov equation. This is sufficient inside algorithms that at least implicitly form the LRCFP, like in Algorithm 4.7 or Algorithm 4.8. Note however that, e.g., in the LR-SRM (see, e.g., Algorithm 7.1) used for balanced truncation MOR, the LRCF directly enters the subsequent computations and therefore a smaller truncation tolerance (like the machine precision itself) must be employed in the column compression.

4.4.2. Hybrid Krylov-ADI Solvers for the Lyapunov Equation

A Galerkin Projection Based Acceleration Technique

Krylov subspace methods for solving large Lyapunov equations are based on the equally named paper [80] by Jaimoukha and Kasenally. There the basic idea is to consider the Schur decomposition $X = U\Sigma U^T$ of the solution X, where $U \in \mathbb{R}^{n \times n}$ is an orthogonal matrix and $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$ is diagonal, due to the symmetry of X. Further the eigenvalues are considered to be ordered such that $|\sigma_1| \geq |\sigma_2| \geq \cdots \geq |\sigma_n|$. Therefore it coincides with the SVD of X. The best rank-m Frobenius-norm approximation is thus given by

$$X_m := U \begin{bmatrix} \Sigma_m & 0 \\ 0 & 0 \end{bmatrix} U^T = U_m \Sigma_m U_m^T. \tag{4.21}$$

Here $\Sigma_m = \operatorname{diag}(\sigma_1, \ldots, \sigma_m)$ and $U_m \in \mathbb{R}^{n \times m}$ consists of the first m columns of U. The basic idea now is to compute X_m via the solution of a projected version of the Lyapunov

equation (4.2)

$$(U_m^T F U_m) Y_m + Y_m (U_m^T F^T U_m) = -U_m^T G G^T U_m, \tag{4.22}$$

on $\mathcal{U}_m$, the column span of U_m, and define X_m as

$$X_m = U_m Y_m U_m^T. \tag{4.23}$$

The basic projection idea described in the above has already been considered by Saad [126] 4 years earlier for general subspaces $\mathcal{U}_m$ to project on. The main concern in [80] and related Krylov subspace methods (e.g., [75, 131, 82, 81, 72]) is then to find or compute an orthogonal basis of a good (Krylov) subspace approximating U_m. The most promising among these methods seems to be the recently proposed Krylov-plus-inverse-Krylov (KPIK) Method by Simoncini [131], which uses a rational Krylov subspace for the projection. A convergence analysis for the projection based solvers is carried out in [132].

We employ the same projection idea here, but replace the critical Krylov subspace by the column span of the current ADI iterate Z_i in the i-th iteration step. Since we need to preserve the Galerkin type approximation features, the orthogonality of the projection is crucial. We are interested in the subspace spanned by Z_i, therefore we compute a QR decomposition

$$Z_i =: Q_i R_i \tag{4.24}$$

and use Q_i for the projection in equation (4.22)

$$(Q_i^T F Q_i) Y_i + Y_i (Q_i^T F^T Q_i) = -Q_i^T B B^T Q_i. \tag{4.25}$$

Now computing a Cholesky factorization of $Y_i = \tilde{R}_i^T \tilde{R}_i$ we define the optimization $\tilde{Z}_i$ of Z_i on the current subspace via

$$\tilde{Z}_i := Q_i \tilde{R}_i. \tag{4.26}$$

We emphasize, that Hammarling's method [67], as well as the sign function method [22] can directly compute the Cholesky factor $\tilde{R}_i$. Note that dropping the original R_i completely is no problem at all, since our primary concern is the subspace information contained in the orthogonal columns of Q_i only. Note however, that in cases where Z_i does not have full rank the column space of a standard QR-decomposition does not necessarily coincide with the column space of Z_i (see [57, Section 5.4]). In those cases it is crucial to use QR with column pivoting to ensure the equality of the column spaces.

A similar method has also been proposed for the more general Silvester equation case in [19]. We have to keep in mind, that the Krylov projection based methods do not work for general Lyapunov equations. They are only applicable for those equations where the projected equation (e.g. (4.22)) remains stable, i.e., the Matrix $U_m^T F U_m$ is asymptotically stable. It can be shown using Bendixson's theorem [103], that $F + F^T < 0$ is sufficient to get $\mathrm{Re}(\lambda) < 0$ for all $\lambda \in \Lambda(U^T F U)$ if $U \in \mathbb{R}^{n \times r}$. This is clearly a restriction, but it holds for any dissipative operator. Dissipativity of the operator is a rather basic

assumption, that has to be made, e.g., in many control problems for parabolic PDEs anyway. The condition $F + F^T < 0$ however can be crucial. For the artificial test matrix from Section 3.2 this condition is not fulfilled. It is easy to check that for this example we have $F + F^T = A + A^T = \text{diag}$[-0.02 0.002 -0.4 -0.2 -0.04 0 -0.02 -0.02 -2:-2:-400] in MATLAB notation. Comparing LRCF-ADI and LRCF-ADI-GP shows, that the Galerkin projection causes the ADI iteration to stagnate for this example. The integration of the above acceleration into LRCF-ADI gives the Galerkin projection accelerated LRCF-ADI (LRCF-ADI-GP) presented in Algorithm 4.3. Note that the last step in the loop is rather arbitrary, since we can not tell which columns are the ones "belonging" to V_1 after computing the updated Z_i. That means that we are more or less free to choose any appropriate number of columns in Z_i. Here we decided to take the last columns since they will in general contain linear combinations of the largest number of orthogonal columns from U_i and thus have most subspace information saved. In this sense this should be seen as a restarted ADI iteration, since we can no longer guarantee the full structure of the factor as in (4.5), but keep as much information as possible for the restart.

Rank Deficiency and Combination With Column Compression. We have noted above, that in case of rank deficient Z_i we need to perform column pivoting while computing the QR decomposition in (4.24). We will now discuss how we can combine this approach with the column compression technique discussed in Section 4.4.1. The basic idea is to replace step 6 in Algorithm 4.3 by an orthogonalize and compress step. The naive approach would be to apply the RRQR to compress the columns first and then use the new factor to compute the orthogonal basis for the projection. Obviously this would require the relatively expensive orthogonalization twice. Since this is unacceptable especially for larger systems, we propose an alternative way, that will not avoid the two orthogonalizations, but apply one of them to a much smaller matrix.

Let $Z \in \mathbb{R}^{n \times r_c}$ and consider $\text{rank}\,(Z, \tau) = r < r_c$ as in Section 4.4.1 for a given $\tau \in \mathbb{R}$. We can now compute the "economy size" QR decomposition with column pivoting

$$Z =: Q_1 R_1 \Pi_1,$$

where $Q_1 \in \mathbb{R}^{n \times r_c}$ and $R_1 \in \mathbb{R}^{r_c \times r_c}$ and Π_1 a permutation matrix. Note that this can be done efficiently using level 3 BLAS [121] via `xGEQP3` routines in recent LAPACK implementations. Also MATLAB uses these routines when called with second input parameter 0 or three output parameters. Now assuming that $r_c \ll n$, R_1 is much smaller than Z. The numerical rank decision can therefore be performed a lot cheaper on R_1 than on Z^T. Hence computing

$$R_1 =: Q_2 R_2 \Pi_2,$$

using the RRQR as in Section 4.4.1, we have a cheap way to perform the rank decision. The final factorization then is

$$Z = Q_1 Q_2 R_2 \Pi_2 \Pi_1.$$

Defining Q as the first r columns of the product Q_1Q_2 we can now proceed as in (4.25) and (4.26) resulting in a compressed and corrected new LRCF iterate $\tilde{Z}$. Note that solving (4.25) now is even cheaper, since the subspace is smaller and so are the matrices. Also the numerical stability of the computation for (4.25) will be better, since due to the truncation the condition numbers of the projected matrices should have decreased.

Note that the numerical results for the Galerkin projection acceleration in Chapter 8 have been acquired by an experimental MATLAB implementation using `orth` to compute the orthogonal basis of span(Z), which uses an SVD approach for the rank decision and truncation. This, although more reliable in terms of the rank approximation, will in general be more time consuming than the RRQR based approach. In the present context it is more favorable to have faster execution, thus we propose the RRQR based approach for efficient implementations. The efficient RRQR based codes and tests are part of the C.M.E.S.S.-implementation and corresponding timings will be given in [84].

Krylov Subspace Interpretation. From the viewpoint of the LRCF-ADI the Galerkin projection is an acceleration technique trying to improve the quality of the iterate on the current column space. That means it can be interpreted as some kind of subspace optimization method applied to the ADI iteration. On the other hand in her thesis Jing-Rebecca Li showed [95, Corollary 1] that the column span of the Lyapunov solution is itself a special type of rational Krylov subspace. The span of its factor is then the same Krylov subspace. Following from the structure of the space and the iteration this holds for the previous iterates as well. Combining that knowledge with the idea of taking rational Krylov subspaces (as in KPIK [131]) for the projection in [80], we immediately see that Algorithm 4.3 can also be interpreted as a certain Krylov subspace projection method. It is thus in fact a hybrid Krylov-ADI-Iteration.

Avoiding the Orthogonalization. The projection method we have just introduced uses orthogonalization of the columns of Z_i to compute the orthogonal projection to the subspace. In general using orthogonal matrices for the projection is a good idea, since these are well conditioned and do not amplify errors in the computations. On the other hand in the proposed method the projection is only an accelerator for the outer iteration. So from an implementational point of view, we may earn more if we compute a slightly worse optimization in notably reduced time, i.e., we can afford to risk numerical stability issues due to non-orthogonality if we can further accelerate the computation. In the following we will therefore extend the above consideration to the case where we implement the orthogonal projection (which we cannot avoid) by Z_i itself instead of Q_i. Note however, that in practice we need to ensure that Z_i is not rank deficient, such that all the following computations are well defined and well conditioned. The general orthogonal projection onto the column space of a matrix Z is

$$P_Z := Z(Z^TZ)^{-1}Z^T.$$

In case of an orthogonal Q obviously this reduces to

$$P_Q := QQ^T,$$

which gives rise to equations (4.22), (4.25). Starting with the projection of F and G by P_Q equation (4.2) becomes

$$P_Q F P_Q^T X_m + X_m P_Q F^T P_Q^T = -P_Q G G^T P_Q^T, \tag{4.27}$$

which becomes (4.22) after multiplication by Q from the right and Q^T from the left and explains (4.23).

Now replacing P_Q with P_Z in the above we need to take an additional step. Starting at

$$P_Z F P_Z^T X_m + X_m P_Z F^T P_Z^T = -P_Z G G^T P_Z^T,$$

and inserting the definition of P_Z, after multiplication with Z^T and Z from let and right as above, we get

$$\begin{aligned} Z^T Z(Z^T Z)^{-1} Z^T F Z(Z^T Z)^{-1} Z^T X_m Z + Z^T X_m Z(Z^T Z)^{-1} Z^T F^T Z(Z^T Z)^{-1} Z^T Z \\ = -Z^T Z(Z^T Z)^{-1} Z^T G G^T Z(Z^T Z)^{-1} Z^T Z. \end{aligned}$$

Now defining $F_m := Z^T F Z$ and $G_m := Z^T G$ we find

$$\begin{aligned} F_m(Z^T Z)^{-1} Z^T X_m Z + Z^T X_m Z(Z^T Z)^{-1} F_m^T &= -G_m G_m^T \\ F_m(Z^T Z)^{-1} Z^T X_m Z(Z^T Z)^{-T}(Z^T Z)^T + (Z^T Z)^T(Z^T Z)^{-T} Z^T X_m Z(Z^T Z)^{-1} F_m^T &= -G_m G_m^T, \end{aligned}$$

and finally

$$F_m \tilde{Y}_m E_m^T + E_m \tilde{Y}_m F_m^T = -G_m G_m^T,$$

where $E_m = Z^T Z$. Again we see that this is a direct extension of the orthogonal case since there $E_m = Q^T Q = I_m$.

Note that computing X_m from Y_m works exactly as above since

$$Y_m = (Z^T Z)^{-T} Z^T X_m Z(Z^T Z)^{-1}$$

and thus

$$Z Y_m Z^T = P_Z X_m P_z^T.$$

So at the cost of solving a generalized projected Lyapunov equation instead of the projected standard Lyapunov equation (4.25), we can avoid the orthogonalization of the current iterate completely.

Generalized Lyapunov Equations. The Galerkin projection technique can easily be extended to the case of generalized Lyapunov equations of the form

$$FXE^T + EXF^T = -GG^T. \tag{4.28}$$

Especially avoiding the orthogonalization can be generalized, where most of the above remains unchanged. We only have to define $E_m := Z^T EZ$. Analogously we define $E_m := Q^T EQ$ in the orthogonal case. More details on the algorithms developed for solving (4.28) can be found in Chapter 5.

KPIK Starting Guesses for the LRCF-ADI

Up to now we have always considered the LRCF-ADI to use an empty or all zero matrix as initial value for the iteration. In the first paragraph we will discuss that both the basic low-rank ADI iteration as introduced by Penzl [116] and the Li/White extension [97] allow for the use of a non-zero initial guess. The second paragraph then describes how to compute those initial values efficiently in the SISO case. We show there that after minor changes to the present behaviour we can calculate them from the data that we need to compute anyway.

Non-Zero Starting Guesses for the LRCF-ADI. The key observation for Penzls derivation of the low-rank ADI iteration is the following. After rewriting the two step ADI process as a one step iteration by inserting one equation into the other and replacing the iterate X_j by its factorization $X_j = Z_j Z_j^T$, he ended up with:

$$\begin{aligned} Z_0 &= 0, \\ Z_1 &= \sqrt{-2p_1}(F + p_1 I)^{-1} G, \\ Z_j &= [\sqrt{-2p_j}(F + p_j I)^{-1} G,\ (F + p_j I)^{-1}(F - p_j I) Z_{j-1}], \qquad j \geq 2. \end{aligned}$$

The main reasons for the $Z_0 = 0$ here were that it is the easiest (and obvious) choice and works flawlessly. It even simplifies the expression for Z_1. If we now assume that we have an admissible initial guess Z_0 given, the above changes to read

$$Z_j = [\sqrt{-2p_j}(F + p_j I)^{-1} G,\ (F + p_j I)^{-1}(F - p_j I) Z_{j-1}], \qquad j \geq 1. \tag{4.29}$$

Defining $S_i := (F + p_i I)^{-1}$ and $T_i := (F - p_i I)$, as in [97] we find that after $J = i_{max}$ ADI steps the iterate is

$$\begin{aligned} Z_J = \Big[S_J \sqrt{-2p_J} G,\ S_J(T_J S_{J-1}) \sqrt{-2p_{J-1}} G,\ \ldots,\ S_J T_J \cdots S_2 (T_2 S_1) \sqrt{-2p_1} G, \\ S_J T_J \cdots S_2 T_2 S_1 T_1 Z_0 \Big]. \end{aligned} \tag{4.30}$$

Algorithm 4.4 Low-rank Cholesky factor ADI iteration with initial guess (LRCF-ADI-S)

Input: F,G defining $FX + XF^T = -GG^T$, a starting guess Z_0 and shift parameters $\{p_1, \dots, p_{i_{max}}\}$
Output: $Z = Z_{i_{max}} \in \mathbb{C}^{n \times t_{i_{max}}}$, such that $ZZ^H \approx X$

1: $V_1 = \sqrt{-2\,\mathrm{Re}\,(p_1)}(F + p_1 I)^{-1} G$
2: $V_0 = (F - p_1 I)(F + p_1 I)^{-1} Z_0$
3: $\tilde{Z}_1 = V_1$
4: $Z_1 = [\tilde{Z}_1 \;\; V0]$
5: **for** $i = 2, 3, \dots, i_{max}$ **do**
6: $\quad V_i = \sqrt{\mathrm{Re}\,(p_i)/\,\mathrm{Re}\,(p_{i-1})}(V_{i-1} - (p_i + \overline{p_{i-1}})(F + p_i I)^{-1} V_{i-1})$
7: $\quad V_0 = V_0 - (p_i + \overline{p_{i-1}})(F + p_i I)^{-1} V_0$
8: $\quad \tilde{Z}_i = [\tilde{Z}_{i-1} \;\; V_i]$
9: $\quad Z_i = [\tilde{Z}_i \;\; V_0]$
10: **end for**

All columns arising from G (first line in (4.30)) are exactly the same as computed for $Z_0 = 0$. Therefore the only difference is the column block arising from Z_0 (second line in (4.30)). That means we can apply the standard Li/White method to the first column block as before and only need to handle the columns computed from Z_0 separately. We summarize these findings in Algorithm 4.4 and note that the acceleration techniques discussed so far can also be applied here. They should however be applied to $\tilde{Z}_i$ and V_0, as well, since comparing the last 2 block columns in (4.30) we see that the column space of V_0 is manipulated by T_1 which the part resulting from G is not. We should therefore separate the subspace information wherever possible. In the case of the column compression this is not problematic. In case of the Galerkin projection we need to apply the projection using the full factor information from Z_i including that from V_0. Unfortunately we will not be able to separate V_0 and V_1 from the resulting corrected factor. But with the same interpretation as in the case of the LRCF-ADI-GP, we can choose the corresponding last column blocks here as well.

Computing the KPIK Starting Guess in the LRCF-ADI Context. Let us restrict ourselves to the SISO system case for the moment, i.e., $G \in \mathbb{R}^n$. We have seen in Section 4.3 that in general we need to approximate the spectrum of F in order to find the shift parameters p_i for the LRCF-ADI. To do so we generally apply an Arnoldi process to F to compute some Ritz values approximating the large eigenvalues of F. For approximation of the small eigenvalues we compute some Ritz values with respect to F^{-1} and take their reciprocals as the desired approximations. Normally we start these two Arnoldi processes with an all ones or random vector b. The two Arnoldi processes are then computing orthogonal bases of the two Krylov subspaces

$$\mathcal{K}_{k_p}(F, b) \qquad \text{and} \qquad \mathcal{K}_{k_m}(F^{-1}, b).$$

On the other hand KPIK [131] forms the augmented rational Krylov space

$$\mathcal{K}_m(F, G) \cup \mathcal{K}_m(F^{-1}, G)$$

to perform the subspace projection (4.22) from the beginning of this section. If we now choose $k_p = k_m = m$ (compare Algorithm 4.2) and $b = G$ for the Arnoldi processes computing the Ritz values, we can directly reuse the orthogonal bases computed there to find the m-th KPIK approximation to the solution of (4.2). Normally this should already be a better approximation than $Z_0 = 0$ and we may save some ADI steps at essentially no additional cost. Clearly the projected Lyapunov equation and its solution need to be computed additionally, but that will generally be cheaper than the execution of several steps of the ADI process.

4.4.3. Software Engineering Aspects

In this section we will especially consider implementation details. The idea is to find the fastest way to implement ADI in different software environments. Especially we emphasize, that the straight forward way is not always the best way to implement things in the different environments.

The part of ADI that allows for the largest gain in computation time is at the same time the part we can loose most time. It is the cyclical usage of the shift parameters. When applying direct solvers, we compute the LU or Cholesky factorization of the shifted matrices and do triangular forward-backward-solves to get the solution computed. This is especially helpful in the MIMO systems case. There we would have to rerun iterative methods for every column in G. Alternatively we would need to find and apply some sort of reuse strategy capable of computing the Krylov-basis for next column in G from the one for the current one (see, e.g. [95, Section 8.8]). In case of direct solvers we can apply the triangular solves to all columns in G at the same time. Also, if we are running a cyclic ADI and we have enough memory available, we would want to save the decomposition of the shifted matrix and reuse the factors the next time the same shift is applied, i.e., in the next cycle. Doing so obviously avoids repetition of the same decomposition every time the shift is used and thus reduces the cost of the solve to the cheaper forward-backward-substitutions. Unfortunately in current MATLAB implementations the computation and storage of the factors (via `lu` or `chol`) shows to be much more expensive than doing multiple solver (\) operations for matrices of larger dimensions. Thus the full solve applying \ is much cheaper in CPU time. Opposing what simple operation counts would tell us, avoiding the pre-computation of the factors and thus repeating the decompositions in every cycle is what should currently be done in MATLAB. See Section 8.4 Table 8.18 for details.

Thinking of a C-implementation on the other hand we can perfectly exploit the existence of factors from previous cycles. Additionally, we can compute these decompositions in parallel on modern multi-processor or multi-core computers. We can further exploit multiple processors and cores especially on shared memory machines when doing the

forward backward solves for MIMO systems, since we can again work on the columns of G in parallel. Note however, that even on shared memory multi-core machines we will not observe a speedup equal to the number of cores. The problem here is that depending on the platform the different processor cores are bound to the memory via the same cache hierarchy or the main memory is partitioned such that there are parts associated to the different cores and accessing the memory of a different core is much slower than accessing the *own* memory. Such that, although no inter-processor-communications are needed, one observes very similar delay effects reducing the efficiency of the parallel computations, as on distributed memory systems. Some fortifying tests can also be found in Section 8.4 (Tables 8.13, 8.14 and 8.15).

A crucial point in memory usage also is the precomputation of the sparse LU decompositions. Here we can massively save memory when using what we call a *single-pattern–multi-value* LU. The basic idea is to apply the same pivoting strategy to F and all shifted matrices $F + p_i I$. Since then all of these matrices will have the same sparsity pattern, one can store all of them in the same structure reusing the pattern vector. Thus one can almost save half the memory for all but one of the decompositions, since in all but one cases only the values need to be stored. Additionally the decompositions can be computed faster since the pattern is already known in advance. Details and numerical tests regarding this effect can be found in 8.4.

4.5. Algebraic Riccati Equations

We consider the algebraic Riccati equation

$$0 = C^T C + A^T X + XA - XBR^{-1}B^T X =: \mathfrak{R}(X). \tag{4.31}$$

Here $A \in \mathbb{R}^{n\times n}$ is supposed to be sparse, B, C may be dense, but B should have very few columns and C only few rows compared to n.

4.5.1. Newtons Method for Algebraic Riccati Equations

The classical approach for solving the algebraic Riccati equation is to tackle its nonlinearity with a Newton type method. The basic Newton step then is

$$\mathfrak{R}'|_X(N_\ell) = -\mathfrak{R}(X_\ell), \qquad X_{\ell+1} = X_\ell + N_\ell. \tag{4.32}$$

Taking a closer look at the Frechét derivative of the Riccati operator $\mathfrak{R}$ at X we observe, that this is the Lyapunov operator

$$\mathfrak{R}'|_X : \quad N \mapsto (A - BR^{-1}B^T X)^T N + N(A - BR^{-1}B^T X). \tag{4.33}$$

This observation gives rise to Algorithm 4.5. In principle we would thus be able to use the theory from Sections 4.1- 4.4 and ensure low-rank solvability of the Riccati equation.

Algorithm 4.5 Newtons Method for Algebraic Riccati Equations – Basic Iteration

Input: A,B,$Q = C^T\hat{Q}C$,R as in (2.19) and an initial guess $X^{(0)}$ for the iterate.
Output: X_∞ solving (2.19) (or an approximation when stopped before convergence).

1: **for** $k = 1, 2, \ldots$ **do**
2: $K^{(k-1)} = X^{(k-1)}BR^{-1}$.
3: Determine the solution $N^{(k)}$ of
4: $(A^T - K^{(k-1)}B^T)N^{(k)} + N^{(k)}(A - BK^{(k-1)^T}) = -\mathfrak{R}(X^{(k-1)})$.
5: $X^{(k)} = X^{(k-1)} + N^{(k)}$.
6: **end for**

Algorithm 4.6 Newtons Method for Algebraic Riccati Equations – Kleinman Iteration

Input: A,B,$Q = C^T\hat{Q}C$,R as in (2.19) and an initial guess $K^{(0)}$ for the feedback.
Output: X_∞ solving (2.19) and the optimal state feedback K_∞ (or approximations when stopped before convergence).

1: **for** $k = 1, 2, \ldots$ **do**
2: Determine the solution $X^{(k)}$ of
3: $(A^T - K^{(k-1)}B^T)X^{(k)} + X^{(k)}(A - BK^{(k-1)^T}) = -Q - K^{(k-1)}RK^{(k-1)^T}$.
4: $K^{(k)} = X^{(k)}BR^{-1}$.
5: **end for**

Unfortunately the residual of the previous iterate on the right hand side is in general an indefinite, full rank matrix. Therefore we cannot guarantee the low-rank structure of the right hand side, which was crucial in the derivation of the LRCF-ADI. The representation of Newtons method proposed by Kleinman [83] on the other hand is mathematically equivalent to the basic Newton iteration and does not have this disadvantage. In [18] a detailed discussion of the advantages of Kleinmans formulation (given in Algorithm 4.6) in the present case can be found. The following theorem provides conditions ensuring the convergence of both methods.

Theorem 4.1 (Convergence to Unique Stabilizing Solution):

If the system (A, B) is stabilizable, then choosing $X^{(0)} = X^{(0)^T} \in \mathbb{R}^{n\times n}$ in Algorithm 4.5 or $K^{(0)} \in \mathbb{R}^{n\times m}$ in Algorithm 4.6 such that $A - BK^{(0)^T}$ is stable, the iterates $X^{(k)}$ and $K^{(k)}$ satisfy the following assertions:

a) For all $k \geq 0$, the matrix $A - BK^{(k)^T}$ is stable and the Lyapunov equations in Algorithms 4.5 and 4.6 admit unique solutions which are positive semidefinite.

b) $\left\{X^{(k)}\right\}_{k=1}^{\infty}$ is a non-increasing sequence satisfying $X^{(k)} \geq X^{(k+1)} \geq 0$ for all $k \geq 1$. Moreover, $X_\infty = \lim_{k\to\infty} X^{(k)}$ exists and is the unique stabilizing solution of the CARE (2.19).

c) There exists a constant $\gamma > 0$ such that

$$\|X^{(k+1)} - X_\infty\| \leq \gamma \|X^{(k)} - X_\infty\|^2, \qquad k \geq 1,$$

i.e., the $X^{(k)}$ converge globally quadratic to X_∞ from any stabilizing initial guess. ◊

A complete proof for the above result can be found, e.g., in [89]. The task of computing the stabilizing initial guesses $X^{(0)}$ and feedbacks $K^{(0)}$ is a numerically challenging task itself. For dense problems (partial) stabilization methods based on pole placement or solving certain Lyapunov equations have been existing for years now. Their extension to the sparse / low-rank case is considered in [4, 52, 122, 11].

In the following we will sketch how the special structure of the closed loop operators in the Lyapunov equations for every Newton step fit into the LRCF-ADI framework of the previous sections. We can then use the low-rank framework for the inner ADI iteration to derive the low rank Cholesky factor Newton method (LRCF-NM) for the ARE summarized in Algorithm 4.7.

Definition 4.2 (splr):
A matrix $F \in \mathbb{R}^{n\times n}$ is called *sparse plus low-rank* or simply *splr* if we can find matrices $A \in \mathbb{R}^{n\times n}$ and $U, V \in \mathbb{R}^{n\times p}$ such that

$$F = A + UV^T. \qquad \text{◊}$$

Let $F = A - BK^{(k-1)^T}$ be the current closed loop operator in both the Newton and Kleinman-Newton iteration. Obviously F is splr. Now recalling which operations are needed in the LRCF-ADI we see that these can be computed in low-rank fashion. For the shift parameter computation we need to multiply and solve with F and in the ADI-step we need to solve a shifted system with F. First we note, that with F also $F + pI$ for a scalar p is splr, since we can simply replace A with $\tilde{A} = A + pI$ to match the definition. For solving the linear systems with the splr matrices we can now apply the Sherman-Morrison-Woodburry formula[1] (e.g. [57])

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^TA^{-1}U)^{-1}V^TA^{-1}. \tag{4.34}$$

Convergence to the Stabilizing Solution. In general the ARE (2.19) has a wide range of solutions besides the stabilizing one we are searching for. As has already been discussed in [18], we cannot naturally expect that the LRCF-NM will always compute the stabilizing solution, since we do no longer solve the Lyapunov equations in each Newton step exactly, but approximate the solution by low-rank factors generated by an iterative process. [18] also explains that by an additional iteration step we can easily

[1] This formula is often referred to as *matrix inversion lemma* in the engineering literature, as well.

Algorithm 4.7 Low-Rank Cholesky Factor Newton Method (LRCF-NM)

Input: $A, B, C, \hat{Q}, R, K^{(0)}$ for which $A - BK^{(0)^T}$ is stable (e.g., $K^{(0)} = 0$ if A is stable)
Output: $Z = Z^{(k_{max})}$, such that ZZ^H approximates the solution X of the CARE (2.19) where $Q = C^T\hat{Q}C$.

1: **for** $k = 1, 2, \ldots, k_{max}$ **do**
2: Determine (sub)optimal ADI shift parameters $p_1^{(k)}, p_2^{(k)}, \ldots$ with respect to the matrix $F^{(k)} = A^T - K^{(k-1)}B^T$ (e.g., [116, Algorithm 1], Algorithm 4.2).
3: $G^{(k)} = \begin{bmatrix} C^T\tilde{Q} & K^{(k-1)}\tilde{R} \end{bmatrix}$
4: Compute matrix $Z^{(k)}$ by Algorithm 4.1 or 4.3, such that the LRCFP $Z^{(k)}Z^{(k)^H}$ approximates the solution of $F^{(k)}X^{(k)} + X^{(k)}F^{(k)^T} = -G^{(k)}G^{(k)^T}$.
5: $K^{(k)} = Z^{(k)}(Z^{(k)^H}BR^{-1})$
6: **end for**

check admissibility of the final iterate. Recently [50] applied inexact Newton approaches to the ARE. There operator inequalities for the residuals of the Lyapunov solutions in Algorithm 4.7 have been derived ensuring the convergence of the outer iteration in a sequence of stabilizing iterates.

4.5.2. Efficient Computation of Feedback Gain Matrices

The LRCFs in the inner loop for solving will grow in the progress of the iteration. Therefore the iteration is becoming more and more expensive and the computation cost of the feedback calculation from the final LRCF in each (outer) Newton step is growing as well. If we are solving the Riccati equation in an LQR context to compute the feedback gain matrix, it is very desirable to iterate on the fixed size feedback itself, since doing so we can overcome the above problems. It was noted in [11] and [117] that this can easily be achieved by rewriting the Newton-Kleinman step.

In the LRCF-NM (Algorithm 4.7) we are updating the feedback iterate regarding

$$K_k = R^{-1}B^TX_k, \tag{4.35}$$

where X_k is the solution N to the Lyapunov equation

$$(A - BR^{-1}B^TX)^TN + N(A - BR^{-1}B^TX) = -C^T\hat{Q}C - K_{k-1}^TR^{-1}K_{k-1}, \tag{4.36}$$

as in Algorithm 4.6. Note that the Algorithm 4.7 is formulated for K^T rather than K. The basic idea in [11] now is to rewrite the Newton iteration such that it works on the difference $Y_k = X_{k-1} - X_k$. Defining $L_k := K_k - K_{k-1}$ then Y_k can be determined as the solution N of

$$(A - BR^{-1}B^TX)^TN + N(A - BR^{-1}B^TX) = -L_k^TL_k.$$

Algorithm 4.8 Implicit Low-Rank Cholesky Factor Newton Method (LRCF-NM-I)

Input: A, B, C, $\hat{Q}$, R, $K^{(0)}$ for which $A - BK^{(0)^T}$ is stable (e.g., $K^{(0)} = 0$ if A is stable)
Output: $K^{(k_{max})}$ approximating $K = ZZ^H BR^{-1}$ the transpose of the optimal feedback as, e.g., in (2.32).

1: Compute $\tilde{R}$, $\tilde{Q}$, such that $R = \tilde{R}\tilde{R}^T$, $\hat{Q} = \tilde{Q}\tilde{Q}^T$
2: **for** $k = 1, 2, \ldots, k_{max}$ **do**
3: Determine (sub)optimal ADI shift parameters $p_1^{(k)}, p_2^{(k)}, \ldots$ with respect to the matrix $F^{(k)} = A^T - K^{(k-1)}B^T$ (e.g., [116, Algorithm 1], Algorithm 4.2).
4: $G^{(k)} = \begin{bmatrix} C^T\tilde{Q} & K^{(k-1)}\tilde{R} \end{bmatrix}$
5: $V_1^{(k)} = \sqrt{-2\,\mathrm{Re}\,(p_1^{(k)})}(F^{(k)} + p_1^{(k)}I)^{-1}G^{(k)}$
6: $K_1^{(k)} = 0$
7: **for** $i = 2, 3, \ldots, i_{max}^{(k)}$ **do**
8: $V_i^{(k)} = \sqrt{\frac{\mathrm{Re}\,(p_i^{(k)})}{\mathrm{Re}\,(p_{i-1}^{(k)})}}(V_{i-1}^{(k)} - \left(p_i^{(k)} + \overline{p_{i-1}^{(k)}}\right)\left((F^{(k)} + p_i^{(k)}I)^{-1}V_{i-1}^{(k)}\right)$
9: $K_i^{(k)} = K_{i-1}^{(k)} + V_i^{(k)}(V_i^{(k)^H}BR^{-1})$
10: **end for**
11: $K^{(k)} = K_{i_{max}^{(k)}}^{(k)}$
12: **end for**

Although providing us with a right hand side of fairly low-rank, this method is numerically not recommendable. The minor drawback is that due to the difference formulation we need an additional iterate to start the iteration. The authors provide a way to get that in [11]. The major drawback however is that this cheap formulation unfortunately shows robustness problems in practice. [50] provides an explanation for this issue in terms of an inexact Newton interpretation. There the authors observe that the residuals of the single steps accumulate during the course of the iteration for this formulation.

We will concentrate on the implicit LRCF-NM as introduced in, e.g., [117] and presented in Algorithm 4.8. The algorithm avoids the difference formulation. Thus it acts on (4.36) directly. However in contrast to Algorithms 4.6 and 4.7 it does not explicitly solve the Lyapunov equation to compute K_k from its solution. It rather exploits the special structure of the iterates in Algorithm 4.1 to find that

$$K_k = lim_{i\to\infty} K_k^{(i)}$$

and thus the inner iteration can be rewritten to implement

$$K_k^{(i)} := R^{-1}B^T Z_k^{(i)} Z_k^{(i)^H} = R^{-1}B^T \sum_{j=1}^{i} V_k^{(j)} V_k^{(j)^H}.$$

Obviously we can then completely avoid forming $Z_k^{(i)}$ during the inner iteration. On the other hand the stopping criteria for the inner iteration need $Z_k^{(i)}$ to compute relative

change or residual norms. These criteria can easily be replaced by

$$\frac{\|K_k - K_{k-1}\|_F}{\|K_k\|_F} < \varepsilon,$$

though. This criterion is very cheap in evaluation if the underlying control system has very few inputs, i.e., $K \in \mathbb{R}^{n\times m}$ where $m \ll n$.

4.5.3. Modified Variants of the LRCF-NM

We will now discuss a handful of modifications of the LRCF-NM that may have desirable properties in one or the other context.

An Inexact Newton Iteration. An interesting, though at the current state not very useful, modification has been discussed in [50]. The authors there interprete the LRCF-NM in the context of an inexact Newton approach and derive a measure for the accuracy that is needed for the ADI process in each Newton step such that the outer Newton's method still converges. The only flaw of this excellent work is, that the accuracy bound is presented in terms of an operator/matrix inequality

$$R_k < C^T C,$$

i.e., a matrix needs to be tested for positive definiteness in each iteration step and it is currently unclear how this can be done in at most super-linear complexity.

Simplified Newton Approaches. An alternative idea would be to use a simplified Newton iteration to accelerate the computations. The idea there is to fix the Jacobian to that of the first step or at least not update it very frequently. The main computational work in the application of the Lyapunov operator are the matrix decompositions. Due to the Sherman-Morrison-Woodburry trick we apply, these decompositions are independent of the feedback part. On the other hand that is the only part updated in the course of the Newton iteration. Thus the main savings resulting from the simplified Newton approach are regarding the shift parameter computation, that needs to be reapplied in every Newton step. Therefore a mixed variant might be more useful, that fixes the shifts from the first Newton step for the whole iteration avoiding the expensive approximations of the current spectrum. On the other hand the cheaper updates of the feedback should still be performed in every Newton step. A further variant of this type would be to use the shifts with respect to the optimal closed loop operator right from the beginning. These can be computed via the Hamiltonian eigenvalue problem for

$$\mathcal{H} = \begin{bmatrix} A & -BB^T \\ -C^T C & -A^T \end{bmatrix} \tag{4.37}$$

via specialized Lanczos algorithms as in [16, 51].

Algorithm 4.9 Quadratic Alternating Directions Implicit Iteration for the Algebraic Riccati Equation (QADI)

Input: $A, B, Q = C^T\hat{Q}C, R$ as in (2.19), a set of shift parameters as in Algorithm 4.1.
Output: X_∞ solving (2.19) (or an approximation when stopped before convergence).
1: **for** $j = 1, 2, \ldots$ **do**
2: $(A^T - X_{j-1}^T BR^{-1}B^T + p_j I)X_{j-\frac{1}{2}}^T = -Q - X_{j-1}^T(A - p_j I)$
3: $(A^T - X_{j-\frac{1}{2}} BR^{-1}B^T + p_j I)X_j = -Q - X_{j-\frac{1}{2}}(A - p_j I)$
4: **end for**

Newton Galerkin Methods It is crucial to state that the LRCF-NM can be combined with all modifications of ADI (like, e.g., column compression, subspace projection ...) in the inner iteration. Test examples for the acceleration of the LRCF-NM via Galerkin projection in the inner loop are discussed in Section 8.2.2. Note that the Galerkin Projection idea can also be incorporated in the outer loop, since the ARE is symmetric as well and allows for the same projection technique to be applied as in the Lyapunov equation case. Here a smaller dense ARE is solved on the column space of the current LRCF.

4.5.4. The Relationship of LRCF-NM and the QADI Iteration

Starting from a Newton-Smith approach [151], Wong and Balakrishnan in a series of conference papers [148, 147, 149] developed a version of the ADI iteration that directly applies to the ARE instead of using a combination of an outer Newton method and an inner Lyapunov-solver. This section is dedicated to showing a connection between the two methods, hopefully giving a better insight to the one or the other. A summary of the QADI work can be found in [152, 150].

The term *quadratic* in the context of quadratic ADI should rather be understood as a naming scheme reflecting the quadratic nature of the Riccati equation it is applied to, than anything else. The authors do not claim to have a quadratically converging method or the like. In fact they state [147] that they expect (super-)linear convergence due to the close relationship to the ADI iteration for the Lyapunov equation.

The quadratic ADI iteration is a generalization of the ADI iteration for Lyapunov equations, as stated in (4.1). Algorithm 4.9 provides the basic QADI iteration. Note that just as in Algorithms 4.7 and 4.8 the matrices on the left hand sides of lines 2 and 3 in Algorithm 4.9 are splr and thus can be handled using the Sherman-Morrison-Woodburry formula (4.34).

For the following discussion for any $k \in \mathbb{Z}$ we define

$$K_k := R^{-1}B^T X_k \tag{4.38}$$

analogously to (4.35). Then using the appropriate iterates X_j from Algorithm 4.9 we can

rewrite the defining equations of the algorithm as

$$(A^T - K_{j-1}^T B^T + p_j I)X_{j-\frac{1}{2}}^T = -Q - X_{j-1}^T(A - p_j I),$$
$$(A^T - K_{j-\frac{1}{2}}^T B^T + p_j I)X_j = -Q - X_{j-\frac{1}{2}}(A - p_j I).$$

If we now add a zero to acquire the closed loop splr matrix on the right hand side as well, we end up with

$$\begin{aligned}(A^T - K_{j-1}^T B^T + p_j I)X_{j-\frac{1}{2}}^T &= -Q - K_{j-1}^T R K_{j-1} - X_{j-1}^T(A - BK_{j-1} - p_j I),\\ (A^T - K_{j-\frac{1}{2}}^T B^T + p_j I)X_j &= -Q - K_{j-\frac{1}{2}}^T R K_{j-\frac{1}{2}} - X_{j-\frac{1}{2}}(A - BK_{j-\frac{1}{2}} - p_j I).\end{aligned} \tag{QADI}$$

Now we consider these as opposed to the equations resulting from the Newton-Kleinman-ADI iteration as it is used in Algorithm 4.7. For the Newton-Kleinman-ADI iteration we have to distinguish the outer iteration index j of the Newton-Kleinman iteration and the inner index k of the ADI iteration.

$$\begin{aligned}(A^T - K_{j-1}^T B^T + p_k I)X_{k-\frac{1}{2}}^T &= -Q - K_{j-1}^T R K_{j-1} - X_{k-1}^T(A - BK_{j-1} - p_k I),\\ (A^T - K_{j-1}^T B^T + p_k I)X_k &= -Q - K_{j-1}^T R K_{j-1} - X_{k-\frac{1}{2}}(A - BK_{j-1} - p_k I).\end{aligned} \tag{NK-ADI}$$

Comparing (QADI) and (NK-ADI) we immediately find two main differences. First (QADI) does not distinguish between inner and outer iterations at all and second the solution to the first equation is used to update all data in the second equation immediately. In contrast to this in (NK-ADI) the closed loop matrices $A-BK_{j-1}$ are only updated at the beginning of each (outer) Newton step. In this sense we can interprete QADI as a maximally updated Newton-Kleinman-ADI iteration. In other words; if we interpret NK-ADI as a full step method in the sense of a Jacobi-type-iteration, then QADI follows the Gauß-Seidel-idea to take as much updated information into account a possible.

4.5.5. Does CFQADI Allow Low-Rank Factor Computations?

Besides the two step QADI iteration Wong and Balakrishnan present a version of the iteration that explicitly computes the j-th iterate as

$$X_j = M_{11} + M_{12}X_{j-1}(I - M_{22}X_{j-1})^{-1}M_{12}^T, \tag{4.39}$$

where, as in [97], $S_j = (A + p_j I)^{-1}$ and

$$\begin{aligned}M_{11} &= -2p_j S_j^T C^T(I + CS_j BB^T S_j^T C^T)^{-1}CS_j,\\ M_{12} &= I - 2p_j S_j^T(I + C^T CS_j BB^T S_j^T)^{-1},\\ M_{22} &= 2p_j S_j B(I + B^T S_j^T C^T CS_j B)^{-1}B^T S_j^T.\end{aligned} \tag{4.40}$$

The authors further argue that all inverses in the above exist under practical assumptions on the Riccati equation. It is easy to see that symmetry is preserved in this iteration and

thus starting from $X_0 = 0$ all iterates will be symmetric as expected. Note that inverses in M_{11} and M_{22} are small under the assumption that the underlying system has only very few inputs and outputs. Note further that the large inverse in M_{12} is splr and thus can be solved exploiting (4.34). So at least the intermediate computations can be kept cheap. Unfortunately X will in general be full such that the algorithm needs to be reformulated in low-rank fashion to be applicable in large scale. For this purpose a Cholesky factor variant is proposed by Wong and Balakrishnan, e.g., in [148]. Sadly they do not give a derivation for it and we do not see a way to assure the apparent derivation in low-rank fashion. We will next state the factorized variant and demonstrate the concerns we find in it. Keeping M_{12} and M_{22} as above and defining

$$\tilde{M}_{11} = \sqrt{-2p_j} S_j^T C^T (I + CS_j BB^T S_j^T C^T)^{-\frac{1}{2}},$$

the new factor is proposed to be

$$Z_j := [\tilde{M}_{11} \quad M_{12} Z_{j-1} (I - Z_{j-1}^T M_{22} Z_{j-1})^{-\frac{1}{2}}], \tag{4.41}$$

assuming that we have the definiteness to ensure the existence of all matrix square roots used. Note that in comparison to [148] we changed the misleading name $M_{11}^{\frac{1}{2}}$ used by the original authors to $\tilde{M}_{11}$, since this is not the matrix square root of M_{11} in the sense of the definition. Now trying to recompute X_j from Z_j we find

$$\begin{aligned}
X_j &= Z_j Z_j^T \\
&= [\tilde{M}_{11} \quad M_{12} Z_{j-1} (I - Z_{j-1}^T M_{22} Z_{j-1})^{-\frac{1}{2}}][\tilde{M}_{11} \quad M_{12} Z_{j-1} (I - Z_{j-1}^T M_{22} Z_{j-1})^{-\frac{1}{2}}]^T \\
&= \tilde{M}_{11} \tilde{M}_{11}^T + M_{12} Z_{j-1} (I - Z_{j-1}^T M_{22} Z_{j-1})^{-\frac{1}{2}} (I - Z_{j-1}^T M_{22} Z_{j-1})^{-\frac{T}{2}} Z_{j-1}^T M_{12}^T \\
&= M_{11} + M_{12} Z_{j-1} (I - Z_{j-1}^T M_{22} Z_{j-1})^{-1} Z_{j-1}^T M_{12}^T.
\end{aligned}$$

Assume that the factors Z_j are not rank deficient. Then $Z_{j-1}^T Z_{j-1}$ is invertible and we can compute

$$\begin{aligned}
Z_{j-1}(I - Z_{j-1}^T M_{22} Z_{j-1})^{-1} Z_{j-1}^T &= Z_{j-1}(Z_{j-1}^T Z_{j-1}(Z_{j-1}^T Z_{j-1})^{-1} - Z_{j-1}^T M_{22} Z_{j-1})^{-1} Z_{j-1}^T \\
&= Z_{j-1} Z_{j-1}^T Z_{j-1} (Z_{j-1}^T Z_{j-1} - Z_{j-1}^T M_{22} Z_{j-1} Z_{j-1}^T Z_{j-1})^{-1} Z_{j-1}^T \\
&= X_{j-1} Z_{j-1} (Z_{j-1}^T (I - M_{22} X_{j-1}) Z_{j-1})^{-1} Z_{j-1}^T
\end{aligned}$$

Now we immediately see that for full rank Cholesky factors and square root matrices Z_{j-1} that allow forming of Z_{j-1}^{-1} the last line is easily rewritten in the form $X_{j-1}(I - M_{22} X_{j-1})^{-1}$ as needed in (4.39). In a low-rank setting however this is prohibitive for the general application of this type of factorized iteration, since it is at least not obvious that the kernel of Z_{j-1} is avoided.

4.6. Stopping Criteria

All algorithms in this chapter have been formulated as if the number of iteration steps to achieve a certain accuracy was a priori known. However, e.g., for the ADI iteration this can only be ensured for the very special case of the Wachspress parameters and even there it is only known for real spectra. Further, also the Wachspress parameters are very sensitive to round off errors, such that even for real spectra the accuracy can not be guaranteed on the computer. Instead the iterations have to be stopped ad hoc. The crucial question then is how to determine good stopping criteria.

Relative Change Based Criteria. In [18] the authors propose the use of relative change criteria. There the change of the current LRCF (for the ADI variants and LRCF-NM) or the feedback operator (in case of LRCF-NM-I) is checked and the iteration is stopped as soon as the contribution of the change in the iterate is small compared to the norm of the current iterate, i.e., expressions like

$$\frac{\|Z_i - Z_{i-1}\|_F}{\|Z_i\|_F} < \varepsilon, \tag{4.42}$$

or

$$\frac{\|K_i - K_{i-1}\|_F}{\|K_i\|_F} < \varepsilon, \tag{4.43}$$

need to be evaluated for a certain stopping tolerance ε. For feedbacks K_i this is clearly cheap in evaluation, since the feedback and all its iterates are supposed to be thin rectangular matrices containing only $\mathcal{O}(n)$ entries even when densely populated. For the LCRF-ADI variants we find that it can be evaluated cheaply as well. The difference between two consecutive factors Z_i and Z_{i+1} following the LRCF-ADI methods is the new column block V_i (compare, e.g., Algorithm 4.1). Thus the numerator in (4.42) is just $\|V_i\|_F$. Further the Frobenius-norm of the factor can be accumulated via

$$\|Z_i\|_F^2 = \|Z_{i-1}\|_F^2 + \|V_i\|_F^2,$$

such that in every step only $\|V_i\|_F$ needs to be computed where again V_i is a thin rectangular matrix that has the same dimensions as B.

Residual Based Criteria. In practical computations however one observes (see, e.g., the tables in Section 8.2.2), that the residual of the current ADI-factor is already much better than the estimation based on the relative change of the factor. Therefore, especially in the inner iterations of Newton's methods, it is desirable to evaluate the residual norm based criteria

$$\|FZ_iZ_i^T + Z_iZ_i^TF^T + GG^T\| < \varepsilon \quad \text{and} \quad \|\Re(Z_iZ_i^T)\| < \varepsilon, \tag{4.44}$$

for a given tolerance ε, as well and stop the iteration after as few steps as possible. Unfortunately direct computation of the Frobenius-norm of the residual would require

forming the residual explicitly and thus lead to unacceptable (i.e., quadratic) memory and computation demands.

Penzl [116, equation (4.7)] provides a cheaper way of computation that avoids the explicit forming of the residual matrix, but still the computation is fairly complex and at a certain point more expensive than the iteration itself. Therefore we propose to *estimate* the residual norm instead.

The key observation to a cheap approximation of the residual norm is that the residual matrices for the Lyapunov and Riccati equations are both symmetric due to the symmetry of the Lyapunov and Riccati operators. Now we recall that the 2-norm of a symmetric matrix coincides with its spectral radius (e.g. [57, Section 2.3.3]), i.e., the absolute value of its largest magnitude eigenvalue and thus can be estimated by the power iteration (as also done in `normest` in MATLAB). Exploiting the structure of the operators, which are formed from sparse, or splr matrices and low-rank factors, we can perform the power iteration in $\mathcal{O}(n)$ complexity as long as the LRCF have only very few (i.e., $\ll n$) columns. Thus here we can decrease the complexity again when applying column compression to the LRCFs.

Note that in the case of the LRCF-NM one may be able to save many ADI steps by adapting the inner stopping tolerance to the outer residual error following the inexact Newton idea in [50]. Unfortunately the bound that guarantees the convergence of the inexact Newton method is an operator inequality limiting the current residual operator by the constant term in the ARE (here C^TC) from above. It is currently an open problem to evaluate this inequality in $\mathcal{O}(n)$ complexity. Thus the adaption needs to be performed heuristically.

Stagnation Based Criteria. In addition to the smallness of the above stopping criteria it can be important to apply some stagnation detection techniques. Those techniques have successfully been used in LyaPack [117]. The idea in the residual based criterion is clear. If the residual error can not be decreased we can stop the iteration. For the relative change criteria this is not so obvious. On the one hand, we can detect periodic or almost behavior of the finite arithmetic implementation caused by round off errors. On the other hand this can be helpful in the case of the usage of column compression, when we add and truncate (almost) the same information over and over again with no real progress in the iteration. Note however that the stagnation detection should incorporate more steps, than the current Galerkin projection frequency. We have observed that in some cases the projection gives a suitable reduction, but the intermediate steps may almost stagnate regarding residual error reduction. Therefore we should consider the iteration as stagnated if even the projected step does not decrease the residual, but not before.

You have brains in your head.
You have feet in your shoes.
You can steer yourself in any direction you choose.
You're on your own.
And you know what you know.
You are the guy who'll decide where to go.

Oh! The Places You'll Go!
Dr. Seuss

CHAPTER

FIVE

GENERALIZED SYSTEMS: HANDLING MASS MATRICES

Contents

In cases where the spatial semi-discretization is executed using a finite element method (FEM) a mass matrix arises in front of the time derivative. In this chapter we will discuss methods to handle this type of generalized systems

$$\begin{aligned} M_h \dot{x}_h &= N_h x_h + B_h \mathbf{u}, \\ y &= C_h x_h. \end{aligned} \tag{5.1}$$

where M_h is invertible (in the FEM case M_h usually is symmetric positive definite (spd)). Needless to say that all techniques presented in this chapter are also applicable to any other large sparse system

$$M\dot{x} = Nx + Bu \tag{5.2}$$
$$y = Cx \tag{5.3}$$

with M invertible.

The basic idea is to rewrite the generalized system in standard state space form. That means to get rid of the mass matrix by an appropriate transformation. The naive way

of doing this obviously would be to multiply by the inverse of M_h from the left. From a theoretical point of view this solves the problem completely. This was already noted in Section 2.2.2. On the other hand in numerical computations it is infeasible to compute $M_h^{-1}A_h$ since this would destroy the sparse structure of the problem and thus make computations impossible for large scale applications.

One way to avoid the full inverse is the reordering of unknowns like in sparse direct solver techniques followed by the application of an LU or Cholesky factorization. Then only one of the factors needs to be inverted (i.e., triangular solves need to be performed) and the other one is used to define a state space transformation. Additionally the reordering reduces the fill-in in the resulting matrices. Section 5.1 shows how to avoid the fill in by the decomposition technique. In Section 5.2 we will review a technique that avoids the computation of $M_h^{-1}A_h$ by exploiting matrix pencil techniques. This idea has also been given in [15] and exploited in [31]. Numerical tests have to be performed to decide whether the one or the other method is more suitable in certain applications. A comparison of different reordering strategies in the case of Section 5.1 can be found in [30].

5.1. Avoiding the Mass Matrix by Matrix Decomposition

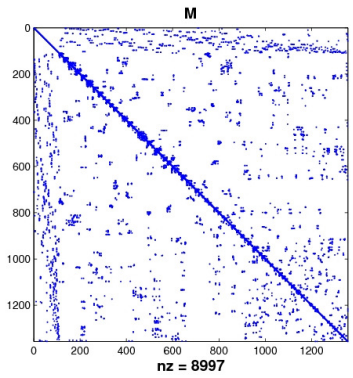
Let us first review how we can rewrite the system (5.1) in standard state space form by decomposing the mass matrix following the method, proposed by Penzl in [117]. For better readability and to reflect the general applicability of the following computations we will neglect the discretization index h for now. Consider the generalized state space system (5.2), (5.3) First we apply a reordering (e.g., approximate minimum degree [3]) to the unknowns in x to reduce the fill in in the resulting matrix factors, i.e., we perform a change of basis with the unitary permutation matrix P:

$$\begin{aligned} P^*MPP^*\dot{x} &= P^*NPP^*x + P^*Bu, \\ y &= CPP^*x. \end{aligned} \tag{5.4}$$

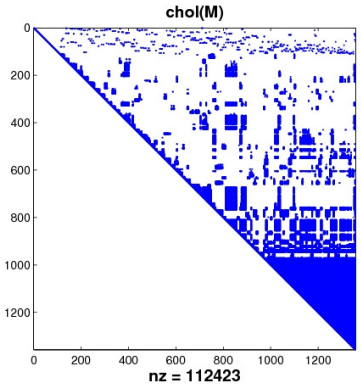
Figure 5.1 demonstrates AMD reordering in comparison to Reverse Cuthill-McKee (RCM) reordering and a non-permuted decomposition. Note especially the drastic differences in the number of non-zero elements in the resulting factors. Both AMD and RCM can have their advantages. In the case of AMD we clearly have the smallest memory demands. On the other hand the RCM reordered factor has banded structure which may be exploited by specialized banded solvers and may lead to better caching properties since one can work more locally in the data it is applied to.

Defining $\tilde{M} := P^*MP$, $\tilde{N} := P^*NP$, $\tilde{B} := P^*B$, $\tilde{C} := CP$ and $\tilde{x} := P^*x$ we end up with

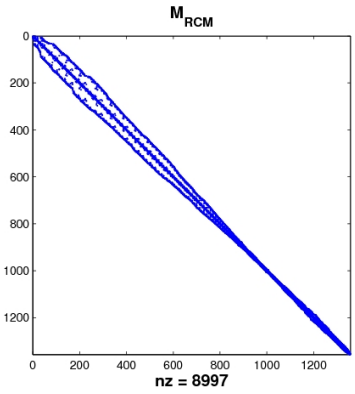
$$\begin{aligned} \tilde{M}\dot{\tilde{x}} &= \tilde{N}\tilde{x} + \tilde{B}u, \\ y &= \tilde{C}\tilde{x}. \end{aligned} \tag{5.5}$$



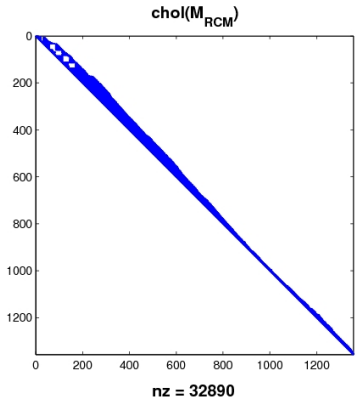
(a) original M



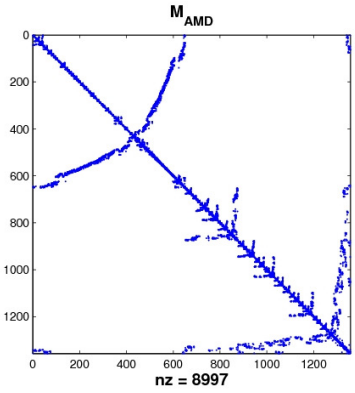
(b) Cholesky factor of M



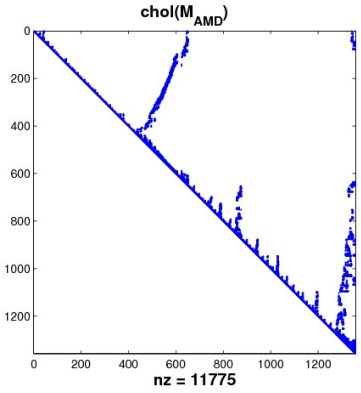
(c) M after Reverse Cuthill-McKee (RCM) reordering



(d) Cholesky factor of RCM reordered M



(e) M after Aproximate Minimum Degree (AMD) reordering



(f) Cholesky factor of AMD reordered M

Figure 5.1.: Sparsity patterns of mass matrix M and its Cholesky factors (steel profile example)

We can now more efficiently decompose the mass matrix into $\tilde{M} = LU$ (or $\tilde{M} = LL^*$ in the self-adjoint case), especially in terms of memory consumption of the factors. Although the mass matrix arising in FEM is in general selfadjoint, we will follow the LU case here for more general applicability. Multiplying by L^{-1} from the left after the decomposition we get

$$\begin{aligned} U\dot{\tilde{x}} &= L^{-1}\tilde{N}U^{-1}U\tilde{x} + L^{-1}\tilde{B}u, \\ y &= \tilde{C}U^{-1}U\tilde{x}. \end{aligned} \tag{5.6}$$

This takes the form of a standard state space system, if we now define $\hat{N} := L^{-1}\tilde{N}U^{-1}$, $\hat{B} := L^{-1}\tilde{B}$, $\hat{C} := \tilde{C}U^{-1}$ and $\hat{x} := U\tilde{x}$:

$$\begin{aligned} \dot{\hat{x}} &= \hat{N}\hat{x} + \hat{B}u, \\ y &= \hat{C}\hat{x}. \end{aligned} \tag{5.7}$$

Note that we have only changed the internal representation of the system, but not the input and output variables. The crucial question now is whether we are able to compute the solution we are interested in from the solution of this transformed system with similar complexity. That question is addressed for the solution of Riccati and Lyapunov equations in the following two section.

5.1.1. Algebraic Riccati Equations and Feedback Computations

Assuming a standard quadratic cost function (2.12) with operators Q and R, we find the associated algebraic Riccati equation

$$\begin{aligned} 0 =& \hat{C}^*Q\hat{C} + \hat{N}^*\hat{X} + \hat{X}\hat{N} - \hat{X}\hat{B}R^{-1}\hat{B}^*\hat{X} \\ =& U^{-*}P^*C^*QCPU^{-1} + U^{-*}P^*NPL^{-*}\hat{X} + \hat{X}L^{-1}P^*NPU^{-1} \\ & - \hat{X}L^{-1}P^*BR^{-1}B^*PL^{-*}\hat{X}, \end{aligned} \tag{5.8}$$

where P is the permutation matrix in (5.4), L and U the LU-factors of $\tilde{M}$ as in and above (5.6). The question now arises, how we can compute the solution X of the ARE associated with the generalized system (5.2), (5.3) from the solution $\hat{X}$ of (5.8). To see this we rewrite (5.8) in a form that allows us direct comparison to the ARE

$$0 = C^*QC + N^*XM + M^*XN - M^*XBR^{-1}B^*XM \tag{5.9}$$

for (5.2), (5.3):

$$\begin{aligned} 0 &= C^*QC + N^*PL^{-*}\hat{X}UP^* + PU^*\hat{X}L^{-1}P^*N - PU^*\hat{X}L^{-1}P^*BR^{-1}B^*PL^{-*}\hat{X}UP^{-1} \\ &= C^*QC + N^*PL^{-*}\hat{X}L^{-1}P^*PLUP^* + PU^*L^*P^*PL^{-*}\hat{X}L^{-1}P^*N \\ &\quad - PU^*L^*P^*PL^{-*}\hat{X}L^{-1}P^*BR^{-1}B^*PL^{-*}PLUP^* \end{aligned} \tag{5.10}$$

Noting that $M = PLUP^*$ and comparing the last line in (5.10) with (5.9) we immediately see that

$$X = PL^{-*}\hat{X}L^{-1}P^*. \tag{5.11}$$

In terms of the low-rank factors $X = ZZ^*$, which we are actually computing following Chapter 4, we obtain $\hat{X} = \hat{Z}\hat{Z}^*$, where

$$Z = PL^{-*}\hat{Z}. \tag{5.12}$$

This also enables us to express the feedback operator KM for (5.2), (5.3) in terms of the one for (5.7), since

$$\begin{aligned} K &= -R^{-1}B^*X = -R^{-1}B^*PL^{-*}\hat{X}L^{-1}P^* \\ \hat{K} &= -R^{-1}\hat{B}^*\hat{X} = -R^{-1}B^*PL^{-*}\hat{X} \end{aligned} \tag{5.13}$$

and thus

$$K = \hat{K}L^{-1}P^*,$$

or more importantly

$$KM = \hat{K}L^{-1}PM = \hat{K}L^{-1}PP^*\tilde{M}P = \hat{K}L^{-1}LUP = \hat{K}UP. \tag{5.14}$$

That means we can easily and efficiently recover the solution and feedback operator of the original problem with the generalized state space system from the computation for the equivalent system in standard state space formulation. Note that implementations should never compute the $\tilde{N}$ matrix explicitly. On the other hand we can safely apply the transformations to the in general dense matrices B and C.

5.1.2. Lyapunov Equations

In analogy to the computations in equations (5.8)-(5.12) we find the transformation rules for the LRCFs for the Lyapunov equation. We only have to distinguish the two types of Lyapunov equations. For the second equation in (5.21) containing the C everything works exactly as for (5.8). For the other equation however the adjoints in the linear terms are exchanged compared to (5.8). This also changes the roles of L and U in the subsequent computations. Thus we have to use U instead of L in the back transformation in (5.12).

5.2. Implicit Handling of the Inverse Mass Matrix

Throughout this thesis we assume, that $M \in \mathbb{R}^{n\times n}$ is invertible. If so, we can formally rewrite the system into standard state space representation. Simply multiplying by M^{-1} from the left results in

$$\dot{x} = \underbrace{M^{-1}N}_{=:\tilde{N}}x + \underbrace{M^{-1}B}_{=:\tilde{B}}u, \qquad y = Cx. \tag{5.15}$$

This, being enough for theoretic considerations, as mentioned above is not adequate in numerical applications, since $\tilde{N} = M^{-1}N$ will be a full matrix even if M and N are

sparse. So we cannot afford to form $\tilde{N}$ explicitly. The remainder of this section will be concerned with avoiding it. Instead we want to apply matrix pencil techniques to rewrite Algorithm 4.1 in a form that works with the original problem data, i.e., the matrices M, N and preferably also B and C.

5.2.1. Algebraic Riccati Equations and Feedback Computations

Assuming a standard quadratic cost function (2.12) with operators Q and R as above and following the presentation in (2.1), (2.19) we find the ARE associated with (5.15) as

$$0 = C^T Q C + \tilde{X}\tilde{N} + \tilde{N}^T\tilde{X} - \tilde{X}\tilde{B}R^{-1}\tilde{B}^T\tilde{X}. \tag{5.16}$$

Now inserting the definitions of $\tilde{N}$ and $\tilde{B}$ followed by multiplications with M^T from the left and M from the right, we realize that (5.16) is equivalent to a generalized ARE;

$$\begin{aligned} 0 &= M^{-T}C^TQCM^{-1} + M^{-T}\tilde{X}M^{-1}N + N^T\tilde{X}M^{-1} - M^{-T}\tilde{X}M^{-1}BR^{-1}B^TM^{-T}\tilde{X}M^{-1} \\ &= M^{-T}C^TQCM^{-1} + XN + N^TX + XBR^{-1}B^TX \\ \Leftrightarrow\ 0 &= C^TQC + M^TXN + N^TXM + M^TXBR^{-1}B^TXM. \end{aligned} \tag{5.17}$$

Especially we learn from (5.17) that the solutions of the ARE for (5.15) and the generalized ARE can be transformed into each other following

$$\tilde{X} = M^TXM, \qquad \text{or} \qquad \tilde{Z} = M^TZ, \tag{5.18}$$

in terms of their factors. Using these we can now express the feedback operator KM for (5.1) in terms of the feedback gain for (5.15),

$$\begin{aligned} K &= -R^{-1}B^TX \\ \tilde{K} &= -R^{-1}\tilde{B}^T\tilde{X} = -R^{-1}B^TM^{-T}M^TXM = -R^{-1}B^TXM = KM \end{aligned} \tag{5.19}$$

Thus the feedback $\tilde{K} = KM$ we compute for (5.15) is exactly the feedback we are interested in when working in the generalized systems case. That means we do not even have to back-transform the results. Now applying Newtons method to (5.16) leads to the Lyapunov operator (compare (4.33))

$$\mathfrak{R}'|_X : \quad Y \mapsto (\tilde{N} - \tilde{B}\tilde{K}_X^T)^TY + Y(\tilde{N} - \tilde{B}\tilde{K}_X^T), \tag{5.20}$$

with $\tilde{K}_X$ defined as in (5.19) and the index corresponding to the X at which it is defined. Lyapunov equations with this type of operator are the subject of the following section (see (5.22)).

5.2.2. Lyapunov Equations and Balancing Based Model Order Reduction

The natural controllability and observability Lyapunov equations for system (5.2), (5.3) are the generalized Lyapunov equations

$$NPM^T + MPN^T = -BB^T, \qquad N^TQM + M^TQN = -C^TC. \tag{5.21}$$

On the other hand following (2.1), (2.37) the controllability and observability Gramians for (5.15) solve the equations

$$\tilde{N}\tilde{P} + \tilde{P}\tilde{N}^T = -\tilde{B}\tilde{B}^T, \qquad \tilde{N}^T\tilde{Q} + \tilde{Q}\tilde{N} = -C^TC. \tag{5.22}$$

Inserting the definitions of $\tilde{N}$ and $\tilde{B}$ we observe that $\tilde{P} = P$, but $\tilde{Q} = M^TQM$. So when rewriting Algorithm 4.1, we need to keep track of all changes carefully to examine whether the final version is actually solving (2.37) or (5.22).

The final goal in modifying the algorithm however is to keep the increase in the per step computations as small as possible. Note especially that transforming the solution of (2.37) to that of (5.22) or vice versa only requires one sparse matrix multiplication or one sparse linear system solve with M, respectively, due to the symmetry of the factorizations we are computing. Both can be computed with $\mathcal{O}(n)$ complexity.

In the following we consider the Lyapunov equation

$$FX + XF^T = -GG^T$$

and distinguish the two cases:

1. $F = \tilde{N} = M^{-1}N$, $G = \tilde{B} = M^{-1}B$, and $X = P$;
2. $F = \tilde{N}^T = N^TM^{-T}$, $G = C^T$, and $X = Q$.

The first is the easy case since we already observed that the solutions of the two Lyapunov equations containing B are equal. Let us now consider the two critical steps in the algorithm. These are the initialization of the LRCF (line 1) and its incrementation (line 6). Starting with the initialization we find:

$$\begin{aligned} V_1 &= \sqrt{-2\operatorname{Re}(p_1)}(F + p_1I)^{-1}G \\ &= \sqrt{-2\operatorname{Re}(p_1)}(N + p_1M)^{-1}MG \\ &= \sqrt{-2\operatorname{Re}(p_1)}(N + p_1M)^{-1}B. \end{aligned}$$

Algorithm 5.1 Generalized Low-rank Cholesky factor ADI iteration (G-LRCF-ADI)

Input: M, N and B, or C as in (5.21) and shift parameters $\{p_1, \dots, p_{i_{max}}\}$.
Output: $Z = Z_{imax} \in \mathbb{C}^{n \times t i_{max}}$, such that $ZZ^H \approx P$, Q in (5.21), respectively.

1: **if** right hand side given is C **then**
2: $N = N^T$, $M = M^T$, $G = C^T$
3: **else**
4: $G = B$
5: **end if**
6: $V_1 = \sqrt{-2\,\mathrm{Re}\,(p_1)}(N + p_1 M)^{-1} G$
7: $Z_1 = V_1$
8: **for** $i = 2, 3, \dots, i_{max}$ **do**
9: $V_i = \sqrt{\mathrm{Re}\,(p_i)/\,\mathrm{Re}\,(p_{i-1})}(V_{i-1} - (p_i + \overline{p_{i-1}})(N + p_i M)^{-1}(MV_{i-1}))$
10: $Z_i = [Z_{i-1} V_i]$
11: **end for**

Analogously for the increment we observe:

$$\begin{aligned} V_i &= \sqrt{\frac{\mathrm{Re}\,(p_i)}{\mathrm{Re}\,(p_{i-1})}}(V_{i-1} - (p_i + \overline{p_{i-1}})(F + p_i I)^{-1} V_{i-1}) \\ &= \sqrt{\frac{\mathrm{Re}\,(p_i)}{\mathrm{Re}\,(p_{i-1})}}(V_{i-1} - (p_i + \overline{p_{i-1}})(M^{-1}N + p_i I)^{-1} V_{i-1}) \\ &= \sqrt{\frac{\mathrm{Re}\,(p_i)}{\mathrm{Re}\,(p_{i-1})}}(V_{i-1} - (p_i + \overline{p_{i-1}})(N + p_i M)^{-1} M V_{i-1}). \end{aligned}$$

Thus, in both steps we can shift with the mass matrix M instead of the identity at the cost of an additional sparse matrix vector product. The cost for the solution of the shifted linear system here does not change considerably. Surely, it will be slightly more expensive to compute the sparse matrix sum $N + p_i M$ in order to set up the coefficient matrix for sparse direct solvers, than just adding p_i to the diagonal of N when no mass matrix is present. On the other hand, since N and M are normally arising in the same context (e.g., from a finite element semi-discretization), they will very often have a sparsity pattern, such that the pattern of M is essentially contained in the pattern of N and thus the computational and memory complexity for the actual solve does not change. For iterative solvers the change comes at the cost of one additional sparse matrix vector product and one vector-vector addition per iteration step. Note especially that we do not even have to compute $\tilde{B} = M^{-1}B$. Instead we can directly initialize the computation with the original B.

For the case where $F = \tilde{N}^T$, we first note that

$$I - (p_i + \overline{p_{i-1}})(F + p_i I)^{-1} = (F + p_i I)^{-1}(F - \overline{p_{i-1}} I),$$

and therefore

$$V_i = \sqrt{\frac{\operatorname{Re}(p_i)}{\operatorname{Re}(p_{i-1})}}\left(V_{i-1} - (p_i + \overline{p_{i-1}})(F + p_i I)^{-1} V_{i-1}\right) = \sqrt{\frac{\operatorname{Re}(p_i)}{\operatorname{Re}(p_{i-1})}}(F + p_i I)^{-1}(F - \overline{p_{i-1}} I)V_{i-1}.$$

Inserting this in Algorithm 4.1 we get

$$\begin{aligned}
V_i &= \sqrt{\frac{\operatorname{Re}(p_i)}{\operatorname{Re}(p_{i-1})}}(F + p_i I)^{-1}(F - \overline{p_{i-1}} I)V_{i-1} \\
&= \sqrt{\frac{\operatorname{Re}(p_i)}{\operatorname{Re}(p_{i-1})}}((N^T + p_i M^T)M^{-T})^{-1}((N^T - \overline{p_{i-1}} M^T)M^{-T})V_{i-1} \\
&= \sqrt{\frac{\operatorname{Re}(p_i)}{\operatorname{Re}(p_{i-1})}} M^T (N^T + p_i M^T)^{-1}(N^T - \overline{p_{i-1}} M^T)M^{-T} V_{i-1} \\
&= M^T \left(\sqrt{\frac{\operatorname{Re}(p_i)}{\operatorname{Re}(p_{i-1})}} \left(I - (N^T + p_i M^T)^{-1} M^T\right)\right) M^{-T} V_{i-1}.
\end{aligned}$$

Now observing that the multiplication with M^T in the i-th step is canceled by the M^{-T} in the $(i+1)$-st step, we see that in this case the actual iteration operator changes exactly the same way as above. For the initialization step (line 1) we also have

$$V_1 = M^T \sqrt{-2\operatorname{Re}(p_1)}(N^T + p_1 M^T)^{-1} C^T.$$

That means the above also holds for $i = 1$. The final multiplication with M^T here determines whether we are actually computing the solution factor for Q from (5.21) or $\tilde{Q}$ from (5.22).

Our result is summarized in Algorithm 5.1. Note that this is a direct extension of Algorithm 4.1 to the generalized state space systems case. The acceleration techniques described in Section 4.4.1 can be extended to this new algorithm with little to no work. The column compression for example works exactly the same way, whereas for the Galerkin projection acceleration one needs to project to a generalized Lyapunov equation regardless of whether the columns of Z are orthogonalized in (4.24), (4.25), or not.

Avoiding M^T completely is obviously the cheapest choice. We then compute the solution of the generalized Lyapunov equation. In balancing based model order reduction this is desirable, since we can directly work on the data for the generalized system computing the same ROM as in the case of prior transformation to standard state space representation. Avoiding the transformation also avoids having to work with the in general dense matrix $M^{-1}N$. See Chapter 7 for more details.

There are no such things as applied sciences, only applications of science.

Louis Pasteur

CHAPTER

SIX

APPLICATION IN OPTIMAL CONTROL OF PARABOLIC PDES

Contents

A rather large part of the results concerning linear-quadratic regulator control of parabolic partial differential equations has been investigated in the context of the optimal control of a cooling process for rail profiles in a rolling mill. The resulting model problem is given as the second example in the model problems chapter (Chapter 3). The main ideas that influence the modeling and control of this example problem can be found there. A rigoros approximation result for the convergence of finite dimensional semi-discretized versions of this system to the infinite dimensional case has been derived in [127]. These results were refined in [27] and are mostly reprinted in Appendix A. The present chapter summarizes the results that have been presented in [17], [29] and proves a novel suboptimality measure for the application of the controls computed for the finite dimensional approximating systems to the original ∞-dimensional one, i.e., the underlying real world problem.

6.1. Tracking Control

We now consider the tracking problem for a standard state space system

$$\dot{x} = Ax + Bu, \qquad y = Cx.$$

In contrast to stabilization problems, where one searches for a stabilizing feedback K (i.e. a feedback such that the closed loop operator $A - BK$ is stable), we are searching for a feedback which drives the state to a given reference trajectory asymptotically. Thus the tracking problem is in fact a stabilization problem for the deviation of the current state from the desired state. We will show in this section, that for linear operators A and B tracking can easily be incorporated into an existing solver for the stabilization problem with only a small computational overhead. The results are reprinted from [17, Section 2.2].

A common trick (see, e.g., [56]) to handle inhomogeneities in system theory for ODEs is the following. Given

$$\dot{x} = Ax + Bu + f,$$

let $\hat{x}$ be a solution of the uncontrolled system

$$\dot{x} = Ax + f,$$

such that

$$f = \dot{\hat{x}} - A\hat{x}.$$

Then

$$\dot{x} - \dot{\hat{x}} = Ax + Bu - A\hat{x},$$

from which we derive a homogenous linear system

$$\dot{z} = Az + Bu, \quad \text{where} \quad z = x - \hat{x}.$$

We want to apply this to the abstract Cauchy problem. Assume $(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$ is a reference pair solving

$$\dot{\tilde{\mathbf{x}}} = \mathbf{A}\tilde{\mathbf{x}} + \mathbf{B}\tilde{\mathbf{u}}.$$

We rewrite the tracking type control system as a stabilization problem for the difference $\mathbf{z} = \mathbf{x} - \tilde{\mathbf{x}}$

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{v}. \tag{6.1}$$

Now imposing the cost functional

$$\mathfrak{J}(\mathbf{v}) := \int_0^\infty (\mathbf{z}, \mathbf{Q}\mathbf{z})_{\mathcal{H}} + (\mathbf{v}, \mathbf{R}\mathbf{v})_{\mathcal{U}} dt, \tag{6.2}$$

where $\mathbf{Q} := \mathbf{C}^*\hat{\mathbf{Q}}\mathbf{C}$ with $\hat{\mathbf{Q}} \geq 0$ (as in (3.7)) and applying the standard derivation we get the optimal feedback control

$$\mathbf{v} = -\mathbf{K}\mathbf{z}. \tag{6.3}$$

Inserting this into equation (6.1) and replacing $\mathbf{z}$ by its definition from the variables $\mathbf{x}$ and $\tilde{\mathbf{x}}$ we find

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{K}\mathbf{x} + \dot{\tilde{\mathbf{x}}} - \mathbf{A}\tilde{\mathbf{x}} + \mathbf{B}\mathbf{K}\tilde{\mathbf{x}}. \tag{6.4}$$

So the only difference between the tracking type and stabilization problems is the *known* inhomogeneity $\mathbf{f} := \dot{\tilde{\mathbf{x}}} - (\mathbf{A} - \mathbf{BK})\tilde{\mathbf{x}}$. Note that the operators do not change at all. That means we have to solve the same Riccati equation (A.3) in both cases. Thus, provided that in the cost function (3.7) $\mathbf{y} = \mathbf{Cx}$ has been replaced by $\mathbf{C}(\mathbf{x} - \tilde{\mathbf{x}})$ as in (6.2) above, one only has to add the inhomogeneity $\mathbf{f}$ to the solver for the closed loop system in the tracking type case. Note that the inhomogeneity $\mathbf{f}$ can be computed once and in advance directly after the feedback operator is obtained. Especially in case of a constant setpoint $\hat{\mathbf{x}}$ this is very convenient and makes evaluations cheap.

6.2. Suboptimality Estimation from Approximation Error Results

We have already noted earlier that the optimal feedback controls for the approximating finite dimensional systems can be applied directly to the infinite dimensional PDE control system. Obviously then they have to be considered suboptimal. In this section we investigate how this suboptimality can be measured. There are in principle three ways of measuring the suboptimality of a control. The easiest way would be to monitor the deviation of the applied control from the optimal control in some norm. The second approach is very similar. Instead of the control deviation one may as well inspect the deviation in the solution trajectories generated by the systems under the application of the optimal and suboptimal controls, respectively. The third and probably most adequate way of measuring the suboptimality is to look directly at the optimization problem, i.e., compare the minima taken in the cost functional.

In the case of the LQR problem the evaluation of the cost functional is much less complicated than in open loop approaches, since by Theorem 2.5 we have a direct method to compute the optimal cost in terms of the solution to the appropriate Riccati equation and the initial state, i.e.,

$$\mathfrak{J}(\mathbf{u}_*) = \frac{1}{2} < \mathbf{x}_0, \mathbf{X}_*(t_0)\mathbf{x}_0 >,$$

and

$$\mathfrak{J}(u_*^N) = \frac{1}{2} < P^N\mathbf{x}_0, X_*^N(t_0)P^N\mathbf{x}_0 >= \frac{1}{2} < x_0^N, X_*^N(t_0)x_0^N >= \mathfrak{J}^N(u_*^N).$$

The following theorem provides our new contribution to this field, the suboptimality measure under the assumption that we have a spatial discretization scheme meeting the requirements in Appendix A. If these requirements are fulfilled we have convergence of both the spatial approximations and the Riccati operators by Theorem A.1. Employing the inner product in $\mathcal{H}$, the corresponding induced norm $\|.\| := \|.\|_{\mathcal{H}} := \sqrt{< ., . >_{\mathcal{H}}}$ and the associated operator norm $\|.\| = \|.\|_{\mathcal{H}}$, we can prove:

Theorem 6.1 (Suboptimality estimate for application of N-d controls in ∞-d systems): Let the assumptions of Theorem A.1 hold and define $\hat{X}_* := P^{N^*} X_*^N P^N$. Then the suboptimality applying the control computed for ($\mathcal{R}^{\mathcal{N}}$) to ($\mathcal{R}^{\mathcal{H}}$) can be estimated as

$$|\mathfrak{J}(\mathbf{u}_*) - \mathfrak{J}(u_*^N)| \leq \zeta\left(\|\mathbf{x}_0 - x_0^N\| + \|\mathbf{X}_* - \hat{X}_*^N\|\right). \tag{6.5}$$

Here ζ is a constant only depending on the norms of the initial value $\mathbf{x}_0$ and Riccati-solution $\mathbf{X}_*$. ◊

Proof. Define the bilinear forms

$$\sigma_{\mathbf{X}_*}(x,y) := \frac{1}{2} < \mathbf{X}_* x, y >, \qquad \sigma_{X_*^N}(x,y) := \frac{1}{2} < X_*^N P^N x, P^N y > .$$

Then

$$\begin{aligned}
\mathfrak{J}(\mathbf{u}_*) - \mathfrak{J}(u_*^N) &= \sigma_{\mathbf{X}_*}(\mathbf{x}_0,\mathbf{x}_0) - \sigma_{X_*^N}(x_0^N,x_0^N) \\
&= \sigma_{\mathbf{X}_*}(\mathbf{x}_0,\mathbf{x}_0) - \sigma_{\mathbf{X}_*}(\mathbf{x}_0,x_0^N) + \sigma_{\mathbf{X}_*}(\mathbf{x}_0,x_0^N) - \sigma_{X_*^N}(x_0^N,x_0^N) \\
&= \sigma_{\mathbf{X}_*}(\mathbf{x}_0,\mathbf{x}_0 - x_0^N) + \sigma_{\mathbf{X}_*}(\mathbf{x}_0,x_0^N) - \sigma_{X_*^N}(x_0^N,x_0^N) \\
&= \sigma_{\mathbf{X}_*}(\mathbf{x}_0,\mathbf{x}_0 - x_0^N) + \sigma_{\mathbf{X}_*}(\mathbf{x}_0,x_0^N) - \sigma_{X_*^N}(\mathbf{x}_0,x_0^N) + \sigma_{X_*^N}(\mathbf{x}_0,x_0^N) - \sigma_{X_*^N}(x_0^N,x_0^N) \\
&= \sigma_{\mathbf{X}_*}(\mathbf{x}_0,\mathbf{x}_0 - x_0^N) + \sigma_{\mathbf{X}_*}(\mathbf{x}_0,x_0^N) - \sigma_{X_*^N}(\mathbf{x}_0,x_0^N) + \sigma_{X_*^N}(\mathbf{x}_0 - x_0^N,x_0^N) \\
&\leq \frac{1}{2}\left(\|\mathbf{X}_*\|\,\|\mathbf{x}_0\|\,\|\mathbf{x}_0 - x_0^N\| + \|\mathbf{x}_0\|\,\|x_0^N\|\,\|\mathbf{X}_* - \hat{X}_*^N\| + \|x_0^N\|\,\|X_*^N\|\,\|\mathbf{x}_0 - x_0^N\|\right).
\end{aligned}$$

In the last step we exploit

$$\begin{aligned}
\sigma_{\mathbf{X}_*}(\mathbf{x}_0,x_0^N) - \sigma_{X_*^N}(\mathbf{x}_0,x_0^N) &= \frac{1}{2}\left(< \mathbf{X}_*\mathbf{x}_0, x_0^N > - < X_*^N P^N \mathbf{x}_0, P^N x_0^N >\right) \\
&= \frac{1}{2} < (\mathbf{X}_* - \hat{X}_*^N)\mathbf{x}_0, x_0^N > \\
&\leq \frac{1}{2}\|\mathbf{x}_0\|\,\|x_0^N\|\,\|\mathbf{X}_* - \hat{X}_*^N\|.
\end{aligned}$$

Now defining

$$\begin{aligned}
c_0 &:= \|\mathbf{X}_*\|\,\|\mathbf{x}_0\|, \qquad & c_0^N &:= \|X_*^N\|\,\|x_0^N\|, \\
c_1 &:= \|\mathbf{x}_0\|, & c_1^N &:= \|x_0^N\|,
\end{aligned}$$

from the convergence assumptions we know, that for $n \to \infty$, we have $c_0^N \to c_0$ and $c_1^N \to c_1$. Thus for sufficiently large N it holds $|c_0 - c_0^N| < 1$ and $|c_1 - c_1^N| < 1$, such that for $\zeta := \frac{1}{2}\max\{2(c_0+1),(c_1+1)^2\}$ we find

$$\begin{aligned}
|\mathfrak{J}(\mathbf{u}_*) - \mathfrak{J}(u_*^N)| &= \frac{1}{2}\left((c_0 + c_0^N)\,\|\mathbf{x}_0 - x_0^N\| + c_1 c_1^N\,\|\mathbf{X}_* - \hat{X}_*^N\|\right) \\
&\leq \zeta\left(\|\mathbf{x}_0 - x_0^N\| + \|\mathbf{X}_* - \hat{X}_*^N\|\right).
\end{aligned}$$

♦

For the infinite time horizon, i.e., the ARE case, and again N sufficiently large, Ito [77] provides this approximation rate for bounded operators in terms of the operator approximation $\|(\mathbf{A}^* - A^{N^*}P^N)\mathbf{X}_*\|$ and $\|(\mathbf{B}^* - B^{N^*}P^N)\mathbf{X}_*\|$ as

$$\|\mathbf{X}_* - X_*^N\| \leq 2\,\|(\mathbf{A}^* - A^{N^*}P^N)\mathbf{X}_*\| + 2\,c\,\|\mathbf{B}\|\,\|(\mathbf{B}^* - B^{N^*}P^N)\mathbf{X}_*\|. \tag{6.6}$$

In concrete examples [77] pulls these back to order h (for parabolic systems) and $\sqrt{h}$ (for hereditary systems) approximations in terms of the mesh width h of the underlying discretization scheme. Kroller and Kunisch [86] find an almost squared approximation rate $h^2 \ln \frac{1}{h}$ under the assumption of an h^2 discretization error for the PDE. Further approximation results can, e.g., be found in [91, 107].

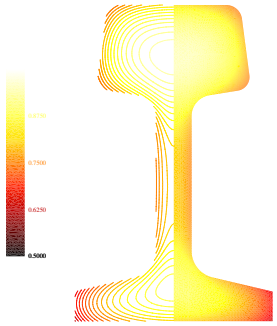
In all the above references the approximation error for the Riccati operator X_*^N is shown to be at most as good as the underlying discretization error. In that sense the term $\|\mathbf{X}_* - X_*^N\|$ in (6.5) will always be the determinative bound for the suboptimality of the N-d feedback applied to the ∞-d control system. We stress this by the following Corollary to the above theorem exploiting the Kroller/Kunisch result for the approximation.

Corollary 6.2:
Let the assumptions of Theorem A.1 hold. Let the discretization provide an h^2 approximation as supposed in [86], then for the suboptimality in applying the control computed for ($\mathcal{R}^{\mathcal{N}}$) to ($\mathcal{R}^{\mathcal{H}}$) we find

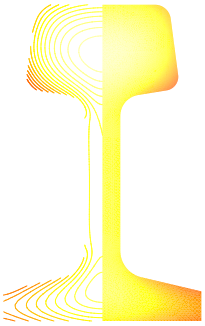
$$|\mathfrak{J}(\mathbf{u}_*) - \mathfrak{J}(u_*^N)| \leq C\left(h^2 + h^2 \log \frac{1}{h}\right) = \mathcal{O}\left(h^2 \log \frac{1}{h}\right) \qquad \lozenge$$

6.3. Adaptive Linear Quadratic Regulator Control of Quasilinear Parabolic PDEs

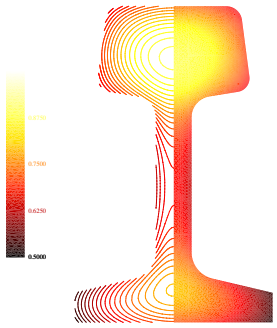
Looking at the steel example of Section 3.3 we find a large interest in controlling quasilinear equations, although the nature of the problem allows us to work with a linearization in the temperature regime of interest. The problem formulation shows that the quasilinearity of the system directly arises from the temperature dependence of the material parameters. We now find from material laws that these dependencies are rather smooth and small. Therefore the idea to adapt these parameters suggests itself immediately. From the numerical implementations point of view this corresponds to a semi-implicit discretization scheme, which has successfully been applied in [49, 27]. On the other hand from the theoretical point of view we loose the invariance of the state equation with respect to time. To overcome this problem we can embed the solution process in a model predictive control (MPC) scheme. Then, following the work of Grüne et al. [61] the crucial ingredient to guarantee the convergence of the scheme is a control Lyapunov function for the system. Their approach especially allows varying sizes of the horizon



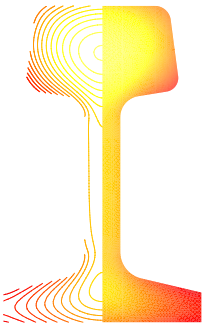
(a) linear model after 20 seconds



(b) nonlinear model after 20 seconds



(c) linear model after 40 seconds



(d) nonlinear model after 40 seconds

Figure 6.1.: Snapshots comparing the optimally controlled temperature distributions on crossections of the steel profile after 20 and 40 seconds for the linear and nonlinear equations

where the local control is applied. In terms of the semi-implicit discretization scheme this allows varying time step sizes.

The control Lyapunov function in the LQR case is well known (e.g., discussed in [78]) to be determined by the solution to the Riccati equation. Thus we can guarantee the convergence of the MPC scheme anytime we are able to compute the LQR feedback control. The required monotonic decrease of the value function [61] can be guaranteed, e.g., by [78, Theorem 2.4]. The next section gives some more details on the embedding of the LQR optimal control into the nonlinear MPC scheme.

6.3.1. Relation to Model Predictive Control

The two most important ingredients of the MPC scheme are the *control horizon* and the *optimization horizon* also called *prediction horizon*. The latter is the time interval on which the future behavior of the system is predicted (e.g., by the nonlinear, or a linearized model) and based on this prediction the optimization takes place, i.e., where the control is computed. The control horizon on the other hand is the time interval on which this control is actually applied ($[t, t + \delta]$ in Figure 6.2). Some authors further distinguish

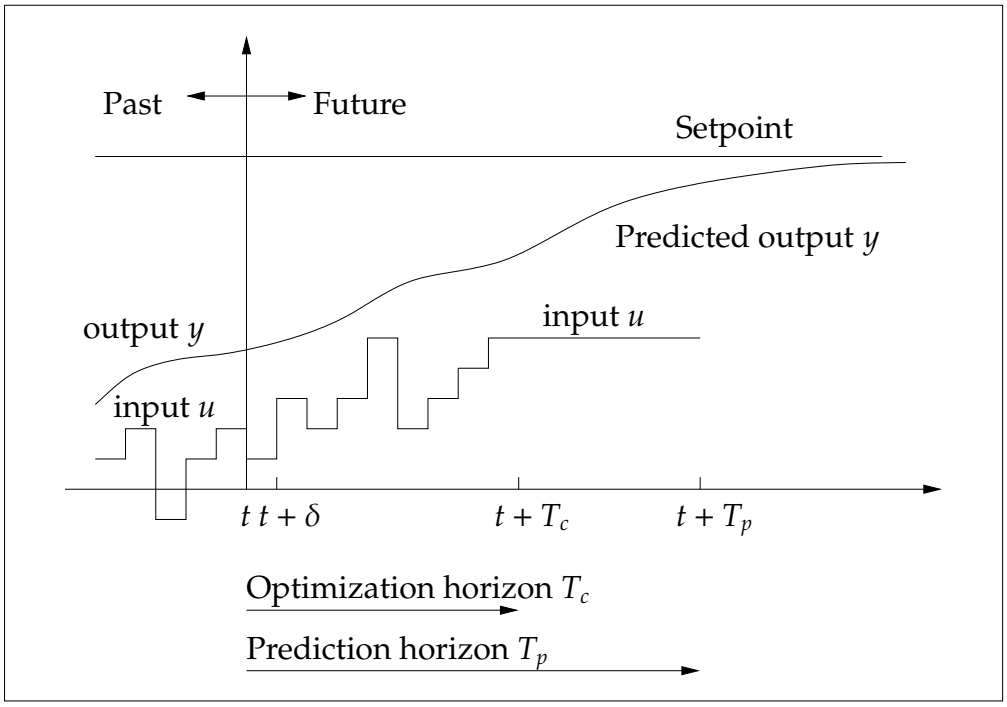


Figure 6.2.: Schematic representation of a model predictive control setting

between optimization and prediction horizons (as in Figure 6.2). Then obviously the prediction horizon needs to be larger than the optimization horizon. The reason for this distinction normally is that simple forward simulations are computationally a lot cheaper and may even be performed nonlinearly, whereas the optimization is the expensive step for which the horizon is ideally as short as possible. Also, the optimization horizon is called control horizon in some publications. Therefore one should carefully check whether the control horizon in a given source is actually $[t, t+\delta]$ as we expect it here, or $[t, t+T_C]$.

Two major approaches to guarantee asymptotic stability of the MPC scheme can be found in the literature. The straight forward approach derives stability from additional terminal constraints on the time frames. This method is widely accepted in the literature and an overview can be found in [104] and references therein. For schemes without stabilizing terminal constraints, results are fairly recent and far less detailed. A consensus has been reached in the corresponding literature that stability can be expected under certain controllability or detectability conditions if the optimization horizon $[t, t+T_C]$ is chosen sufficiently large. This, being a major difficulty in application of open loop approaches, since the optimization is the expensive computational task and thus T_C is desired to be small, makes application of the LQR control even easier, since we can simply choose the infinite time horizon. So we can solve an ARE instead of the more complicated DRE. Additionally we do not need to specify artificial stabilizing terminal

constraints on the single time frames. Note that this is basically the idea of the quasi infinite horizon nonlinear MPC scheme of Chen and Allgöwer [43] that can guarantee stability. Moreover the authors suggest the LQR based feedback gain as the preferential linear optimal control technique in their scheme (see [43, Remark 3.2]).

6.3.2. Identification of Nonlinear MPC Building Blocks

Following the survey in [53] nonlinear MPC consists of three main building blocks:

1. a prediction model,
2. the performance index,
3. a way to compute the control.

Obviously the performance index has to be the quadratic cost functional $\mathfrak{J}(\mathbf{x}, \mathbf{u}, \mathbf{x}(t))$, where the initial value at time t clearly has to be taken as $\mathbf{x}(t)$ rather than $\mathbf{x}_0 = \mathbf{x}(t_0)$. That means we take the final state on the previous application horizon, i.e., the closed loop forward computation, as the initial state $x(t)$ for the current horizon. In the case of the prediction model, we have mainly two choices. We can decide whether to use the full nonlinear simulation model for the prediction, or the linearized version, which is in most cases used for the computation of the control anyway. In our special case we will always use the linearization, since optimization horizon and prediction horizon coincide and we want to compute an LQR based feedback control. Thus the way to compute the control is already determined as the LQR approach.

In summary we linearize the model on short time frames of length δ on which we apply the LQR based feedback control determined by the solution of the ARE, since we choose $T_P = T_C = \infty$. In practical computations δ will be the length of a single upto a few time steps of the simulation method applied. Figure 6.1 illustrates the feasibility of our approach in the context of the optimal cooling of rail profiles model from Section 3.3. More detailed results for this approach can be found in [27].

The monotonicity of the cost functional, as well as the stability of the receding horizon linear-quadratic control of the finite dimensional approximating systems is discussed in [88]. Note that receding horizon control and model predictive control are different notions for similar approaches, that are hardly distinguishable in the literature. One of the most interesting peculiarities in this context is, e.g., the title of the afore cited book: “Receding Horizon Control: Model Predictive Control for State Models”.

> The important thing in science is not so much to obtain new facts as to discover new ways of thinking about them.
>
> Sir William Bragg

CHAPTER

SEVEN

APPLICATION IN MODEL ORDER REDUCTION OF FIRST AND SECOND ORDER SYSTEMS

Contents

The field of model order reduction of linear first order systems is well understood in the literature as far as dense computations are considered. During the recent decade approaches to large scale sparse first order systems have appeared in the literature, that can be summarized as *Smith-type methods for balanced truncation of large scale sparse systems*. Under this title Gugercin and Li [62] have reviewed these types of methods based on the classic balanced truncation MOR. The next section will be dedicated to these methods. We will restrict ourselves to a review of the results found so far and give some comments on the practical issues and observations we have encountered. Besides that we will show how our contributions from Chapters 4 and 5 integrate into the field.

The second section then is dedicated to our novel method for the efficient computation of second order reduction problems exploiting the sparsity and structure of the original second order system matrices while rewriting the system to first order form for the application of the legacy BT approaches.

Algorithm 7.1 Low-Rank Square Root Method (LR-SRM)

Input: (A, B, C) realization of the (large) original state space system,
k the reduced system order

Output: $(\hat{A}, \hat{B}, \hat{C})$ the reduced system realization

1: Solve

$$AX_B + X_B A^T = -BB^T$$

for an LRCF Z_B of X_B.

2: Solve

$$A^T X_C + X_C A = -C^T C$$

for an LRCF Z_C of X_C.

3: Compute the (thin) SVD

$$U_C \Sigma U_B^H = Z_C^H Z_B.$$

4: Define the transformation matrices S_B and S_C according to

$$S_B = Z_B U_B(:, 1:k)\Sigma(1:k, 1:k)^{-\frac{1}{2}}, \qquad S_C = Z_C U_C(:, 1:k)\Sigma(1:k, 1:k)^{-\frac{1}{2}}.$$

{ Note that k can be adapted from Σ for a given error tolerance according to (2.42).}

5: Compute the reduced order realization

$$\hat{A} := S_C^H A S_B, \qquad \hat{B} := S_C^H B, \qquad \hat{C} := C S_B$$

7.1. First Order Systems

7.1.1. Standard State Space Systems

Gugercin and Li [62] have already given an excellent summary on the low-rank solution of Lyapunov equations and the application of the low-rank solution factors in balanced truncation. They especially considered the numerical stability of balanced truncation when the approximating character of the LRCFs is taken into account. Note that they call this approach approximate balanced truncation to reflect this matter. We suggest to rather use this term in the context of H-matrix based balanced truncation, since the approximation is an integral part of the H-matrix approach. Therefore we would suggest using the term low-rank balanced truncation or truncated balanced truncation when LRCFs are applied. The main points from the survey are:

1. low-rank balanced truncation, i.e., balanced truncation based on low-rank factors instead of triangular Cholesky factors can not guarantee the stability of the ROM, although this has not been observed yet in practice.

2. To make the difference between the ROM generated from full rank (i.e., Cholesky)

factors of the Gramians and the ROM computed using low-rank factors small one has to ensure that the full rank factors and low-rank factors are "close" (See formula (2.54) in [62] originally: [63, equation (4.17)]).

3. In [7] a certain decay bound for the eigenvalues of the Gramians is derived, which gives rise to the following remarks regarding the LRCF-ADI:

   a) If the eigenvalues of the system matrix A are clustered in $\mathbb{C}$, choosing the shifts inside the cluster(s) lowers the spectral radius of the iteration matrix (W_J in Section 4.2) and in turn increases the convergence speed of the ADI iteration.

   b) If the eigenvalues of A have mostly dominant real parts, then the decay rate of the eigenvalues of the Gramian is again fast and so is the convergence speed of the ADI iteration.

   c) If the eigenvalues of A have mostly dominant imaginary parts, while the real parts are relatively small, the decay rate for the eigenvalues of the Gramian is small. Then the ADI iteration converges slowly.

Additional Remarks. We want to add some personal remarks here. Losing the guarantee of stability is an issue we have to face in numerical computations even for full rank factors. Thus 1 is an issue on should keep in mind but never overrate. In terms of the LRCF-ADI stopping criteria (Section 4.6) 2. means that we always have to solve very accurately, i.e., with small residuals and small relative change tolerances. Note that in practice the deviations are often first/only observed for very high frequencies in the Bode plot. Note further that 2. especially suggests to choose small truncation tolerances for the column compression (as proposed in Section 4.4.1) applied to the LRCFs. Also we note that [63, equation (4.17)] does not allow for a suggestion for the truncation tolerance, other than the machine precision itself, since none of the data on the right hand side of this inequality is known a priori.

Concerning 3a), in finite arithmetics one has to keep in mind that the step operator at least implicitly contains the term $(A - p_i I)$. Now choosing p_i very close to an eigenvalue one easily increases the condition number of this operator, which in certain applications has been observed to even decrease the convergence speed. For example, for the Gyro example (Section 3.8) computing the heuristic shifts from eigenvalue approximations via `eigs` in MATLAB gives much slower convergence, than the same number of shift computed from a set of Ritz values computed by only a few steps of the Arnoldi iteration. Remark number 3b) additionally supports the strategy to choose only the real parts of the Penzl parameters as the ADI shifts. The convergence should be rather fast in this case anyway and every step of the iteration can be computed a lot faster when complex arithmetics (and memory demands) are avoided. In the case of 3c), if the number of eigenvalues with dominant imaginary parts that are located near the imaginary axis is small, Wachspress [145, 46] suggests to separate them for special treatment and choose the (asymptotically) optimal shifts for the remaining part of the

spectrum.

The low-rank square root method that results from using LRCFs instead of full rank factors is summarized in Algorithm 7.1. Modifications working with the generalized state space representation are given in the following section. The remainder of this section is concerned with the efficient application of this algorithm in terms of both computation speed and accuracy.

Convergence Speed of the LRCF-ADI and MOR accuracy. An important observation that we want to state next, is that in MOR applications the convergence speed in the LRCF-ADI is not a major issue. In fact it can even be helpful or desirable to have slower convergence, because then more iteration steps are taken and thus the factor grows further. This sounding counterintuitive on first glance in turn holds the possibility of adding more subspace information to the factor. Thus for a slower convergence speed the rank of the LRCF may be larger than that of the fast converged one. Now remembering, that the Hankel singular values are computed from the product (2.41) where now S and R are the LRCFs Z_B and Z_C in Algorithm 7.1 we see that the lower of the ranks of Z_B and Z_C limits the rank of the product. Thus it limits the number of nonzero HSVs and with it the order of the ROM. Therefore from a certain point we may be unable to increase the accuracy of the ROM due to missing subspace information in the factors. Therefore we suggest to use the relative change criterion rather than the residual for stopping the ADI iteration in MOR contexts, since this tends to run longer and should catch rank increases better in most cases.

Choice of the Parameters for the Shift Computation. A crucial question when applying any shift parameter computation based on Penzls heuristic method is how many shifts (l_0) to compute and from how many Ritz (k_p) and harmonic Ritz values (k_m) to do so. In [116] Penzl shows, that taking a lot of shifts does not give better convergence results. For an example corresponding to the 2d FDM heat equation model from Section 3.1 without convection, he shows, that for an order 400 problem there are only very small performance gains doubling the number of shifts from 8 to 16 and almost none for further doubling to 32. Thus we can restrict ourselves to a rather small amount of shifts. Taking around $l_0 = 15$ shifts gives good results even in problems of dimension $\mathcal{O}(10^4)$. In MOR applications we can thus even think about smaller l_0, since taking a few more iteration steps can enable us to find more accurate reduced order models. Besides that when looking for smaller numbers of shifts we can also choose k_m and k_p smaller and thus save iteration steps in the preliminary Arnoldi methods.

Dual Lyapunov Solutions. In Algorithm 7.1 we need the solutions to both the observability and controllability Lyapunov equations. The equations are dual to each other, i.e., when writing the LRCF-ADI for the one of them we have to apply the adjoint (in general transpose) operator for the other one. The step operators P_k (see equation (4.6))

Algorithm 7.2 Generalized Low-Rank Square Root Method for Standard ROMs (GS-LR-SRM)

Input: (M, N, B, C) realization of the (large) original state space system, k the reduced system order

Output: $(\hat{A}, \hat{B}, \hat{C})$ the reduced standard state space realization

1: Solve
$$NX_BM^T + MX_BN^T = -BB^T$$
for an LRCF Z_B of X_B.

2: Solve
$$N^TX_CM + M^TX_CN = -C^TC$$
for an LRCF Z_C of X_C.

3: $Z_C = M^HZ_C$ {see Section 5.2}

4: Compute the (thin) SVD
$$U_C\Sigma U_B^H = Z_C^HZ_B.$$

5: Define the transformation matrices S_B and S_C according to
$$S_B = Z_BU_B(:, 1:k)\Sigma(1:k, 1:k)^{-\frac{1}{2}}, \qquad S_C = Z_CU_C(:, 1:k)\Sigma(1:k, 1:k)^{-\frac{1}{2}}.$$
{ Note that k can be adapted from Σ for a given error tolerance according to (2.42).}

6: Compute the reduced order realization
$$\hat{A} := S_C^HM^{-1}NS_B, \qquad \hat{B} := S_C^HM^{-1}B, \qquad \hat{C} := CS_B.$$
{ Note that $S_C^HM^{-1}$ can be precomputed, saving one solve with M.}

in both iterations are therefore transposes in real arithmetic. Now having computed the LU-decomposition of, e.g., $F + p_kI = L_kU_k$ we have the LU decomposition U^TL^T of $F^T + p_kI$ at hand. If our implementation can solve with the transposes at similar cost as with the factors themselves, we can exploit this to save on decomposition per iteration step, when solving for both Gramians simultaneously. Note that in cases where F is self-adjoint we can always compute both Gramians simultaneously at little to no additional cost.

7.1.2. Generalized State Space Systems

In many large scale applications, especially when FEM discretization is applied during the model generation, the system does not arise in standard state space form, but in generalized state space form (2.4). For theoretical considerations it is then sufficient to know that the mass matrix is invertible, such that an equivalent standard state space system can be formed, as described in Section 2.2.2. For small dense systems this is

Algorithm 7.3 Generalized Low-Rank Square Root Method for Generalized ROMs (GG-LR-SRM)

Input: (M, N, B, C) realization of the (large) original state space system, k the reduced system order

Output: $(\hat{M}, \hat{N}, \hat{B}, \hat{C})$ the reduced generalized state space realization

1: Solve
$$NX_BM^T + MX_BN^T = -BB^T$$
for an LRCF Z_B of X_B.

2: Solve
$$N^TX_CM + M^TX_CN = -C^TC$$
for an LRCF Z_C of X_C.

3: Compute the (thin) SVD
$$U_C\Sigma U_B^H = Z_C^HMZ_B$$

4: Define the transformation matrices S_B and S_C according to
$$S_B = Z_BU_B(:, 1:k)\Sigma(1:k, 1:k)^{-\frac{1}{2}}, \qquad S_C = Z_CU_C(:, 1:k)\Sigma(1:k, 1:k)^{-\frac{1}{2}}.$$
{ Note that k can be adapted from Σ for a given error tolerance according to (2.42).}

5: Compute the reduced order realization
$$\hat{M} := S_C^HMS_B, \qquad \hat{N} := S_C^HNS_B, \qquad \hat{B} := S_C^HB, \qquad \hat{C} := CS_B.$$

still applicable and can be performed in quadratic complexity which is easily exceeded by the complexity of the matrix equation solvers for dense systems. Therefore from a complexity point of view these transformations are cheap and applicable there. In large scale sparse applications this is prohibitive since then we loose the sparsity of the state space matrix and memory limitations restrict the problem sizes drastically. In Chapter 5 we have already discussed techniques to avoid this problem. We have seen that we can exploit the sparsity of the original system best by the G-LRCF-ADI algorithm for solving generalized Lyapunov Equations. It has also been discussed there, how these solutions relate to those of the equivalent standard state space representation. Algorithm 7.2 now is the obvious reformulation of Algorithm 7.1 that exploits the matrix pencil approach in the computation of the Gramian factors, then recomputes the Gramian factors for the equivalent standard state space form and uses these to compute the ROM for the standard state space representation. Thus the ROM is in standard state space form. Note, however, that we need to implement the reduction carefully since, e.g., the matrix $\hat{A} := S_C^HM^{-1}NS_B$ needs to be formed and it is crucial to exploit the rectangularity of S_B and S_C by computing $\hat{A}$ as $(M^{-H}S_C)^H(NS_B)$ where $M^{-H}S_C$ and NS_B keep the same dimensions as S_C and S_B. Alternatively we can follow Algorithm 7.3 taking M into account in the SVD instead of forming the equivalent standard state space representation and then

apply the transformation to the generalized state space form that we actually really have given.

Note that Algorithms 7.2 and 7.3 produce the same ROM. First of all, due to the fact that Algorithm 7.2 works with $\tilde{Z}_c = M^H Z_C$ both algorithms compute the same U_C, Σ, U_B. Then in Algorithm 7.2

$$S_C = \tilde{Z}_C U_C(:, 1:k)\Sigma(1:k, 1:k)^{\frac{1}{2}} = M^H Z_C U_C(:, 1:k)\Sigma(1:k, 1:k)^{\frac{1}{2}} =: M^H \hat{S}_C,$$

and $\hat{S}_C$ is exactly the S_C computed in Algorithm 7.3. Further

$$\hat{A} = S_C^H M^{-1} N S_B = \hat{S}_C^H N S_B$$

and

$$\hat{B} = S_C^H M^{-1} B = \hat{S}_C^H B,$$

which obviously coincide with the ones computed in Algorithm 7.3. Finally $\hat{M}$ in Algorithm 7.3 is always $I_k \in \mathbb{R}^{k\times k}$ by construction, since

$$S_C^H M S_B = \Sigma^{\frac{1}{2}} U_C^H Z_C^H M Z_B U_B \Sigma^{\frac{1}{2}} = \Sigma^{-\frac{1}{2}} U_C^H U_C \Sigma U_B^H U_B \Sigma^{-\frac{1}{2}} = \Sigma^{-\frac{1}{2}} \Sigma \Sigma^{-\frac{1}{2}} = I.$$

The dimension $k \times k$ directly follows from the truncation dimension k in the algorithm.

7.2. Second Order Systems

The task of model order reduction is to find a ROM that captures the essential dynamics of the system and preserves its important properties. Since we are considering systems of second order we can essentially follow two paths during the computation of the reduced order model. The natural choice would be to preserve the second order structure of the system and compute a second-order reduced order model of the form

$$\hat{M}\ddot{\hat{x}}(t) + \hat{D}\dot{\hat{x}}(t) + \hat{K}\hat{x}(t) = \hat{B}u(t), \qquad \hat{y}(t) = \hat{C}_v\dot{\hat{x}}(t) + \hat{C}_p\hat{x}(t), \tag{7.1}$$

where $k \ll n$ and $\hat{M}, \hat{D}, \hat{K} \in \mathbb{R}^{k\times k}$, $\hat{B} \in \mathbb{R}^{k\times p}$, $\hat{C}_v, \hat{C}_p \in \mathbb{R}^{m\times k}$ and $\hat{x}(t) \in \mathbb{R}^k$. Unfortunately, the global balanced truncation error bound (2.42) for the reduction is lost if the structure preserving balanced truncation is applied following [108, 41, 124]. Recently it was shown in [153] that it can be reestablished in special cases under additional symmetry assumptions. The basic idea is, that for systems, with input and output matrices being transposes of each other and all matrices defining the differential equation, i.e., M, D and K, are symmetric, one has enough structural information to reestablish the error bound. Although these assumptions on the system may seem rather special, this is a very common setting in simulation and design of electric circuits.

Still many simulation and controller design tools used in applications in the engineering sciences expect the system models to be of first order. Therefore even if the original

system is of second order there is a large demand for the computation of a first-order ROM

$$\hat{M}\dot{\hat{x}}(t) = \hat{A}\hat{x}(t) + \hat{B}u(t), \qquad \hat{y}(t) = \hat{C}\hat{x}(t), \tag{7.2}$$

in practice. Here again $k \ll n$ and $\hat{M}, \hat{A} \in \mathbb{R}^{k\times k}$, $\hat{B} \in \mathbb{R}^{k\times p}$, $\hat{C} \in \mathbb{R}^{m\times k}$ and $\hat{x}(t) \in \mathbb{R}^k$.

The main idea behind both approaches is to rewrite (2.6) in first order representation and apply balanced truncation to the equivalent first-order model, as described in Section 2.2.3. From the previous section in this chapter we know that then $\hat{M}$ will in fact be the identity I_k.

7.2.1. Efficient Computation of Reduced First Order Models

Following the technique presented in Section 2.2.3, we trace the reduction of the second order system back to the reduction of a generalized first order system of double dimension,

$$\mathcal{M}\dot{x}(t) = \mathcal{A}x(t) + \mathcal{B}u(t), \qquad y(t) = \mathcal{C}x(t). \tag{7.3}$$

That means the main task in this section will be to map the required matrix operation to operations with the original system matrices M, G, K of (2.6). Following the derivations in Section 5.2 these operations are $x = \mathcal{M}^{-1}\mathcal{A}f$, $\mathcal{M}^{-1}\mathcal{A}x = f$ and $(\mathcal{A} + p\mathcal{M})^{-1}\mathcal{M}$, as well as $x = (\mathcal{M}^{-1}\mathcal{A})^T f$, $(\mathcal{M}^{-1}\mathcal{A})^T x = f$ and $(\mathcal{A}^T + p\mathcal{M}^T)^{-1}\mathcal{M}^T$. In the following we will always decompose $x, f \in \mathbb{R}^{2n}$ as in

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \text{and} \qquad f = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix},$$

where $x_1, x_2, f_1, f_2 \in \mathbb{R}^n$ to have them fit the block sizes in $\mathcal{M}$ and $\mathcal{A}$.

For the above linear algebra operations involved in the ADI iteration for computing the controllability and observability Gramians, we show in the following list how to perform these operations using original data from the second-order model only:

$x = \mathcal{M}^{-1}\mathcal{A}f$	$\Leftrightarrow \mathcal{M}x = \mathcal{A}f$	$\Leftrightarrow x_1 = f_2,$ $Mx_2 = -Kf_1 - Gf_2$
$x = (\mathcal{M}^{-1}\mathcal{A})^T f$	$\Leftrightarrow x = \mathcal{A}^T\mathcal{M}^{-T} f$	$\Leftrightarrow M^T\tilde{f_2} = f_2,$ $x_1 = -K^T\tilde{f_2},$ $x_2 = f_1 - G^T\tilde{f_2}$
$\mathcal{M}^{-1}\mathcal{A}x = f$	$\Leftrightarrow \mathcal{A}x = \mathcal{M}f$	$\Leftrightarrow x_2 = f_1,$ $Kx_1 = -Mf_2 - Gf_1$
$(\mathcal{M}^{-1}\mathcal{A})^T x = f$	$\Leftrightarrow \mathcal{A}^T\mathcal{M}^{-T}x = f$	$\Leftrightarrow K^T\tilde{x}_2 = -f_1,$ $x_1 = f_2 + G^T\tilde{x}_2$ $x_2 = M^T\tilde{x}_2$
$x = (\mathcal{A} + p\mathcal{M})^{-1}\mathcal{M}f$	$\Leftrightarrow (\mathcal{A} + p\mathcal{M})x = \mathcal{M}f$	$\Leftrightarrow (p^2M - pG + K)x_1 = Gf_1 - M(f_2 + pf_1),$ $x_2 = f_1 - px_1$
$x = (\mathcal{A}^T + p\mathcal{M}^T)^{-1}\mathcal{M}^T f$	$\Leftrightarrow (\mathcal{A}^T + p\mathcal{M}^T)x = \mathcal{M}^T f$	$\Leftrightarrow \tilde{f_2} = M^T f_2,$ $(p^2M^T - pG^T + K^T)x_2 = p\tilde{f_2} - f_1,$ $x_1 = \tilde{f_2} + G^T x_2 - pM^T x_2$

Table 7.2.: Computing the $2n \times 2n$ first order matrix operations in terms of the original $n \times n$ second order matrices

From the rightmost column we see that we can perform all matrix operations needed by Algorithm 5.1 and its preceding parameter computation directly using the original system matrices M, G, K, B, C_p, C_v. Computation of the two matrix polynomials and their usage in sparse direct solvers is *cheap* with the same arguments as in Section 2.2.2. The important message here is that exploiting the block structure of the $2n \times 2n$ matrices in the equivalent first order representation, we can reduce the computational and storage cost to essentially $\mathcal{O}(n)$. That means all system matrices can be stored in $\mathcal{O}(n)$.

A word of warning has to be given regarding the shift parameters. Since linear systems

$$(p^2M^T - pG^T + K^T)x_2 = p\tilde{f_2} - f_1,$$

need to be solved where M, G and K result from the same discretization and therefore in general have similar condition numbers, large shifts need to be avoided. Otherwise the drastic weighting difference in p^2M and K will lead to severe numerical incorrectness

corrupting the results of all subsequent computations. This is no essential restriction, since the information from eigenvalues closer to the imaginary axis is more important for the results in most cases anyway. Especially in the present MOR tasks, where we are interested in covering the poles of the transfer function matrix, it is observed that these closely relate to the imaginary parts of the eigenvalues close to the imaginary axis. That means very large eigenvalues (which have very small or no imaginary parts for most FEM matrices anyway) are of less importance for the reduced order system generation and thus may be neglected in the process of shift parameter computation.

7.2.2. Regaining the Second Order Structure for the Reduced Order Model

Second order balanced truncation has been introduced in [108]. The general idea for reducing the second order system to a second order ROM is essentially as follows: the system (2.6) is equivalently rewritten to first order form (2.8). Then from the first order system the balancing matrices are obtained following Section 2.4. The required second order Gramians in [108] are defined based on the equivalent first order system in standard state space form

$$\begin{bmatrix} \dot{x} \\ \dot{\dot{x}} \end{bmatrix} = \begin{bmatrix} 0 & I_n \\ -M^{-1}K & -M^{-1}G \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1}B \end{bmatrix} u, \qquad y = \begin{bmatrix} C_p & C_v \end{bmatrix} u. \tag{7.4}$$

For this system the Gramians P and Q as in (2.37) are computed. These are compatibly partitioned as

$$P = \begin{bmatrix} P_p & P_o \\ P_o^T & P_v \end{bmatrix}, \qquad Q = \begin{bmatrix} Q_p & Q_o \\ Q_o^T & Q_v \end{bmatrix}. \tag{7.5}$$

The second order *position Gramians* are then given as P_p and Q_p. Analogously P_v and Q_v define the *velocity Gramians* (See [108, 41, 134] for details). Using pairs of these second order Gramians we can now define the *position balanced* (P_p, Q_p), *velocity balanced* (P_v, Q_v), *position-velocity balanced* (P_p, Q_v) and *velocity-position balanced* (P_v, Q_p) ROMs following [124, Definition 2.2]. Now, e.g., the position balancing Gramian pair (P_p, Q_p) takes the role of (P, Q) in the computation of the projectors T_l and T_r in (2.40) and the reduced order system (7.1) is obtained according to

$$\begin{aligned} \hat{M} &= T_l M T_r, \qquad \hat{G} = T_l G T_r, \qquad \hat{K} = T_l K T_r, \\ \hat{B} &= T_l B, \quad \hat{C}_v = C_v T_r, \quad \text{and} \quad \hat{C}_p = C_p T_r. \end{aligned}$$

In order to preserve stability and symmetry of the original system, projection can also be performed by an orthogonal matrix T as in

$$\begin{aligned} \hat{M} &= T^T M T, \qquad \hat{G} = T^T G T, \qquad \hat{K} = T^T K T, \\ \hat{B} &= T^T B, \quad \hat{C}_v = C_v T, \quad \text{and} \quad \hat{C}_p = C_p T, \end{aligned}$$

where T can be obtained, e.g., from the range of T_r. In general for a non-symmetric system we will not have $T_l^T = T_r$ and thus the balancing of the Gramian product (2.41) is no longer ensured. Therefore also the global error bound (2.42) is lost. For systems where M, G, K are symmetric, $C_v = 0$ and $C_p = B^T$ [153] reestablishes the error bound. The key idea there is to use the equivalent first order model

$$\begin{bmatrix} -K & 0 \\ 0 & M \end{bmatrix} \dot{z}(t) = \begin{bmatrix} 0 & -K \\ -K & -G \end{bmatrix} z(t) + \begin{bmatrix} 0 \\ B \end{bmatrix} u(t), \qquad y(t) = \begin{bmatrix} B^T & 0 \end{bmatrix} z(t), \tag{7.6}$$

and thus regain the symmetry in the first order system. Although these conditions might seem rather academic, there is a large class of systems arising in electrical engineering when developing RLCK circuits, which have exactly these properties. Velocity balancing, position-velocity balancing and velocity-position balancing can be applied similarly (see [124] for details). Stykel and Reis [124] in addition prove stability preservation for the position-velocity balancing of symmetric second order systems, with positive definite mass, stiffness and damping matrices. They also note that in general none of the approaches guarantees stability of the ROM.

In the following we show that the low-rank factors of the second order Gramians P_p, P_v, Q_p, Q_v in (7.5) can be formed directly from LRCFs S and R of the first order Gramians P, and Q computed with respect to (7.4). Hence we can avoid building the full Gramian matrices in (7.5) and therefore reduce the expenses to those of the ADI framework.

Let S be a low-rank Cholesky factor of the Gramians Q computed by the (G-)LRCF-ADI Algorithm for either of the two first order representations, e.g., by LRCF-ADI for (7.4). We can now compatibly partition $S^H = [S_1^H \; S_2^H]$ and compute

$$\begin{bmatrix} P_p & P_o \\ P_o^T & P_v \end{bmatrix} = P = SS^H = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} \begin{bmatrix} S_1^H & S_2^H \end{bmatrix} = \begin{bmatrix} S_1 S_1^H & S_1 S_2^H \\ S_2 S_1^H & S_2 S_2^H \end{bmatrix}.$$

Hence, $P_p = S_1 S_1^H$, such that the low-rank Cholesky factor of the position controllability Gramian is directly given as the upper n rows S_1 of the low-rank Cholesky factor S. Analogously, we can compute the LRCF R_1 of the second order position observability Gramian Q_p from the LRCF R of the first order observability Gramian Q. Also, the lower n rows S_2 of the first order Gramian factor form the required LRCF of the second order velocity controllability Gramian in case we want to apply velocity based balancing. Again, in complete analogy, the same holds true for R_2 as the LRCF of Q_v.

Note that the block structure in (7.6) can be exploited analogously to the procedure presented in Section 7.2.1. Note further that when applying G-LRCF-ADI in the back-transformation according to (5.18) we have $\tilde{S}_1 = S_1$ since the $(1,1)$-block in $\mathcal{M}$ is I_n in (2.8). For the transformation (7.6) on the other hand we need to consider $-K$ for transforming the LRCFs S_1 and $\tilde{S}_1$, i.e., $\tilde{S}_1 = -KS_1$, since $K = K^T$.

7.2.3. Adaptive Choice of Reduced Model Order

We have repeatedly regretted the loss of the global error bound for the second order balancing approaches. Another reason why we do so is, that we can no longer use this easy to compute adaption process to choose the reduced model order based on a prescribed tolerance on the model reduction error. Fortunately an alternative adaption method has shown to give good results in first order MOR. There we do not sum up the truncated singular values, but monitor the ratio $\frac{\sigma_k}{\sigma_1}$ assuming that the singular values are decreasingly ordered as usual. As soon as this ratio drops below the prescribed tolerance the truncation is performed.

This method is the way ROM orders are adapted in LyaPack and has shown to provide very similar results to the exact error bound evaluation. Even though we loose the error bound and in practical applications observe that largest HSVs can be smaller than any prescribed tolerance would be chosen (see, e.g. Table 8.11), we can still apply the ratio method to determine the reduced model order in second order balancing approaches. Although we should keep in mind that it truncates the internal HSV decay of the ROM, but does not guarantee any approximation error bound.

By far the best proof is experience.

Sir Francis Bacon

CHAPTER

EIGHT

NUMERICAL TESTS

Contents

The last chapter of the main part of this thesis is dedicated to the numerical verification of the results from the previous chapters. We will avoid reprinting of numerical tests concerning the LQR problem for the steel example here. The interested reader is referred to [17, 27, 26, 127] where extensive testing for the stabilization, tracking and nonlinear stabilization problems have been presented. This chapter is structured as follows. First we repeat some results [20] on the parameter selection for the ADI iteration. We replenish these results with some new observations on a fairly different choice of parameters showing promising behavior in the application in model order reduction.

The second section then illustrates how the acceleration techniques for the ADI and Newton's method presented in Sections 4.4 and 4.5 work in practice. After that we present all model reduction related results; starting with a case study for a very large generalized state space system and ending with the efficient computation of second order ROMs. The final section then compares the different implementations in C and MATLAB and demonstrates some of our new ideas for the efficient memory management and shared memory parallelization of the algorithms in Chapter 4.

8.1. Numerical Tests for the ADI Shift Parameter Selections

For the numerical tests in this section the LyaPack[1] software package [117] was used. A test program similar to `demo_r1` from the LyaPack examples was employed for the computations, with the ADI parameter selection switching between the methods described in Section 4.3. We have concentrated on the case where the ADI shift parameters can be chosen real. Choosing real shifts wherever possible has two major advantages. When choosing complex shifts, the LRCFs computed with these shifts will be complex as well. This leads to doubling the storage requirements on the one hand and approximately quadrupling computation effort on the other hand.

8.1.1. FDM Semi-Discretized Convection-Diffusion-Reaction Equation

Here we consider the finite difference semi-discretized partial differential equation

$$\frac{\partial \mathbf{x}}{\partial t} - \Delta \mathbf{x} - \begin{bmatrix} 20 \\ 0 \end{bmatrix} . \nabla \mathbf{x} + 180\mathbf{x} = \mathbf{f}(\xi)\mathbf{u}(t), \tag{8.1}$$

where $\mathbf{x}$ is a function of time t, vertical position ξ_1 and horizontal position ξ_2 on the square with opposite corners $(0,0)$ and $(1,1)$. The example is taken from the SLICOT collection of benchmark examples for model reduction of linear time-invariant dynamical systems (see [42, Section 2.7] for details). It is given in semi-discretized state space model representation:

$$\dot{x} = Ax + Bu, \quad y = Cx. \tag{8.2}$$

The matrices A, B, C for this system can be found on the NICONET web site[2].

Figure 8.1a,b show the spectrum and sparsity pattern of the system matrix A. The iteration history, i.e., the numbers of ADI steps in each step of Newton's method are plotted in Figure 8.1c. There we can see that in fact the semi-optimal parameters work exactly like the optimal ones by the Wachspress approach. This is what we would expect since the rectangular spectrum is an optimal case for our idea, because the parameters

[1] available from: `http://www.netlib.org/lyapack/` or `http://www.tu-chemnitz.de/sfb393/lyapack/`
[2] `http://www.icm.tu-bs.de/NICONET/benchmodred.html`

a, b and α (see Section 4.3) are exactly (to the accuracy of Arnoldi's method) met here. Note especially that for the heuristic parameters even more outer Newton iterations than for our parameters are required.

8.1.2. FDM Semi-Discretized Heat Equation

In this example we tested the parameters for the finite difference semi-discretized heat equation on the unit square (3.1) from Section 3.1.

The data is generated by the routines `fdm_2d_matrix` and `fdm_2d_vector` from the examples of the LyaPack package. Details on the generation of test problems can be found in the documentation of these routines (comments and MATLAB help). Since the differential operator is symmetric here, the matrix A is symmetric and its spectrum is real in this case. Hence $\alpha = 0$ and for the Wachspress parameters only the largest magnitude and smallest magnitude eigenvalues have to be found to determine a and b. That means we only need to compute two Ritz values by the Arnoldi process (which here is in fact a Lanczos process because of symmetry, but the software currently does not exploit that) compared to about 30 (which seems to be an adequate number of shifts) for the heuristic approach. We used a test example with 400 unknowns here to still be able to compute the complete spectrum using `eig` for comparison.

In Figure 8.2 we plotted the sparsity pattern of A and the iteration history for the solution of the corresponding ARE. We can see (Figure 8.2b) that iteration numbers only differ very slightly. Hence we can choose quite independently which parameters to use. Since for the Wachspress approach it is crucial to have an accurate approximation of the smallest magnitude eigenvalue it can be a good idea to choose the heuristic parameters here (even though they are much more expensive to compute) if the smallest magnitude eigenvalue is known to be close to the origin (e.g. in case of finite element discretizations with fine meshes).

8.1.3. FEM Semi-Discretized Convection-Diffusion Equation

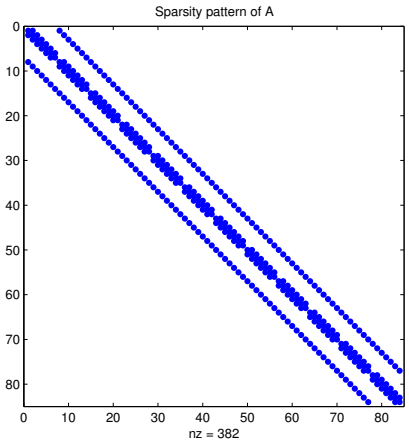
Note, that the heuristic parameters do not appear in the results bar graphics for this example (see Figure 8.4. This is due to the fact that the LyaPack software crashed while applying the complex shift computed by the heuristics. Numerical tests where only the real ones of the heuristic parameters where used lead to very poor convergence in the inner loop, which is generally stopped by the maximum iteration number stopping criterion. This resulted in breaking the convergence in the outer Newton loop. Note that the computations were performed using LyaPack which uses the technique from Section 5.1 to handle the mass matrix. The sparsity patterns of M before and after reordering using Reverse Cuthill-McKee reordering and the Cholesky factor of M after reordering are shown in Figure 8.3. It is illustrating that we have a nice banded structure with only about three times the number of non-zero entries in the factor.

8.1.4. Dominant Pole Shifts and LR-SRM

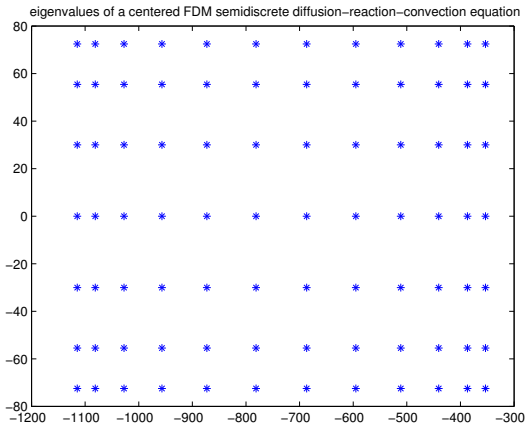
In Section 4.3.3 we motivated the use of dominant poles as ADI shifts in the context of the LR-SRM. Now we want to compare the results of LR-SRM based reductions using dominant poles with the heuristic shift parameters we presented in Section 4.3. The computation of the dominant poles uses the Subspace Accelerated MIMO Dominant Pole Algorithm (SAMDP) presented in [125, Chapter 4]. We test the two shift parameter choices on the CD-Player example from the SLICOT[3] benchmark collection, the spiral inductor from the Oberwolfach Collection and the artificial model from Section 3.2. First we tested all three models with a maximum iteration number of 50 and a residual tolerance of 10^{-10} for the LRCF-ADI, as well as a truncation error tolerance of 10^{-5} and a maximum reduced order of 200. All tests have been carried out without acceleration. The corresponding results are shown in Figure 8.5. Note especially that the area around the minimal relative error for the CD player is almost an exact mirror image of the corresponding peaks in the Bode plot. Also note that the dominant poles give the smallest errors for the spiral inductor (except from very high frequencies $> 10^9$).

Additionally a test with Galerkin projection acceleration in every fifth ADI step has been performed for the artificial model (Section 3.2). Note that the Galerkin projection cannot be supposed to accelerate the computation, since the model does not fulfill $A + A^T < 0$. Still we observe an interesting effect of the projection. From the perfect accordance (see Figure 8.6) of the results for heuristic parameters and the dominant pole shifts, we have to conclude, that the solution factors span the same subspaces onto which the projection is performed.

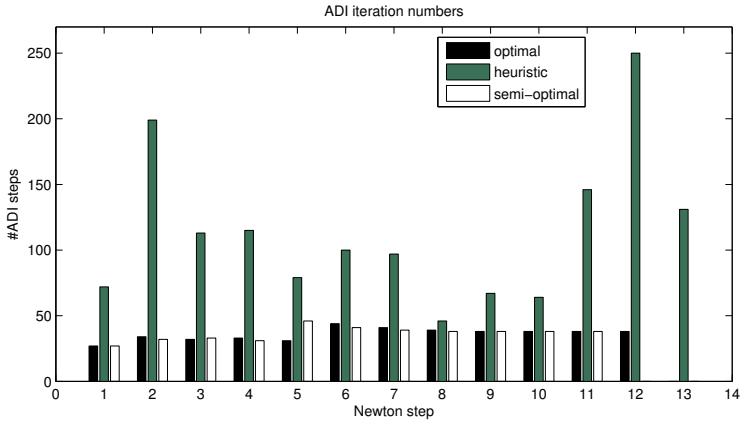
[3] http://www.slicot.org



(a) Sparsity pattern of the FDM semi-discretized operator for equation (8.1)

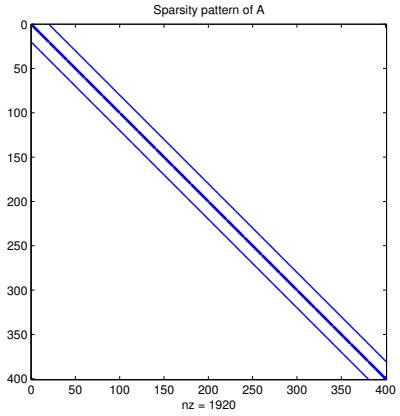


(b) Spectrum of the FDM semi-discretized operator

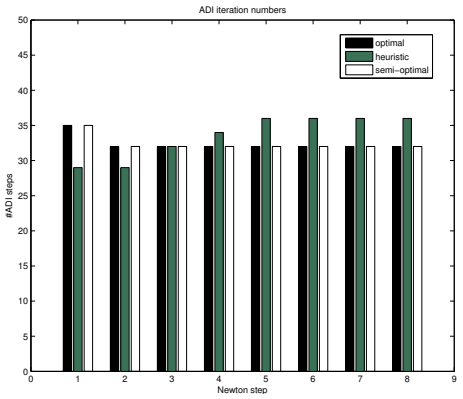


(c) Iteration history for the Newton ADI method applied to (8.1)

Figure 8.1.: Discrete operator and results for the diffusion-convection-reaction equation (FDM)

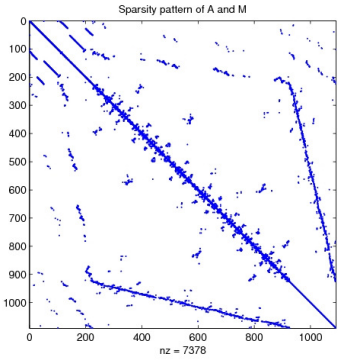


(a) Sparsity pattern of the FDM discretized operator for equation (3.1)

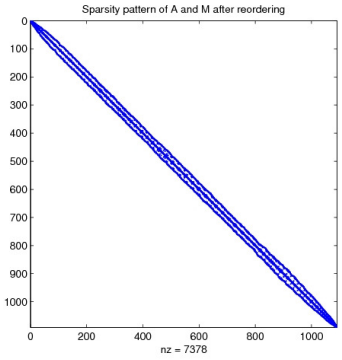


(b) Iteration history for the Newton ADI

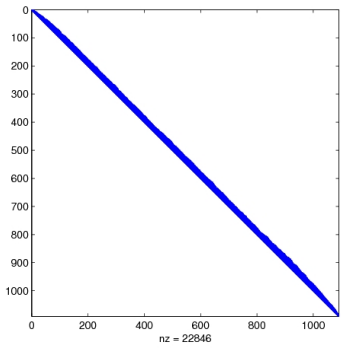
Figure 8.2.: ADI parameters for heat equation (FDM)



(a) Sparsity pattern of A and M in (3.13)

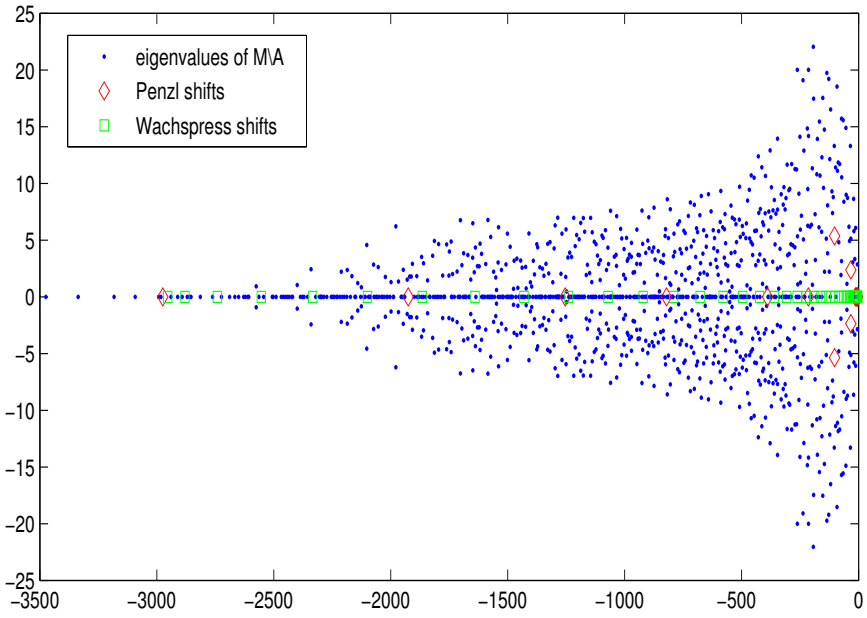


(b) Sparsity pattern of A and M in (3.13) after RCM reordering

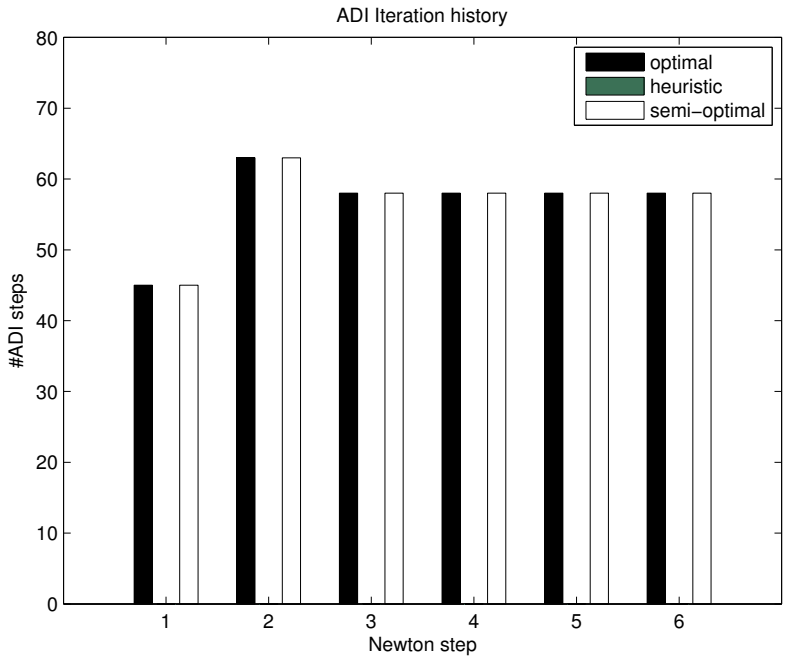


(c) Sparsity pattern of the Cholesky factor of reordered M

Figure 8.3.: The discrete operators for the tube/inflow example

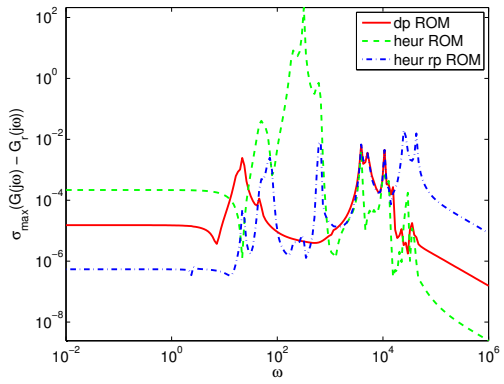


(a) Spectrum and computed shifts for the pencil (A, M) in (3.13)

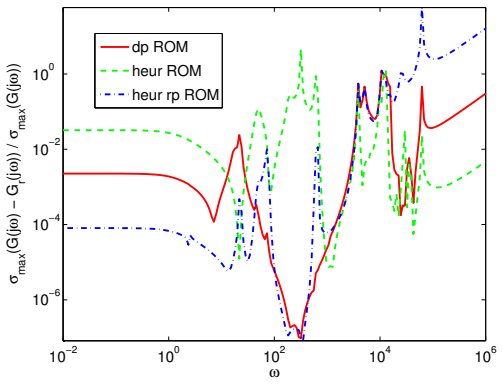


(b) Iteration history for the Newton ADI applied to (3.13)

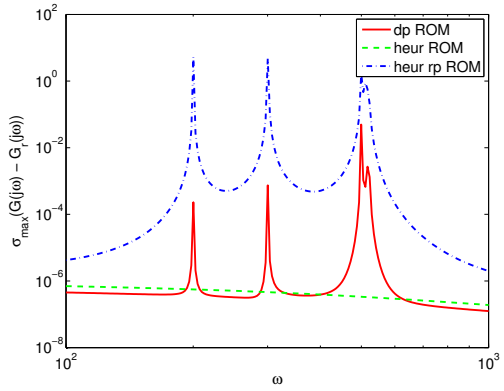
Figure 8.4.: ADI parameters and Newton-ADI iteration history for the tube example



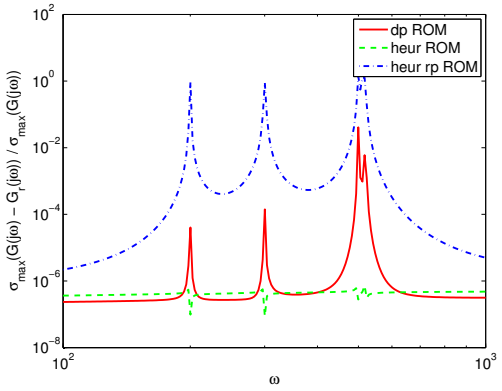
(a) CD Player: absolut error



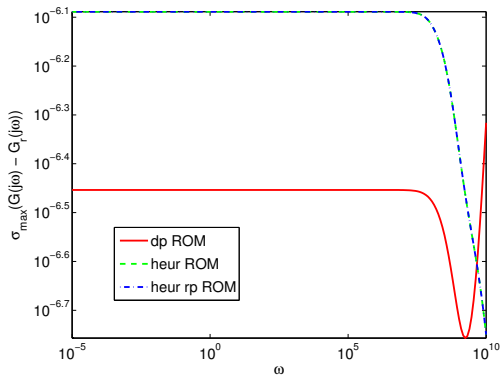
(b) CD Player: relative error



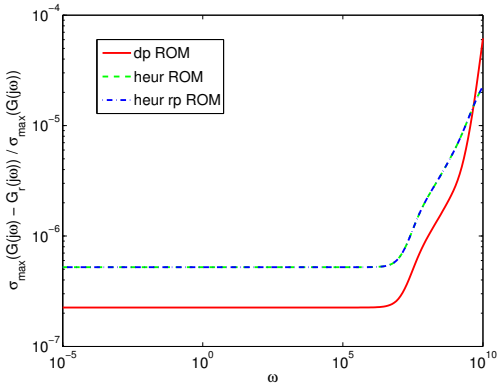
(c) Artificial: absolut error



(d) Artificial: relative error

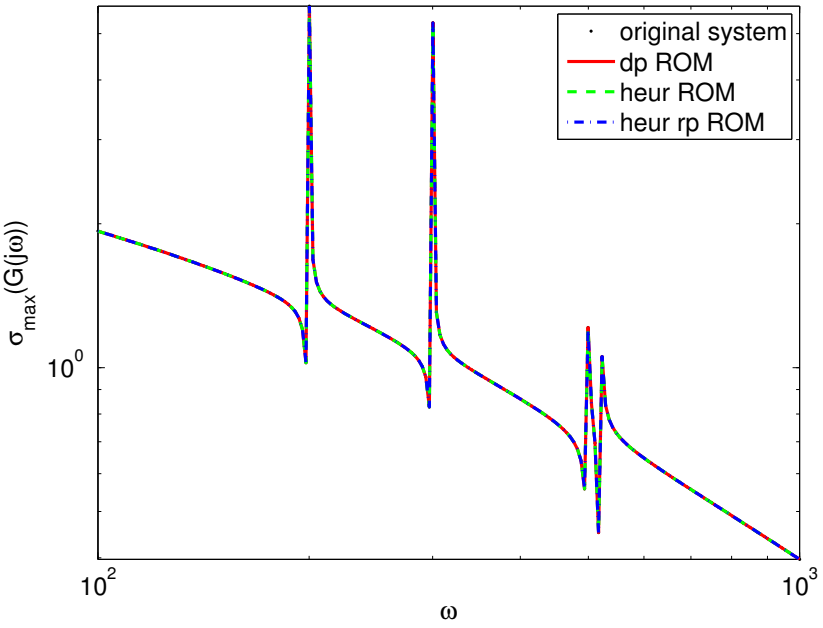


(e) Spiral inductor: absolut error

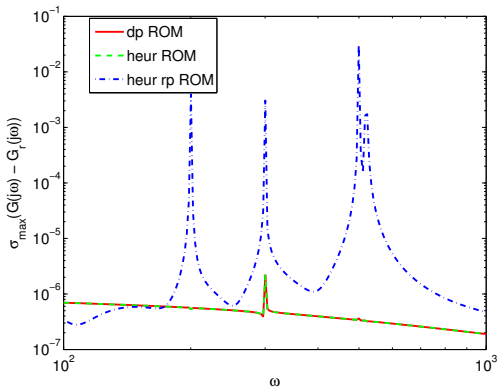


(f) Spiral inductor: relative error

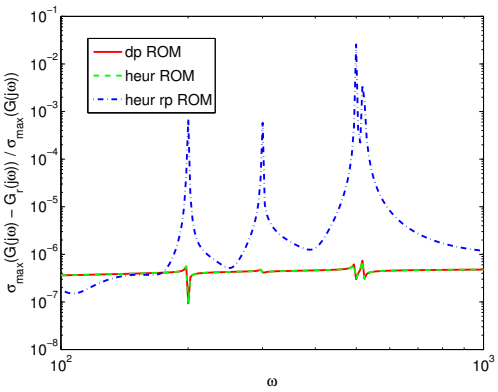
Figure 8.5.: Comparison of dominant pole (dp) based ADI shifts and heuristic based shifts. (heur rp shifts only take the real parts of the heur shifts into account to avoid complex computations)



(a) Bode plots



(b) Absolute errors



(c) Relative errors

Figure 8.6.: LR-SRM reduction of the artificial model with Galerkin projection in every fifth step of the LRCF-ADI

8.2. Accelerating Large Scale Matrix Equation Solvers

The tests for this section have been carried out in MATLAB 2009a on an Intel®Core™ 2 Quad CPU of type Q9400 running at 2.66GHz. MATLAB was running in multithreaded mode where possible. Our test system was equipped with 4GB of main memory and running in 64Bit mode such that these were fully available (although not necessary) in MATLAB.

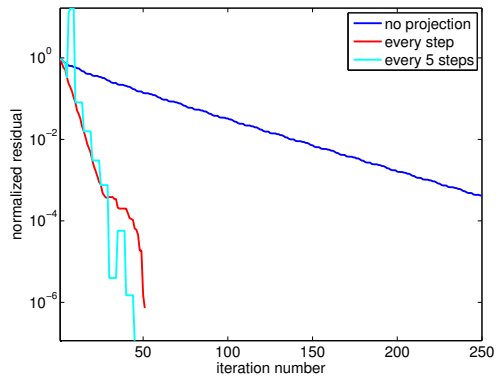
8.2.1. Accelerated Solution of Large Scale Sparse Lyapunov Equations

We demonstrate the efficiency of the Galerkin projection accelerated solution of large scale Lyapunov equations for the generalized Lyapunov equation case. The standard state space case is implicitly covered by the following section, where the same technique is applied to the inner iteration in the Newton-ADI method for the ARE. The two test examples shown in Figure 8.7 are differently sized discretizations of the steel profile model in Section 3.3. This model is especially tough for the projection approach since the costly orthogonalization has to be applied to a relatively high number of columns. The model is a MIMO system with six inputs and seven outputs. Thus in every step even in an optimal implementation (which we do not yet have) one has to apply the orthogonalization to multiple columns, amplifying the part of the step with the highest computational complexity even further. We can learn many things from these pictures. First it is important to note, how close the two projected lines stay to each other, which is a rather common observation we found in many examples. Second we find that the version where the projection is only performed in every fifth step sometime is converging even faster. Thus collecting certain amount of new subspace information seems to be helpful for the computations in finite arithmetics. As a third remark we find that in all cases the less frequent projected version is the fastest in terms of runtime, due to the lower cost compared to the one and better approximation feature compared to the other concurrent method.

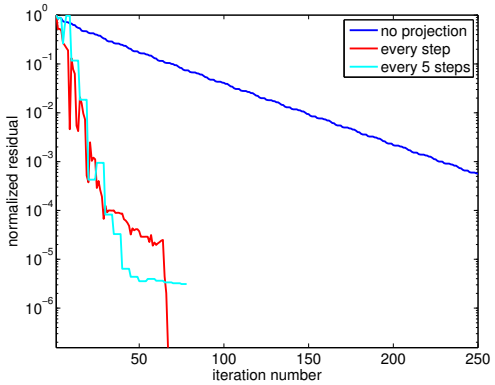
8.2.2. Accelerated Solution of Large Scale Sparse Algebraic Riccati Equations

Here we summarize the tests carried out for the Galerkin projection accelerated LRCF-NM as mentioned in Section 4.5.3. The results we show here are based on the FDM examples as presented in Section 3.1. Both models are of dimension $n = 10000$.

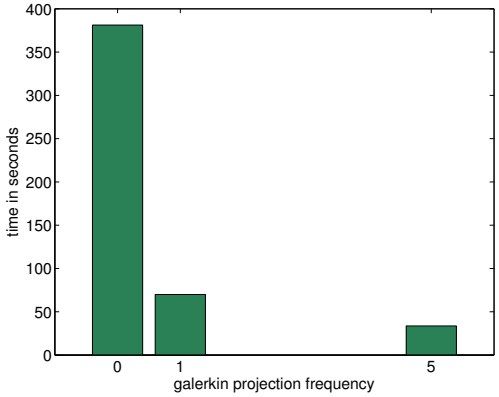
The software is a straight forward implementation of Algorithm 4.7 using Algorithm 4.3 in the inner loop. Heuristic shift parameters have been used. In both cases the 15 shifts have been chosen from 50 Ritz values with respect to the current closed loop operator and 25 for its inverse and updated in every Newton step. The outer Newton's method was stopped whenever either the relative change in the factor $\frac{\|Z_k - Z_{k-1}\|}{\|Z_k\|}$, or the current



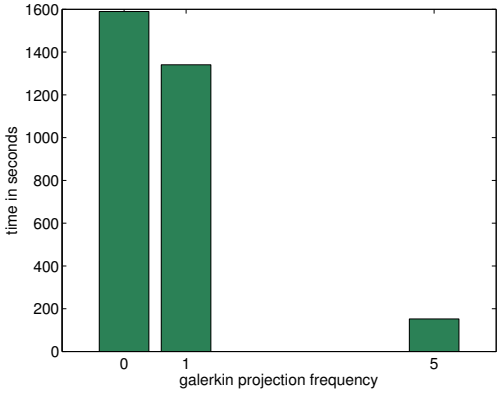
(a) Residual histories controllability LE (dimension 5177)



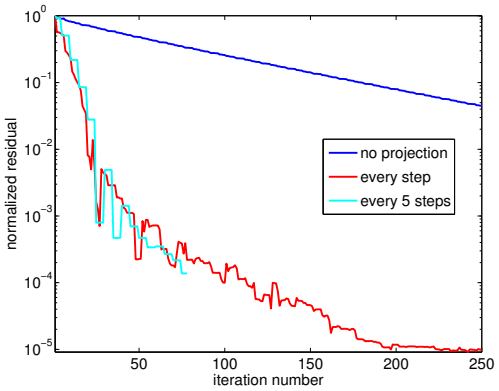
(b) Residual histories observability LE (dimension 5177)



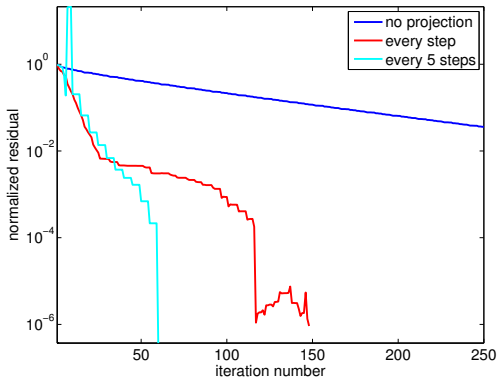
(c) Comparison of runtimes for different projection frequencies (dimension 5177)



(d) Comparison of runtimes for different projection frequencies (dimension 20209)

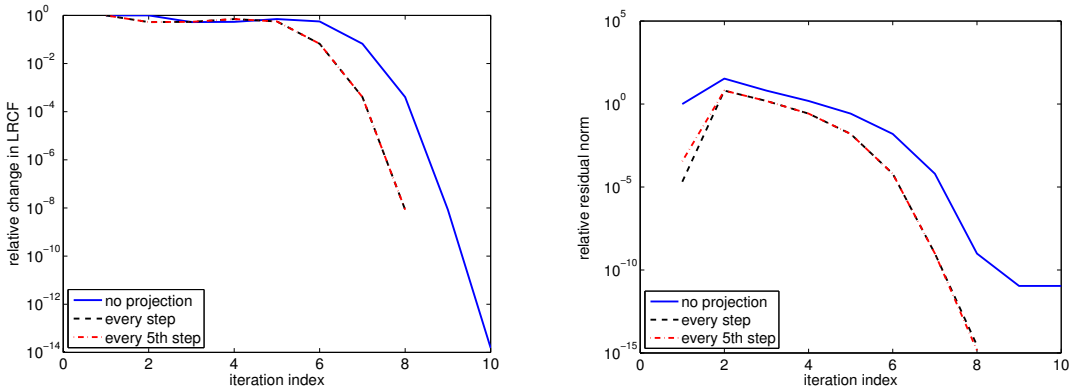


(e) Residual histories observability LE (dimension 20209)



(f) Residual histories controllability LE (dimension 20209)

Figure 8.7.: Galerkin projected solution or controllability and observability Lyapunov equations for the steel profile example in dimensions 5177 and 20209



(a) Relative change in low-rank factors (b) Relative ARE residual

Figure 8.8.: FDM 2d heat equation: LRCF-NM with Galerkin projection

normalized residual $\frac{\|\mathfrak{R}(ZZ^H)\|}{\|C^TC\|}$ was smaller than $n \cdot \texttt{eps}$. The inner ADI iteration is stopped whenever the normalized Lyapunov residual $\frac{\|FZZ^H+ZZ^HF^T+GG^T\|}{\|GG^T\|}$ is smaller than 10^{-10}. Additionally maximum iteration counts of 20 for the Newton iteration and 200 for the ADI process were applied. Section 4.6 explains how these stopping criteria can be evaluated inexpensively.

Galerkin Projection and FDM Semi-Discretized Heat Equation

projection	final ARE residual	final normalized ARE residual	runtime
0	7.608924e-08	1.086989e-11	76.91 seconds
1	2.000888e-11	2.858412e-15	39.62 seconds
5	1.000444e-11	1.429206e-15	38.00 seconds

Table 8.1.: FDM 2d heat equation: Comparison of LRCF-NMs with and without Galerkin projection

First we tested the heat equation without convection, i.e., the symmetric case, where real spectra and real arithmetics can be guaranteed. Table 8.1 shows the comparison of the attained residuals and especially the runtime of the different approaches. The *projection* column tells us how often the projection has been applied, here 0 means never, 1 stands for every step and 5 for every fifth step. Obviously we can compute a more accurate solution in roughly half the time using the projected methods. On the other hand due to the higher costs per iteration step for the projected versions, we do not gain anything when employing the projections to often. Our experience shows that applying this type of subspace optimization in every 5-th step is perfectly enough, which is also very well reflected in this example as we can see from the runtimes and also read off in Tables 8.2 to 8.4

step no.	rel. change in LRCF	rel. LE residual	#ADI iter.
1	1	9.999998e-01	200
2	9.999998e-01	3.405729e+01	23
3	5.249867e-01	6.370599e+00	20
4	5.371225e-01	1.523978e+00	20
5	7.034425e-01	2.639902e-01	23
6	5.573919e-01	1.564753e-02	23
7	6.589515e-02	6.296456e-05	23
8	4.024924e-04	9.681828e-10	23
9	8.452248e-09	1.087860e-11	23
10	1.518166e-14	1.086989e-11	23

Table 8.2.: FDM 2d heat equation: LRCF-NM without Galerkin projection

step no.	rel. change in LRCF	rel. LE residual	#ADI iter.
1	1	2.065203e-05	19
2	5.249864e-01	6.370682e+00	8
3	5.371212e-01	1.524009e+00	8
4	7.034422e-01	2.639984e-01	9
5	5.574055e-01	1.564843e-02	9
6	6.589897e-02	6.297180e-05	10
7	4.025390e-04	9.792769e-10	9
8	8.454352e-09	2.858412e-15	9

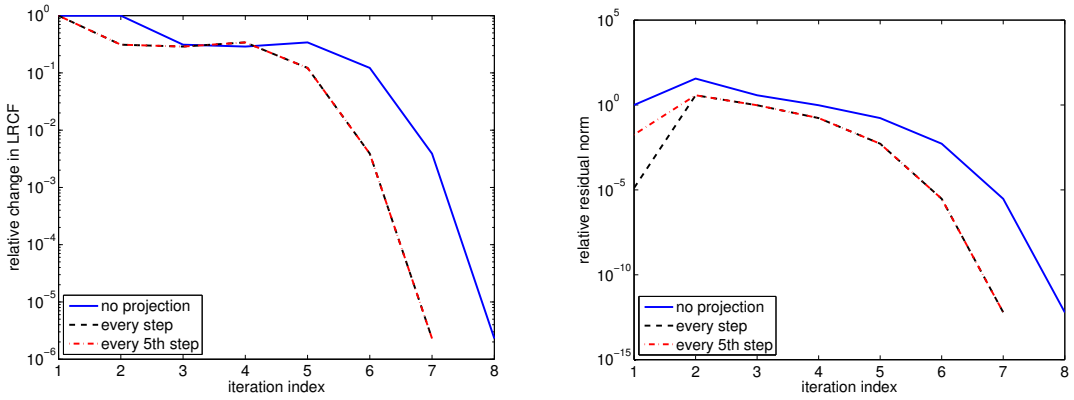
Table 8.3.: FDM 2d heat equation: LRCF-NM with Galerkin projection in every ADI step

Galerkin Projection and FDM Semi-Discretized Convection-Diffusion Equation

Extending the above tests to the non-symmetric case, i.e., adding convection to the heat equation shows very similar results as before. In Table 8.5 we see that the accuracy gain here is negligible, but we can reduce the computation times even slightly more than by a factor of two. Here we can also see that the time needed when projecting in every ADI step is significantly larger, supporting our proposition of a projection frequency of 5 steps. For very large systems even more steps might be taken between subsequent subspace optimizations, due to the fairly high cost for the orthogonalization employed in the projection process. This fact is also nicely reflected in Tables 8.6 to 8.8 again.

step no.	rel. change in LRCF	rel. LE residual	#ADI iter.
1	1	3.559496e-04	20
2	5.249864e-01	6.370682e+00	10
3	5.371210e-01	1.524009e+00	6
4	7.034428e-01	2.639984e-01	10
5	5.574053e-01	1.564842e-02	10
6	6.589894e-02	6.297178e-05	10
7	4.025388e-04	9.792784e-10	10
8	8.454155e-09	1.429206e-15	10

Table 8.4.: FDM 2d heat equation: LRCF-NM with Galerkin projection in every 5-th ADI step



(a) Relative change in low-rank factors

(b) Relative ARE residual

Figure 8.9.: FDM 2d convection-diffusiion equation: LRCF-NM with Galerkin projection

projection	final ARE residual	final normalized ARE residual	runtime
0	4.263711e-09	6.091016e-13	185.91 seconds
1	4.320100e-09	6.171571e-13	83.39 seconds
5	4.295543e-09	6.136491e-13	75.13 seconds

Table 8.5.: FDM 2d convection-diffusion equation: Comparison of LRCF-NMs with Galerkin projection

step no.	rel. change in LRCF	rel. LE residual	#ADI iter.
1	1	9.999999e-01	200
2	9.999999e-01	3.564285e+01	60
3	3.114304e-01	3.716254e+00	39
4	2.882760e-01	9.619409e-01	40
5	3.412568e-01	1.677661e-01	45
6	1.223042e-01	5.246287e-03	42
7	3.882804e-03	2.960483e-06	47
8	2.297297e-06	6.091016e-13	47

Table 8.6.: FDM 2d convection-diffusion equation: LRCF-NM without Galerkin projection

step no.	rel. change in LRCF	rel. LE residual	#ADI iter.
1	1	1.293249e-05	33
2	3.114300e-01	3.716225e+00	16
3	2.882755e-01	9.619435e-01	16
4	3.412566e-01	1.677680e-01	16
5	1.223057e-01	5.246422e-03	17
6	3.882904e-03	2.960637e-06	16
7	2.297416e-06	6.171571e-13	16

Table 8.7.: FDM 2d convection-diffusion equation: LRCF-NM with Galerkin projection in every ADI step

step no.	rel. change in LRCF	rel. LE residual	#ADI iter.
1	1	1.781820e-02	35
2	3.114300e-01	3.716225e+00	15
3	2.882755e-01	9.619435e-01	20
4	3.412566e-01	1.677680e-01	15
5	1.223057e-01	5.246422e-03	20
6	3.882904e-03	2.960637e-06	15
7	2.297416e-06	6.136491e-13	20

Table 8.8.: FDM 2d convection-diffusion equation: LRCF-NM with Galerkin projection in every 5-th ADI step

A Maximum Size Example.

projection	ARE residual	normalized ARE residual	runtime	num. steps
0	8.831953e-05	1.261708e-10	70h	15
5	3.154855e-07	4.506936e-13	82h	14

Table 8.9.: FDM 2d convection-diffusion equation: Comparison of LRCF-NMs with Galerkin projection (dimension 10^6)

As a benchmark for the maximum size computable we tested a Riccati equation for the FDM 2d convection-diffusion equation choosing dimension 10^6. The computations have been carried out in 64Bit MATLAB on the main compute server of MRZ. The machine is a dual CPU dual Core Xeon® 5160 equipped with 64GB RAM. The maximum memory requirement did not exceed 8GB during the computation, though. Therefore this should be considered the largest computable size on 8 to 16GB computers.

Unfortunately the compute server could not be used exclusively for these tests and thus the computation times in Table 8.9 should not be taken to strictly. The main message here is that a Riccati equation of dimension 10^6 could be solved within 70-82 hours using the LRCF-NM. Also, the Galerkin projected version here again gives much better results considering the accuracy of the results in less Newton iteration steps. Note that the 0 and 5 in the leftmost column again represent the no "projection"and "projection in every fifth step" cases, as in the previous examples.

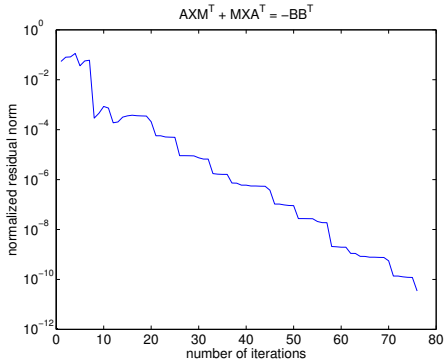
8.3. Model Order Reduction

8.3.1. Reduction of First Order Systems

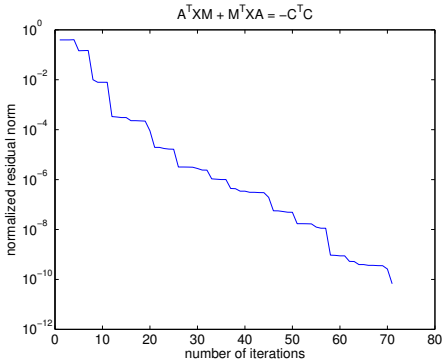
Although the direct contributions in this thesis to the field of MOR for first order systems are rather limited, the new results and techniques for the Lyapunov solvers can be employed in the context of the low-rank square root method (LR-SRM). As a case study we chose the rail model (Section 3.3) of dimension 79841. We compared the LR-SRM using the G-LRCF-ADI without extension with the approaches where first column compression via RRQR is added and then additionally the Galerkin projection acceleration is used. Both extensions are applied in every fifth ADI step. In Figure 8.10

equation	G-LRCF-ADI	G-LRCF-ADI + CC	G-LRCF-ADI + CC + GP
$AXM^T + MXA^T = -BB^T$	622.00 sec	798.11 sec	616.58 sec
$A^TXM + M^TXA = -C^TC$	353.70 sec	489.59 sec	409.11 sec

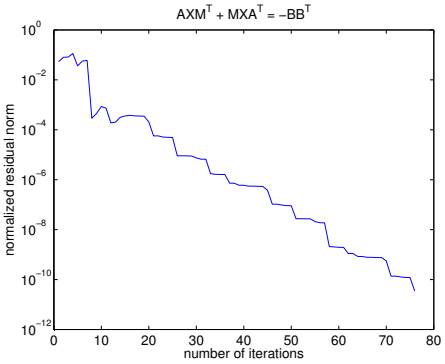
Table 8.10.: Execution times for the G-LRCF-ADI with and without acceleration techniques for the two Lyapunov equations



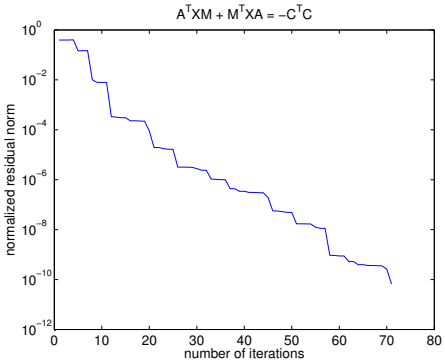
(a) Controllability Lyapunov equation: sole G-LRCF-ADI



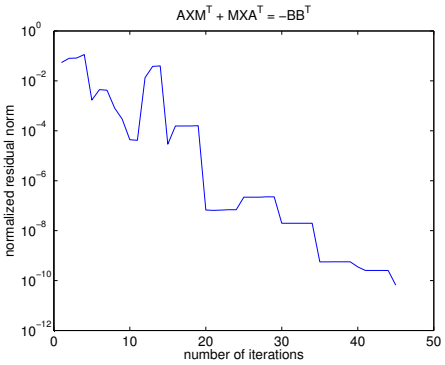
(b) Observability Lyapunov equation: sole G-LRCF-ADI



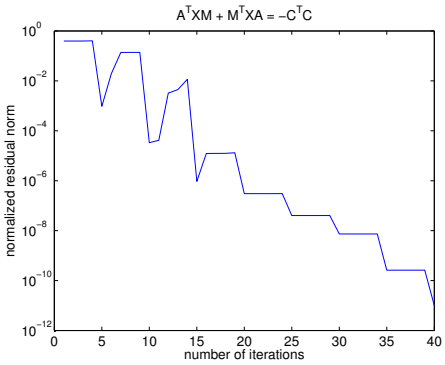
(c) Controllability Lyapunov equation: G-LRCF-ADI + column compression



(d) Observability Lyapunov equation: G-LRCF-ADI + column compression

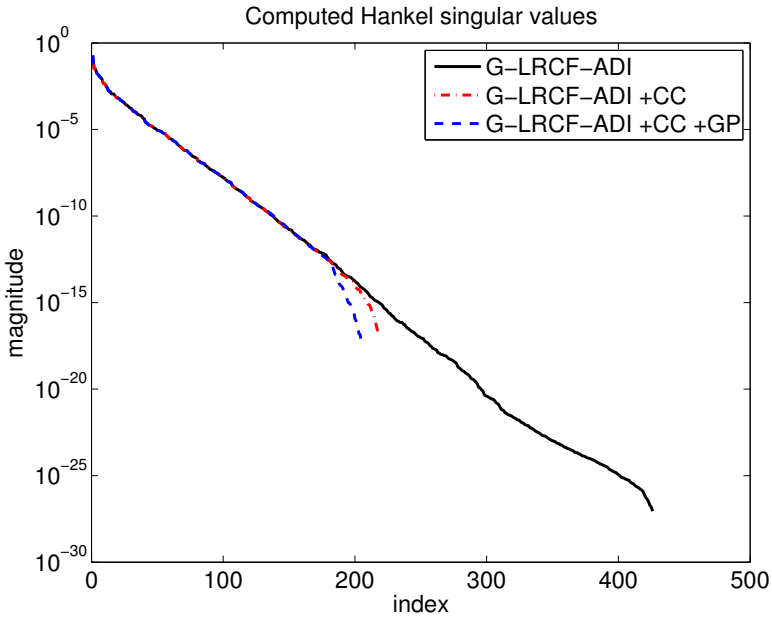


(e) Controllability Lyapunov equation: G-LRCF-ADI + column compression and projection acceleration

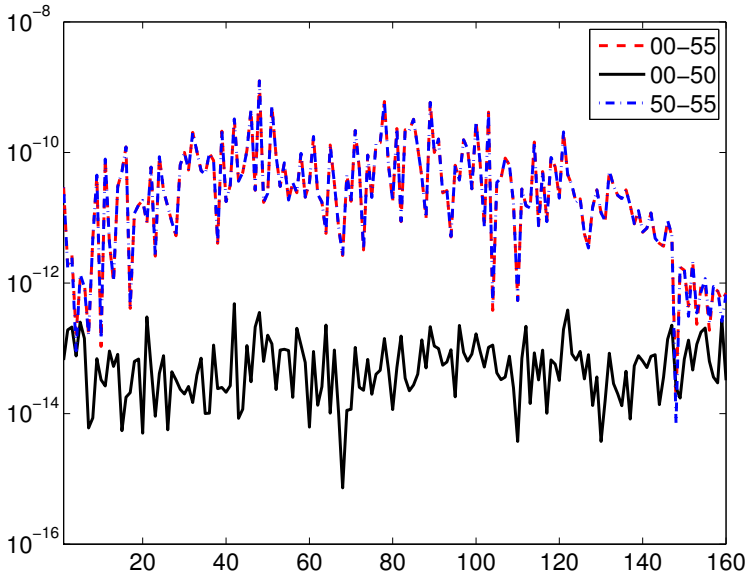


(f) Observability Lyapunov equation: G-LRCF-ADI + column compression and projection acceleration

Figure 8.10.: Comparison of G-LRCF-ADI iteration histories with and without acceleration features for the steel profile example (dimension 79841)

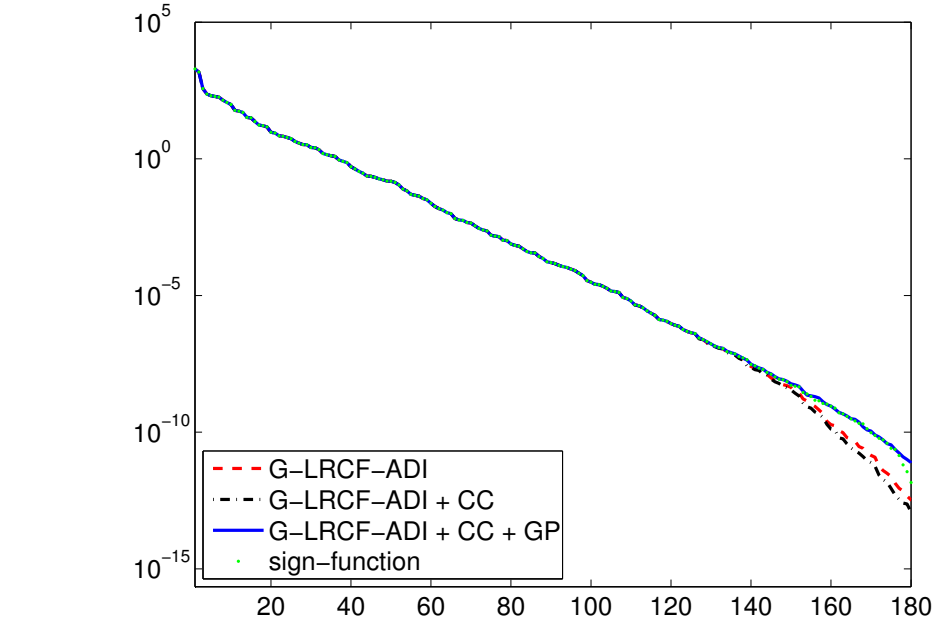


(a) absolute values of the computed HSVs
(CC=column compression; GP=Galerkin projection)

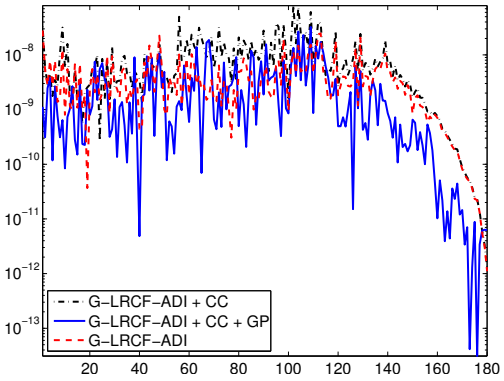


(b) absolute pointwise differences of the computed HSVs
(00=no acceleration;
50=column compression in every fifth step;
55=compression and projection in every fifth step)

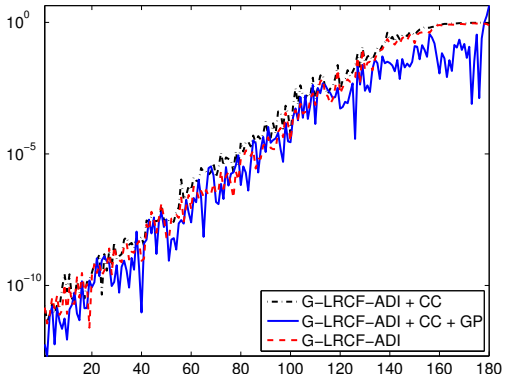
Figure 8.11.: Comparison of HSVs computed with and without acceleration features in G-LRCF-ADI for the steel profile example
(dimension 79841)



(a) Hankel singular values computed from Gramian factors calculated via G-LRCF-ADI with and without accelerations and the matrix sign function approach

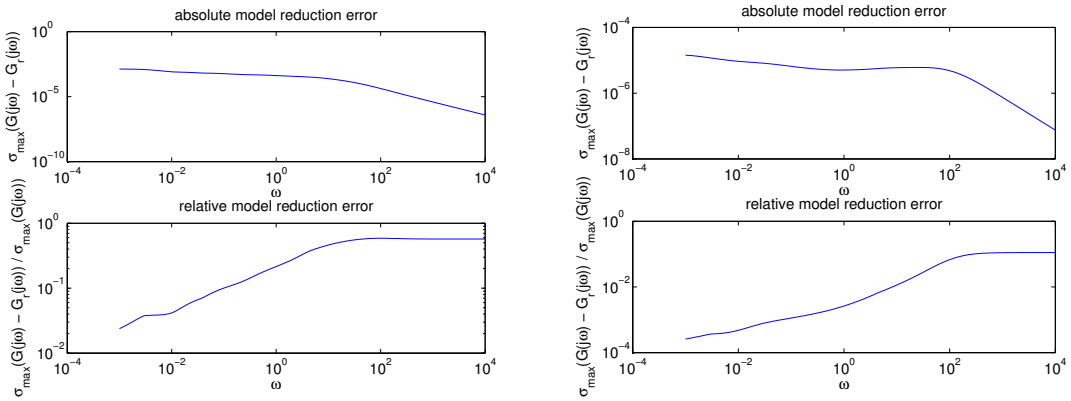


(b) Absolute deviation of the computed Hankel singular values from those computed via the sign function method



(c) Relative deviation of the computed Hankel singular values from those computed via the sign function method

Figure 8.12.: Comparison of Hankel singular value qualities
(CC=column compression; GP=Galerkin projection)

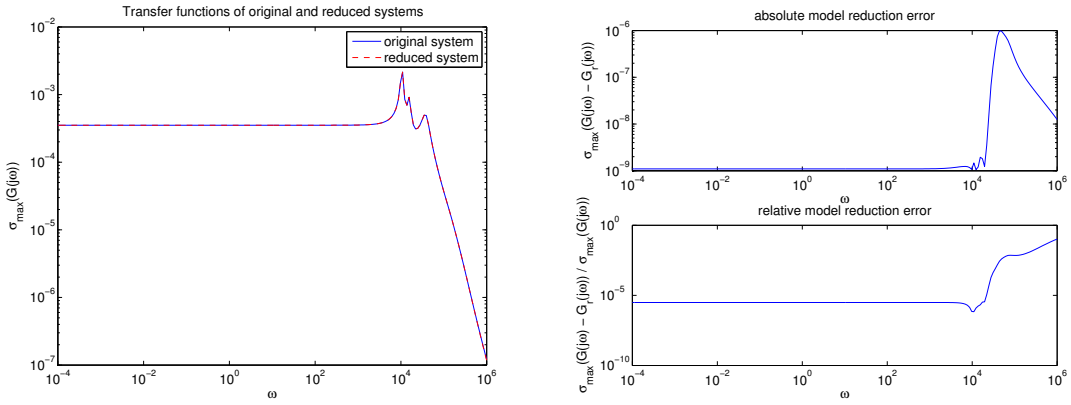


(a) ROM order 20 (b) ROM for error tolerance 10^{-4}

Figure 8.13.: Absolute and relative errors of ROMs for the steel profile example (dimension 79841)

the iteration histories are shown. Figure 8.11 shows a comparison of the computed Hankel singular value decays calculated from the resulting factors. Also we see that the deviation of the first 160 HSVs is very small (especially compare Figure 8.11b). Figure 8.12 shows the results of a similar computation for the rail model of dimension 5177. There we additionally compared the resulting Hankel singular values with the ones computed via the sign function iteration [22] applied for the Gramian computation. The results found there motivate the assumption that the HSV computed via the factors from the G-LRCF-ADI including Galerkin projection are the most accurate in 8.11 as well. Note that we restricted the presentation to the first 180 HSVs that had been computed by all approaches.

The computation times for the dimension 79841 model are collected in Table 8.10. What we learn from the table is that we can save something, but have to choose the acceleration carefully. Also we learn that iteration numbers are only half the bill. Comparing the runtimes to the figures we see that although we save almost half the iteration steps for both equations we do not save as much time due to the expensive orthogonalization involved in both the RRQR and the Galerkin projection. Also the current implementation is still somewhat experimental and does perform the orthogonalization in both techniques separately. Therefore especially when both are applied better runtimes can be achieved in combining them, which should lead to additional time savings. That we can in fact save a lot of time has already been shown in Table 8.1 where the projection has been applied in the inner iteration of the Newton's method. We omit the comparison of the error– and Bode plot here to save some space, since they show no visible differences anyway. Instead we just present (see Figure 8.13) representative error plots, for a reduction to an order 20 model and a reduction to an error bound of 10^{-4}.



(a) Bode plot (b) Error plots

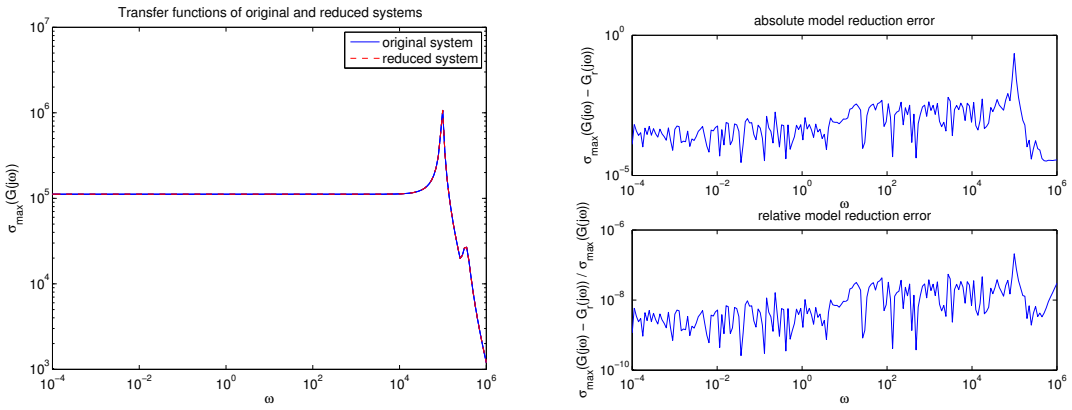
Figure 8.14.: Second order to first order reduction results for the Gyro example

8.3.2. Reduction of Second Order Systems to First Order ROMs

The Butterfly Gyro

The ROM here is of order 18 and has been computed in roughly 1 hour including pre- and postprocessing. Here, preprocessing means assembly of the first order original system and computation of the shift parameters. We used 20 parameters following the heuristic parameter choice as proposed by Penzl (Section 4.3.1). Postprocessing is the computation of the original and reduced order Bode plots, as well as the absolute and relative approximation errors at 200 sampling points. In comparison, the computation of an order 45 ROM on 256 nodes of the CHiC (Chemnitz Heterogeneous Linux Cluster[4]) using PLICMR[5] [25, 24] (which uses a sign function based solver for the Lyapunov equations) plus postprocessing on the same Xeon machine as above can be done in roughly half the time, but results show slightly worse numerical properties, i.e., errors there are slightly larger.

We used a truncation tolerance of 10^{-8} which is not met everywhere in Figure 8.14b. This is due to the LRCFs computed. The order of the ROM is limited by the ranks of those factors. In this example the computation was stopped by a maximum iteration number bound, such that the factors had not fully converged. Therefore not all Hankel singular values are available and the balanced truncation error is enlarged by the error resulting from the truncation of the LRCF computation. It is a common observation, that this additional error normally affects the higher frequencies.



(a) Bode plot (b) Error plots

Figure 8.15.: Second order to first order results for the acceleration sensor example

method	largest HSV
position	$0.3580 \cdot 10^{-12}$
velocity	$0.3457 \cdot 10^{-12}$
position-velocity	$0.3556 \cdot 10^{-17}$
velocity-position	$0.3481 \cdot 10^{-7}$

Table 8.11.: Largest Hankel singular values for the different second order balancing approaches in [124] for the acceleration sensor example

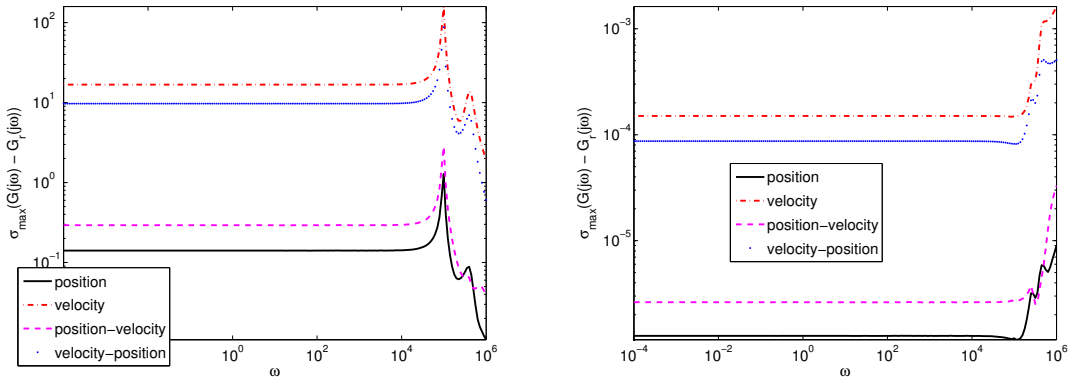
Fraunhofer/Bosch Acceleration Sensor

The single computation steps here are the same as for the gyro example. Here we used 25 shift parameters and the computation took about 1 hour as in the gyro example. The larger dimension is compensated by the fact that here especially the controllability Gramian factor computation converged in only 23 steps to the required accuracy. The ROM, for which the results can be found in Figure 8.15, is of order 25. Here we cannot provide a comparison with the CHiC experiments, since PLICMR crashed for this model due to memory allocation errors in the required ScaLAPACK routines.

8.3.3. Reduction of Second Order Systems to Second Order ROMs

In Section 7.2 we noted that one of the major drawbacks in handling second order systems with balancing based MOR algorithms is that the guaranteed error bound for the approximation error in terms of the transfer function norm is lost. An unfortunate side-effect of this problem is that it can therefore no longer be used to choose the

[4] http://www.tu-chemnitz.de/chic/
[5] http://www.pscom.uji.es/modred/



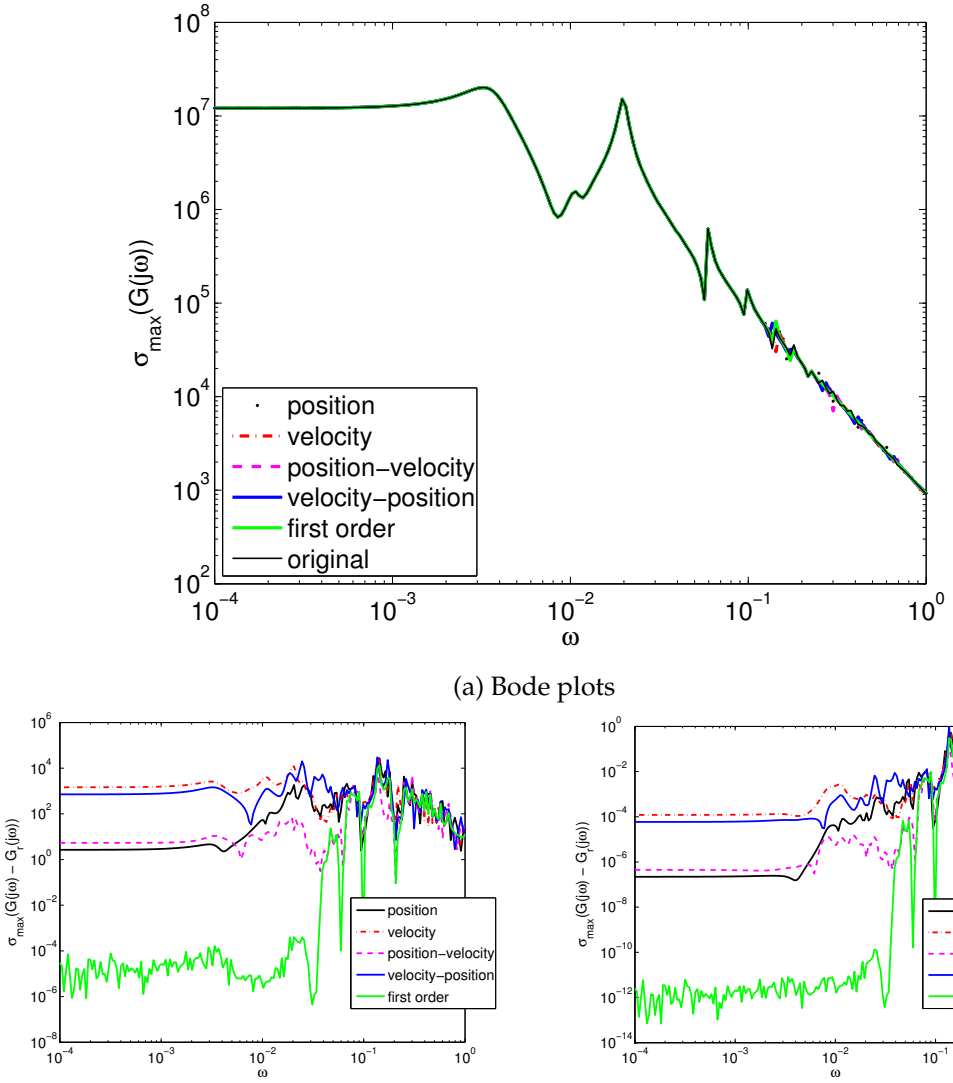
(a) Absolute errors

(b) Relative errors

Figure 8.16.: A comparison of the different second order to second order balancing approaches in [124] for the acceleration sensor example with fixed ROM order 20

reduced model order from the prescribed error tolerance automatically, as well. This problem is not only a theoretical issue, but is observed in concrete applications. For the acceleration sensor (Section 3.9) for example, the reduced system order resulting from this mechanism when position-velocity balancing is applied to the low-rank factors according to the description in Section 7.2.2 is 2 and the ROM does not even barely match the original system behavior. Table 8.11 shows that the largest Hankel singular values for all approaches are already smaller than any realistic error bound one would prescribe. On the other hand prescribing an order 20 for the ROM we end up with a relative error that is smaller than 10^{-4} everywhere at least for two of the approaches. In Figure 8.16 we compare the absolute and relative MOR errors for the four approaches from [124, Definition 2.1] for a ROM of order 20. Note that all of them can be and have been computed from the same Gramian factors generated by Algorithm 7.3 using the representation of the matrix operations from Section 7.2.1 Table 7.2. Since solving the Lyapunov equations is the expensive step in the square root method, we can cheaply compare all methods and use the best result in applications. The Bode plot can not be distinguished visually from the one given in Figure 8.15a and is therefore omitted. Two remarks have to be stated regarding Figure 8.16. First we see that the position-velocity reduced model gives the second best result. Note that this is the model for which [124] proves stability preservation in the symmetric case with positive definite system matrices. Second we observe that the results are ordered as position, position-velocity, velocity-position, velocity from best to worst comparing the plots. That means we see that with rising influence of the velocity part the result is getting worse. Now recalling that the velocity is manipulated by the mass matrix when rewriting the second order system in first order form, we conclude, that we see the influence of the large condition number of the mass matrix here.

As we have mentioned earlier, many software packages in controller design still expect

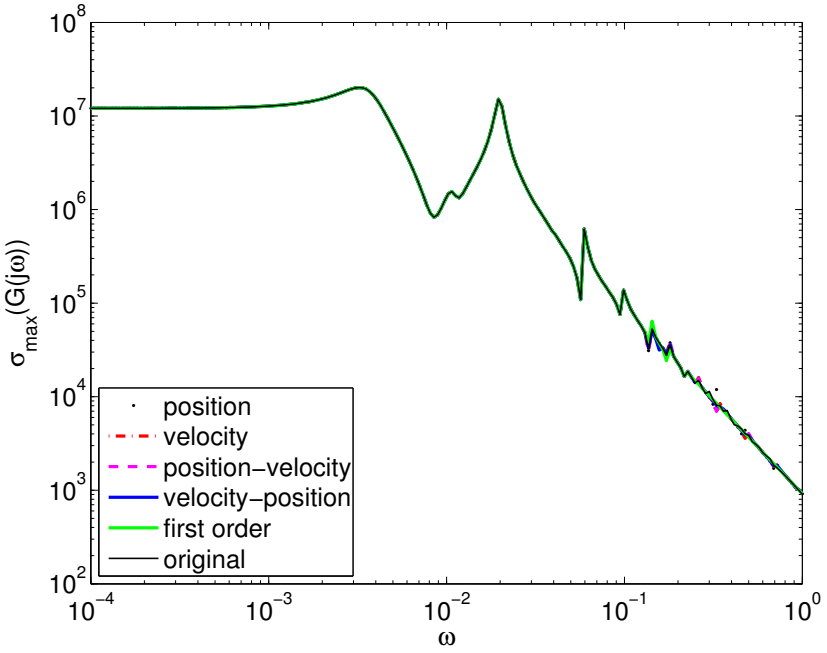


(a) Bode plots

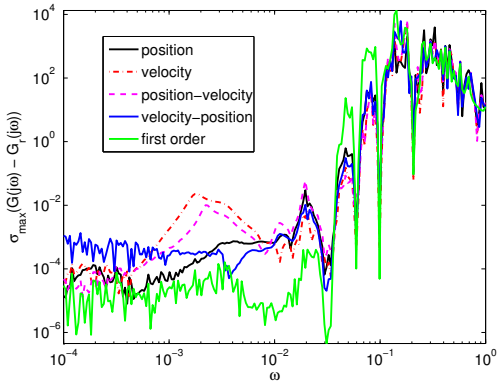
(b) Absolute errors

(c) Relative errors

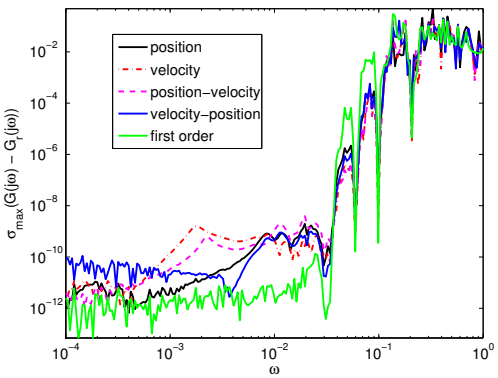
Figure 8.17.: A comparison of the different second order to second order balancing approaches in [124] for the triple chain oscillator example with fixed ROM order 75. (ROM order 150 for the first order ROM)



(a) Bode plots



(b) Absolute errors



(c) Relative errors

Figure 8.18.: A comparison of the different second order to second order balancing approaches in [124] for the triple chain oscillator example with fixed ROM order 150

the system used for the computations to be in first order form. Therefore engineers in practice need first order models for these applications. Now applying the second order to second order reduction in this context will require that we rewrite the second order ROM in first order form. Thus we need to compare the results to a double size first order ROM in this context. Figure 8.17 shows such a comparison for the triple chain coupled oscillator model with $n_1 = 500$ and thus an original model order of 1501. In the figure we compared an order 150 first order ROM computed as in the previous section with the four second order reduced models of order 75. The same comparison has been carried out for Figure 8.18 where all ROMs are of order 150. All computations for the triple chain oscillator have been carried out oa an Intel® Pentium® M Laptop processor with 2.00GHz and 1GB RAM summing up to less than 5 minutes computation time altogether (including the Bode plot sampling at 200 sampling frequencies).

8.4. Comparison of the MATLAB and C Implementations

Matrix	Dimension	non-zeroes Entries	avg. entries per row	mem. usage CRS (32Bit)	mem. usage CRS (64Bit)
DW2048	2 048	10 114	4.94	126.527 kB	174.039 kB
DW8192	8 192	41 746	5.10	521.215 kB	716.289 kB
AF23560	23 560	484 256	20.55	5.632 MB	7.568 MB
E40R5000	17 281	553 956	32.10	6.405 MB	8.585 MB
FIDAPM11	22 294	623 554	27.97	7.221 MB	9.685 MB
FIDAPM37	9 152	765 944	83.69	8.800 MB	11.757 MB
FIDAP011	16 614	1 091 362	65.69	12.553 MB	16.780 MB
SME3DB	29 067	2 081 063	71.60	23.927 MB	31.976 MB
TORSO1	116 158	8 516 500	73.32	97.907 MB	130.838 MB

Table 8.12.: Non-symmetric test matrices and their properties (CRS=Compressed Row Storage)

In this Section we collect some benchmarks comparing MATLAB and C implementations of the algorithms from Chapter 4. Since comparing the Intel and AMD architectural performance influences was a second task of the benchmarks, we will first introduce the two test systems on which all benchmarks were produced. The two test systems are `romulus` and `remus` from the compute server pool of the MRZ (**M**athematisches **R**echen**z**entrum) at the Faculty of Mathematics of TU Chemnitz. Named after the mythologic twin brothers, these computers are almost twins themselves. Both are equipped with the same Hardware as far as possible. They include the same twin SATA harddiscs and both have 16 GB RAM (PC2-5300 DDR2-667ECC). The only difference lies in the computational core hardware. `romulus` is based on 2 Intel® Xeon® CPUs of type 5160 running at 3.0 GHz. Both of these consist of two cores, such that one can access four virtual processors from the Linux (X86-64) operating system. `remus` on the

	# Threads	1	2	4	8
Matrix	System				
DW2048	remus	0.034 (1.00)	**0.036 (0.96)**	**0.025 (1.36)**	**0.046 (0.74)**
	romulus	0.022 (1.00)	0.031 (0.71)	0.029 (0.76)	0.048 (0.46)
DW8192	remus	0.28 (1.00)	**0.15 (1.83)**	**0.09 (3.12)**	**0.13 (2.24)**
	romulus	0.09 (1.00)	0.087 (1.04)	0.06 (1.62)	0.07 (1.31)
AF23560	remus	2.61 (1.00)	1.93 (1.35)	2.41 (1.08)	**1.68 (1.56)**
	romulus	2.74 (1.00)	**0.78 (3.52)**	**1.40 (1.96)**	1.90 (1.44)
E40R5000	remus	3.00 (1.00)	2.17 (1.37)	1.87 (1.60)	1.90 (1.57)
	romulus	2.83 (1.00)	**1.07 (2.63)**	**0.72 (3.93)**	**1.34 (2.10)**
FIDAPM11	remus	3.41 (1.00)	2.50 (1.36)	2.82 (1.20)	2.14 (1.59)
	romulus	3.49 (1.00)	**1.50 (2.32)**	**1.30 (2.68)**	**1.79 (1.95)**
FIDAPM37	remus	3.96 (1.00)	3.05 (1.29)	2.58 (1.53)	2.61 (1.51)
	romulus	3.85 (1.00)	**1.98 (1.94)**	**2.49 (1.54)**	**2.13 (1.80)**
FIPAP011	remus	5.67 (1.00)	4.18 (1.35)	4.00 (1.41)	**3.61 (1.57)**
	romulus	5.70 (1.00)	**3.31 (1.72)**	**3.08 (1.85)**	3.68 (1.54)
SME3DB	remus	13.72 (1.00)	8.70 (1.57)	**6.72 (2.04)**	**7.07 (1.94)**
	romulus	11.89 (1.00)	**7.12 (1.66)**	6.48 (1.83)	7.30 (1.62)
TORSO1	remus	46.53 (1.00)	34.50 (1.34)	30.65 (1.51)	28.59 (1.62)
	romulus	43.60 (1.00)	**27.65 (1.57)**	**27.82 (1.56)**	**26.13 (1.66)**

Table 8.13.: Runtime and speedup measurements using OpenMP (Bold face entries mark best speedups)

other hand is equipped with two Dual-Core AMD Opteron™ 2218 Processors running at 2.6 GHZ. The main difference besides the slightly differing cpu speed is the differing level 2 cache size of 1MB for AMD and 4MB on the Intel system. A second important fact is that in the AMD case every physical processor is exclusively attached to half of the RAM. On the Intel base all cores share the same cache hierarchy. Besides its own memory each AMD processor can also access the memory of the other processor, but this access is comparably slow since it has to pass by the other processor. The maximum over all memory transfer rates of both systems on the other hand coincide again. This makes those 2 machines the perfect platform for a comparison of the 2 architectures.

8.4.1. Shared Memory Parallelization

The first test considers shared memory parallel computation of matrix vector products. The test runs compare different matrix dimensions. Starting from very small matrices the size is successively increased until first the cache size of the Opteron™s and finally even the cache size of the Xeon®s is exceeded. The test is performed with the easy to implement OpenMP shared memory parallelization paradigm, as well as with the more involved MPI standard (using the OpenMPI implementation) known from distributed memory parallelization. The test matrices for the results presented here are all non-symmetric and most of them are available on Matrix Market[6]. The only exceptions are the TORSO1 and SME3DB matrices available from the University of Florida

[6] http://math.nist.gov/MatrixMarket/

	#Threads	1	2	4	8
Matrix	System				
DW2048	remus	0.034 (1.00)	0.084 (.41)	**0.12 (.27)**	0.42 (.08)
	romulus	0.023 (1.00)	**0.053 (.43)**	0.09 (.26)	**0.20 (.11)**
DW8192	remus	0.28 (1.00)	**0.30 (.94)**	**0.39 (.70)**	**1.32 (.21)**
	romulus	0.09 (1.00)	0.23 (.39)	0.36 (.25)	0.82 (.11)
AF23560	remus	2.62 (1.00)	2.33 (1.12)	**2.03 (1.28)**	**3.90 (.67)**
	romulus	2.74 (1.00)	**1.62 (1.69)**	2.44 (1.12)	4.69 (.58)
E40R5000	remus	2.98 (1.00)	2.31 (1.29)	**1.80 (1.65)**	**3.24 (.91)**
	romulus	2.81 (1.00)	**1.57 (1.78)**	2.13 (1.32)	3.85 (.73)
FIDAPM11	remus	3.34 (1.00)	2.67 (1.25)	**2.14 (1.56)**	**3.97 (.84)**
	romulus	3.51 (1.00)	**2.20 (1.59)**	2.80 (1.25)	5.10 (.68)
FIDAPM37	remus	3.97 (1.00)	2.61 (1.51)	**1.65 (2.40)**	**2.72 (1.46)**
	romulus	3.85 (1.00)	**2.15 (1.78)**	2.28 (1.68)	3.25 (1.18)
FIPAP011	remus	5.68 (1.00)	**3.76 (1.51)**	**2.50 (2.27)**	**4.18 (1.35)**
	romulus	5.72 (1.00)	**3.78 (1.51)**	4.16 (1.37)	5.41 (1.05)
SME3DB	remus	13.74 (1.00)	**8.44 (1.62)**	**5.23 (2.62)**	**7.63 (1.80)**
	romulus	11.80 (1.00)	7.90 (1.49)	8.08 (1.46)	10.57 (1.11)
TORSO1	remus	47.03 (1.00)	**29.73 (1.58)**	**18.50 (2.54)**	**32.59 (1.44)**
	romulus	43.79 (1.00)	30.70 (1.42)	33.99 (1.28)	44.88 (.97)

Table 8.14.: Runtime and speedup measurements using OpenMPI (Bold face entries mark best speedups)

Sparse Matrix Collection[7] maintained by Tim Davis. Table 8.12 lists the matrices and their properties. The runtimes and speedups given in Tables 8.13 and 8.14 have been averaged over repeated executions (20 runs with 1000 matrix vector multiplications each) to minimize the influence of side effects caused by the operating system. The non-bracketed numbers represent computation times in seconds. The bracketed numbers are the speedups compared to the single threaded case. We also tested 8 threads on the machines with only 4 virtual CPUs to demonstrate that further increases of thread numbers normally lead to blocking effects that increase the CPU time and decrease

[7] http://www.cise.ufl.edu/research/sparse/matrices/

Matrix	max. speedup	# threads	method	machine type
DW2048	1.36	4	OpenMP	remus / AMD
DW8192	3.12	4	OpenMP	remus / AMD
AF23560	1.96	4	OpenMP	romulus / Intel
E40R5000	3.93	4	OpenMP	romulus / Intel
FIDAPM11	2.68	4	OpenMP	romulus / Intel
FIDAPM37	2.40	4	OpenMPI	remus / AMD
FIDAP011	2.27	4	OpenMPI	remus / AMD
SME3DB	2.62	4	OpenMPI	remus / AMD
TORSO1	2.54	4	OpenMPI	remus / AMD

Table 8.15.: Maximum speedups per matrix

the speedup. Table 8.15 summarizes the best results we could achieve with both approaches. It shows on which machine (i.e., architecture) they have been found and which parallelism paradigm was used.

The tests shown here are only an excerpt of a technical report [84] scheduled for autumn 2009. The main purpose of this section is to demonstrate that exploiting the parallelization properties does make sense in sparse computations although one should not expect better speedups than on distributed memory machines. Note also that the test systems are dual processor dual core computers. On a single processor machine with even more cores sharing the same cache hierarchy, one must expect even smaller speedups. For dense computations with higher complexity on the other hand we expect better speedups due to less data collisions on the cache hierarchy.

dim.	L+U	16 LUs	spmv LU 16 Shifts	savings
100	0.016	0.256	0.168	35.97%
625	0.154	2.464	1.608	34.74%
2 500	0.860	13.760	9.060	34.16%
10 000	4.800	76.800	52.025	32.26%
40 000	25.060	401.000	272.698	32.00%
90 000	67.700	1 083.200	737.884	31.88 %
160 000	130.840	2 094.440	1 427.000	31.87%
250 000	212.840	3 405.440	2 322.260	31.81 %
562 500	536.280	8 580.480	5 854.100	31.81%
1 000 000	1 030.100	16 481.600	11 251.600	31.73%

Table 8.16.: Comparison of memory consumptions using standard and single-pattern–multi-value (spmv) LU on 32bit (all sizes in MB and using AMD reordering)

dim.	L+U	16 LUs	spmv LU 16 Shifts	savings
100	0.022	0.352	0.170	51.70%
625	0.208	3.341	1.661	50.28%
2 500	1.162	18.592	9.354	49.69%
10 000	6.450	103.200	53.675	47.99%
40 000	33.620	537.920	281.257	47.71%
90 000	90.750	1452.000	760.910	47.60 %
160 000	175.280	2 804.500	1 471.500	47.53%
250 000	285.100	4 561.300	2 394.500	47.5 %
562 500	718.000	11 488.000	6 038.000	47.44%
1 000 000	1 379.000	22 064.000	11 604.000	47.41%

Table 8.17.: Comparison of memory consumptions using standard and single-pattern–multi-value (spmv) LU on 64bit (all sizes in MB and using AMD reordering)

8.4.2. Timings C.M.E.S.S. vs. M.E.S.S.

	C.M.E.S.S.			M.E.S.S.		LyaPack	
dim.	1st LU	other LUs	total	total	ratio	total	ratio
100	0.00	0.00	0.02	0.16	6.89	0.12	5.45
625	0.00	0.01	0.04	0.23	5.44	0.10	2.48
2 500	0.01	0.03	0.16	0.99	6.24	0.70	4.43
10 000	0.11	0.13	0.97	5.64	5.85	6.22	6.45
40 000	1.31	1.25	11.09	34.56	3.12	71.48	6.44
90 000	6.36	10.68	34.67	90.49	2.61	418.55	12.07
160 000	19.25	37.75	109.32	219.91	2.01	–	–
250 000	44.60	68.05	193.67	403.76	2.08	–	–
562 500	250.78	295.54	930.14	1216.69	1.31	–	–
1 000 000	1130.99	753.36	2219.95	2428.64	1.09	–	–

Table 8.18.: Runtime comparison C.M.E.S.S. versus M.E.S.S. versus LyaPack (times in seconds, – : out of memory)

Here we compare the implementations of Algorithm 4.1 in the upcoming C.M.E.S.S. with the ones existing in M.E.S.S. or LyaPack respectively. The test problem is the last one described in Section 3.1 and exactly what is implemented in `demo_l1` in M.E.S.S. and LyaPack. The C.M.E.S.S.-implementation especially incorporates the single-pattern–multi-value LU decomposition (see Section 4.4.3). There the first decomposition (for the sole matrix as needed for the shift parameter computations) is the expensive one where all pivoting is performed and the dynamic memory allocation takes place. The further decompositions then can acquire all memory needed en block, since the complete computation is already determined a priori. The computation itself can then proceed through the memory linearly and thus optimally exploit caching effects. Tables 8.16 and 8.17 show the memory savings we achieve using the single-pattern–multi-value LU. The better savings on 64bit result from the fact that the `long int` datatype used for the pattern vector (we can not use `unsigned long int` due to OpenMP limitations), here has the same length as the `double` type, whereas on 32bit it is smaller.

Moreover all LU decompositions for the shifted matrices can be computed independently, i.e., in parallel, which explains the observation in the columns "1st LU" and "other LUs" in Table 8.18. The first LU is the expensive one determining the pivot strategy and the sparsity pattern. The information can then be reused in the further decompositions to allocate the right amount of main memory at once and avoid task switches in the computations. The "total" column shows the over all runtime including the triangular solves employed in the actual ADI iteration. The main difference in the implementations in LyaPack and M.E.S.S. is that LyaPack performs all LU decompositions once, at the beginning of the ADI iteration and stores the factors for reuse when shift parameters are cyclically applied, whereas M.E.S.S. always computes the decompositions on demand and thus saves all the memory space otherwise blocked by

the LU decompositions. Note that C.M.E.S.S. uses the precomputation approach in the specialized storage format and thus saves memory in a different way.

All test have been performed on `romulus` using 16 heuristic shift parameters (see Section 4.3). The "-" signs in the table correspond to cases where LyaPack could no longer be employed due to memory requirements, i.e., the computation ran out of memory while computing all shifted LU decompositions. One may increase the dimension of computable problems there by reducing the number of shifts at the cost of a slower convergence speed.

Especially note that the performance advantage of C.M.E.S.S. over M.E.S.S. reduces with increasing dimension of the matrices. We expect that this is due to the UMFPack solvers used in MATLAB. The multi-frontal solvers employed there can speed up computations for very large matrices drastically. First tests exploiting UMFPack for the LU decompositions in C.M.E.S.S. seem to confirm this expectation, although these are in a very early stage and it is currently not clear how the solver strategies in UMFPack can be combined with the memory gains of the single-pattern–multi-value LU we use to save storage.

Human beings, who are almost unique in having the ability to learn from the experience of others, are also remarkable for their apparent disinclination to do so.

Last Chance to See
Douglas Adams

CHAPTER

NINE

CONCLUSIONS AND OUTLOOK

Contents

9.1. Summary and Conclusions

In this thesis we have examined two important applications of large scale algebraic matrix equations. The balancing based model order reduction on the one hand and the linear-quadratic regulator control of partial differential equations on the other hand. Both of which have, against common believe, been shown to be practically solvable. The key to the efficiency of these methods is the low-rank solution of the matrix Riccati and Lyapunov equations involved. We have reviewed the basic ideas and properties of one important class of those algorithms – the class of low-rank alternating directions implicit (ADI) based algorithms. Starting from the idea [145] to interpret the Lyapunov equation as an ADI model problem, we followed the seminal ideas of Penzl [116, 114] and Li/White [95, 97] to derive the low-rank Cholesky factor ADI (LRCF-ADI) and low-rank Newton ADI methods (LRCF-NM) [18]. We have proposed shift parameter strategies for the ADI algorithm that can be optimal in case the defining matrix F in the Lyapunov equation is symmetric. In many other cases our proposed parameters stay real where the previously used heuristics produce complex shifts. Real shifts enable us to compute real low-rank factors of the solution instead of having to work with complex factors that require double the memory for storage and increase the computational costs at least by a factor of four (current C compilers easily increase the number of atomic processor instructions by a factor of six for the `complex` datatype compared to `double`

computations). Note that [18] also proposes a real arithmetic version of the LRCF-ADI for complex shift parameters. Then again, that version combines two steps of the ADI iteration for a complex conjugate pair of shifts in one step resulting in a quadratic coefficient matrix for linear system of equations to solve per step. That system can still be solved efficiently by iterative solvers, but will in general notably decrease the efficiency of the direct solvers we have proposed to apply. We have also shown some numerical experiments motivating the use of dominant poles of the system as the shift parameters in the context of model order reduction.

In Chapter 4 we have proposed the application of column compression to reduce both memory demands and computational cost for the storage and application of the low-rank factors. Further, we have employed the subspace projection technique used in Krylov projection methods to increase the quality of the of the low-rank factor on the current ADI subspace. Both techniques help to accelerate the ADI iterations convergence. In turn this fact accelerates the LRCF-NM radically since we can save a lot of time in every Newton step. Besides that, due to the projection, the quality of the final ADI iterate often is better, which further increases the convergence speed of the outer Newton method. All these methods have been proven to be efficiently applicable in the case of generalized state space systems (i.e., systems with invertible mass matrix) as well in Chapter 5.

Linear-quadratic regulator problems for parabolic partial differential equations have been discussed in Appendix A and Chapter 6. The appendix reviews the method we introduced in [127, 27] based on the theory proposed by Gibson [54] and refined by Banks and Kunisch [12]. There we basically show that the theory Banks and Kunisch developed for distributed control, which guarantee and preserve exponential stability, can also be extended to boundary control problems. In Chapter 6, we presented an efficient method to solve tracking type problems based on a solver for the stabilization problem. We found a way to rewrite the problem such that the tracking approach appears as an uncontrolled source term in the closed loop system. Numerical results fortifying the effectivity of this approach have been shown and discussed in [17]. The latest contribution in that chapter is the derivation of a suboptimality estimate for the application of a numerically computed feedback control to the real world process. The final section then embeds the LQR system into a nonlinear MPC scheme to apply this technique to the optimal control of nonlinear parabolic PDEs as well. Based on the work of Ito/Kunisch [78] and Chen/Allgöwer[43], we motivate the stabilization feature of this approach. The application to the nonlinear version of the Rail example (Section 3.3) illustrates the practicability of this approach.

The chapter on model order reduction techniques has two major contributions. On the one hand, we discussed the application of low-rank factors in the square root method for balanced truncation model reduction. This gave rise to the truncated or low-rank square root method, which is also called approximate balanced truncation by some authors. We especially reported on the interrelations and choices of the many ADI process parameters, e.g., numbers of shifts, parameters for the shift computation,

truncation tolerances for the column compression. On the other hand, in the second part of the chapter we introduced efficient ways to compute reduced order models for sparse second order systems. These are normally computed applying the model order reduction techniques for first order systems to an equivalent first order representation of the second order system. We have presented a way to exploit the structure of the equivalent first order representations such that we can work with the original sparse second order matrices exploiting their features with the direct solvers applied. Further we have shown that the second order Gramians required by the second order to second order balanced truncation model order reduction can be computed directly from the low-rank factors of the first order Gramians computed for the equivalent first order system representation. Thus we can avoid forming the full $n \times n$ dense Gramians to pick the correct blocks for the second order Gramians.

A wide range of numerical tests has been performed and many features of the methods proposed in the other chapters have been presented in Chapter 8. We have especially proven that Lyapunov and Riccati equations of dimension up to 10^6 can be solved on modern computer hardware.

9.2. Future Research Perspectives

The work we have discussed in this thesis opens the path to a wide range of future research possibilities. Some of them are rather theoretic whereas others lie more in the range of computational scientific aspects. As we have already noted in Sections 4.3.3 and 8.1.4, the usage of dominant poles as ADI shifts shows interesting phenomena that require a rigorous mathematical examination. Also, the usage of the eigenvalues of the optimal closed loop operator as fixed ADI shifts in the context of the LRCF-NM (as suggested in Section 4.5) should be discussed in more detail, although it seems to be the obvious and natural choice especially in view of the projection acceleration. From the remarks on the eigenvalue decay of the solution to the Lyapunov equation in [7] we learn that it is desirable to have the eigenvalues of F in (4.2) cluster in the complex plain and choose the shift parameters inside those clusters. This proposes the pre-conditioning of F such that this clustering is generated. That means one should study whether such a pre-conditioning exists and can be incorporated in the LRCF-ADI at a cost linear in the dimension.

For the LRCF-NM it is a highly interesting question, whether one will be able to find a way to exploit the $R_k < C^T C$ condition in [50] for the residual R_k to limit and reduce the number of required ADI steps per Newton step and still guarantee the convergence of the Newton iteration. Also the question arises whether we can use the knowledge about the close relation of Newton-Kleinman-ADI and the QADI iteration to form a new cheaper low rank QADI version from the current presentation of the LRCF-NM rather than the complicated formulas derived in [151]–[150].

In the context of LQR control for PDEs, the nonlinear equations offer a large field

of applications. The techniques presented will need to be related to other existing techniques like the instantaneous control [71]. Another interesting new field was opened by [123], where the resulting algebraic Riccati equations have structural properties that are not covered by the methods presented in this thesis, e.g., full rank constant terms.

Coupled systems in the context of structural analysis of mechanical systems often result in descriptor system representations. This problem arises especially in applications where rigid bodies play an important role. Therefore the results for the second order MOR need further investigation about their applicability and extensibility in this context.

The efficient implementation of the algorithms we presented throughout this thesis has been performed in MATLAB until now. First steps in the direction of a C language library have been made and the results have been given in Chapter 8. Many new questions arise in the C version where one would not have any influence on their handling in MATLAB anyway. The storage of LU factors in the single-pattern–multi-value way as discussed in Chapter 4 is only one example. Especially the efficient shared memory parallelization and specialized solvers for computers with small main memory have to be examined.

Appendices

I am among those who think that science has great beauty. A scientist in his laboratory is not only a technician: he is also a child placed before natural phenomena which impress him like a fairy tale.

MARIE CURIE

APPENDIX

A

SELECTIVE COOLING OF STEEL PROFILES: EXPONENTIAL STABILIZATION AND DISCRETIZATION

Contents

This appendix chapter is taken into the thesis for completeness reasons. Most of the material has been part of [127, 27]. The manuscript is reprinted as it has been presented in [27]. Some minor comments have been added. The software used in the implementation has been unchanged though. Therefore a general recommendation is to read M.E.S.S. wherever LyaPack is referred to when searching for the most appropriate software to use today.

A.1. Theoretical Background

The theoretical fundament for our approach was set by Gibson [54]. The ideas and proofs used for the boundary control problem considered here closely follow the extension of Gibson's method proposed by Banks and Kunisch [12] for distributed control systems arising from parabolic equations. Similar approaches can be found in [91]. Common to all those approaches is to formulate the control system for a parabolic system as an abstract Cauchy problem in an appropriate Hilbert space setting. For

numerical approaches this Hilbert space is approximated by a sequence of finite dimensional spaces, e.g. by spatial finite element approximations, leading to large sparse systems of ordinary differential equations in $\mathbb{R}^n$. Following the theory in [12] those approximations do not even have to be subspaces of the Hilbert space of solutions, if only they approximate it building a Galerkin scheme of approximating spaces.

A.1.1. Linear-Quadratic Regulator Problems in Hilbert Spaces

In the remainder of this chapter we will assume that the state space $\mathcal{H}$, the input space $\mathcal{U}$ and the output space $\mathcal{O}$ are Hilbert spaces. For operators $\mathbf{A} \in \mathcal{L}(\mathrm{dom}\,(\mathbf{A}), \mathcal{H})$ and $\mathbf{B} \in \mathcal{L}(\mathcal{U}, \mathcal{H})$ with $\mathrm{dom}\,(\mathbf{A}) \subset \mathcal{H}$ and $\mathbf{A}$ the infinitesimal generator of the $\mathcal{C}^0$-semigroup $\mathbf{T}(t)$ on $\mathcal{H}$, we examine the system

$$\begin{array}{rcll} \dot{\mathbf{x}}(t) & = & \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), & \text{for } t > 0, \\ \mathbf{y}(t) & = & \mathbf{C}\mathbf{x}(t), & \text{for } t > 0, \\ \mathbf{x}(0) & = & \mathbf{x}_0. & \end{array} \tag{A.1}$$

Furthermore we consider the cost functional (3.7) for selfadjoint operators $\mathbf{Q} \in \mathcal{L}(\mathcal{H})$ and $\mathbf{R} \in \mathcal{L}(\mathcal{U})$, with $\mathbf{Q} \geq 0$ and $\mathbf{R} > 0$. This completes the LQR-problem:

$$\text{Minimize (3.7) over } \mathbf{u} \in \mathcal{L}^2(0, \infty; \mathcal{U}) \text{ with respect to (A.1).} \tag{$\mathcal{R}^{\mathcal{H}}$}$$

Let $\mathbf{A} : \mathcal{H} \to \mathcal{H}$ be the infinitesimal generator of a $\mathcal{C}^0$-semigroup $\mathbf{T}(t)$, $\mathbf{B} : \mathcal{U} \to \mathcal{H}$ the above input operator. From [54] we know that the solution trajectory $\mathbf{x}_*$ and control input $\mathbf{u}_*$ can be expressed by

$$\begin{array}{rcl} \mathbf{u}_*(t) & = & -\mathbf{R}^{-1}\mathbf{B}^*\mathbf{X}_\infty\mathbf{x}_*(t) \\ \mathbf{x}_*(t) & = & \mathbf{S}(t)\mathbf{x}_0 \end{array} \tag{A.2}$$

iff there exists an admissible control for (3.7),(A.1) for every $\mathbf{x}_0 \in \mathcal{H}$. Here $\mathbf{X}_\infty$ is the minimum nonnegative selfadjoint solution of the operator algebraic Riccati equation:

$$\mathfrak{R}(\mathbf{X}) := \mathbf{Q} + \mathbf{A}^*\mathbf{X} + \mathbf{X}\mathbf{A} - \mathbf{X}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^*\mathbf{X} = 0 \tag{A.3}$$

in the sense that for all $\varphi, \psi \in \mathrm{dom}\,(\mathbf{A})$ it holds

$$(\varphi, \mathbf{Q}\psi)_{\mathcal{H}} + (\mathbf{A}\varphi, \mathbf{X}\psi)_{\mathcal{H}} + (\mathbf{X}\varphi, \mathbf{A}\psi)_{\mathcal{H}} - (\mathbf{B}^*\mathbf{X}\varphi, \mathbf{B}^*\mathbf{X}\psi)_{\mathcal{U}} = 0$$

and $\mathbf{S}(t)$ is the $\mathcal{C}^0$-semigroup generated by $\mathbf{A} - \mathbf{B}\mathbf{R}^{-1}\mathbf{B}^*\mathbf{X}_\infty$. If $\mathbf{Q} > 0$ then $\mathbf{S}$ is even uniformly exponentially stable.

A.1.2. Weak Formulation and Abstract Cauchy Problem

We will now show the relation of the abstract Cauchy problem (A.1) to the partial differential equation model problem from equation (3.6). We therefore consider a variational

formulation of (3.6) tested with $\mathbf{v} \in \mathcal{H}^1(\Omega)$. We will later choose Galerkin approximations of $\mathcal{H}^1(\Omega)$ as test spaces and thus get a finite dimensional approximation of the resulting Cauchy problem by choosing those Galerkin approximations as certain finite-element (fem) spaces. The weak formulation of (3.6) leads to

$$\begin{aligned}(\partial_t \mathbf{x}, \mathbf{v}) &= (\tfrac{1}{c\rho}\nabla.\lambda\nabla\mathbf{x}, \mathbf{v}) \\ &= \int_\Omega \tfrac{1}{c\rho}(\nabla.\lambda\nabla\mathbf{x})\mathbf{v}d\xi = -\int_\Omega \alpha\nabla\mathbf{x}.\nabla\mathbf{v}d\xi + \int_\Gamma \alpha\partial_\nu\mathbf{x}\mathbf{v}d\sigma \\ &\overset{(3.9)}{=} -\int_\Omega \alpha\nabla\mathbf{x}.\nabla\mathbf{v}d\xi - \sum_{k=1}^{7}\int_{\Gamma_k}\tfrac{\kappa_k}{c\rho}(\mathbf{x}-\mathbf{x}_{ext,k})\mathbf{v}d\sigma \\ &= -\int_\Omega \alpha\nabla\mathbf{x}.\nabla\mathbf{v}d\xi - \sum_{k=1}^{7}\left(\int_{\Gamma_k}\tfrac{\kappa_k}{c\rho}\mathbf{x}\mathbf{v}d\sigma - \kappa_k\mathbf{x}_{ext,k}\int_{\Gamma_k}\tfrac{1}{c\rho}\mathbf{v}d\sigma\right).\end{aligned} \tag{A.4}$$

Rewriting (A.4) as

$$(\partial_t\mathbf{x},\mathbf{v})_{\mathcal{L}^2(\Omega)} + \alpha(\nabla\mathbf{x},\nabla\mathbf{v})_{\mathcal{L}^2(\Omega)} + \sum_{k=1}^{7}\frac{\kappa_k}{c\rho}(\mathrm{tr}\,(\mathbf{x}),\mathrm{tr}\,(\mathbf{v}))_{\mathcal{L}^2(\Gamma_k)} - \sum_{k=1}^{7}\frac{\kappa_k}{c\rho}(\mathbf{x}_{ext,k},\mathrm{tr}\,(\mathbf{v}))_{\mathcal{L}^2(\Gamma_k)} = 0 \tag{A.5}$$

with $\mathrm{tr}\,(:)\,\mathcal{H}^1(\Omega) \to \mathcal{L}^2(\partial\Omega)$ the trace operator, and defining

$$\begin{aligned}\mathbf{A}: \quad \mathcal{H}^1(\Omega) &\to \mathcal{H}^1(\Omega)' \\ \mathbf{x} &\mapsto \alpha\left((\nabla\mathbf{x},\nabla.)_{\mathcal{L}^2(\Omega)} - \sum_{k=1}^{7}\tfrac{\kappa_k}{\lambda}(\mathrm{tr}\,(\mathbf{x}),\mathrm{tr}\,(.))_{\mathcal{L}^2(\Gamma_k)}\right) \\ \mathbf{B}: \quad \mathcal{U} &\to \mathcal{H}^1(\Omega)' \\ \mathbf{x}_{ext} &\mapsto \sum_{k=1}^{7}\tfrac{\kappa_k}{c\rho}(\mathbf{x}_{ext,k},\mathrm{tr}\,(.))_{\mathcal{L}^2(\Gamma_k)} \\ \mathbf{M}: \quad \mathcal{H}^1(\Omega) &\to \mathcal{H}^1(\Omega)' \\ \mathbf{x} &\mapsto (\partial_t\mathbf{x},.)_{\mathcal{L}^2(\Omega)'}\end{aligned} \tag{A.6}$$

we obtain the sesquilinear form

$$\sigma_A(\varphi,\psi) :=< \mathbf{A}\varphi,\psi > +\alpha(\varphi,\psi)_{\mathcal{L}^2(\Omega)} \tag{A.7}$$

where as above $< \varphi,\psi >:= \varphi(\psi)$ is the duality product for $\varphi \in \mathcal{H}^1(\Omega)'$ and $\psi \in \mathcal{H}^1(\Omega)$. Note, that analogous to **C** in equation (3.11) the operator **B** is bounded from the boundedness of the domain Ω and the trace operator.

$\sigma_{\mathbf{A}}$ is a continuous and coercive sesquilinear form on $\mathcal{H}^1(\Omega)$ because by definition it holds

$$\sigma_{\mathbf{A}}(\varphi,\varphi) = \alpha\left(\|\varphi\|^2_{1,2} + \sum_{k=1}^{7}\frac{\kappa_k}{\lambda}\|\,\mathrm{tr}\,(\varphi)\,\|^2_{\mathcal{L}^2(\Gamma_k)}\right)$$

so that obviously $\sigma_{\mathbf{A}}(\varphi,\varphi) \geq \alpha\|\varphi\|^2_{1,2}$. The continuity of $\sigma_{\mathbf{A}}$ follows from the continuity of the trace operator (see, e.g., [146]). Now the theorem of Lax-Milgram guarantees the

existence of invertible linear and bounded operators $\mathbf{A}_\alpha \in \mathcal{L}(\mathcal{H}^1(\Omega))$ and $\mathbf{A}_\alpha^* \in \mathcal{L}(\mathcal{H}^1(\Omega))$ such that

$$\begin{aligned} \sigma_{\mathbf{A}}(\varphi,\psi) &= (-\mathbf{A}_\alpha\varphi,\psi)_{\mathcal{H}^1(\Omega)}, \\ \overline{\sigma_{\mathbf{A}}(\varphi,\psi)} &= (-\mathbf{A}_\alpha^*\psi,\varphi)_{\mathcal{H}^1(\Omega)}. \end{aligned} \tag{A.8}$$

This results in a system of the form (A.1) with $\mathcal{H} = \mathcal{H}^1(\Omega)$, for (A.5) together with the initial conditions of the PDE now are:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}_\alpha\mathbf{x} + \mathbf{B}\mathbf{x}_{ext} \quad \text{in } \Omega, \\ \mathbf{x}(0,.) &= \mathbf{x}_0 \quad \text{in } \Omega \end{aligned} \tag{A.9}$$

In the definitions in (A.6) we implicitly used $\mathbf{u}_{ext,k}$ $(k = 1,\dots,7)$ as the controls. If we choose to use the heat transfer coefficients κ_k as controls, we have to define $\mathbf{A}$ and $\mathbf{B}$ as follows

$$\begin{aligned} \mathbf{A}: \quad \mathcal{H}^1(\Omega) &\to \mathcal{H}^1(\Omega)' \\ \mathbf{x} &\mapsto \alpha(\nabla\mathbf{x},\nabla.)_{\mathcal{L}^2(\Omega)} \\ \mathbf{B}: \quad \mathcal{U} &\to \mathcal{H}^1(\Omega)' \\ \kappa_k &\mapsto \sum_{k=1}^{7} \tfrac{\kappa_k}{c\rho}(\operatorname{tr}(\mathbf{x}) - \mathbf{x}_{ext,k}, \operatorname{tr}(.))_{\mathcal{L}^2(\Gamma_k)}. \end{aligned} \tag{A.10}$$

Thus $\mathbf{B}$ is actually $\mathbf{B}(\mathbf{x})$ and the state equation becomes

$$\dot{\mathbf{x}} = \mathbf{A}_\alpha\mathbf{x} + \mathbf{B}(\mathbf{x})\kappa_k \text{ in } \Omega, \tag{A.11}$$

so that in this case we end up with a bilinear system and can not directly apply the linear theory. Eppler and Tröltzsch [49] avoid the bilinear control system by replacing the right hand side of (3.9) by a ficticious heat flux function $\mathbf{v}(t)$. We will later present numerical experiments also for (A.11) when $\mathbf{B}(\mathbf{x})$ is frozen in each time step, similar to [49] where the material parameters λ, c and ϱ are frozen.

From (A.8) and the coercivity of $\sigma_{\mathbf{A}}$ we find that

$$\begin{aligned} \operatorname{Re}(\mathbf{A}_\alpha\varphi,\varphi) &\le -c_1\|\varphi\|_{1,2}^2, \\ \operatorname{Re}(\mathbf{A}_\alpha^*\varphi,\varphi) &\le -c_1\|\varphi\|_{1,2}^2. \end{aligned} \tag{A.12}$$

So $\mathbf{A}_\alpha$ and $\mathbf{A}_\alpha^*$ are densely defined, dissipative linear operators. By [111, Corollary 4.4; Section 1.4] we know that they are infinitesimal generators of $\mathcal{C}^0$-semigroups $\mathbf{T}_\alpha(t)$ and $\mathbf{T}_\alpha^*(t)$. We note that by construction of $\mathbf{T}_\alpha(t)$ the solution semigroup of the uncontrolled system is given by

$$\mathbf{T}(t) = e^{\alpha t}\mathbf{T}_\alpha(t) \tag{A.13}$$

which is generated by $\mathbf{A}_\mathbf{T} = \mathbf{A}_\alpha + \alpha\mathbf{I}$ on the domain of $\mathbf{A}_\alpha$. Analogously we see that

$$\mathbf{T}^*(t) = e^{\alpha t}\mathbf{T}_\alpha^*(t) \tag{A.14}$$

generated by $\mathbf{A}_{\mathbf{T}^*} = \mathbf{A}_\mathbf{T}^* = \mathbf{A}_\alpha^* + \alpha\mathbf{I}$. Furthermore from (A.12) we have

$$\|\mathbf{T}(t)\| \le e^{(\alpha - c_1)t} \text{ for } t \ge 0. \tag{A.15}$$

A.1.3. Approximation by Finite Dimensional Systems

Next we will treat the approximation of (A.9) by finite dimensional systems. A natural requirement for such approximations is

$$\forall \varphi \in \mathcal{H}\ \exists \varphi^N \in \mathcal{H}^N \text{ such that } \|\varphi - \varphi^N\| \leq \varepsilon(N) \text{ and } \varepsilon(N) \to 0 \text{ as } N \to \infty. \tag{C1}$$

This is fulfilled by any Galerkin scheme based approximations, e.g. many finite element approximation schemes. In complete analogy to the procedure in (A.7) to (A.15) the restriction of $\sigma_{\mathbf{A}}$ to $\mathcal{H}^N \times \mathcal{H}^N$ leads to $\mathcal{C}^0$-semigroups $T^N(t)$ and $T^N(t)^*$ generated by operators A^N_T and $A^N_{T^*}$ respectively. Let $P^N : \mathcal{L}^2(\Omega) \to \mathcal{H}^N$ be the canonical orthogonal projection onto $\mathcal{H}^N$. We define the approximations B^N of $\mathbf{B}$ and Q^N of $\mathbf{Q}$ by

$$B^N = P^N\mathbf{B}, \qquad Q^N = P^N\mathbf{Q}.$$

With these we define the approximating LQR-systems as

Minimize:

$$\mathfrak{J}(x_0^N, \mathbf{u}) := \int_0^\infty (x^N, Q^N x^N)_{\mathcal{H}^N} + (\mathbf{u}, \mathbf{R}\mathbf{u})_{\mathcal{U}} dt,$$

with respect to

$$\begin{aligned} M^N \dot{x}^N(t) &= A^N x^N(t) + B^N \mathbf{u}(t), \text{ for } t > 0, \\ x^N(0) &= x_0^N \equiv P^N \mathbf{x}_0 \end{aligned} \tag{$\mathcal{R}^{\mathcal{N}}$}$$

Note that in ($\mathcal{R}^{\mathcal{N}}$) we already wrote the matrix representations of the finite dimensional operators in the fem-basis used for the spatial semi-discretization. This makes the mass matrix M^N appear on the left hand side of the state equation.

A.2. Approximation of Abstract Cauchy Problems

Before stating the main theoretical result we will first collect some approximation prerequisites we will need for the theorem. We call them (BK1) and (BK2) for they were already formulated in [12] (and called H1 and H2 there). The first and natural prerequisite is:

For each $x_0^N \in \mathcal{H}^N$ there exists an admissible control $u^N \in \mathcal{L}^2(0, \infty; \mathcal{U})$ for ($\mathcal{R}^{\mathcal{N}}$) and any admissible control drives the state to 0 asymptotically. (BK1)

Additionally one needs the following properties of the finite dimensional approximations:

(BK2)

(i) For all $\varphi \in H$ it holds $T^N(t)P^N\varphi \to \mathbf{T}(t)\varphi$ uniformly on any bounded subinterval of $[0,\infty)$.

(ii) For all $\phi \in \mathcal{H}$ it holds $T^N(t)^*P^N\phi \to \mathbf{T}(t)^*\phi$ uniformly on any bounded subinterval of $[0,\infty)$.

(iii) For all $\mathbf{v} \in \mathcal{U}$ it holds $B^N\mathbf{v} \to \mathbf{B}\mathbf{v}$ and for all $\varphi \in \mathcal{H}$ it holds $B^{N^*}P^N\varphi \to \mathbf{B}^*\varphi$.

(iv) For all $\varphi \in \mathcal{H}$ it holds $Q^NP^N\varphi \to \mathbf{Q}\varphi$.

With these we can now formulate the main result.

Theorem A.1 (Convergence of the finite dimensional approximations):
Let *(BK1)* and *(BK2)* hold. Moreover let $\mathbf{R} > 0$, $\mathbf{Q} \geq 0$ and $Q^N \geq 0$. Also let X^N be the solutions of the AREs for the finite dimensional systems $(\mathcal{R}^N)$ and let the minimal nonnegative self-adjoint solution $\mathbf{X}$ of $(\mathcal{R})$ on $\mathcal{H}$ exist. Further let $\mathbf{S}(t)$ and $S^N(t)$ be the operator semigroups generated by $\mathbf{A} - \mathbf{B}\mathbf{R}^{-1}\mathbf{B}^*\mathbf{X}$ on $\mathcal{H}$ and $A^N - B^N\mathbf{R}^{-1}B^{N^*}X^N$ on $\mathcal{H}^N$, respectively, with $\|\mathbf{S}(t)\varphi\| \to 0$ as $t \to \infty$ for all $\varphi \in \mathcal{H}$.

If there exist positive constants M_1, M_2 and ω independent of N and t, such that

$$\begin{aligned} \|S^N(t)\|_{\mathcal{H}^N} &\leq M_1 e^{-\omega t}, \\ \|X^N\|_{\mathcal{H}^N} &\leq M_2, \end{aligned} \tag{A.16}$$

then

$$\begin{aligned} X^NP^N\varphi &\to \mathbf{X}\varphi \quad \text{for all } \varphi \in \mathcal{H} \\ S^N(t)P^N\varphi &\to \mathbf{S}(t)\varphi \quad \text{for all } \varphi \in \mathcal{H} \end{aligned} \tag{A.17}$$

converge uniformly in t on bounded subintervals of $[0,\infty)$ and it holds

$$\|\mathbf{S}(t)\| \leq M_1 e^{-\omega t} \text{ for } t \geq 0. \tag{A.18}$$

◊

Sketch of the Proof. This is basically [12, Theorem 2.2] which is formulated in terms of the sesquilinear forms and operator semigroups. In (A.7) to (A.15) we verified that the properties of the sesquilinear form and semigroups are preserved when migrating from distributed control (which Banks and Kunisch used) to boundary control. Therefore the proof boils down to that of [12, Theorem 2.2].

So we only have to verify the prerequisites (BK1), (BK2) and (A.16) here. Considering the fem basis representations (i.e. matrix representation) introduced in $(\mathcal{R}^N)$ we find

that (BK1) follows directly from well known results for finite dimensional regulator systems [99, 133]. The properties (iii) and (iv) of (BK2) are fulfilled by the Galerkin scheme underlying the finite element method used for the implementation. The first and second conditions follow from (A.13) and (A.14) together with

$$\begin{aligned} T_\alpha^N(t)P^N\varphi &\rightarrow \mathbf{T}_\alpha(t)\varphi, \\ T_\alpha^N(t)P^{N^*}\varphi &\rightarrow \mathbf{T}_\alpha(t)^*\varphi, \end{aligned} \tag{A.19}$$

for all $\varphi \in H$ uniformly in t on any bounded subset of $[0,\infty)$. The later is basically an application of the Trotter-Kato Theorem. See [12, Lemma 3.2] and before for details. (A.16) is verified as in [12, Lemma 3.3]. ♦

Theorem A.1 gives the theoretical justification for the numerical method used for the linear problems described in this paper. It shows that the finite-dimensional closed-loop system obtained from optimizing the semi-discretized control problem indeed converges to the infinite-dimensional closed-loop system. Note especially, that the formulation is chosen such that the controls computed for the approximating finite-dimensional systems can directly be applied to the infinite-dimensional system. Certainly they have to be considered suboptimal then. In [91, Chapter 5.2] also convergence rates are given for a very similar approach. From these convergence rates one may also derive sub optimality bounds. Extending this to the given approach is work in progress an will be published in future papers.

Deriving a similar result to the one given in Theorem A.1 for the nonlinear problems presented in Section 3.3.1 and also treated numerically in the following sections is an open problem.

A.3. Implementation Details

We present different strategies of solving the control system here. We can divide them into two approaches. The first of which can be called the ODE approach, for it handles things just the way one would do in classical approaches concerning control systems governed by ordinary differential equations. The second one on the other hand closely follows the philosophy of finite element methods for parabolic partial differential equations and is therefore refered to as the PDE approach. Using the notation of semi-discretizations of partial differential equations one would call the ODE approach a vertical method of lines and the PDE approach a horizontal method of lines (Rothe method).

Both approaches need to solve an ARE to compute the control. This is done by the LyaPack[1] software package [117]. For further details on the theory of solving large sparse AREs and Lyapunov equations we refer the reader to [18, 14, 15, 97, 113, 116].

[1] available at http://www.tu-chemnitz.de/sfb393/lyapack

The basis of the finite dimensional approximations is given by the Galerkin scheme approximating the state space. This is achieved by a finite element semi-discretization of the spatial domain. We used the ALBERTA-1.2 [2] [130] finite element library to do this. The macro triangulation that serves as the basis of the finite element approximation is shown in Figure 3.1. The curved surfaces at head and web of the profile are handled by a projection method, i.e., new boundary points are relocated to their appropriate position on a circular arc. The computational mesh is refined by a bisection refinement method.

We start both approaches with a short uncontrolled forward calculation. During this calculation the profile is cooled down from constant 1000°C until the maximal temperature reaches approximately 990°C to have a more realistic temperature distribution at the control startup. This calculation is done with boundary conditions set up to model cooling in surrounding air of 20°C. See [36, Section 4] for information on the chosen heat transfer coefficients.

Finishing the *pre*-calculation we end up with an initial set of system matrices A^h, B^h, C^h and M^h (the matrix representations of the above finite dimensional operators A^N, B^N, C^N and M^N). These are then used to compute the feedback matrix K^h with the LyaPack software using a MATLAB `mexfunction` implemented for this purpose.

In the ODE approach the matrix K^h is used to establish the closed loop system

$$\begin{aligned} M^h \dot{x}^h(t) &= -(A^h + B^h K^h) x^h(t), \\ x^h(0) &= x_0^h. \end{aligned} \tag{A.20}$$

The solution of the closed loop system can then be calculated by a standard ODE solver like `ode23` of MATLAB.

The PDE approach uses K to set up the boundary conditions. Doing this we have the full space adaption capabilities of ALBERTA at hand to refine or coarsen the mesh as needed. On the other hand we have to pay for this freedom by frequent recalculations of K if the mesh and with it the system matrices M^h, A^h and B^h have changed. At the moment we solve this problem by calculating certain control parameters (i.e. the temperature of the cooling fluid or the intensity parameters for the cooling nozzles depending on the boundary conditions used) and freezing them for a number of timesteps preset by the user.

Numerical calculations with updates after 2, 5 and 10 timesteps show that parameters tend to become *the same* after 10-20 seconds in model time, even if we use an implicit time integration scheme with probably *large* timesteps. Thus all update strategies lead to the same asymptotic behavior. See [27, Figure 2] for a plot of the control parameters (temperatures of the cooling fluid) over time. The term *the same* is to be understood as "equal regarding model accuracy", for temperature differences in the size of deci- or even centi-°C for the cooling fluid should be refered to as equal concerning technical application of the computed controls.

[2] available at `http://www.alberta-fem.de`.

Even using the spraying intensities as controls which leads to a bilinear control system with state dependent input matrix $B^h(x^h)$, can be computed by this method leading to promising results. To do this we linearize the system by choosing $B^h := B^h(x^h(t_n))$ on the time interval $\tau_n := [t_n, t_{n+1}]$. Interpreting this method in terms of instantaneous control or model predictive control is future work and will be presented elsewhere.

APPENDIX

B

THESES

1. This thesis is devoted to the numerical solution of large scale sparse matrix equations of Riccati and Lyapunov type. In particular those matrix equations appearing in the context of linear-quadratic optimal control problems for parabolic partial differential equations and model order reduction of large linear sparse systems of first and second order are discussed in detail.

2. Basic properties of the ADI based iterative methods for the solution of those equations based on the theory developed by Wachspress et al. and their low-rank versions proposed in the seminal work of Li/White and Penzl are discussed.

3. A crucial point in the application of ADI based iterative methods is the choice of "good" shift parameters minimizing the over all spectral radius of the iteration matrix and thus guaranteeing fast convergence of the methods. One of the main contributions is the proposal of three new parameter choices that can be used in appropriate contexts.

4. Sometimes good, or optimal shifts are either not known, or at least not computable with reasonable effort. Then there is a demand for acceleration techniques, that can speed up the convergence of the iteration. A full section of this thesis is dedicated to the development of those techniques. Their effectivity and efficiency is demonstrated in several numerical examples.

5. Classically all results and algorithms have been discussed for matrix equations corresponding to systems in standard state space representation in the open literature. Often the systems arising, e.g., in PDE control are of generalized state space form, i.e., an invertible mass matrix M comes into play. Then the system is multiplied by the inverse to reestablish standard state space form. Alternatively a LU or Cholesky factorization of M is used to define a state space transformation

and eventually transform to standard state space form preserving the symmetry of the system. Both approaches suffer severe fill in and thus explicitly forming the transformed system has to be avoided. For the prior it is shown in this thesis that applying matrix pencil techniques one can almost completely avoid the inversion of M.

6. The efficiency of the methods is demonstrated in two important classes of applications: The linear-quadratic optimal control of parabolic partial differential equations on the one hand and the model order reduction of large sparse linear systems on the other hand.

7. The methods for the stabilization of parabolic PDEs are shown to be efficiently applicable in tracking type control problems, as well. Further first ideas for the treatment of non-linear PDEs are given.

8. For linear parabolic PDEs another main contribution of this thesis is the proof of an estimation of the suboptimality of the numerically computed controls when applied to the real world problem

$$|\mathfrak{J}(\mathbf{u}_*) - \mathfrak{J}(u_*^N)| \leq C\left(\|\mathbf{x}_0 - x_0^N\| + \|\mathbf{X}_* - \hat{X}_*^N\|\right).$$

9. Another chapter is dedicated to the review of the approximate balanced truncation method extending classic balanced truncation to large scale sparse systems by replacing the Cholesky factors in the square root method by the corresponding low-rank factors computed by the low-rank ADI methods in the low-rank square root method. There the single process parameters and their choices in relation to the prescribed errors for the reduction are discussed.

10. The major contribution of this thesis in the model order reduction context is the introduction of a novel method preserving the sparsity and structure of the original system matrices for the iterations applied to second order systems, which are normally transformed to double sized first order form prior to the application of the balanced truncation approach. Besides that the corresponding section of this thesis presents an efficient way to compute the low-rank factorizations of the position and velocity Gramians applied in second-order-to-second-order-balancing from the low-rank factors of the Gramians computed with respect to the equivalent first order system.

11. This thesis has disproved the myth that "solving large scale matrix equations is impossible and impractical". Especially the common believe that LQR problems for linear parabolic PDEs are theoretically well understood but unsolvable in practice has been proven false. With our new codes we have been able to solve matrix equations of dimension up to $10^6 \times 10^6$ with memory demands smaller than 16GBytes.

BIBLIOGRAPHY

[1] H. Abou-Kandil, G. Freiling, V. Ionescu, and G. Jank, *Matrix Riccati Equations in Control and Systems Theory*, Birkhäuser, Basel, Switzerland, 2003. 14

[2] M. Abramovitz and I. Stegun, eds., *Pocketbook of mathematical functions*, Verlag Harry Deutsch, 1984. Abridged edition of "Handbook of mathematical functions" (1964). 9, 45, 49

[3] P. Amestoy, Enseeiht-Irit, T. Davis, and I. Duff, *Algorithm 837: AMD, an approximate minimum degree ordering algorithm*, ACM Transactions on Mathematical Software, 30 (2004), pp. 381–388. 72

[4] L. Amodei and J.-M. Buchot, *A stabilization algorithm based on algebraic Bernoulli equation*, Numer. Lin. Alg. Appl., (2009). submitted. 62

[5] B. Anderson and J. Moore, *Optimal Control – Linear Quadratic Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1990. 13

[6] A. Antoulas, *Approximation of Large-Scale Dynamical Systems*, SIAM Publications, Philadelphia, PA, 2005. 7

[7] A. Antoulas, D. Sorensen, and Y. Zhou, *On the decay rate of Hankel singular values and related issues*, Sys. Control Lett., 46 (2002), pp. 323–342. 40, 91, 135

[8] T. Bagby, *On interpolation by rational functions*, Duke Math. J., 36 (1969), pp. 95–104. 44

[9] A. Balakrishnan, *Boundary control of parabolic equations: L-Q-R theory.*, in Theory of nonlinear operators, Proc. 5th int. Summer Sch., no. 6N in Abh. Akad. Wiss. DDR 1978, Berlin, 1977, Akademie-Verlag, pp. 11–23. 18

[10] H. Banks, ed., *Control and Estimation in Distributed Parameter Systems*, vol. 11 of Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, 1992. 19

[11] H. Banks and K. Ito, *A numerical algorithm for optimal feedback gains in high dimensional linear quadratic regulator problems*, SIAM J. Cont. Optim., 29 (1991), pp. 499–515. 2, 62, 63, 64

[12] H. Banks and K. Kunisch, *The linear regulator problem for parabolic systems*, SIAM J. Cont. Optim., 22 (1984), pp. 684–698. 21, 22, 23, 134, 139, 140, 143, 144, 145

[13] P. Benner, *Computational methods for linear-quadratic optimization*, Supplemento ai Rendiconti del Circolo Matematico di Palermo, Serie II, No. 58 (1999), pp. 21–56. Extended version available as *Berichte aus der Technomathematik*, Report 98–04, Universität Bremen, August 1998, from `http://www.math.uni-bremen.de/zetem/berichte.html`. 7, 9, 13

[14] ——, *Efficient algorithms for large-scale quadratic matrix equations*, Proc. Appl. Math. Mech., 1 (2002), pp. 492–495. 145

[15] ——, *Solving large-scale control problems*, IEEE Control Systems Magazine, 14 (2004), pp. 44–59. 72, 145

[16] P. Benner and H. Fassbender, *An implicitly restarted symplectic Lanczos method for the Hamiltonian eigenvalue problem*, Linear Algebra Appl., 263 (1997), pp. 75–111. 65

[17] P. Benner, S. Görner, and J. Saak, *Numerical solution of optimal control problems for parabolic systems*, in Parallel Algorithms and Cluster Computing. Implementations, Algorithms, and Applications, K. Hoffmann and A. Meyer, eds., vol. 52 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2006, pp. 151–169. 27, 81, 82, 101, 134

[18] P. Benner, J.-R. Li, and T. Penzl, *Numerical solution of large Lyapunov equations, Riccati equations, and linear-quadratic control problems*, Numer. Linear Algebra Appl., 15 (2008), pp. 755–777. 2, 24, 61, 62, 69, 133, 134, 145

[19] P. Benner, R.-C. Li, and N. Truhar, *On the ADI method for Sylvester equations*. submitted to Journal of Computational and Applied Mathematics, 2009. 53

[20] P. Benner, H. Mena, and J. Saak, *On the parameter selection problem in the Newton-ADI iteration for large-scale Riccati equations*, Electronic Transitions on Numerical Analysis, 29 (2008). 41, 101

[21] ——, M.E.S.S. *1.0 User Manual*, tech. rep., Chemnitz Scientific Computing, TU Chemnitz, 2009. in preparation.

[22] P. Benner and E. Quintana-Ortí, *Solving stable generalized Lyapunov equations with the matrix sign function*, Numer. Algorithms, 20 (1999), pp. 75–100. 53, 120

[23] ——, *Model reduction based on spectral projection methods*, in Dimension Reduction of Large-Scale Systems, P. Benner, V. Mehrmann, and D. Sorensen, eds., vol. 45 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2005, pp. 5–45. 52

[24] P. Benner, E. Quintana-Ortí, and G. Quintana-Ortí, *PSLICOT routines for model reduction of stable large-scale systems*, in Proc. 3rd NICONET Workshop on Numerical Software in Control Engineering, Louvain-la-Neuve, Belgium, January 19,

2001, 2001, pp. 39–44. 121

[25] ——, *State-space truncation methods for parallel model reduction of large-scale systems*, Parallel Comput., 29 (2003), pp. 1701–1722. 121

[26] P. Benner and J. Saak, *Efficient numerical solution of the LQR-problem for the heat equation*, Proc. Appl. Math. Mech., 4 (2004), pp. 648–649. 27, 32, 101

[27] ——, *Linear-quadratic regulator design for optimal cooling of steel profiles*, Tech. Rep. SFB393/05-05, Sonderforschungsbereich 393 *Parallele Numerische Simulation für Physik und Kontinuumsmechanik*, TU Chemnitz, D-09107 Chemnitz (Germany), 2005. Available from http://www.tu-chemnitz.de/sfb393/sfb05pr.html. 22, 23, 27, 30, 81, 85, 88, 101, 134, 139, 146

[28] ——, *A semi-discretized heat transfer model for optimal cooling of steel profiles*, in Dimension Reduction of Large-Scale Systems, P. Benner, V. Mehrmann, and D. Sorensen, eds., vol. 45 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2005, pp. 353–356. 27

[29] ——, *Application of LQR techniques to the adaptive control of quasilinear parabolic PDEs*, Proc. Appl. Math. Mech., 7 (2007). DOI: 10.1002/pamm.200700240. 81

[30] ——, *Efficient solution of large scale lyapunov and riccati equations arising in model order reduction problems*, Proc. Appl. Math. Mech., 8 (2008), pp. 10085 – 10088. 72

[31] P. Benner and J. Saak, *Efficient balancing based mor for second order systems arising in control of machine tools*, in Proceedings of the MathMod 2009, I. Troch and F. Breitenecker, eds., no. 35 in ARGESIM-Reports, Vienna, Austria, January 2009, Vienna Univ. of Technology, ARGE Simulation News, pp. 1232–1243. ISBN/ISSN: 978-3-901608-35-3. 72

[32] A. Bensoussan, G. Da Prato, M. C. Delfour, and S. K. Mitter, *Representation and control of infinite dimensional systems*, Systems & Control: Foundations & Applications, Birkhäuser Boston Inc., Boston, MA, second ed., 2007. 21

[33] A. Bensoussan, G. D. Prato, M. Delfour, and S. Mitter, *Representation and Control of Infinite Dimensional Systems, Volume I*, Systems & Control: Foundations & Applications, Birkäuser, Boston, Basel, Berlin, 1992. 18, 21

[34] ——, *Representation and Control of Infinite Dimensional Systems, Volume II*, Systems & Control: Foundations & Applications, Birkäuser, Boston, Basel, Berlin, 1992. 18, 21

[35] C. H. Bischof and G. Quintana-Ortí, *Algorithm 782: Codes for rank-revealing QR factorizations of dense matrices.*, ACM Trans. Math. Softw., 24 (1998), pp. 254–257. 51

[36] M. Böhm, M. Wolff, E. Bänsch, and D. Davis, *Modellierung der Abkühlung von Stahlbrammen, Berichte aus der Technomathematik, Bericht 00-07*. http://www.math.uni-bremen.de/zetem/reports/reports-psgz/report0007.ps.gz, 2000. 146

[37] T. Bonin, J. Saak, A. Soppa, M. Zaeh, P. Benner, and H. Fassbender, *Werkzeugmaschinensimulation auf Basis ordnungsreduzierter FE-Modelle*. In preparation for at-Automatisierungstechnik, 2009. 11

[38] T. Bonin, A. Soppa, J. Saak, M. Zaeh, H. Fassbender, and P. Benner, *Einsatz neuer Verfahren zur Ordnungsreduktion von Finite-Elemente-Modellen für die effiziente Simulation von Werkzeugmaschinen*. To appear, October 2009. 11

[39] J. A. Burns, K. Ito, and R. K. Powers, *Chandrasekhar equations and computational algorithms for distributed parameter systems*, in Proc. 23rd IEEE Conference on Decision and Control., Las Vegas, NV, Dec. 1984. 2

[40] J. L. Casti, *Dynamical systems and their applications: linear theory.*, Mathematics in Science and Engineering, Academic Press, New York - London, first ed., 1977. 7, 13, 14

[41] V. Chahlaoui, K. A. Gallivan, A. Vandendorpe, and P. Van Dooren, *Model reduction of second-order system*, in Dimension Reduction of Large-Scale Systems, P. Benner, V. Mehrmann, and D. Sorensen, eds., vol. 45 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2005, pp. 149–172. 95, 98

[42] Y. Chahlaoui and P. Van Dooren, *A collection of benchmark examples for model reduction of linear time invariant dynamical systems*, Tech. Rep. 2002–2, SLICOT Working Note, Feb. 2002. Available from `www.slicot.org`. 10, 26, 102

[43] C.-H. Chen and F. Allgöwer, *A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability*, Automatica, 34 (1998), pp. 1205–1217. 88, 134

[44] R. Curtain and T. Pritchard, *Infinite Dimensional Linear System Theory*, vol. 8 of Lecture Notes in Control and Information Sciences, Springer-Verlag, New York, 1978. 19, 21

[45] R. Curtain and H. Zwart, *An Introduction to Infinite-Dimensional Linear Systems Theory*, vol. 21 of Texts in Applied Mathematics, Springer-Verlag, New York, 1995. 19, 20, 21

[46] N. S. Ellner and E. L. Wachspress, *Alternating direction implicit iteration for systems with complex spectra.*, SIAM J. Numer. Anal., 28 (1991), pp. 859–870. 91

[47] K.-J. Engel and R. Nagel, *One-parameter semigroups for linear evolution equations.*, Graduate Texts in Mathematics. 194. Berlin: Springer., 2000. 20

[48] K. Eppler and F. Tröltzsch, *Discrete and continuous optimal control strategies in the selective cooling of steel profiles*, Z. Angew. Math. Mech., 81 (2001), pp. 247–248. 28, 29, 30

[49] ——, *Discrete and continuous optimal control strategies in the selective cooling of steel profiles*, Preprint 01-3, DFG Schwerpunktprogramm *Echtzeit-Optimierung*

grošer Systeme, 2001. Available from http://www.zib.de/dfg-echtzeit/Publikationen/Preprints/Preprint-01-3.html. 28, 30, 85, 142

[50] F. Feitzinger, T. Hylla, and E. W. Sachs, *Inexact Kleinman-Newton Method for Riccati Equations*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 272–288. 63, 64, 65, 70, 135

[51] W. Ferng, W.-W. Lin, and C.-S. Wang, *The shift-inverted J-Lanczos algorithm for the numerical solutions of large sparse algebraic Riccati equations*, Comput. Math. Appl., 33 (1997), pp. 23–40. 65

[52] K. Gallivan, X. Rao, and P. VanDooren, *Singular riccati equations stabilizing large-scale systems*, Linear Algebra Appl., 415 (2006), pp. 359–372. 62

[53] C. E. García, D. M. Prett, and M. Morari, *Model predictive control: Theory and Practice - a survey.*, Automatica, 25 (1989), pp. 335–348. 88

[54] J. Gibson, *The Riccati integral equation for optimal control problems in Hilbert spaces*, SIAM J. Cont. Optim., 17 (1979), pp. 537–565. 21, 134, 139, 140

[55] K. Glover, *All optimal Hankel-norm approximations of linear multivariable systems and their L^∞ norms*, Internat. J. Control, 39 (1984), pp. 1115–1193. 24

[56] S. Godunov, *Ordinary Differential Equations with Constant Coefficient*, vol. 169 of Translations of Mathematical Monographs, AMS, Providence, RI, 1997. 82

[57] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, third ed., 1996. 53, 62, 70

[58] A. A. Gonchar, *Zolotarev problems connected with rational functions*, Math USSR Sbornik, 7 (1969), pp. 623–635. 44

[59] L. Grasedyck, *Existence of a low rank or H-matrix approximant to the solution of a Sylvester equation*, Numer. Lin. Alg. Appl., 11 (2004), pp. 371–389. 2, 40

[60] L. Grasedyck, W. Hackbusch, and B. Khoromskij, *Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices*, Computing, 70 (2003), pp. 121–165. 2

[61] L. Grüne, J. Pannek, M. Seehafer, and K. Worthmann, *Analysis of unconstrained nonlinear MPC schemes with varying control horizon*, tech. rep., Universität Bayreuth, 2009. 85, 86

[62] S. Gugercin and J.-R. Li, *Smith-type methods for balanced truncation of large systems*, in Dimension Reduction of Large-Scale Systems, P. Benner, V. Mehrmann, and D. Sorensen, eds., vol. 45 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2005, pp. 49–82. 89, 90, 91

[63] S. Gugercin, D. Sorensen, and A. Antoulas, *A modified low-rank Smith method for large-scale Lyapunov equations*, Numer. Algorithms, 32 (2003), pp. 27–55. 51, 91

[64] M. Günter, U. Feldmann, and J. ter Maten, *Modelling and discretization of circuit problems.*, in Numerical methods in electromagnetics., W. H. A. Schilders et al., ed., vol. XIII of Handbook of numerical analysis, Elsevier/North Holland, Amsterdam, 2005, pp. 523–629. Special volume. 10

[65] J. Haase, S. Reitz, S. Wünsche, P. Schwarz, U. Becker, G. Lorenz, and R. Neul, *Netzwerk- und Verhaltensmodellierung eines mikromechanischen Beschleunigungssensors*, in 6. Workshop "Methoden und Werkzeuge zum Entwurf von Microsystemen", December 1997, pp. 23–30. 37

[66] D. Halliday, R. Resnick, and W. Jearl, *Fundamentals of Physics*, Wiley, 7 ed., 2005. Restricted! Not for sale in the united states. 30

[67] S. Hammarling, *Numerical solution of the stable, non-negative definite Lyapunov equation*, IMA J. Numer. Anal., 2 (1982), pp. 303–323. 53

[68] S. Hein, *MPC-LQG-based optimal control of parabolic PDEs*, PhD thesis, TU Chemnitz, 2009. in preparation. 18, 20

[69] M. Heyouni and K. Jbilou, *An extended block Arnoldi algorithm for large-scale solutions of the continuous-time algebraic Riccati equation*, Electronic Transitions on Numerical Analysis, 33 (2009), pp. 53–62. 2

[70] D. Hinrichsen and A. Pritchard, *Mathematical Systems Theory I*, Springer-Verlag, Berlin, 2005. 7

[71] M. Hinze, *Optimal and instantaneous control of the instationary Navier-Stokes equations*, Habilitationsschrift, TU Berlin, Fachbereich Mathematik, 2002. 18, 136

[72] M. Hochbruck and G. Starke, *Preconditioned Krylov subspace methods for Lyapunov matrix equations*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 156–171. (See also: IPS Research Report 92–17, ETH Zürich, Switzerland (1992)). 53

[73] K.-H. Hoffmann, I. Lasiecka, G. Leugering, J. Sprekels, and F. Tröltzsch, eds., *Optimal control of complex structures. Proceedings of the international conference, Oberwolfach, Germany, June 4–10, 2000*, vol. 139 of ISNM, International Series of Numerical Mathematics, Birkhäuser, Basel, Switzerland, 2002. 18

[74] K.-H. Hoffmann, G. Leugering, and F. Tröltzsch, eds., *Optimal control of partial differential equations. Proceedings of the IFIP WG 7.2 international conference, Chemnitz, Germany, April 20–25, 1998*, vol. 133 of ISNM, International Series of Numerical Mathematics, Birkhäuser, Basel, Switzerland, 1999. 18

[75] D. Hu and L. Reichel, *Krylov-subspace methods for the Sylvester equation*, Linear Algebra Appl., 172 (1992), pp. 283–313. 53

[76] M. P. Istace and J. P. Thiran, *On the third and fourth Zoltarev problems in complex plane*, Math. Comp., (1993). 43

[77] K. Ito, *Strong convergence and convergence rates of approximationg solutions for alge-*

braic Riccati equations in Hilbert spaces, in Distributed parameter systems, Proc. 3rd Int. Conf., Vorau/Austria, 1987, pp. 153–166. 85

[78] K. Ito and K. Kunisch, *Asymptotic properties of receding horizon optimal control problems*, SIAM J. Cont. Optim., 40 (2002), pp. 1585–1610. 86, 134

[79] ——, *Receding horizon control with incomplete observations*, SIAM J. Cont. Optim., 45 (2006), pp. 207–225. 18

[80] I. Jaimoukha and E. Kasenally, *Krylov subspace methods for solving large Lyapunov equations*, SIAM J. Numer. Anal., 31 (1994), pp. 227–251. 52, 53, 55

[81] K. Jbilou, *Adi preconditioned krylov methods for large lyapunov matrix equations.*, tech. rep., L.M.P.A., Nov. 2008. 53

[82] K. Jbilou and A. Riquet, *Projection methods for large Lyapunov matrix equations*, Linear Algebra Appl., 415 (2006), pp. 344–358. 53

[83] D. Kleinman, *On an iterative technique for Riccati equation computations*, IEEE Trans. Automat. Control, AC-13 (1968), pp. 114–115. 2, 16, 61

[84] M. Köhler and J. Saak, *Efficiency improving implementation techniques for large scale matrix equation solvers*, Chemnitz Scientific Computing, TU Chemnitz, 2009. in preparation. 55, 129

[85] R. Krengel, R. Standke, F. Tröltzsch, and H. Wehage, *Mathematisches Modell einer optimal gesteuerten Abkühlung von Profilstählen in Kühlstrecken*, Preprint 98-6, Fakultät für Mathematik TU Chemnitz, November 1997. 28, 29, 30

[86] M. Kroller and K. Kunisch, *Convergence rates for the feedback operators arising in the Linear Quadratic Regulator Problem governed by Parabolic Equations*, SIAM J. Numer. Anal., 28 (1991), pp. 1350–1385. 23, 85

[87] P. Kunkel and V. Mehrmann, *Differential-Algebraic Equations: Analysis and Numerical Solution*, Textbooks in Mathematics, EMS Publishing House, 2006. 10

[88] W. Kwon and S. Han, *Receding Horizon Control: Model Predictive Control for State Models*, Advanced Textbooks in Control and Signal Processing, Springer, London, 1st ed., 2005. 88

[89] P. Lancaster and L. Rodman, *The Algebraic Riccati Equation*, Oxford University Press, Oxford, 1995. 16, 62

[90] I. Lasiecka and R. Triggiani, *Differential and Algebraic Riccati Equations with Application to Boundary/Point Control Problems: Continuous Theory and Approximation Theory*, no. 164 in Lecture Notes in Control and Information Sciences, Springer-Verlag, Berlin, 1991. 18, 19, 21

[91] ——, *Control Theory for Partial Differential Equations: Continuous and Approximation Theories I. Abstract Parabolic Systems*, Cambridge University Press, Cambridge, UK, 2000. 18, 19, 21, 22, 85, 139, 145

[92] ——, *Control theory for partial differential equations: Continuous and approximation theories II. Abstract hyperbolic-like systems over a finite time horizon*, in Encyclopedia of Mathematics and its Applications, vol. 75, Cambridge University Press, Cambridge, 2000, pp. 645–1067. 18, 19

[93] A. J. Laub, M. T. Heath, C. C. Paige, and R. C. Ward, *Computation of system balancing transformations and other applications of simultaneous diagonalization algorithms.*, IEEE Trans. Autom. Control, 32 (1987), pp. 115–122. 23

[94] V. I. Lebedev, *On a Zolotarev problem in the method of alternating directions*, USSR Comput. Math. and Math. Phys., 17 (1977), pp. 58–76. 43

[95] J.-R. Li, *Model Reduction of Large Linear Systems via Low Rank System Gramians*, PhD thesis, Massachusettes Institute of Technology, September 2000. 55, 59, 133

[96] J.-R. Li and M. Kamon, *PEEC model of a spiral inductor generated by Fasthenry*, in Dimension Reduction of Large-Scale Systems, P. Benner, V. Mehrmann, and D. Sorensen, eds., vol. 45 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2005, pp. 373–377. 35

[97] J.-R. Li and J. White, *Low rank solution of Lyapunov equations*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 260–280. 24, 42, 57, 67, 133, 145

[98] J. Lions, *Optimal Control of Systems Governed by Partial Differential Equations*, Springer-Verlag, Berlin, FRG, 1971. 18, 21

[99] A. Locatelli, *Optimal Control: An Introduction*, Birkhäuser, Basel, Boston, Berlin, 2001. 7, 13, 145

[100] A. Lu and E. Wachspress, *Solution of Lyapunov equations by alternating direction implicit iteration.*, Comput. Math. Appl., 21 (1991), pp. 43–58. 44

[101] D. G. Luenberger, *Introduction to dynamic systems. Theory, models, and applications.*, John Wiley & Sons., New York etc., first ed., 1979. 7, 13

[102] J. Macki and A. Strauss, *Introduction to Optimal Control Theory*, Springer-Verlag, 1982. 7

[103] M. Marcus and H. Minc, *A Survey of Matrix Theory and Matrix Inequalities*, Allyn and Bacon, Boston, 1964. 53

[104] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, *Constrained model predictive control: Stability and optimality.*, Automatica, 36 (2000), pp. 789–814. 87

[105] V. Mehrmann, *The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution*, no. 163 in Lecture Notes in Control and Information Sciences, Springer-Verlag, Heidelberg, July 1991. 16

[106] V. Mehrmann and T. Stykel, *Balanced truncation model reduction for large-scale systems in descripter form*, in Dimension Reduction of Large-Scale Systems, P. Benner, V. Mehrmann, and D. Sorensen, eds., vol. 45 of Lecture Notes in Compu-

tational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2005, pp. 83–115. 10

[107] H. Mena, *Numerical Methods for Large-Scale Differential Riccati Equations with Applications in Optimal of Partial Differential Equations*, PhD thesis, Escuela Politécnica Nacional, Quito, Ecuador, 2007. 13, 16, 20, 21, 23, 85

[108] D. G. Meyer and S. Srinivasan, *Balancing and model reduction for second-order form linear systems.*, IEEE Trans. Autom. Control, 41 (1996), pp. 1632–1644. 95, 98

[109] B. Moore, *Principal component analysis in linear systems: Controllability, observability, and model reduction*, IEEE Trans. Automat. Control, AC-26 (1981), pp. 17–32. 23

[110] K. Morris and C. Navasca, *Solution of algebraic Riccati equatons arising in control of partial differential equations.*, in Control and Boundary Analysis., J. P. Zolesio and J. Cagnol, eds., vol. 240 of Lecture Notes in Pure Appl. Math., CRC Press, 2005. 2

[111] A. Pazy, *Semigroups of linear operators and applications to partial differential equations.*, Applied Mathematical Sciences, 44. New York etc.: Springer-Verlag. VIII, 1983. 6, 20, 142

[112] D. Peaceman and H. Rachford, *The numerical solution of elliptic and parabolic differential equations*, J. Soc. Indust. Appl. Math., 3 (1955), pp. 28–41. xix, 40

[113] T. Penzl, *A cyclic low rank Smith method for large, sparse Lyapunov equations with applications in model reduction and optimal control*, Tech. Rep. SFB393/98-6, Fakultät für Mathematik, TU Chemnitz, 09107 Chemnitz, FRG, 1998. Available from `http://www.tu-chemnitz.de/sfb393/sfb98pr.html`. 145

[114] ——, *Numerische Lösung großer Lyapunov-Gleichungen*, Logos–Verlag, Berlin, Germany, 1998. Dissertation, Fakultät für Mathematik, TU Chemnitz, 1998. 24, 133

[115] ——, *Algorithms for model reduction of large dynamical systems*, Tech. Rep. SFB393/99-40, Sonderforschungsbereich 393 *Numerische Simulation auf massiv parallelen Rechnern*, TU Chemnitz, 09107 Chemnitz, FRG, 1999. Available from `http://www.tu-chemnitz.de/sfb393/sfb99pr.html`. 40

[116] ——, *A cyclic low rank Smith method for large sparse Lyapunov equations*, SIAM J. Sci. Comput., 21 (2000), pp. 1401–1418. 24, 41, 43, 44, 46, 57, 63, 64, 70, 92, 133, 145

[117] ——, Lyapack *Users Guide*, Tech. Rep. SFB393/00-33, Sonderforschungsbereich 393 *Numerische Simulation auf massiv parallelen Rechnern*, TU Chemnitz, 09107 Chemnitz, Germany, 2000. Available from `http://www.tu-chemnitz.de/sfb393/sfb00pr.html`. 24, 26, 27, 63, 64, 70, 72, 102, 145

[118] ——, *Algorithms for model reduction of large dynamical systems*, Linear Algebra Appl., 415 (2006), pp. 322–343. (Reprint of Technical Report SFB393/99-40, TU Chemnitz, 1999.). 24

[119] L. Pernebo and L. M. Silverman, *Model reduction via balanced state space represen-*

tations., IEEE Trans. Autom. Control, 27 (1982), pp. 382–387. 24

[120] A. Pritchard and D. Salamon, *The linear quadratic control problem for infinite dimensional systems with unbounded input and output operators.*, SIAM J. Control Optimization, 25 (1987), pp. 121–144. 21

[121] G. Quintana-Ortí, X. Sun, and C. H. Bischof, *A BLAS-3 version of the QR factorization with column pivoting.*, SIAM J. Sci. Comput., 19 (1998), pp. 1486–1494. 54

[122] X. Rao, *Large scale stabilization with linear feedback*, Master's thesis, Florida State University, 1999. 62

[123] J.-P. Raymond, *Local boundary feedback stabilization of the Navier-Stokes equations*, in Control Systems: Theory, Numerics and Applications, Rome, 30 March – 1 April 2005, Proceedings of Science, SISSA, http://pos.sissa.it, 2005. 136

[124] T. Reis and T. Stykel, *Balanced truncation model reduction of second-order systems*, Math. Comput. Model. Dyn. Syst., 14 (2008), pp. 391–406. xiii, xiv, xv, 95, 98, 99, 122, 123, 124, 125

[125] J. Rommes, *Methods foreigenvalue problems with applications in model order reduction*, PhD thesis, Universiteit Utrecht, June 2007. 50, 104

[126] Y. Saad, *Numerical solution of large Lyapunov equation*, in Signal Processing, Scattering, Operator Theory and Numerical Methods, M. A. Kaashoek, J. H. van Schuppen, and A. C. M. Ran, eds., Birkhäuser, 1990, pp. 503–511. 53

[127] J. Saak, *Effiziente numerische Lösung eines Optimalsteuerungsproblems für die Abkühlung von Stahlprofilen*, Diplomarbeit, Fachbereich 3/Mathematik und Informatik, Universität Bremen, D-28334 Bremen, Sept. 2003. 27, 30, 81, 101, 134, 139

[128] J. Sabino, *Solution of Large-Scale Lyapunov Equations via the Block Modified Smith Method*, PhD thesis, Rice University, Houston, Texas, June 2007. available from: http://www.caam.rice.edu/tech_reports/2006/TR06-08.pdf. 44

[129] S. B. Salimahrami, *Structure Preserving Order Reduction of Large Scale Second Order Models*, PhD thesis, TU München, 2005. available from: http://www.rt.mw.tum.de/salimbahrami/BehnamThesis.pdf. 10

[130] A. Schmidt and K. Siebert, *Design of Adaptive Finite Element Software. The Finite Element Toolbox ALBERTA*, vol. 42 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin-Heidelberg, 2005. 146

[131] V. Simoncini, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288. 2, 26, 53, 55, 59

[132] V. Simoncini and V. Druskin, *Convergence analysis of projection methods for the numerical solution of large Lyapunov equations*, SIAM Journal on Numerical Analysis,

47 (2009), pp. 828–843. 53

[133] E. Sontag, *Mathematical Control Theory*, Springer-Verlag, New York, NY, 2nd ed., 1998. 9, 13, 145

[134] D. Sorensen and A. Antoulas, *On model reduction of structured systems*, in Dimension Reduction of Large-Scale Systems, P. Benner, V. Mehrmann, and D. Sorensen, eds., vol. 45 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2005, pp. 117–130. 98

[135] G. Starke, *Optimal alternating directions implicit parameters for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 28 (1991), pp. 1431–1445. 43, 44

[136] ——, *Rationale Minimierungsprobleme in der komplexen Ebene im Zusammenhang mit der Bestimmmung optimaler ADI-Parameter*, dissertation, Fakultät für Mathematik, Universität Karlsruhe, Germany, 1993. In German. 44, 46

[137] U. Storch and H. Wiebe, *Textbook of mathematics. Vol. 1: Analysis of one variable. (Lehrbuch der Mathematik. Band 1: Analysis einer Veränderlichen.)*, Spektrum Akademischer Verlag, Heidelberg, 3 ed., 2003. (German). 49

[138] H. Tanabe, *Equations of evolution. Translated from Japanese by N. Mugibayashi and H. Haneda.*, vol. 6 of Monographs and Studies in Mathematics., Pitman Publishing Ltd , London - San Francisco - Melbourne, 1979. 20

[139] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Review, 43 (2001), pp. 235–286. 10

[140] J. Todd, *Applications of transformation theory: A legacy from Zolotarev (1847-1878)*, in Approximation Theory and Spline Functions, S. P. S. et al., ed., no. C 136 in NATO ASI Ser., Dordrecht-Boston-Lancaster, 1984, D. Reidel Publishing Co., pp. 207–245. Proc. NATO Adv. Study Inst., St. John's/Newfoundland 1983. 43

[141] M. Tombs and I. Postlethwaite, *Truncated balanced realization of a stable non-minimal state-space system*, Internat. J. Control, 46 (1987), pp. 1319–1330. 23

[142] F. Tröltzsch, *Optimale Steuerung partieller Differentialgleichungen - Theorie, Verfahren und Anwendungen*, Vieweg, Wiesbaden, 2005. In German. 18

[143] F. Tröltzsch and A. Unger, *Fast solution of optimal control problems in the selective cooling of steel*, Z. Angew. Math. Mech., 81 (2001), pp. 447–456. 28, 29, 30

[144] N. Truhar and K. Veselic, *An efficient method for estimating the optimal dampers' viscosity for linear vibrating systems using Lyapunov equation*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 18–39. 35

[145] E. Wachspress, *The ADI model problem*, 1995. Available from the author. xix, 41, 43, 44, 91, 133

[146] J. Wloka, *Partial differential equations.*, Cambridge University Press. XI, 1987. Transl. from the German by C. B. and M. J. Thomas. 6, 31, 141

[147] N. Wong and V. Balakrishnan, *Fast balanced stochastic truncation via a quadratic extension of the alternating direction implicit iteration*, in Proc. Int. Conf. Computer Aided Design 05, 2005, pp. 801–805. 66

[148] ——, *Quadratic alternating direction implicit iteration for the fast solution of algebraic Riccati equations*, in Proc. Int. Symposium on Intelligent Signal Processing and Communication Systems, 2005, pp. 373–376. 66, 68

[149] ——, *Multi-shift quadratic alternating direction implicit iteration for high-speed positive-real balanced truncation*, in Proc. Design Automation Conference (DAC) 2006, 2006, pp. 257–260. 66

[150] ——, *Fast positive-real balanced truncation via quadratic alternating direction implicit iteration*, IEEE Trans. CAD Integr. Circuits Syst., 26 (2007), pp. 1725–1731. 66, 135

[151] N. Wong, V. Balakrishnan, C.-K. Koh, and T.-S. Ng, *A fast Newton/Smith algorithm for solving algebraic Riccati equations and its application in model order reduction*, in Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, vol. 5, 2004, pp. 53–56. 66, 135

[152] ——, *Two algorithms for fast and accurate passivity-preserving model order reduction*, IEEE Trans. CAD Integr. Circuits Syst., 25 (2006). 66

[153] B. Yan, S. X.-D. Tan, and B. McGaughy, *Second-order balanced truncation for passive-order reduction of RLCK circuits*, IEEE Trans. Circuits Syst. II, 55 (2008), pp. 942–946. 95, 99

[154] J. Zabczyk, *Remarks on the algebraic Riccati equation*, Appl. Math. Optim., 2 (1976), pp. 251–258. 16, 21

INDEX