EFFICIENT ALGORITHMS FOR DISTRIBUTED CONTROL: A STRUCTURED MATRIX APPROACH

Justin Rice

PhD Thesis
September 6, 2010

Front Cover: Fractal Cabbage, by Chi Nguyen Thanh
Back Cover: Octopussy Night, by Justin Rice

EFFICIENT ALGORITHMS FOR DISTRIBUTED CONTROL: A STRUCTURED MATRIX APPROACH

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op

maandag 6 september 2010 om 12:30 uur

door

Justin RICE

Mechanical Engineer, Cornell University, U.S.A.
geboren te Independence, Missouri, U.S.A.

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. ir. M. Verhaegen

Samenstelling promotiecommisie:

Rector Magnificus,	voorzitter
Prof. dr. ir. M. Verhaegen,	Technische Universiteit Delft, promotor
Prof. dr. ir. A.J. van der Veen,	Technische Universiteit Delft
Prof. dr. ir. P.M. Dewilde,	Technische Universiteit Delft
Prof. dr. B. Bamieh,	University of California, Santa Barbara
Prof. dr. S. Chandrasekaran,	University of California, Santa Barbara
Prof. dr. M. Jovanovic,	University of Minnesota
Prof. dr. H. Sandberg,	KTH Royal Institute of Technology

Paranimfen:

Luat Vuong
Fedor Leeman

The work presented in this thesis has been supported by the Dutch Technology Foundation (STW) under project number 05636.

ISBN: ...

Copyright © 2010 by Justin K. Rice under the Creative Commons license:
http://creativecommons.org/licenses/by-nc-nd/3.0/
You may copy and distribute this work if you attribute it to Justin K. Rice, but only for noncommercial purposes, and you may not alter or transform it in any way.

Printed in The Netherlands

Acknowledgments

I have many people to thank for their contributions to this thesis. Without your help and support, I might have quit my PhD 100 times before making it this far.

First, I must of course thank Michel, for his close and occasionally stern guidance, but also for the extraordinary freedom he gave me this last four years to explore my field (and other fields). I don't think it's been easy for either of us, but I couldn't have done a PhD any other way. I also thank Rufus and Tamas for their patience with me and curiosity with my work, and Carsten for always giving me time to ask my ridiculous questions and for resisting the urge to tell me that it's all trivial. Thanks to Jan-Willem for being my friendly neighbor and de facto daily supervisor for 4 years now.

Thanks to David Delchamps and Adam Bojanczyk for their infinite patience in teaching me linear systems theory and large matrix computations back in Ithaca, and for continuing to provide me with technical support, years later. Thanks to Raff D'Andrea for first inspiring me to study feedback control, and to Steven Strogatz for making dynamical systems fun. Thanks to Matt Ulinski for giving me time to explore, and for filling me in on all the stuff my engineering degree left out. Thanks to Lucas Parra for giving me a chance, in spite of my inexperience.

Thankyou to Cedric, Geir, Lars, and Orest for their time discussing both their work and my own, and for explaining any and all technical questions I asked. Thanks to Niels, Lassi, Masako, Carlos, and Ixbalank for making distributed parameter systems conferences tolerable, and even fun. Thanks to Andrey, Dang, Soloman, Navin, and Balasz for making all the other conferences fun, and for being great traveling companions. Thank you Paolo, for being my friend, workmate, and companion through this whole crazy PhD thing, for always playing Sancho Panza to my Don Quixote, and for teaching me that when the going gets tough, it's best to relax.

Thanks to Aydin for loaning me his basketball shoes, even if they were too small and made both of my big toenails turn purple and fall off. Thanks to Mathieu for showing me that pretty much anything can be fun if you want it to be, even fighting off a midnight kitchen flood in your pajamas. Thanks to Diederick, Eric, and Jelmer for 4 years of lunch, for listening to and often solving all of my problems with and in the Netherlands, for translating my most ridiculous compositions, and for being all around great guys. Thanks to Rudy for his legal advice, and Redouanne for his great thesis template.

Thanks to Andrea for his unquestioning and irrevocable friendship; we will have many more crazy adventures together. Thanks to Pavel for Scopa. Thanks to Okonor, wherever he may be, for being my friend, and showing me that it was possible for someone to fit in to Dutch culture even worse than myself. Thanks to Robert, Olaf, Alessandro, and the rest of the DCSC football team for the fun and exercise. Thanks to Ilhan, in his capacity as a bear, for not eating me, and in his capacity as a person, for being my superb cover photographer and for helping to carry my desk back from Gamma. Thanks to Ivo for always solving my computer problems. Thanks to Hakan for trying his best to answer with kindness all of my ill-posed, spur of the moment, and badly-thought-up questions, and for never balking at my unorthodoxy.

Thanks to Hong Song, for proving that one can play superb basketball without ever running or jumping, to Jianfei for providing a constant reminder of the kind of perfect posture that is possible if you really try, to Snezsana for showing me how to laugh in the face of beaurocratic adversity, and to Ali for demonstrating how to be dignified yet indignant with adversity of another kind. Thanks to Gabriel for being my partner in the movie-club, and for organizing volleyball. Thanks to Daniele Corona for inspiring me to escape my desk.

Thanks to STW for funding my research, and thanks to my STW user's committee for always trying to be interested in my results. Thanks to Professor Wieringa, for arranging to have my tuxedo paid for by the university.

Huge thanks to the secretaries, and especially Kitty, for always being over-competent, patient with my own incompetence, and usually 3 steps ahead of me in everything. Thanks to the Service Point for nothing, thanks to Human Resources for less than nothing, and thanks to the IT migration for wasting 4 months of my time.

Thanks to my PhD committee, Alle-Jan, Bassam, Henrik, Mihailo, Patrick, and Shiv. You've all been very helpful as advisors throughout my final half-year, and even before, by always being interested in my research and giving me useful advice. Thankyou also for agreeing to carefully read my thesis, fly to the Netherlands, and patiently sit through the traditional Dutch defense ceremony, all motivated only by a sense of scientific duty, and maybe some interest in my work.

Thanks to Chi, Julio, Winny, Takeshi, Uli and Zwaanjte for being great 14th floor neighbors, friends, and cooks, and for making my first year great. Thanks to Fedor for being my first and best friend in the Netherlands, and for teaching me all the Dutch I know (and more); if we hadn't ended every lesson with whisky, perhaps I would be fluent by now. Thanks to Andrea Sekulovic for her super-powers, to Valentine for the hammer and chicken-wing, and to Alberto for freeing me from my enslavement to that evil overlord, Microsoft. Thanks to Aurelie for being my running partner, confidante, and friend, and for all the cakes.

Thanks to my Poptahof neighbors, for adopting me in spite of all of our differences, and for smoking me at ping-pong in 30 different national styles.

Thanks to Jasper for igniting his socks, Lizzette for teaching me how to use a sewing machine, Eva for her motherliness, Sandro for his Tiramisu, Mei-Ling for the brownies and Flux, Merel for letting me win at Mario-Bros, Rita for her explanation of the Dutch people, Sebastiaan for his Tai-Chi lessons, Leonard for his impromptu raps, and all the other Nieuwelaaners for being awesome neighbors and great friends.

For being beautiful, flavorful, and welcoming, I thank Barcelona, my second home.

Thanks to Neeraj for always being my brother in Boston, and Yusufi for replying to my emails. Thanks to Barbara for an unforgettable day. Thanks to Timmy! Reissman for showing me what *not* to do as a PhD student, how *not* to ride a skateboard, and how *not* to buy groceries. Thanks also for teaching me how to use an oscilloscope, how to be a good teaching assistant, and how to play a mean game of darts. Thank you Roger, for valuing my work as a researcher more than I did myself, and reminding me of the possibilities in life. Thanks to Caroline for believing in me, inspiring me, and encouraging me to take risks.

Thanks to Noah for teaching me about Quantum Dots, and of course for the ark. Thanks to Evan for being himself. Thanks to Christina Murray for accidentally turning my future on its head, in the best possible way. Thanks to the email Book Club/Magazine Club members for all the good discussions; don't worry, it will rise again. Thanks to Sander, Arturo, and Vessie for the Delft book club; we never had a meeting, but it was fun anyway. Thanks to Laura, Sarah, and Shelly for the gin on the rocks, knots advice, friendship, songs, and photos. I'll see you all at sea.

Thanks to X.T. for lending my his accordion, without which I might have gone mad during my third year, and Bik-Kee for worrying about my health and diet when no one else did and for sending me turkey jerky and almonds. Thanks to Ben and Luan for spontaneously but sincerely adopting me every time I was in the DC area. Thanks to the whole Vuong family for recovering my lost Master's degree from their garage and rush-shipping it to me.

Finally, thanks to my parents for their patience, support, and their willingness to indulge my curiosity in everything when I was young. Thanks to my grandparents and aunts and uncles for sending me encouragement half way around the world. Thanks to Brian for pushing me to be mathematically rigorous from a very early age. Thanks to Luat for her simultaneous support and adversity, and for never letting me go soft.

Justin

When I heard the learn'd astronomer,
When the proofs, the figures, were ranged in columns before me,
When I was shown the charts, the diagrams, to add, divide, and measure them,
When I sitting heard the learned astronomer where he lectured with much applause in the lecture room,
How soon unaccountable I became tired and sick,
Till rising and gliding out I wander'd off by myself,
In the mystical moist night-air, and from time to time,
Look'd up in perfect silence at the stars.

-Walt Whitman

Contents

Acknowledgments **v**

1 Introduction **1**

1.1 Fish and Birds: Distributed Systems 1

1.2 Scope and Contributions of this Thesis 5

1.3 Structure of this Thesis 7

List of Symbols and Notation **12**

2 Mixed Causal Input-Output Operators and their Arithmetic **13**

2.1 Introduction 13

2.1.1 The Idea of Structured Matrix Arithmetic 14

2.1.2 The Idea of Mixed Causal Systems 16

2.2 Discrete LTI Systems and Laurent Matrices with Rational Symbols . 17

2.2.1 Fourier Transform, the Operator Symbol, and Useful Equivalencies 18

2.2.2 $\mathcal{S}$-realizations 21

2.2.3 Addition of L_r Matrices with $\mathcal{S}$ Realizations 23

2.2.4 Multiplication of L_r Matrices with $\mathcal{S}$ Realizations 24

2.2.5 Order Reduction of L_r Matrices with $\mathcal{S}$ Realizations 27

2.2.6 Transpose and Permutations of L_r Matrices with $\mathcal{S}$ Realizations 28

2.2.7 Inversion of L_r Matrices with $\mathcal{S}$ Realizations 30

2.2.8 l_2 Induced Norm of L_r Matrices with $\mathcal{S}$ Realizations 34

2.2.9 Calculating the Path Integral of the Symbol of an $\mathcal{S}$ Realization Over the Unit Circle 35

2.2.10 Summary, L_r Matrices with $\mathcal{S}$ Realizations 36

2.3 Discrete LTV Systems and SSS matrices 36

2.3.1 SSS Addition 38

2.3.2 SSS Multiplication 38
2.3.3 SSS Inversion 39
2.3.4 SSS Order Reduction 40
2.3.5 SSS Transpose and Permutation 41
2.3.6 Calculating Norms of SSS Matrices 41
2.3.7 Partial Derivatives of an SSS Matrix-Vector Product 42
2.3.8 SSS Inversion $\mathcal{O}(N)$ example 44
2.3.9 Summary, SSS matrices 44
2.4 Discrete LTI Finite Length Systems and Almost Toeplitz SSS Matrices 46
2.4.1 ATSSS Addition 47
2.4.2 ATSSS Multiplication 47
2.4.3 ATSSS Inversion 48
2.4.4 ATSSS Order Reduction 49
2.4.5 ATSSS Transpose & Permutations & Norm Calculation . . . 49
2.4.6 Convergence of ATSSS Arithmetic 49
2.4.7 Example 50
2.4.8 Summary, ATSSS matrices 51
2.5 Continuous LTI Systems and Their Transfer Matrices 51
2.5.1 $\mathcal{S}_c$ Addition 52
2.5.2 $\mathcal{S}_c$ Multiplication 53
2.5.3 $\mathcal{S}_c$ Inversion 53
2.5.4 $\mathcal{S}_c$ Order Reduction 55
2.5.5 $\mathcal{S}_c$ Transpose, Permutation, and Norm 55
2.5.6 Summary, $\mathcal{S}_c$ realizations 56
2.6 Conclusion 56
2.7 Appendix: Algorithm for finding SSS matrix representations of banded matrices 56
2.8 Appendix: Algorithms for Building Dense matrices out of SSS representations in $\mathcal{O}(N^2)$ 57

3 Structure Preserving Iterations 59
3.1 Introduction 59
3.1.1 History 60
3.1.2 Useful Definitions 60
3.1.3 Useful Facts 61
3.2 Applications 64

3.2.1 Matrix Stability and Square Root 65
3.2.2 Lyapunov and Riccati Equations 65
3.2.3 Block Diagonalization 66
3.2.4 Balanced Model Order Reduction 67
3.2.5 H_2 and H_∞ Analysis and Controller Synthesis 68
3.3 Numerical Problems and What to Do about Them 71
3.3.1 Numerical Problems 72
3.3.2 What to Do about Them 72
3.4 Application to this Thesis 73
3.4.1 Structure Preserving Property and Convergence Results . . . 74
3.4.2 Summary 77
3.5 Multi-level Structured Operators 77
3.5.1 From 1-D to 2-D to n-D 77
3.5.2 Dealing with Approximation Errors on Multiple Levels 80
3.6 Conclusion 81
3.7 Appendix: Special Control Related Results for Laurent Matrices . . 81

4 Heterogeneous Distributed Systems 85
4.1 Heterogeneous String Interconnected Systems ⇔ SSS Matrices . . . 85
4.1.1 Interconnected System $\Longrightarrow$ SSS Matrices 86
4.1.2 SSS Matrices $\Longrightarrow$ Interconnected System 87
4.1.3 Distributed Computation of SSS Algorithms 89
4.1.4 Summary 90
4.2 Example: Stability Check 90
4.3 Example: H_2 Control of a Vehicle Platoon 92
4.3.1 $\mathcal{O}(N)$ Demonstration 93
4.3.2 Distributedness 94
4.4 Example: H_∞ control of the Wave Equation 94
4.4.1 Introduction/Discretization 95
4.4.2 $\mathcal{O}(N)$ H_∞ Synthesis 95
4.4.3 Distributed Implementation 96
4.5 Example: SSS Structure Preserving Model Order Reduction 97
4.6 Example: Fast Structured System ID 99
4.6.1 EKF for Parametric ID 99
4.6.2 Numerical Demonstration 101
4.6.3 Discussion 104

4.7 Counterexample: Control of a Smart Blade 104
  4.7.1 Distributed Model 104
  4.7.2 Finite Difference Discretization 106
  4.7.3 Computational Difficulties 107
4.8 Chapter Conclusion 107

5 Homogeneous Distributed Systems 109
5.1 Subsystems and Interconnected System 110
5.2 Scalar Example: LQR of a Thin Heated Rod 111
5.3 Matrix Example: LQR for a Square Heated Bar 112
5.4 Matrix Example: H_2 Control of a Platoon, with Comparison to Other Techniques 113
  5.4.1 Exact H_2 Solution 114
  5.4.2 Numerical Comparisons 115
5.5 Conclusion 116
5.6 Appendix: Proof of Lemma 5.1 116

6 Homogeneous Distributed Systems with Boundary Conditions 119
6.1 Almost Toeplitz Systems 120
6.2 Example: H_2 control of a heated rod 121
6.3 Conclusion 123

7 Uncertain Distributed Systems 125
7.1 Subsystem model/Interconnection Structure 125
7.2 Computational Methods 127
  7.2.1 Upper bound for the matrix structured singular value 127
  7.2.2 D-K iteration for Decoupled Uncertainty Structure 128
7.3 Other uncertainty structures 130
  7.3.1 Parametric Uncertainty in Σ_s 130
  7.3.2 Dynamically Coupled Uncertainties 131
7.4 Example: $\mathcal{O}(N)$ D-K iterations 132
  7.4.1 Spatially Homogeneous, Nominal 133
  7.4.2 Spatially Heterogeneous, Nominal 133
  7.4.3 Heterogeneous, Uncertain 134
7.5 Conclusions 134
7.6 Appendix: Absorbing D-scalings 135

8 Multi-D Distributed Systems 137
8.1 Multi-D structure . . . 138
8.2 Numerical Example: Heat Flow on a Flat Plate . . . 139
8.3 Conclusion . . . 140
8.4 Appendix: 2-level L_r operator ⇒ 2-D interconnected system . . . 141

9 Repetitive and Iterative Learning Control 143
9.1 Background/Survey . . . 143
9.2 Formulation of ILC/RC into an Output Feedback Problem . . . 144
9.2.1 SSS structure of RC problem . . . 147
9.3 Illustrative Example . . . 149
9.3.1 Closed Loop Simulations . . . 150
9.3.2 Computational Complexity and Accuracy of the SSS Structure Preserving Techniques . . . 152
9.4 Conclusions . . . 153

10 LPV Analysis and Synthesis 155
10.1 Background/Survey . . . 155
10.2 Background, notation, and conversion to $\mathcal{S}_c$ realizations . . . 156
10.3 Use for LPV analysis and synthesis . . . 158
10.4 Airfoil Flutter Example . . . 159
10.5 Conclusion . . . 160

11 Reflections, Recommendations, and Conclusions 161
11.1 Research Results . . . 161
11.2 Research Recommendations . . . 166
11.3 Reflections . . . 170

Bibliography 171

Summary 183

Samenvatting 184

Curriculum Vitae 185

1 Introduction

We will try to convey the broad scope of this thesis by reordering the information from our various papers into a learnable structure. Hence it is organized a bit differently than most PhD theses, with chapters differentiated by theme instead of according to where the information was published. For example, this introductory chapter will be completely devoted to motivating the research problem and outlining the thesis, and should be readable by my parents, thus there will be almost no discussion herein of mathematics or control techniques. The remaining thesis is divided up into two main parts, "Part I" dealing with the mathematical methods, derivations, and proofs, and "Part II" dealing only with Part I's use in control of dynamical systems. Hence readers interested in computational methods can just read Part I, and those interested just in control applications can read only Part II if they choose.

In this chapter we'll first overview the motivations of distributed control and the inherent difficulties and research problems, and provide a brief summary of previous attempts at resolving them by other researchers in the field, relegating all details to Part II. In section 1.2 we'll then very briefly describe our own approach to the problems, and outline our specific goals and contributions of this thesis, after which we'll finish with a detailed description of the organization of the following chapters, with some recommendations for the reader on how best to proceed.

1.1 Fish and Birds: Distributed Systems

Distributed dynamical systems are all around us, from the flocks of birds in the sky and schools of fish in the sea to the traffic jams at rush hour and in server networks to even the fluid dynamics of the air we breath and the immune response of our bodies. Sometimes we would like to design, analyze, or control such systems, and this has been a hot research topic for around 40 years, with applications such as boundary layer and transition control in fluid mechanics [1], [2], flexible structures [3], heat conduction [4], highway traffic control [5] and vehicle platooning [6], iterative circuit networks [7], building anti-earthquake systems [8], aircraft and satellite formation flight [9], [10], large adaptive telescope mirrors [11], image processing [12], paper processing [13], irrigation networks [14], tissue development

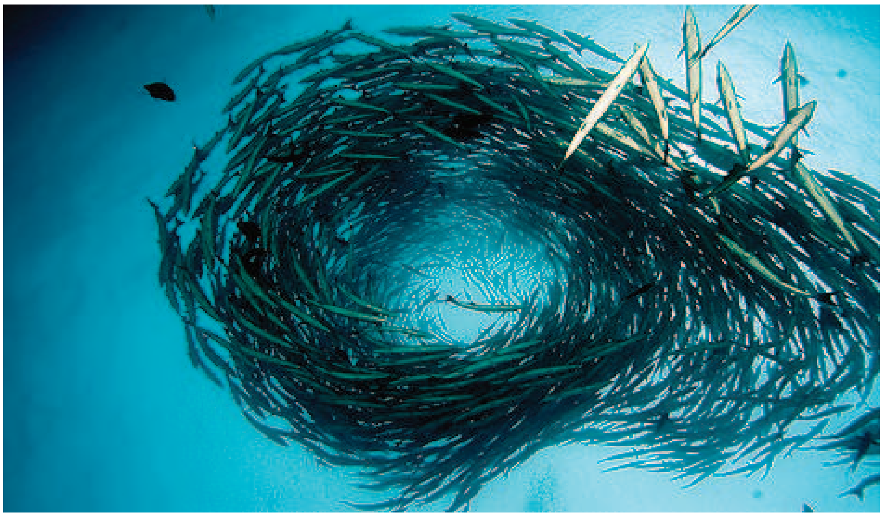


Figure 1.1: School of Barracuda. Photo by Robin Hughes, used under the creative commons license: http://www.flickr.com/photos/robinhughes/404457553/

from stem-cells [15], biochemical reactions [16], wind turbine farms [17][18], and varmint population control [19].

What makes these problems so interesting is their size; such systems consist of thousands or hundreds of thousands of individual parts, all moving and interacting in a dynamic way, leading to very rich and often unexpected behavior, making them difficult to predict or control using standard methods. Even when the subsystems themselves are relatively simple (e.g. linear and spatially homogeneous), complicated results can arise. For example, in [20] it is shown that the *linearized* Navier Stokes equations can exhibit turbulent ‘streaking’ and other features that have long been considered to be purely ‘nonlinear behavior’. This ‘emergent’ complexity makes distributed systems difficult, but the flip side of the coin is that simple control methods, when combined together on a large scale, can have very high performance. For example, no one has ever accused a herring of being intelligent, but a *school* of herring are certainly brilliant when dodging an attacking orca or porpoise, and the most advanced distributed control methods in existence can’t compare to their performance for controlling a 3-D, nonlinearly coupled, dynamically changing configuration of fish (see figure 1.1).

Likewise with the octopus (see figure 1.2). Octopuses are often cited as being the most intelligent invertebrates, and they can control their noodly appendages, consisting of literally thousands of actuators and sensors connected nonlinearly, with an extraordinary degree of speed and precision which the control community has been hard pressed to match [21], using distributed processing (octopuses keep most of their brains and much of their memory in their legs![22]).

Humans are also capable of distributed control, e.g. Rockettes-style dancers (see figure 1.3) can, without warning, pass a kick wave down the line with a dancer to dancer propogation rate twice as fast as the minimum human visual reaction time,

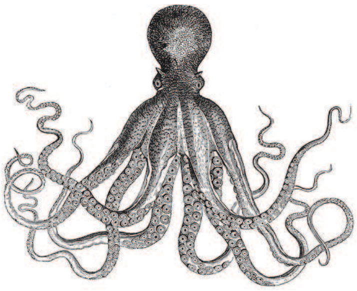


Figure 1.2: Octopus. Photo by Margaret, used under the creative commons license: http://www.flickr.com/photos/flashmaggie/3196980819/

Figure 1.3: Rockettes: Photo by Seth Vidal, used under the creative commons license: http://www.flickr.com/photos/skvidal/2239922329/

we're just not as good at it; birds with no formal training can pass maneuvers in any direction through a 3-D flock at *three times* their visual reaction rate[23], so fast that it lead early observers to believe that they used some type of electromagnetic or extra-sensory communication (telepathy!).

So we are constantly being reminded of the fantastic potential of distributed control, but due to the incredible size and complexity of such systems, engineering methods haven't caught up yet. Because of these complications, special theoretical and numerical techniques for dealing with distributed systems are being developed. In this thesis we will focus on the numerical aspects.

The challenge has been in the computational cost. The system matrix describing the input-state-output behavior of N interconnected subsystems (ODE's), each of size (order) n, will be $nN \times nN$, and thus most matrix arithmetic operations will require $\mathcal{O}(n^3N^3)$ floating point operations, making traditional robust or optimal controller design prohibitively expensive (slow) for fine discretizations or large numbers of discrete subsystems.

Many special approaches have been developed to surmount this obstacle. Researchers have used the special matrix structure to 'decouple' the systems from eachother for easier control [24][25][26][10], or tackle the structure directly for faster computations [27][28][29][30]. Other approaches model the systems as interconnections of smaller subsystems, and use special LMI techniques to compute topologically similar interconnected controllers[4][31][32][33][34]. There have also

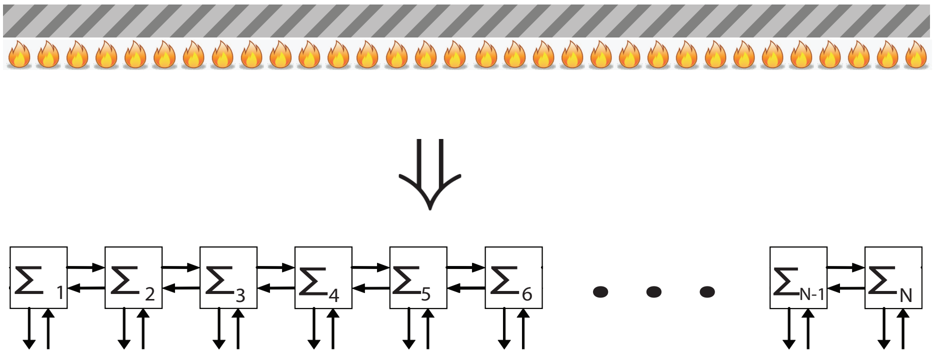


Figure 1.4: locally heated rebar can be approximated via finite difference as a set of interconnected linear systems

been recent results on actually *computing* distributed controllers in a distributed way, see e.g. [35][36]. These results approach half of what we might call 'distributed intelligence'; the ability to rely entirely on distributed sensors, actuators, and processors to handle a large scale distributed system from start to finish; the other half of the necessary results involve System Identification (SysID).

In multiagent systems, where each subsystem is *dynamically decoupled* from its neighbors (see e.g. [37]), structured SysID is easy: each subsystem is separately identified using whichever method is preferred, then they are coupled together using the (user designed) cost function. However, in distributed systems where the subsystems are dynamically connected, such as PDE's and many of the applications listed in the first paragraph of this subsection, SysID seems to be much more difficult than control or estimation, and there has not been much progress in comparison. In the 70's and early 80's, there were many methods developed for SysID of spatially homogeneous PDE's and 'inverse' problems (see [38] for an overview), and in [39][40][41] more general circulant and spatially invariant state space models are efficiently identified (see [42] for extensions to more general interconnection structures, and also the recent [43]). Unfortunately, often systems are not homogeneous in practice, and blindly applying standard SysID methods won't work very well, as they will be too computationally complex (e.g. Subspace ID and Output Error Methods [44] are at the very least $\mathcal{O}(N^3)$) and anyway will produce an identified model without any distributed structure, such that the above distributed controller synthesis methods will be of no use.

Hence in spite of these recent advances, for many distributed systems problems the computational techniques are still not satisfactory in speed or resulting controller performance, often with $\mathcal{O}(N^3)$ complexity or significant conservatism or both. To illustrate how important this issue of complexity is, we will now discuss a simple conceptual example.

Rebar, the thick steel wires used to reinforce concrete, are formed by pouring liquid steel into extremely long casters. The rebar is then 'tempered' by reheating to a certain temperature and allowing it to slowly cool. In tempering processes, the rate and homogeneity of the temperature is very important to the overall strength of the steel, so it might be desirable to use real-time control to monitor the temperature and adjust local heating or cooling to maximize the strength (in reality, rebar is dirt-cheap, so no one would actually do this, but the concept extends to more realistic applications). By approximating the heat conduction PDE, using standard finite difference methods, as a set of coupled ODE's or equivalently interconnected subsystems, as shown in figure 1.4, the problem of finding a real time controller to regulate the local heating can be written as a standard centralized or distributed control problem. However, the longer the section of rebar is, and the more mini-heaters under consideration, the longer it will take to perform the computations necessary for controller design. Using the standard centralized optimal control methods, or the special structured LMI methods mentioned above, the number of flops necessary to design the controller will be $\mathcal{O}(N^3)$, where N is the number of local heaters. For $N = 80, 100, 200, 300, 400, 500, 600$, the corresponding computation times are shown for standard centralized control design in figure 1.5 as a solid line. As we see, the computational time grows alarmingly with increasing N, and if we extend this $\mathcal{O}(N^3)$ trend to $N = 10,000$, the computation will take about 5.3 *days*, which is just too long. What we would like to find is *efficient* methods for control design for such problems; i.e. methods that grow in computational complexity proportional to the amount of information in the problem. If the rebar and local heater system is *heterogeneous*, that is, if each of the mini-heaters is different from its neighbors, or the rebar changes in thickness or composition along its length, then the problem will contain $\mathcal{O}(N)$ data, and we could hope for $\mathcal{O}(N)$ complexity algorithms, as shown as a dotted line in figure 1.5 (leading to a time of $\sim$ 15 minutes for $N = 10,000$). If the rebar and heater system is mostly *homogeneous*, that is to say, if all of the heaters are identical, and the rebar is constant in composition and thickness over its entire length (except at the ends), the problem will contain $\mathcal{O}(1)$ data, and we could hence hope for $\mathcal{O}(1)$ complexity algorithms, as shown as dash-dotted line in figure 1.5 (with a time of $\sim$ 7.2 seconds for $N = 10,000$). As we will show in this thesis, such efficient control synthesis computations actually *are* possible (the dotted and dash-dotted lines in figure 1.5 were produced with algorithms developed in this thesis, and discussed in Chapters 4 and 6).

1.2 Scope and Contributions of this Thesis

Our approach to the problem of computing distributed controllers can be summarized as follows. Interconnections of subsystems, such as those shown above, induce a special structure in their 'lifted' system matrices, for which we can develop 'efficient' (e.g. $\mathcal{O}(N)$ or $\mathcal{O}(1)$) structure preserving arithmetics: routines for calculating matrix addition, multiplication, inversion, and norm. Such arithmetics can then be used in special iterative algorithms (e.g. the sign iteration for solving Riccati equations) that preserve the structure, leading to computational methods for efficient design of controllers with this same matrix structure, which can be 'redistributed' into a set of subcontrollers linked in the same interconnection topology as the orig-

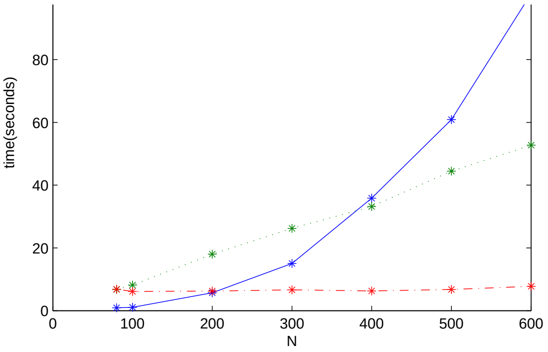


Figure 1.5: Controller synthesis times for heat conduction

inal subsystems. These techniques can be used for efficient stability and H_2, H_∞ performance analysis and H_2, H_∞ controller synthesis for fully homogeneous, homogeneous with boundary conditions, and fully heterogeneous systems. They can also be extended to D-scalings and 'D-K' iterations for robustness analysis and synthesis, structured model order reduction, and to multiple spatial dimensions, in a similarly efficient way.

For fully heterogeneous distributed systems, exploitation of the special matrix structure also makes efficient ($\mathcal{O}(N)$) parametric system identification possible, where the resulting system model is in the correct form for analysis and controller design using the above methods. Furthermore, for fully heterogeneous problems, due to the special form taken by the arithmetic, it's actually possible to perform all of these sysID, analysis and synthesis computations in a distributed manner, on a linear interconnection of microprocessors with distributed memory. Hence, given an unknown heterogeneous distributed system on a cartesian grid, it should now be possible to distribute microprocessors with local memory, sensing, actuation, and communication abilities, and have them first perform a distributed system identification, then using the resulting model, a distributed controller synthesis, and then to analyze the resulting closed loop performance, all in linear computational complexity ($\mathcal{O}(N)$ for N subsystems on a line)[1].

In this thesis we will lay out clearly all of the steps necessary for other students, researchers, or practitioners in the field to proceed from a fundamental knowledge of control and linear algebra to develop all of the results necessary to implement their own numerical Riccati solvers and distributed controller synthesis routines. Some of this material will be review, but much of it has been newly developed within this PhD project and has either recently been published or is first published in this thesis. These contributions are (in the order in which they will appear):

1. New derivations of structure preserving arithmetics of Laurent matrices with rational symbols, SSS matrices, and stable realizations of continuous domain

---

[1] Note that the $\mathcal{O}(N)$ complexity is *proven* in this thesis for all of these computations except for SysID, for which it is only observed in examples, see Chapter 4

transfer functions, and the new development of a structure preserving arithmetic of SSS matrices with almost Toeplitz structure (Chapter 2).

2. Proofs of the convergence of the above structured operators and matrices under the matrix sign iteration, and hence of the structure of solutions of Riccati and Lyapunov equations, and H_2 and H_∞ optimal controllers (Chapter 3).
3. Using the structured solutions to Riccati and Lyapunov equations above, the structured arithmetics in 1. are extended to multiple levels (Chapter 3)
4. Application of 1. and 2. to distributed systems with heterogeneous, homogeneous, and finite homogeneous structures, to develop computationally efficient and structure preserving distributed analysis, controller synthesis, model order reduction, and system identification routines. (Chapters 4-6).
5. For the fully heterogeneous case, a method to perform the computations in 4. on a distributed linear processor array. (Chapter 4).
6. Extensions of 4. to distributed systems with uncertainty and distributed systems in multiple spatial dimensions. (Chapters 7,8).
7. Application of 1. and 2. to computationally efficient synthesis and analysis of repetitive and iterative learning controllers (Chapter 9).
8. Application of 1. and 2. to analysis and synthesis of linear parameter varying systems and controllers (Chapter 10).

The last two items, ILC/RC and LPV control, of course don't have much to do with distributed control, but the computational methods of this thesis may also be applied to them, so we include these chapters as a bonus.

1.3 Structure of this Thesis

This thesis is divided into two main parts. Part I is devoted to the explanation and development of the fundamental tools, such as the structure preserving matrix arithmetics and sign iterations, and Part II is devoted to applying the resulting methods to different types of control and analysis problems. We will show how each application leads to a control problem involving one of the structures in Chapter 2, on which we then use the iterative techniques in Chapter 3 to design controllers of the same structure, demonstrating the computational efficiency on a practical example.

There are no physical applications in Part I and (almost) no proofs in Part II, and readers can selectively choose their chapters of interest accordingly. In the following we will give a detailed description of the contents of each of the chapters. At the end of each chapter listing, we will reference our publications in which this work first appeared, which are in turn listed at the end of this chapter.

- Part I: Fundamentals

  - Chapter 2: “Mixed Causal Input-Output Operators and their Arithmetic” is the densest of the thesis. After a background overview of structured matrices, the paradigm of mixed causal systems is used to fully derive a structure preserving arithmetic of Laurent matrices with rational symbols. In the following sections, these results are extended (using the same derivation method) to SSS matrices, SSS matrices with almost Toeplitz structure, and stable realizations of transfer matrices on the imaginary axis. First appeared in: [D][F][G][J]
  - Chapter 3: “Structure Preserving Iterations” begins with an overview of the matrix sign function and iteration, and some applications in systems and control(some of which are new). The matrix sign function is then applied to the structured matrices and operators from Chapter 2, and different types of convergence are proven, providing results on e.g. the structure of Riccati solutions and the structure of H_∞ controllers. These results are in turn used to extend the arithmetics of Chapter 2 to multiple levels, and thus multiple physical dimensions. The effects of numerical rounding and approximation errors, and techniques for overcoming them, are also discussed. First appeared in: [A][D][F][H][L][J]

- Part II: Applications of Part I to different classes of large scale systems
  - Chapter 4: “Heterogeneous Distributed Systems” deals with arbitrarily spatially varying systems connected on a line, with examples of H_2 and H_∞ control synthesis, model order reduction, system identification, and distributed computing, all in $\mathcal{O}(N)$ computational complexity. First appeared in: [A][B][H][I][L][M].
  - Chapter 5: “Doubly Infinite Homogeneous Distributed Systems” treats perfectly homogeneous systems on a line, with scalar and matrix examples of LQR synthesis for heat conduction and a comparison with some other distributed control techniques on a car platooning example. Relevant publications: [D][E][J].
  - Chapter 6: “Homogeneous Distributed Systems with Boundary Conditions”, deals with systems that are mostly homogeneous, with heterogeneities at the boundaries, in $\mathcal{O}(1)$ computational complexity. Shows a scalar example of H_2 synthesis. First appeared in: [F].
  - Chapter 7: “Uncertain Distributed Systems”, extends chapters 4,5,6 to systems with uncertainties, with a matrix example of robust synthesis (H_∞ and D-scalings) for a heterogeneous system on a line in $\mathcal{O}(N)$ computational complexity. First appeared in: [K].
  - Chapter 8: “Multi Dimensional Distributed Systems” treats systems on cartesian grids, with a 2-D scalar example of LQR synthesis. First appeared in: [D].
  - Chapter 9: “Repetitive and Iterative Learning Control” is a bonus chapter, which shows how to fit RC/ILC for LTV systems into the SSS structure, leading to RC/ILC design in $\mathcal{O}(N)$ computational complexity, with an example of LQG control of a beamer. First appeared in: [C].

  - Chapter 10: “LPV Analysis and Synthesis”. Another bonus chapter, illustrating how the results of Part I can be used with stable realizations of transfer functions on the imaginary axis to perform some basic LPV computations. First appeared in: [G].

The thesis ends with Chapter 11: Reflections, Recommendations, and Conclusions.

As stated at the beginning of the chapter, this thesis is organized with the reader in mind, and we thus have some advice for effective reading order. As shown in figure 1.6, we recommend that the reader first thoroughly absorb the background methods in Part I, before splitting off to read about distributed systems in Track 1 or other applications in Track 2. However, Part I is probably too boring for most people who pick up this thesis, so we've made it possible to understand the chapters in Part II without fully comprehending the underlying math. One of the chapters 4,5, or 6 should be read as background before hitting chapters 7 or 8. For the particularly time-pressed reader, it is also probably possible to skip Chapters 2-10 completely, but gain enough from the summary in Chapter 11 to pretend to have read the whole thesis, but we don't condone it.

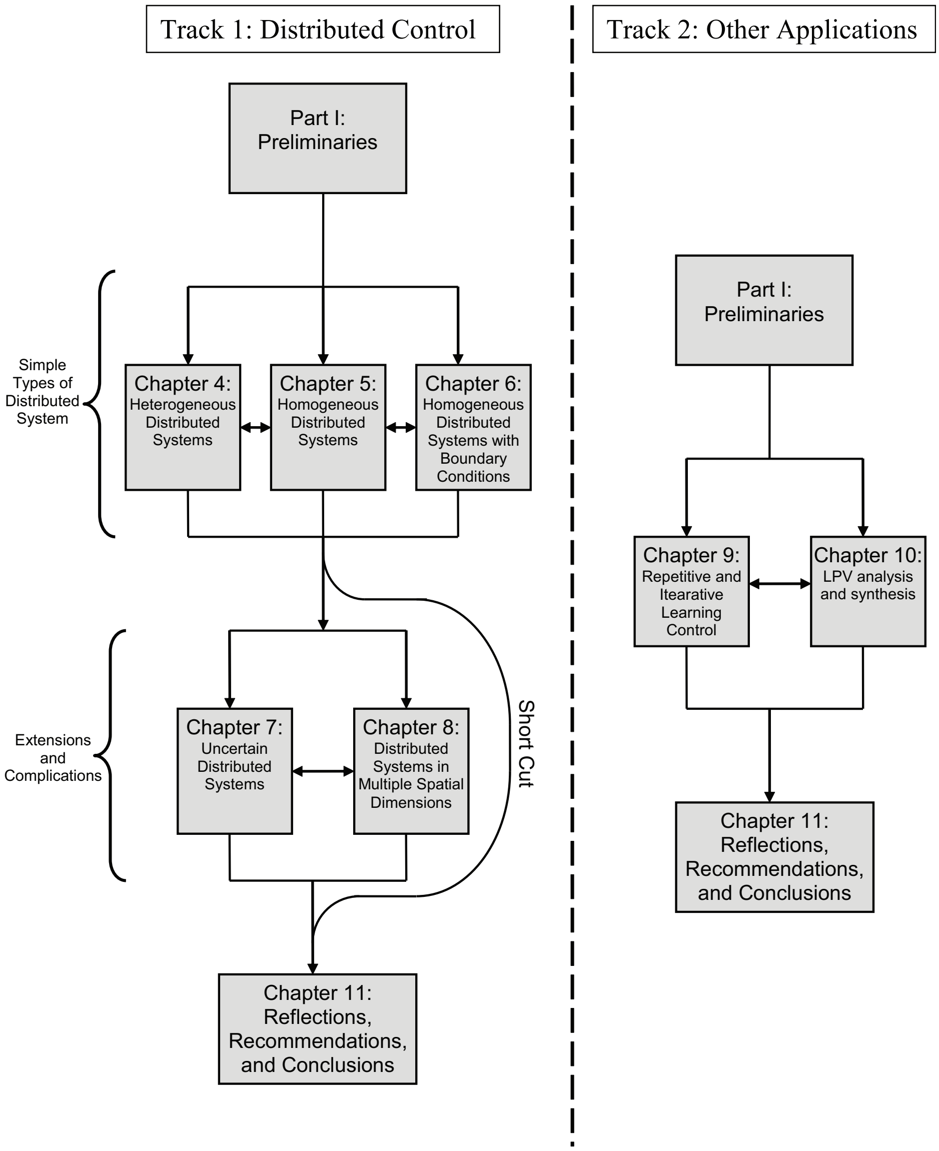

Track 1: Distributed Control
Track 2: Other Applications
Part I: Preliminaries
Simple Types of Distributed System
Chapter 4: Heterogeneous Distributed Systems
Chapter 5: Homogeneous Distributed Systems
Chapter 6: Homogeneous Distributed Systems with Boundary Conditions
Extensions and Complications
Chapter 7: Uncertain Distributed Systems
Chapter 8: Distributed Systems in Multiple Spatial Dimensions
Short Cut
Chapter 11: Reflections, Recommendations, and Conclusions
Part I: Preliminaries
Chapter 9: Repetitive and Itearative Learning Control
Chapter 10: LPV analysis and synthesis
Chapter 11: Reflections, Recommendations, and Conclusions

Figure 1.6: Recommended Reading Trajectories

Publications

Journal Papers and Book Chapters

[A] Justin K. Rice and Michel Verhaegen, “Distributed Control: A Sequentially Semi-Separable Approach for Heterogeneous Linear Systems”, *IEEE Transactions on Automatic Control*, 2009, vol. 54, pp. 1270:1283.

[B] Justin K. Rice and Michel Verhaegen, “Robust and Distributed Control of a Smart Blade”, *Wind Energy*, in press.

[C] Justin K. Rice and Michel Verhaegen, “A Structured Matrix Approach to Efficient Calculation of LQG Repetitive Learning Controllers in the Lifted Setting”, *International Journal of Control*, in press.

[D] Justin K. Rice and Michel Verhaegen, “Distributed Control in Multiple Dimensions: A Structure Preserving Computational Technique”, *IEEE Transactions on Automatic Control*, Accepted for Publication.

[E] Justin K. Rice, Paolo Massioni, Tamas Keviczky, and Michel Verhaegen, “Distributed Control Methods for Structured Large-Scale Systems”, *Efficient Modeling and Control of Large-Scale Systems*, Springer, in press.

Conference Papers

[F] Justin K. Rice and Michel Verhaegen, “Exploiting Semi-Separable and Toeplitz Structure in Distributed Control of Homogeneous Systems with Arbitrary Boundary Conditions”, *IEEE Int. Conf. on Control & Aut.*, 2010, in press.

[G] Justin K. Rice and Michel Verhaegen, “LPV Analysis and Control Using Fast Iterative Solutions to Rationally Parametric Lyapunov and Riccati Equations”, *American Control Conference*, 2010, in press.

[H] Justin K. Rice and Michel Verhaegen, “Structure Preserving Model Order Reduction of Heterogeneous 1-D Distributed Systems”, *American Control Conference*, 2009.

[I] Justin K. Rice and Michel Verhaegen, “Distributed Computations and Control in Multi-Agent Systems”, *Int. Conf. on Autonomous Robots and Agents*, 2009.

[J] Justin K. Rice and Michel Verhaegen, “Distributed Control of Spatially Invariant Systems Using Fast Iterative Solutions to Rationally Parametric Matrix Problems”, *IEEE Conference on Decision and Control*, 2009.

[K] Justin K. Rice and Michel Verhaegen, “Robust Control of Distributed Systems with Sequentially Semi-Separable Structure”, *European Control Conference*, 2009.

[L] Justin K. Rice and Michel Verhaegen, “Distributed Control: A Sequentially Semi-Separable Approach”, *IEEE Conference on Decision and Control*, 2008.

[M] Justin K. Rice and Michel Verhaegen, “Efficient System Identification of Heterogeneous Distributed Systems via a Structure Exploiting Extended Kalman Filter”, in preparation.

Notation

$\mathbb{R}$	set of Real numbers
$\mathbb{C}$	set of Complex numbers
$\mathbb{N}$	set of Natural numbers
$\mathbb{Z}$	set of Integers
$\Re(\cdot)$	Real part of $\cdot$
$\Im(\cdot)$	Imaginary part of $\cdot$
$\mathbb{C}_-$	Open left half plane
$\mathbb{C}_+$	Open right half plane
$\mathbb{R}_-$	set of strictly negative real numbers
$\mathbb{R}_+$	set of strictly positive real numbers
$\mathbb{T}$	unit circle: $\mathbb{T} = \{e^{\sqrt{-1}\theta} \forall \theta \in [0, 2\pi)\}$
$Tr(\cdot)$	trace operator
$(\cdot)^T$	transpose operator
$(\cdot)^*$	complex conjugate transpose operator
$\Vert \cdot \Vert_F$	Frobenius norm
$\Vert \cdot \Vert_\infty$	H_∞ norm of a system
$\Vert \cdot \Vert_2$	2-norm of a vector or induced 2-norm of a matrix or $\mathcal{H}_2$-norm of a system
$\lambda(X)$	spectrum of operator X
$X \succ 0$	indicates positive definiteness: $X = X^T$, $\lambda(X) > 0$
$\rho(X)$	spectral radius: $\rho(X) = \sup_{\lambda_0 \in \lambda(X)} \vert\lambda_0\vert$
$X \otimes Y$	Kronecker product between the matrices X and Y
$l_2 = l_2(\mathbb{Z}, \mathbb{C}^p)$	the set of sequences, $x \in l_2$ with norm: $\Vert x\Vert_{l_2}^2 = \sum_{k=-\infty}^{\infty} \Vert x(k)\Vert_2^2 < \infty$
l_1	the set of sequences, $x \in l_1$ with norm: $\Vert x\Vert_{l_1} = \sum_{k=-\infty}^{\infty} \vert x(k)\vert < \infty$
l_∞	the set of sequences, $x \in l_\infty$ with norm: $\Vert x\Vert_{l_\infty} = \sup_k \vert x(k)\vert < \infty$
$\mathcal{L}_\infty(\mathbb{T})$	the space of matrix valued functions essentially bounded on $\mathbb{T}$
$\mathcal{RL}_\infty(\mathbb{T})$	the real proper rationally parametric subspace of $\mathcal{L}_\infty(\mathbb{T})$
$\mathcal{L}_2(\mathbb{T})$	$\mathcal{L}_2(\mathbb{T}) = \{F(z) : \Vert F(z)\Vert_{\mathcal{L}_2} = \frac{1}{2\pi}\int_0^{2\pi} Tr(F^*(e^{i\theta})F(e^{i\theta}))d\theta < \infty\}$
$\bar{(\cdot)}$	indicates a lifted vector or structured matrix $\cdot$
I	identity matrix of arbitrary dimension
I_m	identity matrix of defined dimension $l \in \mathbb{N}$
$\mathcal{F}_z$	Fourier Transform

2 Mixed Causal Input-Output Operators and their Arithmetic

> Not everything that is more difficult is more meritorious.
>
> -Saint Thomas Aquinas

2.1 Introduction

As we discussed in Chapter 1, in the modeling of distributed systems we often encounter very large matrices. Multiplying and inverting such large matrices is in turn very 'computationally expensive' i.e. it takes a long time, even on a fast computer, and this makes control design or analysis difficult or impossible. However, in certain cases, matrices have a special 'structure' that can be used to dramatically speed up computations, permitting control design for systems of even the most enormous dimensions. In this chapter we will investigate a few types of related matrix structure, deriving very fast computational methods for their addition, multiplication, and inversion. As we will see in Part II, these structured matrices represent distributed and repetitive systems, and can be used with the iterative methods in Chapter 3 for very fast controller design and analysis. This chapter is structured such that readers who are unfamiliar with complex structured matrices should be able to gradually get a feel for them. After a brief overview of the idea of structured matrices, we'll discuss mixed causal systems, which we will use in section 2.2 to derive the arithmetic of Laurent matrices with rational symbols. We will rather painstakingly show every step of the derivations in this section, so that they can then be used as blueprints for the derivations of the arithmetics of SSS matrices in section 2.3, 'almost-Toeplitz' SSS matrices in section 2.4, and stable realizations of transfer matrices on the imaginary axis in section 2.5.

First we should get a clear idea of what we mean by computational complexity. We will measure complexity in terms of 'flops' that is, floating-point arithmetic computations. The number of flops necessary to solve a certain computational problem is a good measure of the time required. Since we will be concerned in this thesis with very large problems, we are more interested in how the computational cost grows as a function of the size of the problem than on the precise number of flops itself. To represent the functional dependency in computational complexity, we will use 'big O' notation,

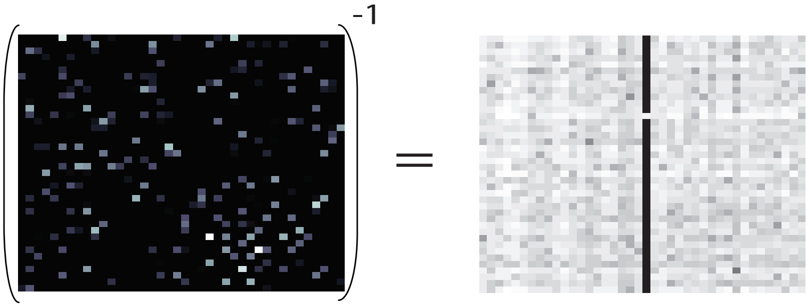
Definition 2.1 *[45] A positive function is $f(N) \in \mathcal{O}(N^{\alpha})$ if there exist finite positive constants, $\infty > c, \kappa > 0$ such that $f(N) < cN^{\alpha}, \forall N > \kappa$.*

Informally, we will say that a procedure 'is' $\mathcal{O}(g)$ if it can be computed in $f(N) \in \mathcal{O}(g)$ flops. $\mathcal{O}(1)$ will mean bounded from above by a constant. Often, complexity is rather arbitrarily broken up into the categories of 'tractable' for problems with any degree of polynomial complexity ($\mathcal{O}(N^a)$, $a \in \mathbb{R}$) and 'intractable' for others, such as exponentially complex problems. However, high-degree polynomial complexity (e.g. $\mathcal{O}(N^{10})$) is often just as useless for large scale problems as exponential or combinatorial complexity, whereas decreasing a solution method from cubic ($\mathcal{O}(N^3)$)to linear($\mathcal{O}(N)$) complexity makes a huge difference in practice. It is this sort of improvement that we will focus on in this work.

2.1.1 The Idea of Structured Matrix Arithmetic

For dense unstructured matrices in $\mathbb{R}^{N \times N}$, trace is $\mathcal{O}(N)$, transpose, Frobenius norm, and addition are $\mathcal{O}(N^2)$, multiplication, inversion, and 2-norm are $\mathcal{O}(N^3)$. For many applications with $N = 5$ or $N = 10$, this doesn't matter, but for some of the very large distributed systems discussed in chapter 1, we might have $N > 10000$, for which $\mathcal{O}(N^3)$ is completely impractical. However, there are some special types of matrices for which these complexities can be improved. The special matrices we will discuss next will all fit in the category of 'data sparse'. (Note that most of the information in this section is well known, and unless specified otherwise, can be found in Golub and van Loan's famous [46] or even *Wikipedia*).

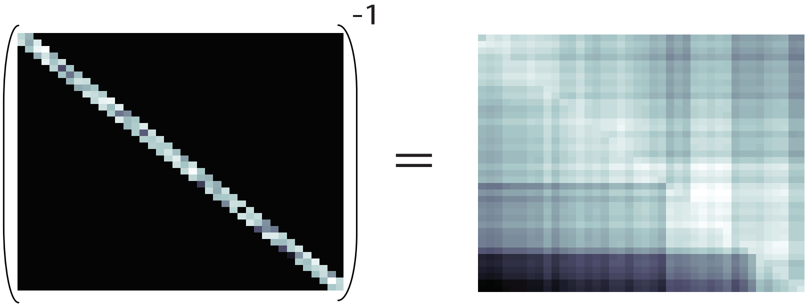
Traditionally, 'sparse' matrices are considered to be matrices in $\mathbb{R}^{N \times N}$ with only $\mathcal{O}(N)$ non-zero entries. Obviously, addition, and Frobenius norm are $\mathcal{O}(N)$ for such matrices, but addition and multiplication generally decrease the 'sparsity' of the matrices, and the inverse of a sparse matrix might be 'full' (having $\mathcal{O}(N^2)$ non-zero entries) as we see here:



Comment 2.1 *The picture above is an image of a matrix made with Matlab's 'imagesc' function. In this section, for illustrative purposes, we will always display the logarithm of the absolute value of the elements of matrices, i.e. in Matlab*

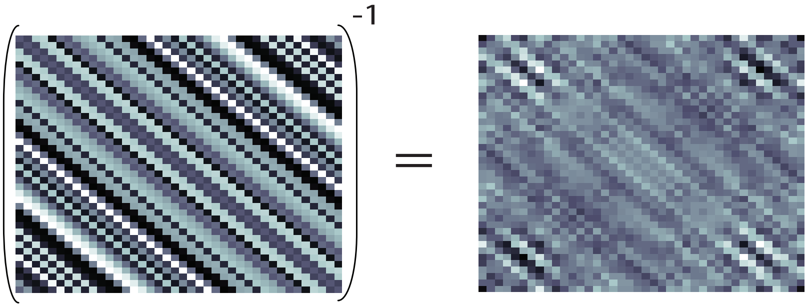
notation $\log_{10}(abs(X))$ *to better show the zero(black) and non-zero(gray and white, depending on intensity) areas.*

But this is not true of all sparse matrices. A simple example is the class of diagonal matrices; diagonal matrices in $\mathbb{R}^{N\times N}$ have $\mathcal{O}(N)$ complexity for all of the operations listed above. Furthermore, arithmetic preserves the diagonal structure; e.g. if you multiply two diagonal matrices together and then invert the product, you still have a diagonal matrix. The same holds for block diagonal matrices. Such matrices aren't very general in distributed systems though, and a more useful type of matrix, the 'banded' matrix, e.g. tridiagonal, does not have all of these nice characteristics. While tridiagonal matrices may be added and multiplied in $\mathcal{O}(N)$, the inverse of a tridiagonal matrix is generally full:



This motivates the more useful characterization of such matrices as 'data-sparse', which means that a matrix in $\mathbb{R}^{N\times N}$ can be stored using only $\mathcal{O}(N)$ data. This class obviously contains 'sparse' matrices, but also some full matrices, for example, a low rank matrix $A \in \mathbb{R}^{N\times N}$ may be factored into the form $A = U\Sigma V$ where $U \in \mathbb{R}^{N\times m}$, $V \in \mathbb{R}^{m\times N}$, $\Sigma \in \mathbb{R}^{m\times m}$, $m \ll N$, and cheaply stored this way, although A may very well be a full matrix. Low rank matrices may also be added and multiplied in $\mathcal{O}(N)$, although they often gain rank and complexity through these operations, and of course they cannot be inverted!

Other types of nice 'data-sparse' matrices, which are also often used to model distributed systems, are Toeplitz(Hankel) matrices, which have the virtue of being constant along their diagonals(anti-diagonals). While such matrices can be added in $\mathcal{O}(N)$, and multiplied and inverted in $\mathcal{O}(N^2)$, they unfortunately lose their structure under multiplication and inversion.



The fact that Toeplitz, low rank, and banded matrices all lose their structure under some of these basic arithmetic operations is unfortunate, since it prevents the efficient use of iterative algorithms. This motivates new types of data sparse matrices that do preserve their structure under these operations, and still have very low computational complexity. Research into such matrices began in the late 90's, and resulted in the class of Sequentially Semi-Separable(SSS) Matrices [47][48] with $\mathcal{O}(N)$ complexity, and the class of Hierarchical Matrices($\mathcal{H}$-matrices) [49] with $\mathcal{O}(N\log(N))$ complexity. Both classes of matrices preserve their structure through $+$, $\times$, and $^{-1}$, and additionally LU and QR factorizations, and can also have their Frobenius norms calculated quickly.

$\mathcal{H}$ matrices have already been used in structure preserving iterations[28][50] involving discretizations of PDEs, and we will not discuss them further in this section (see Chapter 11 for more perspective), but will only discuss SSS matrices and near relatives.

2.1.2 The Idea of Mixed Causal Systems

The SSS matrices and their infinitely large operator counterparts that we will be working with happen to have interpretations as the input-output maps of 'mixed causal' linear systems, and thinking of them in this way will help us to derive their arithmetic, and also to have some intuition as for how and why it works. In the linear systems and control field, we're used to dealing with systems that move forward in time, e.g. the discrete time system:

$$x(t+1) = f_c(x(t)) \tag{2.1}$$

where $f_c()$ is some function, t is time, and x is the state. Because our model steps forward in time; the state at time t is dependent on the state at previous times $t-1, t-2, ...$, and not vice-versa, we call such a system 'causal' with respect to time. Conversely, we can imagine systems that are 'anti-causal' and move backwards in time:

$$x(t-1) = f_a(x(t)) \tag{2.2}$$

We as humans don't encounter such systems very often; we generally think of time as only moving in one direction. However, anti-causal systems still work out mathematically, basically in the same way as causal systems (except backwards), and for relating inputs $u(t) \in l_2$, to outputs $y(t)$, an anti-causal system representation might be just as valid as a causal one.

Example 2.1 *Consider the causal linear LTI system:*

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{2.3}$$

where A is assumed invertible. If we define a new state variable as a backwards shift of the old one: $\hat{x}(k) = x(k+1)$ then the same input-output behavior can be *captured by the anti-causal system:*

$$\begin{array}{rcl} \hat{x}_{k-1} & = & A^{-1}\hat{x}_k - A^{-1}Bu_k \\ y_k & = & CA^{-1}\hat{x}_k + (D - CA^{-1}B)u_k \end{array} \tag{2.4}$$

That is to say, for some input signal u, the outputs of the two systems will be identical.

As we said before, systems moving backwards in time aren't commonly encountered, but anti-causal systems in other variables, such as spatial, can be very useful, as we will see later on.

In the following, we won't require any fancy results of mixed causal systems, we'll just use the intuition to derive a structure preserving arithmetic for SSS matrices and three other closely related classes of matrix and operators in an easy and constructive way. We will start with Laurent operators with rational symbols because they are the simplest class that we consider, and because the derivation techniques we use for them can be easily extended to work for all of the other classes.

2.2 Discrete LTI Systems and Laurent Matrices with Rational Symbols

Consider the discrete linear time invariant(LTI) mixed causal system:

$$\begin{bmatrix} x^a_{i-1} \\ x^c_{i+1} \end{bmatrix} = \begin{bmatrix} W & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} x^a_i \\ x^c_i \end{bmatrix} + \begin{bmatrix} V \\ Q \end{bmatrix} u_i, \qquad y_i = \begin{bmatrix} U & P \end{bmatrix} \begin{bmatrix} x^a_i \\ x^c_i \end{bmatrix} + Du_i \tag{2.5}$$

for $i \in \mathbb{Z}$. The state variables here, x^a_i and x^c_i, are extraneous, so called 'latent variables' in behavioral systems jargon, and we can eliminate them by substitution to obtain an equation that just relates the outputs y to the inputs, $u \in l_2$:

$$\underbrace{\begin{bmatrix} \vdots \\ y_i \\ y_{i+1} \\ y_{i+2} \\ y_{i+3} \\ y_{i+4} \\ \vdots \end{bmatrix}}_{\bar{y}} = \underbrace{\begin{bmatrix} \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \ddots & D & UV & UWV & UW^2V & UW^3V & \ddots \\ \ddots & PQ & D & UV & UWV & UW^2V & \ddots \\ \ddots & PRQ & PQ & D & UV & UWV & \ddots \\ \ddots & PR^2Q & PRQ & PQ & D & UV & \ddots \\ \ddots & PR^3Q & PR^2Q & PRQ & PQ & D & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}}_{\bar{A}} \underbrace{\begin{bmatrix} \vdots \\ u_i \\ u_{i+1} \\ u_{i+2} \\ u_{i+3} \\ u_{i+4} \\ \vdots \end{bmatrix}}_{\bar{u}} \tag{2.6}$$

$\bar{A}$ is called a 'Laurent matrix' because it is doubly infinite and has a Toeplitz structure, which is to say that it is constant down the diagonals (see [51] and [52]

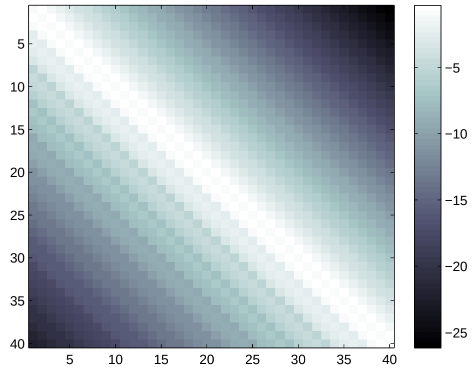


Figure 2.1: the log of the absolute value of a 40 by 40 section of a Laurent Matrix(we can't show the whole thing, since it is infinite!). In Matlab notation: $log_{10}(abs(\bar{A}(i:i+40, i:i+40)))$ for any $i \in \mathbb{Z}$

for a very nice introduction to Laurent matrices). Assuming that $\rho(R), \rho(W) < 1$, the absolute values of the entries moving away from the diagonal on either side are bounded by an exponential decay. For example, in figure 2.1 we show a section of one of these Laurent matrices. However, $\bar{A}$ in (2.6) actually has even more than just Laurent structure, in that the formulas for the diagonals are related to eachother. Such an $\bar{A}$ is said to have a 'rational symbol', as we will next discuss.

2.2.1 Fourier Transform, the Operator Symbol, and Useful Equivalencies

First we'll need a some mathematical background. $\mathcal{F}_z$ will indicate the Fourier transform mapping a (matrix) function $X(k) \in l_2(\mathbb{Z})$ to a (matrix) function $X(z) \in \mathcal{L}_2(\mathbb{T})$

$$\mathcal{F}_z X(k) = X(z) = \sum_{k=-\infty}^{\infty} X(k) z^k, \qquad \mathcal{F}_z^{-1} X(z) = X(k) = \frac{1}{2\pi i} \oint_{\mathbb{T}} X(z) z^{k-1} dz$$

If we apply this Fourier transform to the input-output equation above (assuming that $\rho(R), \rho(W) < 1$):

$$\mathcal{F}_z \bar{y} = \mathcal{F}_z \bar{A} \mathcal{F}_z^{-1} \mathcal{F}_z \bar{u} \tag{2.7}$$

Then we get the transfer function form, in the Laplace domain:

$$y(z) = \underbrace{[P(zI-R)^{-1}Q + D + U(z^*I-W)^{-1}V]}_{A(z)} u(z), \qquad z \in \mathbb{T} \tag{2.8}$$

Where the transfer matrix $A(z)$ is called the 'symbol' of the Laurent operator $\bar{A}$. The nice thing about such rational symbols is that if you have a stable realization of it ($\{P, R, Q, D, U, W, V\}$ with $\rho(R), \rho(W) < 1$), you can construct the Laurent matrix directly. Hence we will use the notation $\bar{A} = L_r\{P, R, Q, D, U, W, V\}$ as shorthand for the infinite dimensional Laurent matrix $\bar{A}$, so

$$L_r\{P, R, Q, D, U, W, V\} = \begin{bmatrix} \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \ddots & D & UV & UWV & UW^2V & \ddots \\ \ddots & PQ & D & UV & UWV & \ddots \\ \ddots & PRQ & PQ & D & UV & \ddots \\ \ddots & PR^2Q & PRQ & PQ & D\ddots & \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \tag{2.9}$$

Because the Fourier transform is an isometric isomorphism, this gives us a way to construct an arithmetic of these infinite Laurent matrices [51]:

$$\begin{aligned} \bar{A} + \bar{B} &= \mathcal{F}_z^{-1} A(z) \mathcal{F}_z + \mathcal{F}_z^{-1} B(z) \mathcal{F}_z = \mathcal{F}_z^{-1} (A(z) + B(z)) \mathcal{F}_z \\ \bar{A}\bar{B} &= \mathcal{F}_z^{-1} A(z) \mathcal{F}_z \mathcal{F}_z^{-1} B(z) \mathcal{F}_z = \mathcal{F}_z^{-1} (A(z) B(z)) \mathcal{F}_z \\ \bar{A}^{-1} &= (\mathcal{F}_z^{-1} A(z) \mathcal{F}_z)^{-1} = \mathcal{F}_z^{-1} (A(z)^{-1}) \mathcal{F}_z \end{aligned} \tag{2.10}$$

That is to say, the computations of addition, multiplication, and inversion of Laurent matrices can equivalently be performed on the symbols of the matrices. There are also nice equivalencies between the norm and spectra [51][52](the spectral equivalence on the right is only guaranteed for rational symbols):

$$\|\bar{X}\| = \|X(z)\|_\infty, \qquad \lambda(\bar{X}) = \lambda(X(\mathbb{T})) = \bigcup_{z \in \mathbb{T}} \lambda(X(z)) \tag{2.11}$$

(The first is familiar for causal systems, ($\{P, R, Q, D, 0, 0, 0\}$), where the transfer function H_∞ norm is equal to the system induced l_2 norm.) The last formula is interesting to contemplate: the spectrum of the Laurent matrix, $\bar{X}$, is just equal to the spectrum of the transfer matrix $X(z)$ everywhere on the unit circle $z \in \mathbb{T}$, or equivalently, the union of the spectrums of the matrices $X(z_0)$ at each point $z_0 \in \mathbb{T}$ (for those readers who are familiar: $\lambda(\bar{X})$ is the Nyquist plot of $X(z)$). Since $X(z)$ is a rational function (2.8)and has no poles on the unit circle, it is therefore a continuous function of $z \in \mathbb{T}$. Since the eigenvalues of some complex matrix $X(z_0)$ are continuous functions of the entries of that matrix, $\lambda(X(z))$ is a continuous function of z and thus forms a closed curve in the complex plane (or curves, if $X(z)$ is a matrix).

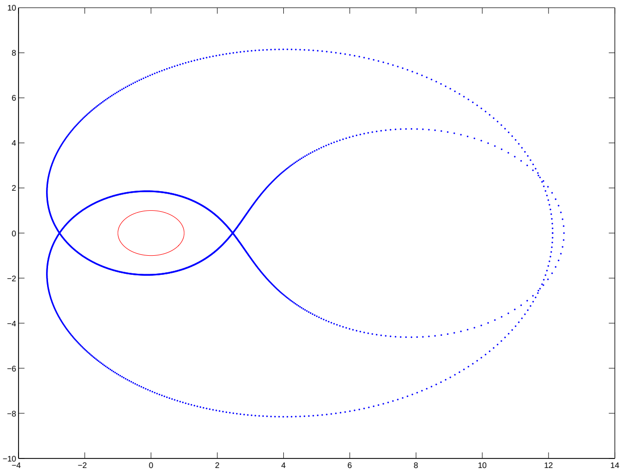


Figure 2.2: 2000 point discretization of the spectrum of $\bar{A}$, with the unit circle shown for reference

Example 2.2 *Consider:* $\bar{A} = L_r\{P, R, Q, D, 0, 0, 0\}$ *with* $D = 0.2120$ *and*

$$P = \begin{bmatrix} -0.7126 & -0.5247 & 0.7653 \end{bmatrix}, R = \begin{bmatrix} 0.2144 & 1.0358 & 1.5484 \\ -0.6355 & 0.9121 & -1.2734 \\ -0.1434 & -0.7232 & -0.9274 \end{bmatrix}, Q = \begin{bmatrix} -0.5810 \\ -0.1888 \\ -0.8399 \end{bmatrix}$$

To find $\|\bar{A}\|$*, since it is causal (lower triangular), we can simply use the Bounded Real Lemma:*

$$\|\bar{A}\| = \|A(z)\|_\infty = \| \left[\begin{array}{c|c} R & Q \\ \hline P & D \end{array} \right] \|_\infty = 12.4657 \tag{2.12}$$

as implemented in Matlab for discrete time systems: $Asymbol = ss(R, Q, P, D, -1)$; $Anorm = norm(Asymbol, \text{'}inf')$. *This method would also work if* $\bar{A}$ *were anti-causal(upper triangular), but note that for 'full' Laurent matrices with both lower and upper sides, we will need to be more creative, as we will see in section 2.2.8.*

As for the spectrum, to get a good picture of $\lambda(\bar{A})$*, we can calculate* $\lambda(A(z_0))$ *sampled over the unit circle* $z_0 \in \mathbb{T}$*, as in figure 2.2. Note that our 2000 points* z_0 *are uniformly sampled on the unit circle, but this doesn't lead to a uniform distribution of points in* $\lambda(\bar{A})$*, and herein lies the danger. Even though we've used alot of points in our drawing above, there are still large gaps of about 0.3 at the right end of the curve. We know that our spectrum will be continuous, but it can otherwise be arbitrarily quickly varying, and needs not even be Lipschitz smooth [53], so there could be cusps and other evil things in our curve that we might miss, even with very fine sampling. For this reason, we'll have to develop other techniques for reliably calculating or bounding* $\rho(\bar{A})$ *(see Chapter 3).*

Note that much wilder curves are possible, and if $A(z)$ *were in* $\mathbb{C}^{n\times n}$*, there* *would be n closed curves in its spectrum instead of 1, additionally confounding the problem (see example 2.4).*

So, while the infinite dimensional matrices may be puzzling, we've seen that we can drastically simplify things by looking at their symbols. In the next subsection we'll show a way to make symbol arithmetic even easier.

2.2.2 $\mathcal{S}$-realizations

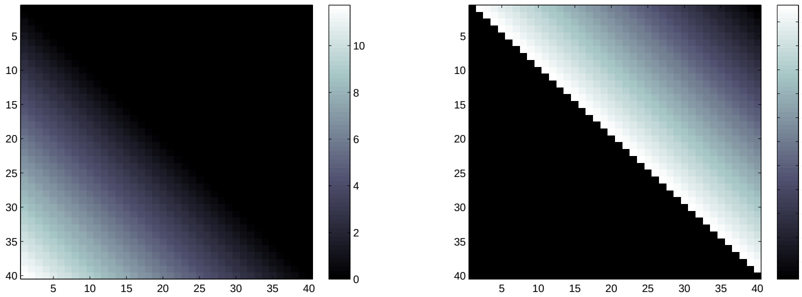
In the last subsection, we motivated using the rational symbols of L_r matrices, instead of the big, infinite extent, unwieldy matrices themselves. However, the symbols come in the form of transfer matrices, and transfer matrix arithmetic is computationally slow (especially inversion, the available algorithms for which are either ill-conditioned, or use symbolic algebra[54][55]) and often leads to unnecessarily high orders, so it is undesirable for us to actually use the transfer matrix form of the symbols of our Laurent operators. Instead we will work only in the state space of stable realizations of these symbols, for which we will develop a simple and efficient arithmetic in this section. This will make our technique fast and efficient in practice, and easy to program in software like Matlab. Before starting any definitions, we will first motivate the need to differentiate between 'stable' and 'unstable' realizations.

Note that a bounded L_r matrix, $\bar{X}$, with structure as in (2.6), has a unique symbol, $X(z) \in \mathcal{RL}^\infty(\mathbb{T})$ [51], but can be represented by many different state space realizations $L_r\{P, R, Q, D, U, W, V\}$. We can say that:

Lemma 2.1 *An L_r matrix $\bar{X}$ is a bounded operator if and only if there exists a realization of matrices $\bar{X} = L_r\{P, R, Q, D, U, W, V\}$ each bounded in norm, with $\rho(R) < 1$ and $\rho(W) < 1$.* ***Proof:*** *We start with sufficiency: $\|\bar{X}\| = \|X(z)\|_\infty \leq \|D\| + \|P(zI - R)^{-1}Q\|_\infty + \|V^*(zI - W^*)^{-1}U^*\|_\infty$. For necessity, we know that $\bar{X}$ is bounded if and only if its symbol, $X(z)$ is bounded on the unit circle: $X(z) \in \mathcal{RL}^\infty(\mathbb{T})$([51], theorem 1.1). When this is the case, $X(z)$ can always be split up into $X(z) = D + X^{out}(z) + X^{in}(z)$ with D constant, $X^{out}(z) \in \mathcal{RH}_\infty^\perp$ analytic inside the unit circle and $X^{in}(z) \in \mathcal{RH}_\infty$ analytic outside the unit circle. Bounded strictly stable realizations (P, R, Q) and (U, W, V) can then be found such that $X^{in}(z) = P(zI - R)^{-1}Q$ and such that $X^{out}(z) = U(z^{-1}I - W)^{-1}V$.* □

Furthermore, some rational matrix function $A(z) \in \mathcal{RL}^\infty(\mathbb{T})$ can be the symbol of many Laurent matrices, not all of them bounded (in fact, only one of them bounded).

Example 2.3 *Consider the symbol $A(z) = \frac{z}{z+2}$. This could have an unstable realization of $L_r\{1, -2, -2, 1, 0, 0, 0\}$, leading to an unbounded lower triangular Laurent matrix, or a stable realization of $L_r\{0, 0, 0, 0, \frac{1}{2}, -\frac{1}{2}, 1\}$ leading to a bounded upper triangular Laurent matrix. See how these matrices would look in (2.9)) or in the $log_{10}|\cdot|$ pictures of their sections below:*



The matrix on the left is using the first, unstable realization, while the right is the stable realization. If these finite sections were extended to be doubly infinite Laurent matrices, the left would be unbounded and the right would be bounded

An unbounded Laurent matrix would be difficult to physically implement, so it is essential that we only work with stable realizations. However, for high rational orders and large input and output dimensions, actually calculating a stable L_r realization of some transfer matrix $X(z)$ is computationally expensive (the so called 'realization problem', see e.g. [56]). In the following we will thus assume that we are given realizations of bounded L_r matrices (a fair assumption in distributed systems), and from then on we will only work with these stable state-space realizations and operations thereon. This motivates the following definition:

Definition 2.2 (Stable Realization) *Realizations of the form of* $\{P, R, Q, D, U, W, V\}$ *with stable multiplier terms (with* $\rho(R), \rho(W) < 1$*) will be called stable realizations. The space of such realizations will be denoted by* $\mathcal{S}$*. The norm and spectrum of an* $\mathcal{S}$ *realization will be defined as those of its* L_r *operator.*

With some abuse of notation, we will use $\bar{X}$ to refer both to the bounded L_r matrix and to a specific realization. Hence $\bar{X} = \mathcal{S}\{P, R, Q, D, U, W, V\}$ will denote both the L_r matrix $\bar{X}$ itself in equation (2.6), and the specific stable realization: $\{P, R, Q, D, U, W, V\} \in \mathcal{S}$, so:

$$\mathcal{S}\{P, R, Q, D, U, W, V\} = L_r\{P, R, Q, D, U, W, V\} \tag{2.13}$$

Note that L_r matrices with $\mathcal{S}$ realizations are clearly exponentially spatially decaying operators [57], and when square, generate C_0 semi-groups. We also note that this type of matrix is actually bounded on a larger set of signals:

Lemma 2.2 *All* L_r *operators in* $\mathcal{S}$ *are in the Wiener Algebra, and hence are bounded on* l_∞*.* ***Proof:*** $\mathcal{S}$ *matrices have stable causal and anticausal parts, with finite* H_∞ *norms. The* H_∞ *norm and the system order can be used to bound the row* l_1 *norm from above [58], thus guaranteeing inclusion in the Wiener Algebra.* □

In the next few subsections, we will basically develop a way to add, multiply, invert, transpose, and calculate the norm of these mixed causal transfer functions,

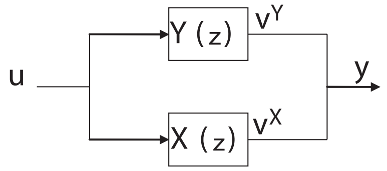


Figure 2.3: system representations of $\bar{X}$ and $\bar{Y}$ connected in parallel

using only their stable realizations. The result will be an easily programmable, fast, efficient structure preserving arithmetic of L_r matrices.

2.2.3 Addition of L_r Matrices with $\mathcal{S}$ Realizations

Before getting started with the actual derivations, let's put this in perspective. We're trying to develop a method of adding, multiplying, and inverting (block)Laurent matrices with rational symbols (that is, to develop an arithmetic). Since Laurent matrices are doubly infinite, we can't just start multiplying rows by columns as with finite matrices. Fortunately, as we saw in section 2.2.1, Laurent matrix arithmetic can equivalently be performed on the *symbols* of the Laurent matrices, which are just rational transfer matrices in our case, for which arithmetic is trivial. However, arithmetic of transfer matrices is slow, and it's crucial for rebuilding our Laurent matrices that we keep separate the stable and anti-stable components of the symbols (as we saw in example 2.3). One way to do this is to store the transfer functions as stable causal+anticausal realizations, as in Definition 2.2. One benefit of doing this is that we can use the mixed causal LTI systems interpretation of such realizations (equation (2.5)) and our own LTI systems background knowledge to derive simple formulas for this arithmetic, as we will now demonstrate.

To start with, adding two L_r matrices with input-output operator interpretations is like putting the system realizations of their symbols in parallel, hence we can use mixed causal systems theory to find a formula for $\mathcal{S}$-realization addition:

Lemma 2.3 *Given*

$$\bar{X} = \mathcal{S}\{P_X, R_X, Q_X, D_X, U_X, W_X, V_X\}$$
$$\bar{Y} = \mathcal{S}\{P_Y, R_Y, Q_Y, D_Y, U_Y, W_Y, V_Y\}$$

Then a realization of the sum: $\bar{Z} = \bar{X} + \bar{Y}$ *is:*

$$\bar{Z} = \mathcal{S}\{\begin{bmatrix} P_X^T \\ P_Y^T \end{bmatrix}^T, \begin{bmatrix} R_X & 0 \\ 0 & R_Y \end{bmatrix}, \begin{bmatrix} Q_X \\ Q_Y \end{bmatrix}, (D_X + D_Y), \begin{bmatrix} U_X & U_Y \end{bmatrix}, \begin{bmatrix} W_X & 0 \\ 0 & W_Y \end{bmatrix}, \begin{bmatrix} V_X \\ V_Y \end{bmatrix}\}$$

Proof: *We'll use the diagram in figure 2.3 with the two mixed causal system realizations:*

$$\begin{bmatrix} x_{i-1}^a \\ x_{i+1}^c \end{bmatrix} = \begin{bmatrix} W^X & 0 \\ 0 & R^X \end{bmatrix} \begin{bmatrix} x_i^a \\ x_i^c \end{bmatrix} + \begin{bmatrix} V^X \\ Q^X \end{bmatrix} u_i, \qquad v_i^X = \begin{bmatrix} U^X & P^X \end{bmatrix} \begin{bmatrix} x_i^a \\ x_i^c \end{bmatrix} + D^X u_i \quad (2.14)$$

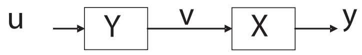


Figure 2.4: system representations of $\bar{X}$ and $\bar{Y}$ connected in series

$$\begin{bmatrix} z^a_{i-1} \\ z^c_{i+1} \end{bmatrix} = \begin{bmatrix} W^Y & 0 \\ 0 & R^Y \end{bmatrix} \begin{bmatrix} z^a_i \\ z^c_i \end{bmatrix} + \begin{bmatrix} V^Y \\ Q^Y \end{bmatrix} u_i, \qquad v^Y_i = \begin{bmatrix} U^Y & P^Y \end{bmatrix} \begin{bmatrix} z^a_i \\ z^c_i \end{bmatrix} + D^Y u_i \tag{2.15}$$

Now $y_i = v^X_i + v^Y_i$*, so:*

$$\begin{bmatrix} z^a_{i-1} \\ x^a_{i-1} \\ z^c_{i+1} \\ x^c_{i+1} \end{bmatrix} = \begin{bmatrix} W^Y & 0 & 0 & 0 \\ 0 & W^X & 0 & 0 \\ 0 & 0 & R^Y & 0 \\ 0 & 0 & 0 & R^X \end{bmatrix} \begin{bmatrix} z^a_i \\ x^a_i \\ z^c_i \\ x^c_i \end{bmatrix} + \begin{bmatrix} V^Y \\ V^X \\ Q^Y \\ Q^X \end{bmatrix} u_i$$

$$y_i = v^X_i + v^Y_i = \begin{bmatrix} U^Y & U^X & P^Y & P^X \end{bmatrix} \begin{bmatrix} z^a_i \\ x^a_i \\ z^c_i \\ x^c_i \end{bmatrix} + (D^Y + D^X) u_i \tag{2.16}$$

□

The result is also easily verifiable by inspection.

We note that $\bar{Z} \in \mathcal{S}$ since R_X, R_Y, W_X, W_Y are stable since $\bar{X}, \bar{Y} \in \mathcal{S}$, so not just the rational Laurent structure, but also the stability of the realization is preserved.

2.2.4 Multiplication of L_r Matrices with $\mathcal{S}$ Realizations

Likewise, multiplication of two Laurent matrices with input-output operator interpretations is like putting the system realizations of their symbols in series: (as in figure 2.4) By thinking of the operation in this way, we can use our systems knowledge to derive a formula for the resulting composite system, and thus the product of the two L_r matrices.

Lemma 2.4 *Given:*

$$\bar{X} = \mathcal{S}\{P_X, R_X, Q_X, D_X, U_X, W_X, V_X\} \tag{2.17}$$

$$\bar{Y} = \mathcal{S}\{P_Y, R_Y, Q_Y, D_Y, U_Y, W_Y, V_Y\} \tag{2.18}$$

$\bar{X}\bar{Y} = \bar{Z} = \mathcal{S}\{P_Z, R_Z, Q_Z, D_Z, U_Z, W_Z, V_Z\}$ with

$$\begin{aligned}
D_Z &= D_X D_Y + P_X S V_Y + U_X T Q_Y \\
P_Z &= \begin{bmatrix} D_X P_Y + U_X T R_Y & P_X \end{bmatrix}, \qquad U_Z = \begin{bmatrix} D_X U_Y + P_X S W_Y & U_X \end{bmatrix}, \\
R_Z &= \begin{bmatrix} R_Y & 0 \\ Q_X P_Y & R_X \end{bmatrix}, \quad W_Z = \begin{bmatrix} W_Y & 0 \\ V_X U_Y & W_X \end{bmatrix} \\
Q_Z &= \begin{bmatrix} Q_Y \\ Q_X D_Y + R_X S V_Y \end{bmatrix}, V_Z = \begin{bmatrix} V_Y \\ V_X D_Y + W_X T Q_Y \end{bmatrix}
\end{aligned}$$

where S and T are the unique solutions to the Stein equations:

$$S = R_X S W_Y + Q_X U_Y, \quad T = W_X T R_Y + V_X P_Y$$

Proof: *This will work the same as in the above proof of addition; by considering $\bar{X}$ and $\bar{Y}$ as input output operators for LTI systems:*

$$\begin{bmatrix} x^a_{i-1} \\ x^c_{i+1} \end{bmatrix} = \begin{bmatrix} W_X & 0 \\ 0 & R_X \end{bmatrix} \begin{bmatrix} x^a_i \\ x^c_i \end{bmatrix} + \begin{bmatrix} V_X \\ Q_X \end{bmatrix} v_i, \qquad y_i = \begin{bmatrix} U_X & P_X \end{bmatrix} \begin{bmatrix} x^a_i \\ x^c_i \end{bmatrix} + D_X v_i \tag{2.19}$$

$$\begin{bmatrix} z^a_{i-1} \\ z^c_{i+1} \end{bmatrix} = \begin{bmatrix} W_Y & 0 \\ 0 & R_Y \end{bmatrix} \begin{bmatrix} z^a_i \\ z^c_i \end{bmatrix} + \begin{bmatrix} V_Y \\ Q_Y \end{bmatrix} u_i, \qquad v_i = \begin{bmatrix} U_Y & P_Y \end{bmatrix} \begin{bmatrix} z^a_i \\ z^c_i \end{bmatrix} + D_Y u_i \tag{2.20}$$

connecting them in series, and calculating the resulting LTI system, hence $\bar{Z}$. Connecting $Y(z)$ and $X(z)$ together gives us:

$$\begin{bmatrix} x^a_{i-1} \\ z^a_{i-1} \\ x^c_{i+1} \\ z^c_{i+1} \end{bmatrix} = \begin{bmatrix} W_X & V_X U_Y & 0 & V_X P_Y \\ 0 & W_Y & 0 & 0 \\ 0 & Q_X U_Y & R_X & Q_X P_Y \\ 0 & 0 & 0 & R_Y \end{bmatrix} \begin{bmatrix} x^a_i \\ z^a_i \\ x^c_i \\ z^c_i \end{bmatrix} + \begin{bmatrix} V_X D_Y \\ V_Y \\ Q_X D_Y \\ Q_Y \end{bmatrix} u_i$$

$$y_i = \begin{bmatrix} U_X & D_X U_Y & P_X & D_X P_Y \end{bmatrix} \begin{bmatrix} x^a_i \\ z^a_i \\ x^c_i \\ z^c_i \end{bmatrix} + (D_X D_Y) u_i \tag{2.21}$$

Now this is fine, except there are coupling terms between the causal and anticausal parts that we need to get rid of. To do this, we first rewrite the system as:

$$\begin{bmatrix} -W_X & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & -R_X & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} x^a_i \\ z^a_{i-1} \\ x^c_i \\ z^c_{i+1} \end{bmatrix} = \begin{bmatrix} -I & V_X U_Y & 0 & V_X P_Y \\ 0 & W_Y & 0 & 0 \\ 0 & Q_X U_Y & -I & Q_X P_Y \\ 0 & 0 & 0 & R_Y \end{bmatrix} \begin{bmatrix} x^a_{i-1} \\ z^a_i \\ x^c_{i+1} \\ z^c_i \end{bmatrix} + \begin{bmatrix} V_X D_Y \\ V_Y \\ Q_X D_Y \\ Q_Y \end{bmatrix} u_i$$

now solve Stein equations:

$$S = R_X S W_Y + Q_X U_Y, \quad T = W_X T R_Y + V_X P_Y$$

and perform a state transformation:

$$\begin{bmatrix} x_{i-1}^a \\ z_i^a \\ x_{i+1}^c \\ z_i^c \end{bmatrix} = \begin{bmatrix} I & 0 & 0 & T \\ 0 & I & 0 & 0 \\ 0 & S & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}_{i-1}^a \\ z_i^a \\ \hat{x}_{i+1}^c \\ z_i^c \end{bmatrix} \tag{2.22}$$

But from the original equations, we know that, on the left side of the resulting equation, the terms: $-(W_X T + T)z_{i+1}^c = -(W_X T + T)R_Y z_i^c + -(W_X T + T)Q_Y u_i$ *and* $-(R_X S + S)z_{i-1}^a = -(R_X S + S)W_Y z_i^a + -(R_X S + S)V_Y u_i$*. Using these, and since* T *and* S *solve the Stein equations, we then have:*

$$\begin{bmatrix} -W^X & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & -R^X & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}_i^a \\ z_{i-1}^a \\ \hat{x}_i^c \\ z_{i+1}^c \end{bmatrix} = \begin{bmatrix} -I & V_X U_Y & 0 & 0 \\ 0 & W_Y & 0 & 0 \\ 0 & 0 & -I & Q_X P_Y \\ 0 & 0 & 0 & R_Y \end{bmatrix} \begin{bmatrix} \hat{x}_{i-1}^a \\ z_i^a \\ \hat{x}_{i+1}^c \\ z_i^c \end{bmatrix} + \begin{bmatrix} V_X D_Y + W_X T Q_Y \\ V_Y \\ Q_X D_Y + R_X S V_Y \\ Q_Y \end{bmatrix} u_i$$

or

$$\begin{bmatrix} \hat{x}_{i-1}^a \\ z_{i-1}^a \\ \hat{x}_{i+1}^c \\ z_{i+1}^c \end{bmatrix} = \begin{bmatrix} W_X & V_X U_Y & 0 & 0 \\ 0 & W_Y & 0 & 0 \\ 0 & 0 & R_X & Q_X P_Y \\ 0 & 0 & 0 & R_Y \end{bmatrix} \begin{bmatrix} \hat{x}_i^a \\ z_i^a \\ \hat{x}_i^c \\ z_i^c \end{bmatrix} + \begin{bmatrix} V_X D_Y + W_X T Q_Y \\ V_Y \\ Q_X D_Y + R_X S V_Y \\ Q_Y \end{bmatrix} u_i$$

But we also need to fix the output equation. From the state transformation, we see that:

$$\begin{bmatrix} x_i^a \\ x_i^c \end{bmatrix} = \begin{bmatrix} I & 0 & 0 & T \\ 0 & S & I & 0 \end{bmatrix} \begin{bmatrix} \hat{x}_i^a \\ z_{i-1}^a \\ \hat{x}_i^c \\ z_{i+1}^c \end{bmatrix} \tag{2.23}$$

but $z_{i-1}^a = W_Y z_i^a + V_Y u_i$ *and* $z_{i+1}^c = R_Y z_i^a + Q_Y u_i$*, so:*

$$x_i^a = \hat{x}_i^a + T R_Y z_i^c + T Q_Y u_i \tag{2.24}$$

$$x_i^c = \hat{x}_i^c + S W_Y z_i^a + S V_Y u_i \tag{2.25}$$

substituting these into the output equation, we then get:

$$\begin{aligned} y_i &= \begin{bmatrix} U_X, & D_X U_Y + P_X S W_Y, & P_X, & D_X P_Y + U_X T R_Y \end{bmatrix} \begin{bmatrix} \hat{x}_i^a \\ z_i^a \\ \hat{x}_i^c \\ z_i^c \end{bmatrix} + \\ & (D_X D_Y + U_X T Q_Y + P_X S V_Y)u_i \end{aligned}$$

which matches our above formulas for $Z(z)$ *and* $\bar{Z}$*.* □

These formulas are also easy to verify by inspection once we realize that $S = \sum_{k=0}^{\infty} R_X^k Q_X U_Y W_Y^k$ and $T = \sum_{k=0}^{\infty} W_X^k V_X P_Y R_Y^k$. As with the addition, note that the L_r operator structure, and also the stability of the realizations, is preserved under this multiplication: $\bar{X}, \bar{Y} \in \mathcal{S} \Rightarrow \bar{Z} \in \mathcal{S}$.

2.2.5 Order Reduction of L_r Matrices with $\mathcal{S}$ Realizations

The reader will have noticed that through the above two operations (L_r matrix addition and multiplication), $\bar{Z}$ will have a rational symbol with 'order' larger than $\bar{X}$ or $\bar{Y}$.

Fortunately, since we represent $\bar{Z}$ as the sum of a lower and upper stable L_r operators $\bar{Z} = \bar{L} + \bar{U}$ with stable LTI causal and anticausal interpretations, respectively, we can efficiently perform an order reduction on $\bar{Z}$ by performing standard LTI state space model order reductions on its upper and lower triangular parts separately. If we perform order reductions such that $\|L(z) - \tilde{L}(z)\|_\infty < e_L$, $\|U(z) - \tilde{U}(z)\|_\infty < e_U$, then the reduced order L_r operator $\tilde{\bar{Z}} = \tilde{\bar{L}} + \tilde{\bar{U}}$ has error bound $\|\bar{Z} - \tilde{\bar{Z}}\| < e_L + e_U$.

As for actually calculating such an order reduction, there are many possible methods, e.g. balanced truncation, Hankel optimal, etc. We will generally use balanced truncation since it can be simply extended to LTV systems and multiple dimensions (as we will see in Chapter 3). Balanced truncation was proposed in the early 1980's ([59]), and stability and the error bound were proven shortly thereafter (see [60] for an overview), and is now very well known, but we will state the result here for the reader's reference, as well as for comparison with our later use of similar LTV and multidimensional results.

Lemma 2.5 *([60]) Suppose you have a stable system* $L(s) = \left[\begin{array}{c|c} A & B \\ \hline C & 0 \end{array}\right]$ *with* $A \in \mathbb{R}^{n\times n}$*. Solve the Lyapunov equations:*

$$\begin{aligned} APA^* - P + BB^* &= 0 \\ A^*QA - Q + C^*C &= 0 \end{aligned}$$

$P \succeq 0$*,* $Q \succeq 0$*, so* $R = PQ$ *has a real and non-negative spectrum. Find a diagonalization* $V^{-1}RV = \Lambda$ *where the eigenvalues in* Λ *are ordered to be non-increasing down the diagonal:* $\lambda_i \geq \lambda_{i+1}$*. Perform a state transformation to get the balanced realization:*

$$L(s) = \left[\begin{array}{c|c} V^{-1}AV & V^{-1}B \\ \hline CV & 0 \end{array}\right] = \left[\begin{array}{cc|c} \hat{A}_{11} & \hat{A}_{12} & \hat{B}_1 \\ \hat{A}_{21} & \hat{A}_{22} & \hat{B}_2 \\ \hline \hat{C}_1 & \hat{C}_2 & 0 \end{array}\right] \tag{2.26}$$

where $\hat{A}_{11} \in \mathbb{R}^{q\times q}$*.*

Then the truncated qth order system $\hat{L}(s) = \left[\begin{array}{c|c} \hat{A}_{11} & \hat{B}_1 \\ \hline \hat{C}_1 & 0 \end{array}\right]$ *is guaranteed stable and the* H_∞ *error bound:* $\|\hat{L}(s) - L(s)\|_\infty \leq 2(\lambda_{q+1} + \lambda_{q+2} + \ldots\lambda_n)$ *holds.*

As we said before, using this on the $\left[\begin{array}{c|c} R & Q \\ \hline P & 0 \end{array}\right]$ and $\left[\begin{array}{c|c} W^T & U^T \\ \hline V^T & 0 \end{array}\right]$ parts of an $\mathcal{S}$ realization $\mathcal{S}(P, R, Q, D, U, W, V)$ yields a reduced order $\mathcal{S}$ realization with a guaranteed error bound.

2.2.6 Transpose and Permutations of L_r Matrices with $\mathcal{S}$ Realizations

Finding the transpose of L_r matrices is conceptually and computationally easy: looking at $\bar{X}$ in (2.6), we see that $\bar{X}^T = \mathcal{S}(V^T, W^T, U^T, D^T, Q^T, R^T, P^T)$.

We can also permute L_r operators together to form block L_r operators, and vice versa (in the same way that block Toeplitz matrices may be permuted to form Toeplitz block matrices), where the Laurent spatial orders will increase only additively; i.e. we can move between the two equivalent representations:

$$\begin{bmatrix} \bar{e} \\ \bar{f} \end{bmatrix} = \begin{bmatrix} \bar{W} & \bar{X} \\ \bar{Y} & \bar{Z} \end{bmatrix} \begin{bmatrix} \bar{g} \\ \bar{h} \end{bmatrix}, \quad \overline{\begin{bmatrix} e \\ f \end{bmatrix}} = \bar{P} \overline{\begin{bmatrix} g \\ h \end{bmatrix}} \tag{2.27}$$

These operations are very useful later on, because they enable us to avoid using 'block arithmetic' i.e. if we want to invert $\begin{bmatrix} \bar{W} & \bar{X} \\ \bar{Y} & \bar{Z} \end{bmatrix}$ we first permute it into $\bar{P}$, calculate a single inversion: $\bar{P}^{-1}$, then permute back to find $\begin{bmatrix} \bar{W} & \bar{X} \\ \bar{Y} & \bar{Z} \end{bmatrix}^{-1}$, instead of trying to do some type of block inversion using Schur complements and many additions, multiplications, and inversions.

Lemma 2.6 *Given Laurent operators $\overline{R}, \overline{X}, \overline{Y}, \overline{Z}$ with $\mathcal{S}$ realizations:*

$$\begin{aligned} \overline{R} &= \mathcal{S}(B^{Rm}, W^{Rm}, C^{Rm}, A^R, B^{Rp}, W^{Rp}, C^{Rp}) \\ \overline{X} &= \mathcal{S}(B^{Xm}, W^{Xm}, C^{Xm}, A^X, B^{Xp}, W^{Xp}, C^{Xp}) \\ \overline{Y} &= \mathcal{S}(B^{Ym}, W^{Ym}, C^{Ym}, A^Y, B^{Yp}, W^{Yp}, C^{Yp}) \\ \overline{Z} &= \mathcal{S}(B^{Zm}, W^{Zm}, C^{Zm}, A^Z, B^{Zp}, W^{Zp}, C^{Zp}) \end{aligned}$$

and lifted vectors $\overline{e}, \overline{f}, \overline{g}, \overline{h}$, the relations (2.27) are equivalent, to within a permutation, where $\overline{P} = \mathcal{S}(\hat{B}^m, \hat{W}^m, \hat{C}^m, \hat{A}, \hat{B}^p, \hat{W}^p, \hat{C}^p)$ with

$$\begin{aligned} \hat{A} &= \begin{bmatrix} A^R & A^X \\ A^Y & A^Z \end{bmatrix}, \quad \hat{C}^* = \begin{bmatrix} diag(C^{R*}, C^{X*}) \\ diag(C^{Y*}, C^{Z*}) \end{bmatrix} \\ \hat{B}^* &= diag(\begin{bmatrix} B^{R*} & B^{X*} \end{bmatrix}, \begin{bmatrix} B^{Y*} & B^{Z*} \end{bmatrix}) \\ \hat{W}^* &= diag(W^{R*}, W^{X*}, W^{Y*}, W^{Z*}) \end{aligned}$$

and the $*$'s are held constant as m or p in each term. ***Proof:*** *This is easily verified with permutation matrices such that*

$$\Pi_L \begin{bmatrix} \bar{R} & \bar{X} \\ \bar{Y} & \bar{Z} \end{bmatrix} \Pi_R = \bar{P}$$

with realizations:

$$\Pi_L = \left[\mathcal{S}(0,0,0,\begin{bmatrix} I \\ 0 \end{bmatrix},0,0,0), \quad \mathcal{S}(0,0,0,\begin{bmatrix} 0 \\ I \end{bmatrix},0,0,0)\right]$$

$$\Pi_R = \begin{bmatrix} \mathcal{S}(0,0,0,\begin{bmatrix} I & 0 \end{bmatrix},0,0,0) \\ \mathcal{S}(0,0,0,\begin{bmatrix} 0 & I \end{bmatrix},0,0,0) \end{bmatrix}$$

□

The transformation from $\begin{bmatrix} \vdots \\ e_{-2} \\ e_{-1} \\ e_0 \\ e_1 \\ e_2 \\ e_3 \\ \vdots \\ f_{-2} \\ f_{-1} \\ f_0 \\ f_1 \\ f_2 \\ f_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} \overline{e} \\ \overline{f} \end{bmatrix} \rightarrow \overline{\begin{bmatrix} e \\ f \end{bmatrix}} = \begin{bmatrix} \vdots \\ e_{-2} \\ f_{-2} \\ e_{-1} \\ f_{-1} \\ e_0 \\ f_0 \\ e_1 \\ f_1 \\ e_2 \\ f_2 \\ e_3 \\ f_3 \\ \vdots \end{bmatrix}$ could be called a 'shuffle' permutation, since $\overline{e}$ and $\overline{f}$ are shuffled like a deck of cards to get $\overline{\begin{bmatrix} e \\ f \end{bmatrix}}$. The reverse operation is also possible:

Lemma 2.7 *Given* $\overline{P} = \mathcal{S}(\begin{bmatrix} B^{1m} \\ B^{2m} \end{bmatrix}, W^m, \begin{bmatrix} C^{1m} & C^{2m} \end{bmatrix}, \begin{bmatrix} A^{11} & A^{12} \\ A^{21} & A^{22} \end{bmatrix}, \begin{bmatrix} B^{1p} \\ B^{2p} \end{bmatrix}, W^p, \begin{bmatrix} C^{1p} & C^{2p} \end{bmatrix})$*, the relations (2.27) are equivalent, to within a permutation, where:*

$$\begin{array}{rcl} \overline{R} & = & \mathcal{S}(B^{1m}, W^m, C^{1m}, A^{11}, B^{1p}, W^p, C^{1p}) \\ \overline{X} & = & \mathcal{S}(B^{1m}, W^m, C^{2m}, A^{12}, B^{1p}, W^p, C^{2p}) \\ \overline{Y} & = & \mathcal{S}(B^{2m}, W^m, C^{1m}, A^{21}, B^{2p}, W^p, C^{1p}) \\ \overline{Z} & = & \mathcal{S}(B^{2m}, W^m, C^{2m}, A^{22}, B^{2p}, \hat{W}^p, C^{2p}) \end{array}$$

and everything is assumed conformably partitioned. ***Proof:*** *This is easily verified by permutation matrices such that*

$$\Phi_L \bar{P} \Phi_R = \begin{bmatrix} \bar{R} & \bar{X} \\ \bar{Y} & \bar{Z} \end{bmatrix}$$

If all of the generators of the operators in Lemmas 2.6 and 2.7 are appropriately *sized, then* $\Pi_L = \Pi_R^T = \Phi_L^T = \Phi_R$. □

2.2.7 Inversion of L_r Matrices with $\mathcal{S}$ Realizations

So far, we have shown the closure of $\mathcal{S}$ under addition and multiplication and developed easy formulas for performing these operations, but for our iterative methods, we will also need inversion, which now follows. We won't try to use the LTI systems interpretation directly as before, but the proof is still simple, and only requires basic algebra:

Lemma 2.8 *Given a Laurent operator* $\overline{Y} = \mathcal{S}\{P, R, Q, D, U, W, V\}$, *if the non-symmetric discrete algebraic Riccati equation(DARE):*

$$S = RSW + \underbrace{(Q - RSV)}_{\Pi}\underbrace{(X - PSV)^{-1}}_{\Phi}\underbrace{(U - PSW)}_{\Psi} \tag{2.28}$$

has a stabilizing *solution, e.g., an* S *for which both*

$$W^F = W - V\Phi\Psi, \qquad R^F = R - \Pi\Phi P$$

are stable, then $\overline{Y}$ *has an inverse* $\bar{Y}^{-1} = \bar{F} \in \mathcal{S}$ *which has the form*

$$\bar{F} = \mathcal{S}\{P^F, R^F, Q^F, D^F, U^F, W^F, V^F\} \tag{2.29}$$

where $D^F = \Phi + U^F T Q^F$ *and*

$$\begin{aligned} U^F &= \Phi\Psi, & Q^F &= \Pi\Phi \\ V^F &= -V\Phi + W^F T Q^F, & P^F &= -\Phi P + U^F T R^F \end{aligned}$$

where T *is the (unique) solution to the Stein equation*

$$T = W^F T R^F + V\Phi P \tag{2.30}$$

Proof: *Let's first assume that* $\overline{Y}$ *may be factored into* $\overline{Y} = \bar{L}\bar{U}$, *which are upper and lower triangular, respectively:*

$$\overline{L} = \mathcal{S}\{C, A, B, D, 0, 0, 0\}, \qquad \overline{U} = \mathcal{S}\{0, 0, 0, E, F, G, H\}$$

and let's further assume that they respectively have bounded lower and upper triangular inverses; both D *and* E *are each invertible and both* $\rho(A - BD^{-1}C) < 1$, $\rho(G - HE^{-1}F) < 1$, *hence:*

$$\begin{aligned} \bar{L}^{-1} &= \mathcal{S}\{-D^{-1}C, A - BD^{-1}C, BD^{-1}, D^{-1}, 0, 0, 0\} \\ \bar{U}^{-1} &= \mathcal{S}\{0, 0, 0, E^{-1}, E^{-1}F, G - HE^{-1}F, -HE^{-1}\} \end{aligned}$$

Now obviously

$$\overline{Y}^{(-1)} = \bar{U}^{-1}\bar{L}^{-1} := \mathcal{S}\{P_Z, R_Z, Q_Z, X_Z, U_Z, W_Z, V_Z\} \tag{2.31}$$

where (using Lemma 2.4), we can calculate that

$$\begin{array}{rcll} Q_Z &=& BD^{-1}, & U_Z = E^{-1}F \\ R_Z &=& (A - Q_Z C), & W_Z = (G - HU_Z) \\ P_Z &=& -(DE)^{-1}C + U_Z T R_Z & V_Z = -H(DE)^{-1} + W_Z T Q_Z \\ X_Z &=& (DE)^{-1} + U_Z T Q_Z & \end{array} \tag{2.32}$$

where

$$T = W_Z T R_Z + H(DE)^{-1}C \tag{2.33}$$

and also we can calculate, again using Lemma 2.4, that

$$\overline{Y} = \bar{L}\bar{U} = \mathcal{S}\{C, A, BE + ASH, DE + CSH, DF + CSG, G, H\}$$

where

$$S = BF + ASG \tag{2.34}$$

but we know that this must match $\overline{Y} = \mathcal{S}\{P, R, Q, X, U, W, V\}$, giving us the following constraints on A, B, C, D, E, F, G, H:

$$\begin{array}{rcl} C &=& P, \quad A = R, \quad G = W, \quad H = V \\ DE &=& X - PSV, \quad B = QE^{-1} - RSVE^{-1}, \quad F = D^{-1}U - D^{-1}PSW \end{array} \tag{2.35}$$

If we substitute these constraints into (2.34), (2.33), (2.32) and perform a bit of algebra, then we get the Sylvester and Riccati equations in Lemma 2.8, which are only in terms of the original P, R, Q, X, U, W, V.

Now we first assumed that such a fortuitous $\bar{L}\bar{U}$ pair existed, but notice that if the Riccati equation (2.28) has a stabilizing *solution S; then S stabilizes both*

$$\begin{array}{rcl} R_Z &=& R - (Q - RSV)(X - PSV)^{-1}P = A - BD^{-1}C \\ W_Z &=& W - V(X - PSV)^{-1}(U - PSW) = G - HE^{-1}F \end{array}$$

and provides $X - PSV$ invertible from which we can chose some full rank factors $DE = X - PSV$, which, along with (2.35), give us all the appropriate A, B, C, D, E, F, G, H in $\bar{L}$ and $\bar{U}$, thus guaranteeing that they exist (by construction), completing the proof. ☐

Note that the statement only goes one way; the Riccati equation (2.28) *will not* have a stabilizing solution for every invertible $\bar{Y}$, however, in the Hermitian case, we can do better:

Lemma 2.9 *Given a Hermitian $\bar{X} = \mathcal{S}\{P, R, Q, Y, Q^*, R^*, P^*\}$, the Riccati equation*

$$S = RSR^* + \underbrace{(Q - RSP^*)}_{\Pi}\underbrace{(Y - PSP^*)^{-1}}_{\Phi}\Pi^* \tag{2.36}$$

will have a stabilizing solution, S, if and only if $\bar{X} \succ 0$, in which case $\bar{X}^{-1} = \bar{Z} = \mathcal{S}\{P_Z, R_Z, Q_Z, Y_Z, Q_Z^*, R_Z^*, P_Z^*\}$, where

$$P_Z = -\Phi P + (\Pi\Phi)^* T(R - \Pi\Phi P), \qquad R_Z = R - \Pi\Phi P$$
$$Q_Z = \Pi\Phi, \qquad Y_Z = \Phi + (\Pi\Phi)^* T(\Pi\Phi)$$

where T is the unique solution to the Stein equation:

$$T \;=\; P^*\Phi P + (R - \Pi\Phi P)^* T(R - \Pi\Phi P) \tag{2.37}$$

Proof: *The equivalence of the existence of a stabilizing S and $\bar{X} \succ 0$ is exactly the Riccati equation version of the discrete time Positive Real Lemma (see e.g. [61], Lemma 8.C.2). The existence of $\bar{Z} \in \mathcal{S}$ and its formula follow as a special case of Lemma 2.8.* □

Since Lemma 2.9 will always work for $\bar{X} \succ 0$, this also provides a method that will always work for non-Hermitian operators:

Lemma 2.10 *Assume $\bar{A} \in \mathcal{S}$. Then $\exists \bar{A}^{-1} \in \mathcal{S} \Leftrightarrow 0 \notin \lambda(\bar{A})$. Furthermore, we can calculate it using the formulas in Lemma 2.9.* ***Proof:*** *First assume that $0 \notin \lambda(\bar{A})$, then clearly $0 \prec \bar{A}\bar{A}^* \in \mathcal{S}$, and we can use Lemmas 2.4 and 2.9 to calculate $\bar{A}^{-1} = \bar{A}^*(\bar{A}\bar{A}^*)^{-1} \in \mathcal{S}$*
Now assume that $\exists \bar{A}^{-1} \in \mathcal{S}$, hence $\bar{A}^{-1}$ is bounded which always implies $0 \notin \lambda(\bar{A})$. □

We will call such $\bar{A} \in \mathcal{S}$ with $\bar{A}^{-1} \in \mathcal{S}$ 'regular'. Note that we have been assuming $\bar{A}$ square, but these results could easily be extended to nonsquare $\bar{A}$ using left and right inverses.

We also note that the rational order of the symbol of $\bar{A}^{-1}$, as calculated in Lemma 2.10, will be generally 3 times the rational order of $\bar{A}$.However, this is not the case for the method in Lemma 2.8, and hence we use it whenever possible in order to speed up computations.

Example 2.4 *As an example, we'll invert the following non-symmetric L_r operator:*

$$\bar{A} = \mathcal{S}(\begin{bmatrix} -0.7839 & -0.9566 \\ 1.5162 & 0.3232 \end{bmatrix}, \begin{bmatrix} 0.1075 & -0.0889 \\ 0.5697 & 0.4035 \end{bmatrix}, \begin{bmatrix} -0.3552 & 0.5337 \\ 0.1482 & -0.6705 \end{bmatrix},$$
$$\begin{bmatrix} 1.6969 & 0.7925 \\ 0.7260 & 0.6034 \end{bmatrix}, \begin{bmatrix} -0.4356 & 0.6894 \\ 0.9516 & -1.6815 \end{bmatrix}, \begin{bmatrix} 0.0178 & 0.0327 \\ 0.0053 & -0.0255 \end{bmatrix}, \begin{bmatrix} 0.1650 & -0.7986 \\ 0.4571 & 0.1393 \end{bmatrix})$$

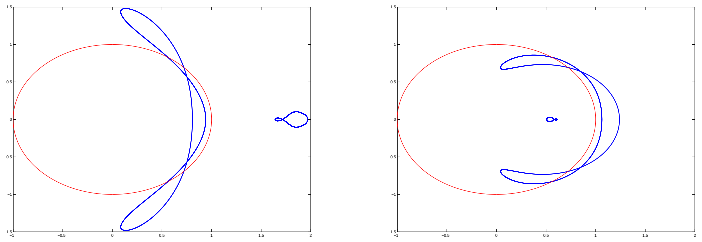
The nonsymmetric Riccati solution is:

$$S = \begin{bmatrix} 0.9323 & -1.6527 \\ -0.6405 & 1.1630 \end{bmatrix} \tag{2.38}$$

which stabilizes R^F *and* W^F*:* $\rho(R^F) = 0.2930$, $\rho(W^F) = 0.6863$. *Hence the Lyapunov equation has a unique solution:* $T = \begin{bmatrix} -0.8416 & -0.3807 \\ -0.1634 & -0.2149 \end{bmatrix}$, *leading to the generators:*

$$\bar{A}^{-1} = \mathcal{S}(\begin{bmatrix} 0.6535 & 0.5894 \\ -0.9272 & -0.3618 \end{bmatrix}, \begin{bmatrix} -1.4548 & -1.0115 \\ 1.6536 & 1.0907 \end{bmatrix}, \begin{bmatrix} -0.7469 & 0.6443 \\ 0.5779 & -0.4161 \end{bmatrix},$$
$$\begin{bmatrix} 0.6239 & -0.0779 \\ -0.2939 & 0.4205 \end{bmatrix}, \begin{bmatrix} -0.7095 & 1.2267 \\ 1.0764 & -1.9402 \end{bmatrix}, \begin{bmatrix} 0.9945 & -1.7192 \\ 0.1796 & -0.3159 \end{bmatrix}, \begin{bmatrix} -0.3305 & 0.3413 \\ -0.2420 & -0.0246 \end{bmatrix})$$

where everything has been rounded off at 4 decimal points. In the figure below we have the spectrum of $\bar{A}$ *on the left and that of* $\bar{A}^{-1}$ *on the right(with the unit circle for reference):*



Note how each point on the spectrum of $\bar{A}$ *is inverted to get the spectrum of the inverse of* $\bar{A}$. *This makes sense from (2.10) and (2.8).*

It's interesting to notice that Lemma 2.8 makes almost no assumptions on $\bar{A}$: D need not be square, and R and W can be different sizes even, leading to a nonsquare Riccati equation!

However, in the next example we'll emphasize that this inversion problem is more subtle than it might seem, and the nonsymmetric Riccati equation doesn't always have the answer:

Example 2.5 *Consider* $\bar{A} = \mathcal{S}(0, 0, 0, 0, 1/2, -1/2, 1)$ *from example 2.3. If we were to try to use our finite dimensional intuition on* $\bar{A}$, *we would conclude that since it is strictly upper triangular, with a zero diagonal, it shouldn't be invertible at all. This is false though; the spectrum is in figure 2.5. Since* $0 \notin \lambda(\bar{A})$, *we can invert it according to Lemma 2.10. However, the nonsymmetric Riccati equation in Lemma 2.8 doesn't work, so we use the trick in Lemma 2.10 to turn it into a symmetric problem. The symmetric Riccati equation does have a solution (as it must), and yields:* $\bar{A}^{-1} = \mathcal{S}(-1, 0, -2, 1, 0, 0, 0)$. *So not only is the strictly upper triangular Laurent matrix invertible, but its inverse is lower triangular (this could never happen with finite dimensional matrices)! This example shows that these infinite dimensional Laurent matrices can indeed be tricky, and we should be careful.*

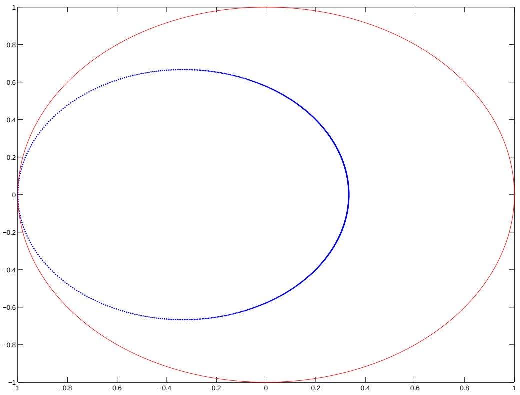


Figure 2.5: spectrum of $\bar{A}$ in Example 2.5

2.2.8 l_2 Induced Norm of L_r Matrices with $\mathcal{S}$ Realizations

Later, in Chapter 3, we'll use the arithmetic we just derived in iterative algorithms, for which we'll need some measure of convergence. We'll use the induced l_2 norm. Unlike in example 2.2, where $\bar{A}$ was lower triangular, for general $\bar{A} \in \mathcal{S}$, $\|\bar{A}\| = \|A(z)\|_\infty$ is not trivial to calculate, due to the presence of both 'causal' and 'anticausal' components. First we need an outer factorization:

Lemma 2.11 (Outer Factorization) *Assume we have a Hermitian* $\bar{X} = \mathcal{S}\{P, R, Q, Y, Q^*, R^*, P^*\}$, $\bar{X} \succ 0$. *Then* $\exists$

$$\begin{aligned}\bar{L} &= \mathcal{S}\{P_L, R_L, Q_L, D_L, 0, 0, 0\}\\ \bar{L}^{-1} &= \mathcal{S}\{(D_L^{-1}P_L), (R_L - Q_L D_L^{-1} P_L), (-Q_L D_L^{-1}), D_L^{-1}, 0, 0, 0\}\end{aligned}$$

with D_L *invertible such that* $\bar{L}\bar{L}^* = \bar{X}$. *Furthermore, such an* $\bar{L}$ *can be calculated as:*

$$P_L = P, \quad R_L = R, \quad Q_L = (Q - RSP^*)D_L^{-*}, \quad D_L D_L^* = Y - PSP^*$$

Where $S \succeq 0$ *is the stabilizing solution to the Riccati equation(2.36) such that* $Y - PSP^* \succ 0$, *and* D_L *can thus be calculated via Cholesky factorization.* ***Proof:*** *Follows from the proofs of Lemmas 2.8 and 2.9* □

If we have some non-Hermitian $\bar{A} \in \mathcal{S}$, (maybe singular), and we want to find the norm $\|\bar{A}\|$, then we can equivalently calculate: $\|\bar{A}\| = \sqrt{\|\bar{L}\bar{L}^*\| - 1}$, where $\bar{L}\bar{L}^* = \bar{A}^*\bar{A} + I \succ 0$ is an outer factorization, and hence $\bar{L}$ has a stable lower(causal) representation, $L(z)$. Since $\|\bar{L}\bar{L}^*\| = \|\bar{L}\|^2 = \|L(z)\|_\infty^2$ we can then use the Bounded Real Lemma for stable discrete LTI systems [61] to find $\|L(z)\|_\infty$ and thus $\|\bar{A}\|$.

2.2.9 Calculating the Path Integral of the Symbol of an $\mathcal{S}$ Realization Over the Unit Circle

We have one more calculation to discuss: the integral of the symbol of an L_r matrix over the unit circle, which has the following very nice result:

Lemma 2.12 *Assume that* $\bar{X} \in \mathcal{S}$ *has symbol:* $X(z) = P(zI - R)^{-1}Q + D + U(z^*I - W)^{-1}V$ *over* $z \in \mathbb{T}$*. Then*

$$\frac{1}{2\pi}\int_0^{2\pi} Tr[X(e^{j\theta})]d\theta = Tr(D) \tag{2.39}$$

Proof: *We can rewrite the integral, pulling out the constant and using a change of variables(*$z = e^{i\theta}$*;* $dz = ie^{i\theta}d\theta$*) as:*

$$\frac{1}{2\pi}\int_0^{2\pi} Tr[D]d\theta + \frac{1}{2\pi i}\oint_{\mathbb{T}} \frac{Tr[P(zI-R)^{-1}Q + U(z^*I-W)^{-1}V]}{z}dz \tag{2.40}$$

or, trivially solving the first integral and splitting up the remaining integral to make it easier to deal with:

$$= Tr[D] + Tr[\frac{1}{2\pi i}\oint_{\mathbb{T}} \frac{P(zI-R)^{-1}Q}{z}dz] + Tr[\frac{1}{2\pi i}\oint_{\mathbb{T}} \frac{U(z^*I-W)^{-1}V}{z}dz] \tag{2.41}$$

Since we are evaluating z *on the unit circle,* $z^* = z^{-1}$*, and making a change of variables in the second integral to* $z = \frac{1}{\zeta}$ *and* $dz = \frac{-1}{\zeta^2}d\zeta$ *where* ζ *is also on the unit circle, but going clockwise, hence the negative sign cancels out:*

$$= Tr[D] + Tr[\frac{1}{2\pi i}\oint_{\mathbb{T}} \frac{P(zI-R)^{-1}Q}{z}dz] + Tr[\frac{1}{2\pi i}\oint_{\mathbb{T}} \frac{V^*(zI-W^*)^{-1}U^*}{z}dz] \tag{2.42}$$

We will first consider the first integral, then generalize to the second. Since $\mathbb{T}$ *is a Jordan curve, we can consider it to be counter-clockwise inclosing all of the poles in* R*, or alternatively clockwise encircling the rest of the extended complex plane, including* ∞*, where it might have a residue. Thus according to the residue theorem [62]:*

$$\frac{1}{\pi i}\oint_{\mathbb{T}} \frac{P(zI-R)^{-1}Q}{z}dz = -2Res[\frac{P(zI-R)^{-1}Q}{z}, \infty] \tag{2.43}$$

To calculate the residue at ∞*, we write out the Laurent series expansion of the integrand around the neighborhood of* ∞*. We have:*

$$\frac{P(zI-R)^{-1}Q}{z} = P(\frac{I}{z^2} + \frac{R}{z^3} + \frac{R^2}{z^4} + \frac{R^3}{z^5} + ...)Q \tag{2.44}$$

Where the series expansion is valid because z *is large(in the neighborhood of infinity) and* R *is stable. So the coefficient corresponding to* z^{-1} *is* 0 *as is the residue and hence the integral. This same idea, since* W *is stable, applies to the second integral,* *which is thus also zero, completing the proof.* □

The utility of this formula isn't immediately obvious, but it will be essential for calculating the spatiotemporal H_2 norm in chapter 5.

2.2.10 Summary, L_r Matrices with $\mathcal{S}$ Realizations

In this section we've studied bounded Laurent operators with rational symbols and gained some intuition about them. We've seen how they can be thought of as input output operators of mixed causal LTI systems, and learned about their strange (compared to finite dimensional matrices) spectrums. We've also used the LTI system idea to derive $+$, $\times$, and $^{-1}$ operations for them, based only on their stable realizations, which can be performed very fast (compared to treating them as matrices of transfer functions) using Matlab. We saw how under these calculations, the 'order' of the operators (the rational orders of their symbols) increase, but that we can use simple LTI model-order reductions to find close approximations using lower orders. We also developed fast and easy formulas for the transpose, the shuffle permutation, and a way to calculate the l_2 induced norm. The following sections will have similar results for other matrix structures, so we showed the detailed derivations of all of the formulas in this chapter to save on space (readers can derive the later formulas by themselves).

Next, we will study Sequentially Semi-Separable matrices, which are the finite time LTV analog of the infinite time LTI systems used to generate the L_r matrices. This is where the results get interesting, because SSS matrices can be used to represent more realistic systems!

2.3 Discrete LTV Systems and SSS matrices

In the last section we studied and developed a structure preserving arithmetic for Laurent matrices with rational symbols, which can be thought of as the input-output operators for mixed causal LTI systems over all time $t \in \mathbb{Z}$. In this section, we will do the same thing, but for the input-output matrices of mixed causal linear time varying(LTV) systems over a finite time interval. This type of structured matrix will be called 'Sequentially Semi-Separable'(SSS).

Our LTV system over finite time steps will take the form:

$$\begin{bmatrix} x^a_{i-1} \\ x^c_{i+1} \end{bmatrix} = \begin{bmatrix} W_i & 0 \\ 0 & R_i \end{bmatrix} \begin{bmatrix} x^a_i \\ x^c_i \end{bmatrix} + \begin{bmatrix} V_i \\ Q_i \end{bmatrix} u_i, \qquad y_i = \begin{bmatrix} U_i & P_i \end{bmatrix} \begin{bmatrix} x^a_i \\ x^c_i \end{bmatrix} + D_i u_i \tag{2.45}$$

for $i \in \{1, ..., N\}$. We are trying to use very similar form and notation to the previous section, in order to indicate the parallels between the two structures. As before, the state variables x^a_i, x^c_i are extraneous, and we can cancel them out to

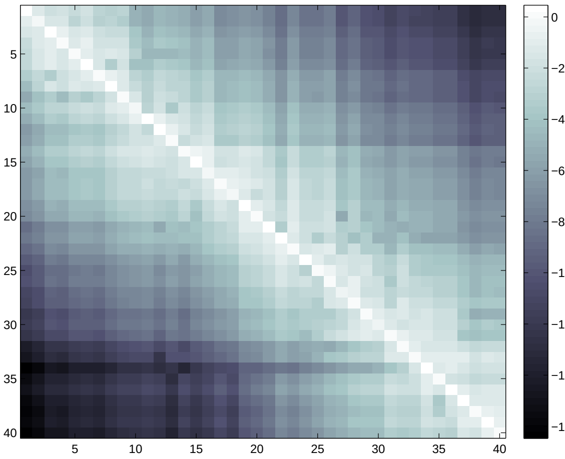


Figure 2.6: example of an SSS matrix ($\log_{10} |\cdot|$)

write down the input output mapping over time $i \in \{1, ..., N\}$:

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_N \end{bmatrix}}_{\bar{y}} = \underbrace{\begin{bmatrix} D_1 & U_1 V_2 & U_1 W_2 V_3 & U_1 W_2 W_3 V_4 & \ddots & \vdots \\ P_2 Q_1 & D_2 & U_2 V_3 & U_2 W_3 V_4 & \ddots & \vdots \\ P_3 R_2 Q_1 & P_3 Q_2 & D_3 & U_3 V_4 & \ddots & \vdots \\ P_4 R_3 R_2 Q_1 & P_4 R_3 Q_2 & P_4 Q_3 & D_4 \ddots & \vdots & \\ \ddots & \ddots & \ddots & \ddots & \ddots & U_{N-1} V_N \\ \cdots & \cdots & \cdots & \cdots & P_N Q_{N-1} & D_N \end{bmatrix}}_{\bar{A}} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \vdots \\ u_N \end{bmatrix}}_{\bar{u}} \qquad (2.46)$$

$\bar{A}$ is an SSS matrix, which we will abbreviate using the notation $\bar{A} = \mathcal{SSS}(P_s, R_s, Q_s, D_s, U_s, W_s, V_s)$ from now on. To compare to the Laurent matrices in the last section, see figure 2.6, an example of an SSS matrix with $N = 40$. Notice how it's finite, and not Toeplitz, but the exponential decay away from the diagonal is similar to that of the section of the L_r matrix in figure 2.1.

Sequentially Semi-Separable matrices (also called 'quasi-separable' and 'matrices of low Hankel rank') were first discussed in [47] and [63], but their roots go back through semi-separable matrices to semi-separable integral kernels in the 1960's in Kailath's paper [64], and much of the SSS arithmetic was worked out as far back as [65]. The interest of [47] was in applications to LTV systems theory, but since then SSS matrices have been studied with respect to scattering theory [48], super-fast Toeplitz solvers[66], Szego polynomials, [67][68] and for their own sake as interesting structured matrices [69][70][71][72][73][74][75][76][77][78]. The result of all of this research is essentially a 'superfast', linear computational complexity ($\mathcal{O}(N)$) arithmetic of SSS matrices, including +, ×, inverse, transpose, and Cholesky and QR factorizations. In this thesis, we will mostly just *use* this arithmetic, although we will also state the relevent formulas in the next few subsections to show their similarity to those for rationally symboled L_r matrices, and the methods used to derive the L_r matrix arithmetic in the last section can be used identically to derive the SSS arithmetic in this section. Notice that these formulas are *extremely* similar to those in the previous section, except in the points where one might expect them to be different: e.g. where for the inversion of L_r operators a Riccati equation is solved, for the inversion of SSS matrices a Riccati recursion is iterated from 1 to N.

2.3.1 SSS Addition

To add together two conformably sized SSS matrices, $\overline{X} = \mathcal{SSS}(P_s^X, R_s^X, Q_s^X, D_s^X, U_s^X, W_s^X, V_s^X)$ and $\overline{Y} = \mathcal{SSS}(P_s^Y, R_s^Y, Q_s^Y, D_s^Y, U_s^Y, W_s^Y, V_s^Y)$ and obtain the SSS generator form of their sum: $\overline{X} + \overline{Y} = \overline{Z} = \mathcal{SSS}(P_s^Z, R_s^Z, Q_s^Z, D_s^Z, U_s^Z, W_s^Z, V_s^Z)$, simply perform the same computation as in Lemma 2.3:

$$P_s^Z = \begin{bmatrix} P_s^X & P_s^Y \end{bmatrix}, \quad D_s^Z = D_s^X + D_s^Y, U_s^Z = \begin{bmatrix} U_s^X & U_s^Y \end{bmatrix}$$

$$V_s^Z = \begin{bmatrix} V_s^X \\ V_s^Y \end{bmatrix}, \qquad Q_s^Z = \begin{bmatrix} Q_s^X \\ Q_s^Y \end{bmatrix}, \qquad ,W_s^Z = \begin{bmatrix} W_s^X & 0 \\ 0 & W_s^Y \end{bmatrix}, R_s^Z = \begin{bmatrix} R_s^X & 0 \\ 0 & R_s^Y \end{bmatrix}$$

for each $0 < s < N$. This formula can also be found in [48], and can be derived using the methods in section 2.2 by treating the SSS matrices as the input output matrices of two LTV systems put in parallel.

2.3.2 SSS Multiplication

To multiply together two conformably sized SSS matrices, $\overline{X} = \mathcal{SSS}(P_s^X, R_s^X, Q_s^X, D_s^X, U_s^X, W_s^X, V_s^X)$ and $\overline{Y} = \mathcal{SSS}(P_s^Y, R_s^Y, Q_s^Y, D_s^Y, U_s^Y, W_s^Y, V_s^Y)$ and obtain the SSS generator form of their product: $\overline{XY} = \overline{Z} = \mathcal{SSS}(P_s^Z, R_s^Z, Q_s^Z, D_s^Z, U_s^Z, W_s^Z, V_s^Z)$, there are essentially two calculations iterated, one forwards, and one backwards. We will call them $\mathfrak{M}_1$, and $\mathfrak{M}_2$, and henceforth these symbols will denote the following calculations:

$$\mathfrak{M}_1 : \left(\begin{array}{c} R_s^Z = \begin{bmatrix} R_s^Y & 0 \\ Q_s^X P_s^Y & R_s^X \end{bmatrix} \\ W_s^Z = \begin{bmatrix} W_s^Y & 0 \\ V_s^X U_s^Y & W_s^X \end{bmatrix} \\ Q_s^Z = \begin{bmatrix} Q_s^Y \\ Q_s^X D_s^Y + R_s^X S_s V_s^Y \end{bmatrix} \\ U_s^Z = \begin{bmatrix} D_s^X U_s^Y + P_s^X S_s W_s^Y & U_s^X \end{bmatrix} \\ S_{s+1} = Q_s^X U_s^Y + R_s^X S_s W_s^Y \\ s = s + 1 \end{array} \right.$$

and

$$\mathfrak{M}_2 : \left(\begin{array}{c} P_s^Z = \begin{bmatrix} D_s^X P_s^Y + U_s^X T_s R_s^Y & P_s^X \end{bmatrix} \\ D_s^Z = D_s^X D_s^Y + P_s^X S_s V_s^Y + U_s^X T_s Q_s^Y \\ V_s^Z = \begin{bmatrix} V_s^Y \\ V_s^X D_s^Y + W_s^X T_s Q_s^Y \end{bmatrix} \\ T_{s-1} = V_s^X P_s^Y + W_s^X T_s R_s^Y \\ s = s-1 \end{array} \right.$$

To perform the SSS multiplication, we just calculate $\mathfrak{M}_1$ for $s = 1 : N$ starting with $S_0 = 0$, and then $\mathfrak{M}_2$ for $s = N : 1$ starting with $T_N = 0$, leading to $\bar{Z}$ with $\mathcal{O}(N)$ computational complexity. Notice the similarity to Lemma 2.4: here we just iterate Lyapunov recursions through each s, instead of solving steady state equations. A similar set of formulas can be found in [48]. This can also be derived using the methods in section 2.2, by treating the SSS matrices as the input output matrices of two LTV systems put in series.

2.3.3 SSS Inversion

We will next find a formula to calculate the generators of the inverse of an SSS matrix $\overline{X} = \mathcal{SSS}(P_s, R_s, Q_s, D_s, U_s, W_s, V_s)$. As with the SSS multiplication, there are two essential iterations performed in this calculation:

$$\mathfrak{I}_1 : \left(\begin{array}{c} S_{s+1} = R_s S_s W_s + \underbrace{(Q_s - R_s S_s V_s)}_{\Pi_s} \underbrace{(D_s - P_s S_s V_s)^{-1}}_{\Phi_s} \underbrace{(U_s - P_s S_s W_s)}_{\Psi_s} \\ U_s^F = \Phi_s \Psi_s, \qquad Q_s^F = \Pi_s \Phi_s \\ W_s^F = W_s - V_s \Phi_s \Psi_s, \qquad R_s^F = R_s - \Pi_s \Phi_s P_s \\ s = s+1 \end{array} \right.$$

$$\mathfrak{I}_2 : \left(\begin{array}{c} U_s^F = \Phi_s \Psi_s, \qquad Q_s^F = \Pi_s \Phi_s \\ V_s^F = -V_s \Phi_s + W_s^F T_s Q_s^F, \qquad P_s^F = -\Phi_s P_s + U_s^F T_s R_s^F \\ D_s^F = \Phi_s + U_s^F T_s Q_s^F \\ T_{s-1} = W_s^F T_s R_s^F + V_s \Phi_s P_s \\ s = s-1 \end{array} \right.$$

Starting with $S_1 = 0$, iterate $\mathfrak{I}_1$ for $s = 1 : N$, then starting with $s = N$ and $T_s = 0$, iterate $\mathfrak{I}_2$ for $s = N : 1$, at which point the SSS inversion is finished, resulting in $\bar{X}^{-1} = \bar{F} = \mathcal{SSS}(P_s^F, R_s^F, Q_s^F, D_s^F, U_s^F, W_s^F, V_s^F)$.

Our formula above is believed to be similar to that in [63], and can also be derived using the methods in section 2.2, by first assuming an LU factorization, then solving for the appropriate generators. Note the similarity to the $\mathcal{S}$ realization inversion formula in Lemma 2.8: here we iterate a Riccati recursion from $s = 1 : N$ whereas for the $\mathcal{S}$ realizations we solve a steady state Riccati equation. We should also note that there are many other methods of performing $\mathcal{O}(N)$ inversions of SSS matrices, based on ULV and QR factorizations etc. [47][48][71], which may be numerically better, and even this method might be improved in many ways, e.g.

by using a square root implementation of the Riccati recursion [79], but we use this version for its clear and easy derivation, and connection to our other arithmetics.

We can also prove something interesting about the iteration:

Lemma 2.13 $\overline{X} = \mathcal{SSS}(C_s, A_s, B_s, D_s, B_s^T, A_s^T, C_s^T) \succ 0$ *implies that* $D_s \succ 0$, *and* $S_s \succeq 0$ *in* $\mathfrak{I}_1$. ***Proof:*** *The fact that* $\overline{X} \succ 0$ *implies that all of its principal minors are also positive definite, hence* $D_s \succ 0$. *Also, the first leading principal minor implies* $D_0 \succ 0$, *and the second, using the Schur complement, implies* $D_1 - C_1 B_0 D_0^{-1} B_0^T C_1^T \succ 0$, *etc. In fact, these positive definite matrices are exactly the* $(D_s - C_s S_s C_s^T)$ *terms occurring in the iteration on* S_s. *Since* $S_0 = 0 \succeq 0$, *and each* S_{s+1} *is a sum of semi-positive quadratic products of* S_s *and* $(D_s - C_s S_s C_s^T)^{-1} \succ 0$, *hence by induction all* $S_s \succeq 0$. □

Among other things, this lemma proves that for positive definite symmetric $\overline{X}$, this algorithm will *always* work, hence providing a method that can invert any invertible $\bar{A}$, in the same way as for L_r matrices, using the pseudoinverse formulas: either $\bar{A}^T(\bar{A}\bar{A}^T)^{-1}$ or $(\bar{A}^T\bar{A})^{-1}\bar{A}^T$.

2.3.4 SSS Order Reduction

As with the L_r model order reduction, we can split up the SSS matrix into its lower and upper triangular parts, and reduce each separately. This is by no means optimal with respect to the matrix norm, but it works pretty well in practice.

Given some SSS matrix $\overline{X} = \mathcal{SSS}(C_s, A_s, B_s, D_s, E_s, F_s, G_s)$ where the sizes $A_s, F_s \in \mathbb{R}^{n_x \times n_x}$ are larger than is practical for fast computations, one might like to find a close SSS approximation $\overline{X} \approx \overline{Z} = \mathcal{SSS}(P_s, R_s, Q_s, D_s, U_s, W_s, V_s)$ of lower 'SSS order': $R_s, W_s \in \mathbb{R}^{n_z \times n_z}$ where $n_z < n_x$.For brevity, we will only show the 'lower triangular' version; the algorithm for the C_s, A_s, B_s part of the matrix, since D_s is not affected, and the 'upper triangular' E_s, F_s, G_s is just a transposed version of the lower triangular and can thus be easily derived.

As with the SSS multiplication and inversion, there are two essential iterations performed in this calculation:

$$\mathfrak{R}_1 : \left(\begin{array}{c} S_{s+1} = A_s S_s A_s^T + B_s B_s^T \\ s = s+1 \end{array} \right.$$

$$\mathfrak{R}_2 : \left(\begin{array}{c} T_s = A_s^T T_{s+1} A_s + C_s^T C_s \\ V_s \Lambda_s V_s^{-1} = S_s T_s \\ \hat{A}_s = V_{s+1}^{-1} A_s V_s \\ \hat{C}_s = C_s V_s \\ \hat{B}_s = V_{s+1}^{-1} B_s \\ R_s = \hat{A}_s(1:n_z, 1:n_z) \\ P_s = \hat{C}_s(:, 1:n_z) \\ Q_s = \hat{B}_s(1:n_z, :) \\ s = s-1 \end{array} \right.$$

where $V_s\Lambda_s V_s^{-1}$ in the second line of $\mathfrak{R}_2$ is an ordered eigenvalue decomposition $\Lambda = diag(\lambda_1, \lambda_2, \lambda_3...)$ with $\lambda_j \geq \lambda_{j+1}$, and we have used MATLAB notation in lines 6 through 8 of $\mathfrak{R}_2$ to indicate the selected rows and columns of the matrices. The iteration proceeds as follows: Starting with $s = 1$ and $S_s = 0$, iterate $\mathfrak{R}_1$ for $s = 1 : N$. Then starting with $T_N = 0$ and $s = N$, iterate $\mathfrak{R}_2$ for $s = N : 1$.

If we compare this with Lemma 2.5 in section 2.2 for L_r matrices, we see that this procedure is equivalent to a balanced truncation method of model order reduction performed on an LTV system ($\left[\begin{array}{c|c} A_s & B_s \\ \hline C_s & 0 \end{array}\right]$) over time steps $s = 1 : N$, for which error bounds are also available, see [80]. Other methods for SSS order reduction are based on SVD's [48] or optimally minimizing the error in the Hankel norm[47].

2.3.5 SSS Transpose and Permutation

Calculating the transpose of SSS matrices is easy, just like for $\mathcal{S}$ realizations, except the computation must be performed at each $s \in [1, 2, 3...N]$:

$$\bar{X}^T = \mathcal{SSS}(V_s^T, W_s^T, U_s^T, D_s^T, Q_s^T, R_s^T, P_s^T) \tag{2.47}$$

Likewise, the shuffle permutation of SSS matrices just consists of using the earlier formulas in Lemmas 2.6 and 2.7, once at each $s \in [1, 2, 3...N]$.

2.3.6 Calculating Norms of SSS Matrices

For L_r matrices, we used an outer factorization and the Bounded Real Lemma to efficiently calculate the induced l_2 norm, and something similar is possible for the matrix 2-norm, or 'spectral norm' of SSS matrices. It is clear that $\|\bar{A}\| < \gamma$ if and only if $\bar{A}\bar{A}^T - \gamma^2 I \succ 0$, and we can check if $\bar{A}\bar{A}^T - \gamma^2 I$ is positive definite in the following way:

Lemma 2.14 *$\bar{X} = \mathcal{SSS}(P_s, R_s, Q_s, Y_s, Q_s^T, R_s^T, P_s^T)$ is positive definite if and only if, when performing the Riccati recursion:*

$$S_{s+1} = R_s S_s R_s^T + (Q_s - R_s S_s P_s^T)(Y_s - P_s S_s P_s^T)^{-1}(Q_s - R_s S_s P_s^T)^T \tag{2.48}$$

(in $\mathfrak{I}_1$), $(Y_s - P_s S_s P_s^T)^{-1} \succ 0$, $\forall s \in [0, N]$. ***Proof:*** *Necessity comes from the proof of Lemma 2.13. Sufficiency goes as follows: If $(Y_s - P_s S_s P_s^T)^{-1} \succ 0$, then we can calculate a Cholesky factorization of $\bar{X} = \bar{L}\bar{L}^T$:*

$$\bar{L} = \mathcal{SSS}(C_s, A_s, B_s, D_s, 0, 0, 0) \tag{2.49}$$

where

$$C_s = P_s, \qquad A_s = R_s, \qquad B_s = (Q_s - R_s S_s P_s^T) \tag{2.50}$$

and $Y_s - P_s S_s P_s^T = D_s D_s^T$ *is a Cholesky factorization. The existence of the Cholesky factors* $\bar{L}$ *prove that* $\bar{X} \succ 0$. $\square$

Of course, the above norm calculation involves a bisection on γ (so does that in the Laurent operator case, once you get to the Bounded Real Lemma), where each step is $\mathcal{O}(N)$, but we can actually calculate the Frobenius norm much faster: $\|\bar{A}\|_F^2 = Trace(\bar{A}\bar{A}^T)$ with just an SSS transpose, multiplication, and trace, each of which is $\mathcal{O}(N)$, without any bisection iterations. Unless we specifically need the 2-norm (as in H_∞ synthesis in Chapter 3), we will often use the Frobenius norm for this reason.

2.3.7 Partial Derivatives of an SSS Matrix-Vector Product

We have one more calculation to derive for SSS matrices in this section, which isn't much related to the previous or following calculations, but will be used only later on in Chapter 4: a partial derivative of the product $\bar{A}\bar{x}$ with respect to each of the generators of $\bar{A}$. Given

$$\begin{aligned}\bar{A} &= \mathcal{SSS}(P_k, R_k, Q_k, D_k, U_k, W_k, V_k) \\ &= \begin{bmatrix} D_1 & U_1V_2 & U_1W_2V_3 & U_1W_2W_3V_4 & \ddots & \vdots \\ P_2Q_1 & D_2 & U_2V_3 & U_2W_3V_4 & \ddots & \vdots \\ P_3R_2Q_1 & P_3Q_2 & D_3 & U_3V_4 & \ddots & \vdots \\ P_4R_3R_2Q_1 & P_4R_3Q_2 & P_4Q_3 & D_4 \ddots & \vdots & \\ \ddots & \ddots & \ddots & \ddots & \ddots & U_{N-1}V_N \\ \dots & \dots & \dots & \dots & P_NQ_{N-1} & D_N \end{bmatrix}\end{aligned}$$

and $\overline{x} = \begin{bmatrix} x_1^T & x_2^T & \dots & x_N^T \end{bmatrix}^T$, where we will assume that the generators of the SSS matrix are constantly sized for $s = 1 : N$ (although this doesn't have to be the case) and that the upper triangular U_s, W_s, V_s side has the same order, n_w, as the lower triangular P_s, R_s, Q_s side: $U_s, P_s \in \mathbb{R}^{n_r \times n_w}$, $R_s, W_s \in \mathbb{R}^{n_w \times n_w}$, $Q_s, V_s \in \mathbb{R}^{n_w \times n_c}$, $D_s \in \mathbb{R}^{n_r \times n_c}$, where n_r is the number of rows in each block and n_c the number of columns, hence $x_s \in \mathbb{R}^{n_c}$, $\bar{x} \in \mathbb{R}^{Nn_c}$, and $\bar{A} \in \mathbb{R}^{Nn_r \times Nn_c}$. It's then easy to see that

$$\bar{A}\bar{x} = \begin{bmatrix} D_1x_1 + U_1V_2x_2 + U_1W_2V_3x_3 + U_1W_2W_3V_4x_4 + U_1W_2W_3W_4V_5x_5 + \dots \\ P_2Q_1x_1 + D_2x_2 + U_2V_3x_3 + U_2W_3V_4x_4 + U_2W_3W_4V_5x_5 + \dots \\ P_3R_2Q_1x_1 + P_3Q_2x_2 + D_3x_3 + U_3V_4x_4 + U_3W_4V_5x_5 + \dots \\ P_4R_3R_2Q_1x_1 + P_4R_3Q_2x_2 + P_4Q_3x_3 + D_4x_4 + U_4V_5x_5 + \dots \\ P_5R_4R_3R_2Q_1x_1 + P_5R_4R_3Q_2x_2 + P_5R_4Q_3x_3 + P_5Q_4x_4 + D_5x_5 + \dots \\ \vdots \end{bmatrix}$$

and now we would like to calculate $\frac{\partial[\bar{A}\bar{x}]}{\partial\theta_\bullet}$ for each of

$$\bar{\theta}_P = \begin{bmatrix} \mathrm{vec}(P_1^T) \\ \mathrm{vec}(P_2^T) \\ \vdots \\ \mathrm{vec}(P_N^T) \end{bmatrix}, \quad \bar{\theta}_R = \begin{bmatrix} \mathrm{vec}(R_1^T) \\ \mathrm{vec}(R_2^T) \\ \vdots \\ \mathrm{vec}(R_N^T) \end{bmatrix}, \quad \bar{\theta}_Q = \begin{bmatrix} \mathrm{vec}(Q_1^T) \\ \mathrm{vec}(Q_2^T) \\ \vdots \\ \mathrm{vec}(Q_N^T) \end{bmatrix}, \quad \bar{\theta}_D = \begin{bmatrix} \mathrm{vec}(D_1^T) \\ \mathrm{vec}(D_2^T) \\ \vdots \\ \mathrm{vec}(D_N^T) \end{bmatrix},$$

$$\bar{\theta}_U = \begin{bmatrix} \mathrm{vec}(U_1^T) \\ \mathrm{vec}(U_2^T) \\ \vdots \\ \mathrm{vec}(U_N^T) \end{bmatrix}, \quad \bar{\theta}_W = \begin{bmatrix} \mathrm{vec}(W_1^T) \\ \mathrm{vec}(W_2^T) \\ \vdots \\ \mathrm{vec}(W_N^T) \end{bmatrix}, \quad \bar{\theta}_V = \begin{bmatrix} \mathrm{vec}(V_1^T) \\ \mathrm{vec}(V_2^T) \\ \vdots \\ \mathrm{vec}(V_N^T) \end{bmatrix}$$

It's then easy to derive by hand that:

$$\begin{aligned} \frac{\partial[\bar{A}\bar{x}]}{\partial\bar{\theta}_P} &= \mathcal{SSS}(0,0,0,I_{n_r} \bigotimes \Phi_k^T, 0,0,0) \\ \frac{\partial[\bar{A}\bar{x}]}{\partial\bar{\theta}_R} &= \mathcal{SSS}(P_s, R_s, I_{n_w} \bigotimes \Phi_k^T, 0,0,0,0) \\ \frac{\partial[\bar{A}\bar{x}]}{\partial\bar{\theta}_Q} &= \mathcal{SSS}(P_s, R_s, I_{n_w} \bigotimes x_k^T, 0,0,0,0) \end{aligned}$$

where

$$\Phi_1 = 0, \quad \Phi_{k+1} = R_k\Phi_k + Q_k x_k \tag{2.51}$$

and

$$\frac{\partial[\bar{A}\bar{x}]}{\partial\bar{\theta}_D} = \mathcal{SSS}(0,0,0,I_{n_r} \bigotimes x_k^T, 0,0,0,0) \tag{2.52}$$

For the upper triangular side, the formulas are very similar (although not transposed, as one might expect) as follows.

$$\begin{aligned} \frac{\partial[\bar{A}\bar{x}]}{\partial\bar{\theta}_U} &= \mathcal{SSS}(0,0,0,I_{n_r} \bigotimes \beta_k^T, 0,0,0) \\ \frac{\partial[\bar{A}\bar{x}]}{\partial\bar{\theta}_W} &= \mathcal{SSS}(0,0,0,0,I_{n_w} \bigotimes \beta_k^T, W_k, V_k) \\ \frac{\partial[\bar{A}\bar{x}]}{\partial\bar{\theta}_V} &= \mathcal{SSS}(0,0,0,0,I_{n_w} \bigotimes x_k^T, W_k, V_k) \end{aligned}$$

where

$$\beta_N = 0, \quad \beta_{k-1} = W_k\beta_k + V_k x_k \tag{2.53}$$

So we see that the derivatives of $\bar{A}\bar{x}$ with respect to each of the $\bar{\theta}_\bullet$ vectors of generators is actually a low order SSS matrix with N blocks. Actually, they have even more in common than that: they all share common upper and lower multiplier terms: each having either 0's or R_s and W_s respectively. Hence when all of the

$\bar{\theta}_\bullet$'s are 'shuffled' together to form

$$\bar{\theta} = \begin{bmatrix} \breve{\theta}_1 \\ \breve{\theta}_2 \\ \vdots \\ \breve{\theta}_N \end{bmatrix}, \quad \breve{\theta}_k = \begin{bmatrix} \mathrm{vec}(P_k^T) \\ \mathrm{vec}(R_k^T) \\ \mathrm{vec}(Q_k^T) \\ \mathrm{vec}(D_k^T) \\ \mathrm{vec}(U_k^T) \\ \mathrm{vec}(W_k^T) \\ \mathrm{vec}(V_k^T) \end{bmatrix} \tag{2.54}$$

then the corresponding partial derivative of $\bar{A}\bar{x}$ with respect to $\bar{\theta}$ will also have a very low order, which can be calculated explicitly as:

$$\frac{\partial[\bar{A}\bar{x}]}{\partial\bar{\theta}} = \mathcal{SSS}(P_s, R_s, \breve{Q}_s, \breve{D}_s, U_s, W_s, \breve{V}_s) \tag{2.55}$$

Where

$$\begin{aligned} \breve{Q}_s &= \begin{bmatrix} 0_{n_w \times n_w n_c} & I_{n_w} \bigotimes \Phi_k^T & I_{n_w} \bigotimes x_k^T & 0_{n_w \times (n_r n_c + n_w(n_c+n_r) + n_w^2)} \end{bmatrix} \\ \breve{D}_s &= \begin{bmatrix} I_{n_r} \bigotimes \Phi_k^T & 0_{n_r \times (n_w^2 + n_w n_c)} & I_{n_r} \bigotimes x_k^T & I_{n_r} \bigotimes \beta_k^T & 0_{n_r \times (n_w^2 + n_w n_c)} \end{bmatrix} \\ \breve{V}_s &= \begin{bmatrix} 0_{n_w \times (n_w(2n_c+n_r) + n_w^2 + n_r n_c)} & I_{n_w} \bigotimes \beta_k^T & I_{n_w} \bigotimes x_k^T \end{bmatrix} \end{aligned}$$

We should note that $\frac{\partial[\bar{A}\bar{x}]}{\partial\bar{\theta}}$ has Nn_w rows, $N(2n_w^2 + 2n_w(n_r+n_c) + n_r n_c)$ columns, and SSS order n_w, hence its computational complexity for arithmetic will be something like $\mathcal{O}(n^5 N)$ (where $n_w, n_r, n_c \sim n$). Fortunately, since the iterations for Φ and β are both $\mathcal{O}(N)$, the SSS generators of $\frac{\partial[\bar{A}\bar{x}]}{\partial\bar{\theta}}$ can be computed and populated in $\mathcal{O}(N)$.

We'll leave this computation for now, coming back in Chapter 4 to use it for fast distributed system identification.

2.3.8 SSS Inversion $\mathcal{O}(N)$ example

Now that we've derived the arithmetic and other computations for SSS matrices, we'll actually demonstrate the $\mathcal{O}(N)$ complexity on a simple example. We randomly generated an SSS matrix and extended it to be longer and longer (in a Toeplitz manner), and recorded the times to invert it and the resulting error (as measured by $\|\bar{X}\bar{X}^{-1} - I\|$) for different lengths, as compared to MATLAB's standard inversion solver($\mathcal{O}(N^3)$). As we see in figure 2.7, the SSS inversion is indeed $\mathcal{O}(N)$ in practice, and thus much faster than standard dense solvers, without sacrificing accuracy.

2.3.9 Summary, SSS matrices

So, we've now extended our structure preserving arithmetic of doubly infinite Toeplitz L_r operators in the last chapter to finite dimensional, non-Toeplitz SSS

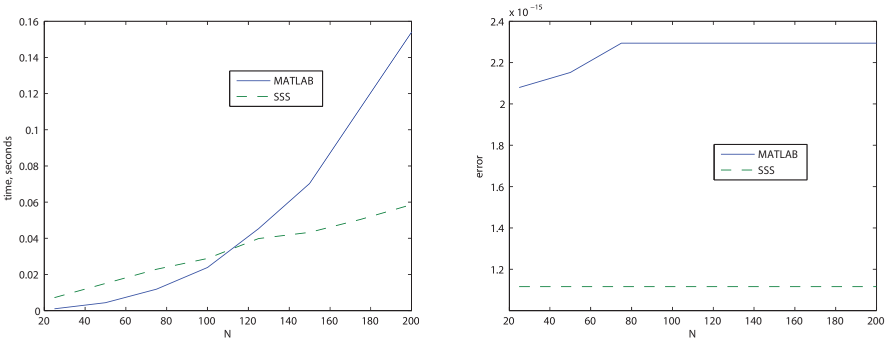


Figure 2.7: Computational complexity and error comparisons

matrices in this section. The arithmetic includes $+$, $\times$, $^{-1}$, and LU factorization, and is only $\mathcal{O}(N)$, as opposed to the usual $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$ complexity for dense matrices. We can also calculate the transpose, shuffle permutation, and Frobenius and 2-norm (to within some tolerance) in only $\mathcal{O}(N)$. As we saw in the example, this complexity is real, and quickly ($N \approx 110$) becomes faster than the usual MATLAB dense matrix computations.

We also note that any banded matrix of bandwidth $2n + 1$ can be put into SSS structure of order n using the algorithm in appendix 2.7 in only $\mathcal{O}(Nn^3)$ computational complexity. So the inverse of the tridiagonal matrix shown in section 2.1.1 isn't tridiagonal, but it is SSS.

Also, if trying to actually form an SSS matrix as a large dense matrix, a naive method of filling in each of the N^2 entries individually will be high $\mathcal{O}(N^3)$ complexity, but there are two clever methods for doing it in only $\mathcal{O}(N^2)$ (the minimal possible complexity) in appendix 2.8.

In our formulation of the SSS structure, we assumed that the generators at each $s \in [0, N]$ could be arbitrarily different from eachother. In this case, it makes sense that we cannot make an arithmetic that is any faster than $\mathcal{O}(N)$, since the matrix is defined by $\mathcal{O}(N)$ data, and each piece of data should be used at least once. However, in many cases this assumption is too general; often all of the generators in $s \in [0, N]$ are identical, except for a few at the ends. This is the case, for example, for the input-output operators of LTI systems over a finite horizon, and in certain distributed control problems for homogeneous systems with boundary conditions. In this case we require only order $\mathcal{O}(1)$ data to store our matrix, and so would hope for an $\mathcal{O}(1)$ arithmetic, independent of the size, N. It turns out that this is possible, using a specialization of the SSS arithmetic, as we will show in the next section.

2.4 Discrete LTI Finite Length Systems and Almost Toeplitz SSS Matrices

In the first section, we considered LTI systems over infinite time, in the second section, we considered LTV systems over finite time, and in this section, we will look at an intermediate case: LTI systems over finite time. These will induce SSS matrices with an additional 'almost Toeplitz' structure that will enable arithmetic calculations that are even faster than $\mathcal{O}(N)$.

Our mixed causal system will take the same form as in the last section:

$$\begin{bmatrix} x^a_{i-1} \\ x^c_{i+1} \end{bmatrix} = \begin{bmatrix} W_i & 0 \\ 0 & R_i \end{bmatrix} \begin{bmatrix} x^a_i \\ x^c_i \end{bmatrix} + \begin{bmatrix} V_i \\ Q_i \end{bmatrix} u_i, \qquad y_i = \begin{bmatrix} U_i & P_i \end{bmatrix} \begin{bmatrix} x^a_i \\ x^c_i \end{bmatrix} + D_i u_i \tag{2.56}$$

for $i \in \{1, ..., N\}$, but with all generators constant for $N_T < i < N - N_B$, and only varying in $s \in [0, N_T]$ and $s \in [N - N_B, N]$. As in the previous sections, we can resolve the state variables, leading to the input-output relationship:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_N \end{bmatrix} = \underbrace{\begin{bmatrix} D_1 & U_1V_2 & U_1W_2V_3 & U_1W_2W_3V_4 & \ddots & \vdots \\ P_2Q_1 & D_2 & U_2V_3 & U_2W_3V_4 & \ddots & \vdots \\ P_3R_2Q_1 & P_3Q_2 & D_3 & U_3V_4 & \ddots & \vdots \\ P_4R_3R_2Q_1 & P_4R_3Q_2 & P_4Q_3 & D_4 & \ddots & \vdots \\ \ddots & \ddots & \ddots & \ddots & \ddots & U_{N-1}V_N \\ \cdots & \cdots & \cdots & \cdots & P_NQ_{N-1} & D_N \end{bmatrix}}_{\bar{A}} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \vdots \\ u_N \end{bmatrix} \tag{2.57}$$

which has the same SSS structure as in the last section, but additionally with all generators constant for $N_T < i < N - N_B$, meaning that the submatrix $\bar{A}(N_T : N - N_B, N_T : N - N_B)$ is Toeplitz, as we see in an example in figure 2.8.

Our definition of 'almost Toeplitz' SSS matrices will basically be the type of matrices produced by 'lifting' the LTI mixed causal system over N steps to produce equation (2.57). They will of course still have the SSS structure, but the generators of the SSS matrices can be separated into three sections; the top, interior, and bottom. For example, for $\bar{A} = \mathcal{SSS}(P_s, R_s, Q_s, D_s, U_s, W_s, V_s)$ above, the generators $(P_s, R_s, Q_s, D_s, U_s, W_s, V_s)$ in the top, for $0 < s < N_T$, and in the bottom, for $N - N_B < s < N$ can be arbitrarily varying, but in the interior will be constant for all $N_T < s < N - N_B$, which we will denote by $(P_\infty, R_\infty, Q_\infty, D_\infty, U_\infty, W_\infty, V_\infty)$.

The great thing about such matrices, hereafter referred to as 'Almost Toeplitz Sequentially Semi Separable' (ATSSS), is that their structure is closed under addition, multiplication, and inversion, although the sizes of N_T and N_B may change a little bit. Furthermore, the Toeplitz structure of the interior can be used to break the usual $\mathcal{O}(N)$ complexity of standard SSS arithmetic when $N_B + N_T \ll N$. The crux of our method is that because the interior of the SSS matrix is Toeplitz, the iterations $\mathfrak{M}_1, \mathfrak{M}_2, \mathfrak{I}_1, \mathfrak{I}_2, \mathfrak{R}_1$ in the SSS arithmetic formulas will quickly converge to a steady state behavior, allowing us to skip to the other boundary, avoiding iterating through the rest of the interior, and thus the bulk of the $\mathcal{O}(N)$ SSS

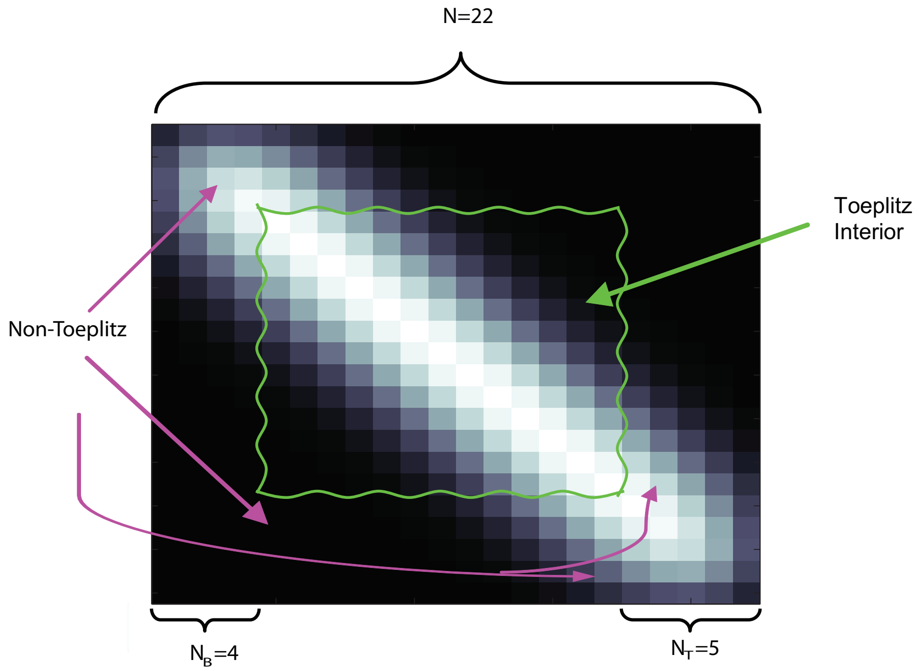


Figure 2.8: Schematic diagram of an 'almost Toeplitz' SSS matrix

computational load, leading to $\mathcal{O}(1)$ complexity (assuming $N_T + N_B \in \mathcal{O}(1)$).

In the following, we will show this arithmetic, which is identical to the standard SSS arithmetic shown in the previous section, except modified to take advantage of the convergence of the iterations in the interior. This convergence is proven in section 2.4.6.

2.4.1 ATSSS Addition

To add together two conformably sized ATSSS matrices, $\overline{X} = \mathcal{SSS}(P_s^X, R_s^X, Q_s^X, D_s^X, U_s^X, W_s^X, V_s^X)$ and $\overline{Y} = \mathcal{SSS}(P_s^Y, R_s^Y, Q_s^Y, D_s^Y, U_s^Y, W_s^Y, V_s^Y)$ and obtain the ATSSS generator form of their sum: $\overline{X}+\overline{Y} = \overline{Z} = \mathcal{SSS}(P_s^Z, R_s^Z, Q_s^Z, D_s^Z, U_s^Z, W_s^Z, V_s^Z)$, does not involve any iterations, and is thus identical to the usual SSS calculation, with the exception that the interior calculation of the generator matrices only needs to be performed once, instead of $N - N_B - N_T$ times. Simply perform: $\mathfrak{A}_1$ above for each $0 < s < N_T$, $s = \infty$ and $N - N_B < s < N$.

2.4.2 ATSSS Multiplication

To multiply together two conformably sized ATSSS matrices, $\overline{X} = \mathcal{SSS}(P_s^X, R_s^X, Q_s^X, D_s^X, U_s^X, W_s^X, V_s^X)$ and $\overline{Y} = \mathcal{SSS}(P_s^Y, R_s^Y, Q_s^Y, D_s^Y, U_s^Y, W_s^Y, V_s^Y)$ and obtain the ATSSS generator form of their product: $\overline{XY} = \overline{Z} = \mathcal{SSS}(P_s^Z, R_s^Z, Q_s^Z, D_s^Z, U_s^Z,$

W_s^Z, V_s^Z), there are two calculations iterated, just as in SSS multiplication, started variously with different initial conditions.

For a standard SSS matrix without any Toeplitz structure, we would just calculate $\mathfrak{M}_1$ for $s = 1 : N$ starting with $S_0 = 0$, and then $\mathfrak{M}_2$ for $s = N : 1$ starting with $T_N = 0$, leading to $\bar{Z}$ with $\mathcal{O}(N)$ computational complexity, as in section 2.3.2. When $\bar{X}$ and $\bar{Y}$ have the ATSSS structure though, the iterations on S and T converge(see Lemma 2.16), and we can skip most of the interior of the matrix, saving alot of time when $N_T + N_B \ll N$.

The iteration proceeds as follows, given some small positive tolerance, γ (chosen heuristically to be very small, e.g. $\gamma = 10^{-12}$). Starting with $s = 1$ and $S_s = 0$, iterate $\mathfrak{M}_1$ while $\|S_{s+1} - S_s\| > \gamma$. The lowest s such that $\|S_{s+1} - S_s\| < \gamma$ will be N_T for $\bar{Z}$. Having achieved convergence, we then skip the rest of the interior, by setting $S_\infty = S_{N_T}, R_\infty^Z = R_{N_T}^Z, W_\infty^Z = W_{N_T}^Z, Q_\infty^Z = Q_{N_T}^Z, U_\infty^Z = U_{N_T}^Z$ and finish at the bottom by setting $s = N - N_B$, $S_s = S_\infty$, and iterating $\mathfrak{M}_1$ until $s = N$.

Starting with $s = N$ and $T_s = 0$, we should then iterate $\mathfrak{M}_2$ while $\|T_{s-1} - T_s\| > \gamma$. The lowest $N - s$ such that $\|T_{s-1} - T_s\| < \gamma$ will be N_B for $\bar{Z}$. Having achieved convergence, we then skip the interior, by setting $T_\infty = T_{N_B}, P_\infty^Z = P_{N_B}^Z, D_\infty^Z = D_{N_B}^Z, V_\infty^Z = V_{N_B}^Z$, and finish at the top, by setting $s = N_T$, $T_s = T_\infty$, and iterating $\mathfrak{M}_2$ until $s = 0$, at which point the ATSSS multiplication is finished, resulting in $\bar{Z}$.

2.4.3 ATSSS Inversion

For ATSSS inversion, we will only show the method for positive definite matrices, since we can prove convergence of the iterations in this case (see section 2.4.6) and since all inversion problems can be turned into positive definite inversion problems via pseudoinverse formulas: $\overline{Y}^{-1} = \overline{Y}^T(\overline{YY}^T)^{-1}$, where $\overline{YY}^T = \overline{X} = \mathcal{SSS}(C_s, A_s, B_s, D_s, B_s^T, A_s^T, C_s^T)$ is positive definite, and thus this method is not restrictive. However, experiments show that the non-symmetric SSS inversion method, when applied to ATSSS methods, also converges in practice.

As with the ATSSS multiplication, there are two essential iterations performed in this calculation, just as in the SSS version in section 2.3.3: $\mathfrak{I}_1$, $\mathfrak{I}_2$.

Starting with $s = 1$ and $S_s = 0$, iterate $\mathfrak{I}_1$ while $\|S_{s+1} - S_s\| > \gamma$. The lowest s such that $\|S_{s+1} - S_s\| < \gamma$ will be the N_T for $\bar{Z}$. We then skip the interior, by setting $S_\infty = S_{N_T}, R_\infty = R_{N_T}, Q_\infty = Q_{N_T}$ and finish at the bottom by setting $s = N - N_B$, $S_s = S_\infty$, and iterating $\mathfrak{I}_1$ until $s = N$.

Starting with $s = N$ and $T_s = 0$, we should then iterate $\mathfrak{I}_2$ while $\|T_{s-1} - T_s\| > \gamma$. The lowest $N - s$ such that $\|T_{s-1} - T_s\| < \gamma$ will be the N_B for $\bar{Z}$. We then skip the interior, by setting $T_\infty = T_B, P_\infty = P_{N_B}, F_\infty = F_{N_B}$ and finish at the top, setting $s = N_T$, $T_s = T_\infty$, and iterating $\mathfrak{I}_2$ until $s = 0$, at which point the ATSSS inversion is finished, resulting in $\bar{X}^{-1} = \bar{Z} = \mathcal{SSS}(P_s, R_s, Q_s, F_s, Q_s^T, R_s^T, P_s^T)$.

2.4.4 ATSSS Order Reduction

Given some ATSSS matrix $\overline{X} = \mathcal{SSS}(C_s, A_s, B_s, D_s, E_s, F_s, G_s)$ where the sizes $A_s, F_s \in \mathbb{R}^{n_x \times n_x}$ are larger than is practical for fast computations, one might like to find a close ATSSS approximation $\overline{X} \approx \overline{Z} = \mathcal{SSS}(P_s, R_s, Q_s, D_s, U_s, W_s, V_s)$ of lower 'SSS order': $R_s, W_s \in \mathbb{R}^{n_z \times n_z}$ where $n_z < n_x$. Just as for the SSS version in section 2.3.4, for brevity, we will only show the 'lower triangular' version.

As with the ATSSS multiplication and inversion, there are two essential iterations performed in this calculation, started variously with different initial conditions: $\mathfrak{R}_1, \mathfrak{R}_2$. The iteration proceeds as follows:

Starting with $s = 1$ and $S_s = 0$, iterate $\mathfrak{R}_1$ while $\|S_{s+1} - S_s\| > \gamma$. The lowest s such that $\|S_{s+1} - S_s\| < \gamma$ will be N_T for $\bar{Z}$. We then skip the interior, by setting $S_\infty = S_{N_T}$ and finish at the bottom, setting $s = N - N_B$, $S_s = S_\infty$, and iterating $\mathfrak{R}_1$ until $s = N$.

Starting with $T_N = 0$ and $s = N$, we should then iterate $\mathfrak{R}_2$ while $\|T_{s-1} - T_s\| > \gamma$. The lowest $N - s$ such that $\|T_{s-1} - T_s\| < \gamma$ will be N_B for $\bar{Z}$. We then skip the interior, by setting $T_\infty = T_{N_B}, C_\infty = C_{N_B}, A_\infty = A_{N_B}, B_\infty = B_{N_B}$ and finish at the top, setting $s = N_T$, $T_s = T_\infty$, and iterating $\mathfrak{R}_2$ until $s = 0$, at which point the ATSSS order reduction is finished, resulting in $\bar{Z}$.

2.4.5 ATSSS Transpose & Permutations & Norm Calculation

For ATSSS matrices, the transpose and permutation algorithms are obvious extensions of the SSS versions: just perform the same calculations as for the regular SSS methods, but only at each $0 < s < N_T$, $s = \infty$ and $N - N_B < s < N$.

As for the norm calculations, both the $\|\cdot\|_2$ and $\|\cdot\|_F$ norm work as in the SSS version, except with the same convergence as in the inversion and multiplication.

2.4.6 Convergence of ATSSS Arithmetic

There are only two types of iterations in the arithmetic calculations above that need to converge in the interior for the ATSSS techniques to work; Sylvester iterations on S_s in $\mathfrak{M}_1$ and $\mathfrak{R}_1$ and on T_s in $\mathfrak{M}_2$, $\mathfrak{I}_2$, and $\mathfrak{R}_2$ and a Riccati iteration on S_s in $\mathfrak{I}_1$. When these matrices converge, so do all of the generators, allowing us to skip the large interior of the matrices, reducing the computational complexity.

Lemma 2.15 *Assume that the ATSSS matrix* $\overline{X} = \mathcal{SSS}(C_s, A_s, B_s, D_s, B_s^T, A_s^T, C_s^T)$ *is positive definite,* $\overline{X} \succ 0$*, then in the interior of* $\bar{X}$*,* S_s *in* $\mathfrak{I}_1$ *converges exponentially fast to the the unique positive semi-definite stabilizing solution of the Riccati equation:*

$$S_\infty = A_\infty S_\infty A_\infty^T + (B_\infty - A_\infty S_\infty C_\infty^T)(D_\infty - C_\infty S_\infty C_\infty^T)^{-1}(B_\infty - A_\infty S_\infty C_\infty^T)^T \tag{2.58}$$

Proof: *Using the fact that $D_\infty \succ 0$ from Lemma 2.13, and assuming that A_∞ is stable, $(A_\infty - B_\infty D_\infty^{-1} C_\infty, C_\infty)$ is detectable and $(A_\infty - B_\infty D_\infty^{-1} C_\infty, -B_\infty D_\infty^{-1/2})$ is stabilizable, satisfying the conditions of Lemma 14.5.8 of [61], and combining this with our result that $S_s \succeq 0 \quad \forall s \in \mathbb{N}$, from our Lemma 2.13, the generators of $\bar{X}$ thus satisfy the conditions in Theorem 14.5.2 of [61], hence proving that $\|S_s - S_\infty\|$ converges exponentially fast.* □

Note that above we assumed that $\rho(\bar{A}_\infty) < 1$, which can be justified by physical considerations inherent in the type of systems that we would like to consider (see Chapter 6) and the fact that the ATSSS arithmetic derived above preserves this stability (trivially for addition, multiplication, and model order-reduction, and for inversion, we proved in Lemma 2.15 that S_∞ is the *stabilizing* solution, hence R_∞ is stable).

Now we can treat the Sylvester iterations. Once the iterations have left the boundaries and entered the interior, they all take the basic form of:

$$X_{s+1} = AX_sB + C \tag{2.59}$$

where $\rho(A), \rho(B) < 1$.

Lemma 2.16 *X_s in equation (2.59) converges exponentially fast to the unique solution $X_\infty = AX_\infty B + C$, regardless of the initial condition X_0.* ***Proof:*** *Subtracting X_∞ from equation (2.59), we get $X_{s+1} - X_\infty = A(X_s - X_\infty)B$. Hence $\|X_s - X_\infty\| \leq \|A^s\|\|B^s\|\|X_0 - X_\infty\|$. There exists an r and an M such that $\rho(A)\rho(B) < r < 1$ and $\|A^s\|\|B^s\| < Mr^s$, for all s (see [81]), completing the proof.* □

From Lemma 2.15 and Lemma 2.16 we see that once the iterations in the SSS calculations enter the homogeneous interior of an ATSSS matrix, they quickly (exponentially fast) approach a steady state solution, which can be used for the entire interior, allowing one to skip the rest of the iterations in the interior and go to the other boundary, decreasing the computational complexity, and preserving the 'almost Toeplitz'-ness. Note that we wouldn't expect the iterations to reach a steady state while still in one of the boundaries (although this can happen if the boundary is almost homogeneous), so usually, after each ATSSS arithmetic operation, the sizes of the heterogeneous boundaries N_T and N_B of the resulting ATSSS matrix will be greater than before. However, since the iterations converge *exponentially fast*, the values of N_T and N_B will not grow very much. Thus when $N_B + N_T \ll N$, considerable savings are possible.

2.4.7 Example

We now extend the example in section 2.3.8 to our ATSSS algorithms. Using the same matrices as before, we again recorded the time to invert them and the error

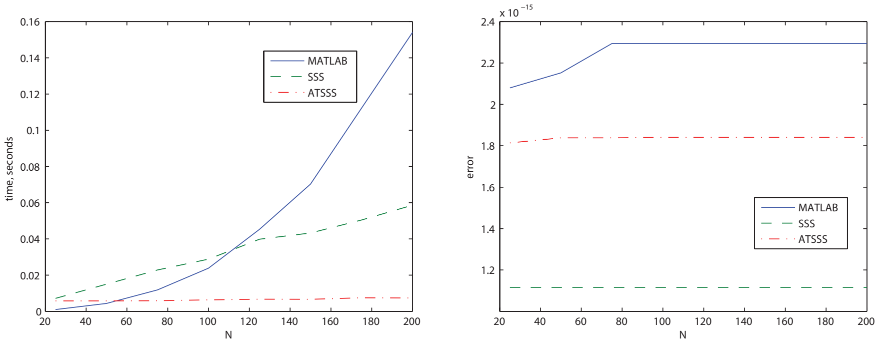


Figure 2.9: Computational complexity and error comparisons

(as measured by $\|\bar{X}\bar{X}^{-1} - I\|$) for different lengths, and compared them to the SSS routines and to MATLAB's standard inversion solver, as we see in figure 2.9. Note that, using the ATSSS routines, it doesn't take any longer to invert the matrix of size $N = 200$ than for $N = 25$ because the iterations converge and most of the interior is skipped.

2.4.8 Summary, ATSSS matrices

So, in this section we've shown that when the SSS matrices of the previous section have an additional 'almost Toeplitz' structure, the recursions in the SSS arithmetic converge exponentially fast in the Toeplitz interior, allowing one to skip to the next boundary, cutting out most of the computational load. The result is an arithmetic for ATSSS matrices with $\mathcal{O}(1)$ complexity, that is, independent of the size, N. We saw that this really works in the example.

We also note that these ATSSS methods are very close in concept to the 'IVI' systems of equations in [79], which deals with infinite matrices that are shift invariant in either direction off to infinity, but varying in the middle. See Chapter 11 for further extensions of these ideas.

2.5 Continuous LTI Systems and Their Transfer Matrices

In the previous three sections, we've always been talking about mixed causal systems operating in discrete time, which induce input-output operators of Laurent matrices, or SSS matrices. However, in section 1 we saw that all of our arithmetic on Laurent matrices could just as well be thought of as arithmetic on stable realizations of transfer functions over the unit circle. This of course begs the question: can we do the same thing for transfer functions on the imaginary axis? The answer is yes, as we'll show in this section, and such a transfer function arithmetic is even useful, as we'll see later in Chapter 10 for LPV systems (and likely also for continuum distributed systems).

Our mixed causal continuous time systems will take the form:

$$\begin{bmatrix} \dot{x}^a(t) \\ \dot{x}^c(t) \end{bmatrix} = \begin{bmatrix} -W & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} x^a(t) \\ x^c(t) \end{bmatrix} + \begin{bmatrix} V \\ Q \end{bmatrix} u(t), \qquad y(t) = \begin{bmatrix} U & P \end{bmatrix} \begin{bmatrix} x^a(t) \\ x^c(t) \end{bmatrix} + Du(t)$$

for $t \in \mathbb{R}$, where R and W are Hurwitz; R is stable and $-W$ is anti-stable. Instead of thinking of this as a system moving forward in time with stable and antistable parts, we can think of it as a system with a stable part moving forward in time, $x^c(t)$, and a stable part moving backward in time, $x^a(t)$. Notice the similarity between this and equations (2.5) in section 2.2.

Just as before, we can resolve the state variables to write this just in terms of the inputs $u(t)$ and outputs $y(t)$, although we don't get a matrix anymore, but a convolution operator.Hence we can't use our matrix intuition like before, but by using a Fourier transform, we can just think of this convolution operator as a transfer function:

$$A(s) = D + P(sI - R)^{-1}Q + U(s^*I - W)^{-1}V \tag{2.60}$$

where s is on the imaginary axis and R and W are both strictly stable. We'll abbreviate such transfer functions as:

$$\overline{A} = \mathcal{S}_c(P, R, Q, D, U, W, V) \tag{2.61}$$

and we will use the notation $\bar{A} \in \mathcal{S}_c$ to indicate that $\bar{A}$ is a mixed causal realization of $A(s)$ with stable (with respect to the imaginary axis) causal and anticausal parts.

By again using our LTI systems thinking, except now in continuous time, we can construct a structure preserving arithmetic of such realizations, as follows. The following formulas and results can be derived analogously to those in section 2.2, and are very similar.

2.5.1 $\mathcal{S}_c$ Addition

Lemma 2.17 *Given*

$$\begin{aligned} \bar{X} &= \mathcal{S}_c\{P_X, R_X, Q_X, D_X, U_X, W_X, V_X\} \\ \bar{Y} &= \mathcal{S}_c\{P_Y, R_Y, Q_Y, D_Y, U_Y, W_Y, V_Y\} \end{aligned}$$

Then a realization of the sum: $\bar{Z} = \bar{X} + \bar{Y}$ *is:*

$$\bar{Z} = \mathcal{S}_c\{ \begin{bmatrix} P_X^T \\ P_Y^T \end{bmatrix}^T, \begin{bmatrix} R_X & 0 \\ 0 & R_Y \end{bmatrix}, \begin{bmatrix} Q_X \\ Q_Y \end{bmatrix}, (D_X + D_Y), \begin{bmatrix} U_X & U_Y \end{bmatrix}, \begin{bmatrix} W_X & 0 \\ 0 & W_Y \end{bmatrix}, \begin{bmatrix} V_X \\ V_Y \end{bmatrix} \}$$

Proof: *This is verifiable by inspection, and follows that in discrete time in Lemma 2.3* □

2.5.2 $\mathcal{S}_c$ Multiplication

Lemma 2.18 *Given:*

$$\bar{X} = \mathcal{S}_c\{P_X, R_X, Q_X, D_X, U_X, W_X, V_X\} \tag{2.62}$$

$$\bar{Y} = \mathcal{S}_c\{P_Y, R_Y, Q_Y, D_Y, U_Y, W_Y, V_Y\} \tag{2.63}$$

$\bar{X}\bar{Y} = \bar{Z} = \mathcal{S}_c\{P_Z, R_Z, Q_Z, D_Z, U_Z, W_Z, V_Z\}$ *with*

$$D_Z = D_X D_Y, \quad P_Z = \begin{bmatrix} P_X & D_X P_B + U_X S \end{bmatrix}$$

$$R_Z = \begin{bmatrix} R_X & Q_X P_Y \\ 0 & R_Y \end{bmatrix}, \quad Q_Z = \begin{bmatrix} Q_X D_Y + T V_Y \\ Q_Y \end{bmatrix}$$

$$U_Z = \begin{bmatrix} U_X & D_X U_Y + P_X T \end{bmatrix}, W_Z = \begin{bmatrix} W_X & V_X U_Y \\ 0 & W_Y \end{bmatrix}, V_Z = \begin{bmatrix} V_X D_Y + S Q_Y \\ V_Y \end{bmatrix}$$

where S and T are the unique solutions to the Sylvester equations:

$$W_X S + S R_Y + V_X P_Y = 0, \quad R_X T + T W_Y + Q_X U_Y = 0$$

Proof: *This is easy to verify using LTI continuous time systems theory, where we consider $\overline{X}$ and $\overline{Y}$ to be mixed-causal system realizations that we put in series. The derivation follows that in discrete time as in Lemma 2.4.* ☐

Note that the proper rational transfer function structure, and also the stability of the realizations, is preserved under these addition and multiplication algorithms: $\bar{X}, \bar{Y} \in \mathcal{S}_c \Rightarrow \bar{Z} \in \mathcal{S}_c$.

2.5.3 $\mathcal{S}_c$ Inversion

We have shown the closure of $\mathcal{S}_c$ under addition and multiplication, but we will also need inversion.

Lemma 2.19 *Given $\overline{A} = \mathcal{S}_c\{P, R, Q, D, U, W, V\}$, if the nonsymmetric CARE:*

$$(R - QD^{-1}P)T + T(W - VD^{-1}U) + TVD^{-1}PT + QD^{-1}U \quad = \quad 0 \tag{2.64}$$

has a stabilizing *solution, e.g., a T for which both*

$$\begin{aligned} W^F &= W - VD^{-1}(U - PT) \\ R^F &= R - (Q - TV)D^{-1}P \end{aligned}$$

are stable, then $\overline{A}$ has an inverse $\bar{A}^{-1} = \bar{F} = \mathcal{S}_c\{P^F, R^F, Q^F, D^F, U^F, W^F, V^F\}$ where

$$D^F = D^{-1}, \quad U^F = D^F(U - PT), \quad Q^F = (Q - TV)D^F$$

$$V^F = SQ^F - VD^F, \quad P^F = U^F S - D^F P$$

where S is the (unique) solution to the Sylvester equation

$$W^F S + SR^F + VD^F P = 0 \tag{2.65}$$

Proof: *can be derived similarly to Lemma 2.8 by first assuming an outer factorization $\bar{A} = \bar{L}\bar{U}$ then constraining the resulting product.* □

Note that whereas in the $\mathcal{S}$ version in Lemma 2.8 we had a discrete algebraic Riccati equation and a Stein equation, here we have a continuous algebraic Riccati equation and a Lyapunov type equation. Similarly to the $\mathcal{S}$ case, the Riccati equation (2.64) *will not* have a stabilizing solution for every invertible $\bar{A}$, but in the symmetric case it does:

Lemma 2.20 *Given a Hermitian $\bar{X} = \mathcal{S}_c\{P, R, Q, Y, Q^*, R^*, P^*\}$. Then $\bar{X} \succ 0$ if and only if the Riccati equation*

$$RG + GR^* + \Phi Y \Phi^* = 0 \tag{2.66}$$

where $\Phi = (Q - GP^*)Y^{-1}$ and $\Pi = R - \Phi P$, has a stabilizing solution G, in which case the Lyapunov equation

$$\Pi^* H + H\Pi + P^* Y^{-1} P = 0 \tag{2.67}$$

has a unique solution H, and $\bar{X}^{-1} = \bar{Z} = L_r\{P_Z, R_Z, Q_Z, Y_Z, Q_Z^T, R_Z^T, P_Z^T\}$, may be calculated as:

$$P_Z = \Phi^* H - Y^{-1}P, \quad R_Z = \Pi, \quad Q_Z = \Phi, \quad Y_Z = Y^{-1}$$

Proof: *Set $F(s) = \frac{1}{2}Y + P(sI - R)^{-1}Q$, and this follows directly from the Positive Real Lemma ([60], Lemma 13.27). Note that $X(s) \succ 0 \forall s \in \Im$ implies $Y \succ 0$, and that $\lambda(R - \Phi P) \in \mathbb{C}_-$ is sufficient for (2.67) to have a unique solution and for $\bar{Z}$ to be stable. Note that the adjoint is characterized as: $\mathcal{S}_c\{P, R, Q, D, U, W, V\}^* = \mathcal{S}_c\{V^*, W^*, U^*, D^*, Q^*, R^*, P^*\}$. The inversion formula then follows as a special case of Lemma 2.19* □

Just as with the L_r matrices, we can now extend to the nonsymmetric case:

Lemma 2.21 *Assume $\bar{A} \in \mathcal{S}_c$. Then $\exists \bar{A}^{-1} \in \mathcal{S}_c \Leftrightarrow 0 \notin \lambda(\bar{A})$. Furthermore, we can calculate it using the formulas in Lemma 2.20.* ***Proof:*** *$\Leftarrow$ Then clearly $0 \prec \bar{A}\bar{A}^* \in \mathcal{S}_c$, and we can use Lemma 2.20 to calculate $\bar{A}^{-1} = \bar{A}^*(\bar{A}\bar{A}^*)^{-1} \in \mathcal{S}_c$ $\Rightarrow$ A bounded $\bar{A}^{-1}$ always implies $0 \notin \lambda(\bar{A})$.* □

We will call such $\bar{A} \in \mathcal{S}_c$ with $\bar{A}^{-1} \in \mathcal{S}_c$ ‘regular’. Note that we have been assuming $\bar{A}$ square, but these results could easily be extended to nonsquare $\bar{A}$ using left and right inverses.

We also note that the rational order of $\bar{A}^{-1}$, as calculated above in Lemma 2.21, will be generally 3 times the rational order of $\bar{A}$. However, often this can be avoided by inverting $\bar{A}$ directly, if(2.64) does have a stabilizing solution (which can always be calculated using the sign iteration [82]), notice that the resulting $\bar{F}$ does not increase in order from $\bar{A}$. This is often the case in practice, allowing us to greatly speed up our iterative computations.

2.5.4 $\mathcal{S}_c$ Order Reduction

Through the above two operations, $Z(s)$ will be a rational transfer matrix with order larger than $X(s)$ or $Y(s)$. In the same way as for the L_r matrices in section 2.2, since we represent $\bar{Z}$ as the sum of $Z(s) = L(s) + U(s)$ with LTI causal and anticausal interpretations, respectively, we can efficiently perform order reduction on Z by performing standard LTI state space model order reduction on its upper and lower triangular parts separately. Since each is stable, using, e.g. balanced truncation, we also obtain an H_∞ bound on the error of the transfer function(see e.g. [60]). If we perform balanced truncations such that $\|L(s) - \tilde{L}(s)\|_\infty < e_L$, $\|U(s) - \tilde{U}(s)\|_\infty < e_U$, then the reduced order realization $\tilde{\bar{Z}} = \tilde{\bar{L}} + \tilde{\bar{U}}$ has error bound $\|\bar{Z} - \tilde{\bar{Z}}\| < e_L + e_U$. This is just the continuous domain analog of the procedure for L_r matrices in section 2.2, and formulas for continuous domain balanced truncation can be found in [60], so we will omit them here.

2.5.5 $\mathcal{S}_c$ Transpose, Permutation, and Norm

The transpose and permutation formulas are exactly those in section 2.2.6 of the Laurent matrix section. The norm is also computed in the same way as in section 2.2.8, only the outer-factorization is different on the imaginary axis from on the unit circle:

Lemma 2.22 (Outer Factorization) *Assume we have a Hermitian* $\bar{X} = \mathcal{S}_c\{P, R, Q, Y, Q^*, R^*, P^*\}$, $\bar{X} \succ 0$. *Then* $\exists$

$$\begin{aligned} \bar{L} &= \mathcal{S}_c\{P_L, R_L, Q_L, D_L, 0, 0, 0\} \\ \bar{L}^{-1} &= \mathcal{S}_c\{(D_L^{-1}P_L), (R_L - Q_L D_L^{-1} P_L), (-Q_L D_L^{-1}), D_L^{-1}, 0, 0, 0\} \end{aligned}$$

with D_L *invertible such that* $\bar{L}\bar{L}^* = \bar{X}$. *Furthermore, such an* $\bar{L}$ *can be calculated as:* $P_L = P$, $R_L = R$, $D_L D_L^* = Y$, $Q_L = (Q - GP^*)Y^{-1}D_L$, *where* $G \succeq 0$ *is the stabilizing solution to the Riccati equation(2.66).* D_L *can be calculated via Cholesky factorization.* ***Proof:*** *The derivation is part of that of the inversion. Since* $\bar{X} \succ 0$, *a stabilizing* G *exists by the positive real lemma.* $\bar{L} \in \mathcal{S}_c$ *since* $R_L = R$ *is stable, and* $\bar{L}^{-1} \in \mathcal{S}_c$ *since* G *is stabilizing and thus* $(R_L - Q_L D_L^{-1} P_L)$ *is stable.* $\square$

As for the $\mathcal{L}_2$ induced norm of an $\mathcal{S}_c$ realization, we can use the Bounded Real Lemma for stable continuous LTI systems [60] to find the infinity norm of $L(s)$ and thus $A(s)$, where $\bar{L}\bar{L}^* = \bar{A}^*\bar{A} + I \succ 0$, is an outer factorization as explained in section 2.2.8 for L_r matrices.

2.5.6 Summary, $\mathcal{S}_c$ realizations

So, in this section we have basically replicated the results of transfer functions on the unit circle in section 2.2 for transfer functions on the extended imaginary axis. All of the results are very nicely similar; continuous domain Riccati equations instead of discrete time Riccati equations, Lyapunov equations instead of Stein equations, etc. However, our uses of these arithmetics will be very different, as we will see in chapters 5 and 10.

2.6 Conclusion

In this chapter we have seen that the input output operators of mixed-causal systems have a special structure ($\mathcal{S}$, SSS, ATSSS, or $\mathcal{S}_c$, depending on the domain and properties of the system), and that by using basic linear algebra and basic linear systems theory, we can develop an arithmetic($+, \times, ^{-1}$) of such operators that preserves the special structure (in contrast to the other structures in subsection 2.1.1). In the cases of the rationally symbolled Laurent matrices and the mixed causal transfer functions over the imaginary axis, this arithmetic took the form of using finite matrix calculations on stable realizations to equivalently do calculations for infinite dimensional operators, while for SSS and ATSSS matrices, we saw that similarly by performing all calculations on the small 'generator matrices', such operations could be reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}(N)$ and $\mathcal{O}(1)$ complexity, respectively.

Next, in Chapter 3, we will show how these results can be used not just for arithmetic, but that we can build structured Riccati and Lyapunov solvers with them using iterative algorithms, and thus ultimately do controller synthesis and analysis very efficiently. In Part II (Chapters 4-10) of this thesis, we will then see how the structured matrices of this chapter can be profitably used to represent repetitive, LPV, and varius kinds of distributed systems.

2.7 Appendix: Algorithm for finding SSS matrix representations of banded matrices

For full matrices, there are already published $\mathcal{O}(N^2n^2)$ methods for finding the SSS generators [47][48]. However, since in distributed systems applications, matrices are often banded, we will print here a $\mathcal{O}(Nn^3)$ specialization of these methods to such matrices.

Say we have a banded matrix, $A \in \mathbb{R}^{N\times N}$, with n nonzero bands directly on either side of the diagonal, which we would like to find an SSS representation for: $\bar{A} = \mathcal{SSS}(P_s, R_s, Q_s, D_s, U_s, W_s, V_s)$. We assume that this is a scalar, not a block, SSS matrix for simplicity here, and will just show the Matlab pseudocode for the upper triangular U, W, V side, since lower triangular can be found just by a transpose of this. The routine is as follows:

```
initialize variables:
    $U = 0(1,n,N); W = 0(n,n,N); V = 0(n,1,N);$
    $H = A(1,2:n+1);$
    $[L,M,G^T] = svd(H);$
    $L = [L,0(1,n-1)];$
    $M = [M,0(1,n-1);0(n-1,n),0(n-1,n-1)];$
    $G = [G;0(n-1,n)];$
    $U(:,:,1) = L(1,1:n);$
    $V(:,:,2) = M(1:n,1:n)*G(1:n,1:n_c);$
    $T = M(1:n,1:n)*G(1:n,n_c+1:size(G,2));$
now run the loop:
    for $i = 2:N-1$
        $H = [[T,0(n,size(H2,2)-size(T,2))];A(i,1+i:min(i+n,N))];$
        $[L,M,G^T] = svd(H);$
        $L = [L,0(size(L,1),n-1)];$
        $M = [M,0(size(M,1),n-1);0(n-1,size(M,2)),0(n-1,n-1)];$
        $G = [G;0(n-1,size(G,2))];$
        $W(:,:,i) = L(1:n,1:n);$
        $U(:,:,i) = L(1+n:size(L,1),1:n);$
        $V(:,:,i+1) = M(1:n,1:n)G(1:n,1);$
        $T = M(1:n,1:n)G(1:n,2:size(G,2));$
    end                                                        (2.68)
```

Where $svd()$ is a singular value decomposition, $min()$ is the minimum of two arguments, and $size()$ finds the dimensions of a matrix. This routine outputs N length arrays of matrices $U(:,:,1:N)$, $W(:,:,1:N)$, and $V(:,:,1:N)$ which are the SSS generators of the upper triangular part of $\bar{A}$. The loop cycles N times, each time computing some matrix multiplications and an SVD of size $n \times n$, and hence is $\mathcal{O}(Nn^3)$ complexity.

2.8 Appendix: Algorithms for Building Dense matrices out of SSS representations in $\mathcal{O}(N^2)$

This appendix will provide an algorithm for doing basically the opposite of the last appendix; given an SSS representation: $\bar{A} = \mathcal{SSS}(P,R,Q,D,U,W,V)$, we'd now like to build a full matrix, filling in each of it's $\mathcal{O}(N^2)$ entries, as fast as possible. It turns out that this is possible in only $\mathcal{O}(N^2)$, using two different methods.

The first, thought of by Ivo Houtzager, is to employ the $\mathcal{O}(N)$ SSS matrix-vector routine in [48] to multiply $\bar{A}$ by each of the N columns of the $N \times N$ identity matrix. This is fast and easy to program, and works best when the 'SSS order' (the dimensions of the R and W matrices) is much larger than row size, n_r, of P and U and the column size, n_c, of Q and V (empirically, 15 times larger).

When the SSS order is not so large, there is another routine that works much faster, as follows. We will show the routine in Matlab pseudo-code, and only for the lower triangular P, R, Q side of the matrix, since the other half can be built using a transpose of this method. The routine is as follows:

```
              initialize variables:
      $L$  =  $zeros(n_r N, n_c N)$;
              fill in the first diagonal:
for   $k$  =  $2 : N$
                $r = k$;   $c = k - 1$;
                $L((n_r(r-1)+1) : n_r(r), (n_c(c-1)+1) : n_c(c)) = P(:,:,r)Q(:,:,c)$;
  end
              initialize variables:
      $\Pi$  =  $zeros(size(R,2), size(R,2), N-1)$;
for   $i$  =  $1 : N-1$
                $\Pi(:,:,i) = I$;
  end

            fill in the rest of the matrix:
for  $k$  =  $2 : N$
              $z = 1$;   $\hat{\Pi} = 0$;
              for  $i = k : N-1$
                $\hat{\Pi}(:,:,z) = R(:,:,i)\Pi(:,:,z)$;
                $r = i + 1;,$      $c = i - (k-1)$;
                $L((n_r(r-1)+1) : n_r(r), (n_c(c-1)+1) : n_c(c)) = P(:,:,r)\hat{\Pi}(:,:,z)Q(:,:,c)$;
                $z = z + 1$;
              end
              $\Pi = \hat{\Pi}$;
  end
```

The result of this routine will be L, the lower triangular half of A.

3

Structure Preserving Iterations

Having developed computationally fast and structure preserving arithmetics for four types of large matrices and operators in Chapter 2, in this chapter we will investigate controller synthesis and analysis routines which can be built on these arithmetics. This progression will eventually lead to computationally efficient system analysis and controller synthesis routines for all of the structures in Chapter 2, and also for operators with multiple levels, each level having one of the aforementioned structures. Such computation methods will then be employed in Part II for distributed and repetitive control systems, the system matrices of which can be shown to have these structures.

3.1 Introduction

Why consider structure preserving iterations at all? Because many of the traditional methods for control and analysis scale very badly to large problems. Consider as an example H_∞ synthesis for a linear system with state, input, and output dimensions N.

Perhaps the most used approachs these days are the LMI approach [83] of $\mathcal{O}(N^6)$ complexity, and the Riccati equation approach [84] of $\mathcal{O}(N^3)$ complexity(due both to the necessity of solving Riccati equations (probably using a QZ method) and calculating SVDs). These techniques both work very well on small problems ($N < 50$), but not so well on large ones; the computational load and memory necessary just become too much for practical use; no one wants to spend 3 weeks of computer time to design a controller.

Furthermore, LMIs, QZ methods, and SVDs all require ‘parsing’ of the data or an ordering of eigenvalues that generally destroys any nice matrix structure. Hence even for large problems ($N > 500$) that have realizations of matrices possessing some of the special structures we discussed in Chapter 2, this cannot be used to any advantage.

However, there are techniques that can be used to solve Riccati equations which preserve certain matrix structures, and it turns out that, with care, such techniques can be used to perform each step of H_∞ analysis and synthesis in a structure preserving way. In this chapter, we'll discuss such a method, called the matrix sign function. Our motive for investigating it is that it will turn out that at least one method for calculating the matrix sign function (called the matrix sign iteration) can be performed extremely efficiently, and in a structure preserving way, for each of the special operator structures discussed in the previous chapter.

The beginning of this chapter will just be an overview of the matrix sign function, its history and characteristics. Section 3.2 will be devoted to finding clever ways to use the matrix sign function to do the jobs usually done by QZ, eigenvalue decompositions, and SVDs. The end of the chapter (sections 3.4, 3.5) will then connect this work to the rest of this thesis; it will show how the matrix sign function preserves the special matrix structures in Chapter 2, allowing for very fast and efficient computation of large scale structured control problems, on one or multiple levels.

3.1.1 History

There are many types of algorithms that are called 'structure preserving', but we will only talk about a few which happen to converge very fast, with easily computable bounds, and which solve all of the problems that we are presented with in a common framework. Specifically, most of our discussion in this chapter will be focused on the 'matrix sign function', its calculation, uses, and properties. We will also briefly discuss the matrix sign iteration's discrete domain sibling, the doubling algorithm, in section 3.1.3.

The matrix sign function was first discovered for use in control circa 1971 by J.D. Roberts and made privately available in the form of an internal report, but not published in a publicly available journal until [85](although the matrix sign function was evidently used as early as 1877 for other purposes (see [86] for a history)). Since then it has been used in control and many other fields involving eigenvalue decompositions and matrix roots. For a comprehensive survey up to 1995, see [86], and for more recent work, [87]

3.1.2 Useful Definitions

The matrix sign function is actually a generalization to matrices of the 'sign' function generalized to the open complex plane

$$sign(x) = \begin{cases} -1, & \text{if} \quad \Re(x) < 0 \\ 0, & \text{if} \quad \Re(x) = 0 \\ 1, & \text{if} \quad \Re(x) > 0 \end{cases} \tag{3.1}$$

but restricted to a domain everywhere but on the imaginary axis $\mathcal{D} = \mathbb{C}_- \bigcup \mathbb{C}_+$. Hence the Jordan decomposition definition of the matrix sign function is:

Definition 3.1 *[86] Given matrix X with Jordan decomposition* $X = P \begin{bmatrix} L & 0 \\ 0 & R \end{bmatrix} P^{-1}$

where $\lambda(L) \in \mathbb{C}_-$, $\lambda(R) \in \mathbb{C}_+$,*the matrix sign of* X *is defined as* $sign(X) = P \begin{bmatrix} -I_L & 0 \\ 0 & I_R \end{bmatrix} P^{-1}$ *where* I_L *and* I_R *are the same size as* L *and* R*, respectively.*

Note that this is not an elementwise generalization of a scalar function to matrices, but instead a generalization via the spectrum of the matrix (see e.g. the excellent book [87]). For the scalar sign function, it's clear that $sign(x)$ is infinitely continuously differentiable on $x \in \mathcal{D}$, hence the matrix sign function, $sign(X)$, is a continuous function on the set of matrices $X \in \mathbb{C}^{n\times n}$ with spectrum in $\mathcal{D}$ ([87], Theorem 1.19).

It turns out that the matrix sign function, for matrices X with $\lambda(X) \in \mathbb{C}_- \bigcup \mathbb{C}_+$, can equivalently be defined by a Newton iteration, usually called 'the matrix sign iteration':

Algorithm 3.1 (Sign Iteration [85])

$$\begin{aligned} & Z_0 = X & \\ Z_{k+1} &= \frac{1}{2}(Z_k + Z_k^{-1}) & \textit{for } k = 0, 1, 2, ... \\ sign(X) &= \lim_{k\to\infty} Z_k & \end{aligned}$$

which is fortunate, since Jordan Decompositions are impractical for actual computations. The matrix sign function also has other equivalent definitions based on integral representations, the matrix square root, and Green's functions, but we won't use any of them, so just refer the reader to [86].

3.1.3 Useful Facts

The following facts (1:4 are from [86]), are all easily confirmed using Definition 3.1:

1. X and $sign(X)$ commute: $X sign(X) = sign(X)X$
2. if V is nonsingular: $sign(V^{-1}XV) = V^{-1}sign(X)V$
3. if c is a scalar in $\mathcal{D}$, then $sign(cX) = sign(c)sign(X)$
4. $sign(X^T) = sign(X)^T$
5. $sign(sign(X)) = sign(X)$
6. $\forall k \in \mathbb{N}$: $sign(Z_k) = sign(X)$
7. $X \in \mathbb{R}^{n\times n}$ has $\frac{Trace(sign(X))+n}{2}$ eigenvalues in $\mathbb{C}_+$ and $\frac{Trace(sign(X))-n}{2}$ eigenvalues in $\mathbb{C}_-$

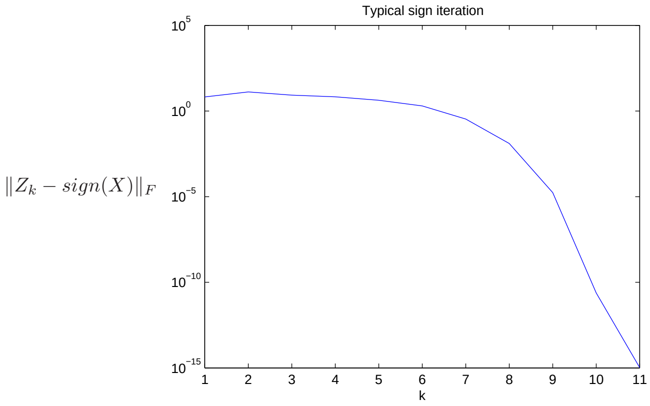


Figure 3.1: Typical sign iteration convergence

Comment 3.1 (Equivalence to Doubling Algorithm) *Another really nice thing about the sign iteration is that, while it may not be immediately obvious, it is equivalent to a doubling algorithm, as first shown by Anderson [88]. Define* $T_k = (Z_k + I)(Z_k - I)^{-1}$*, or alternatively* $Z_k = (T_k + I)(T_k - I)^{-1}$*, where* Z_k *is the iteration in Algorithm 1 above, then it is not too difficult to prove that* $T_{k+1} = T_k^2$*;* T_k *is being squared at each step* k*. Note that the transformation between* Z_k *and* T_k *is a bilinear transformation mapping the imaginary axis to the unit circle. So it is no surprise that using bilinear transformations, one can use the sign iteration to solve discrete time control problems, and doubling algorithms to solve continuous time control problems, but this doesn't seem useful, as the convergence (and hence complexity, in the case of the matrix structures discussed in chapter 2) is the same.*

Another important characteristic of the sign iteration is its quadratic rate of convergence:

Lemma 3.1 *([87], Theorem 5.6) Let* X *have no eigenvalues on the imaginary axis, then* Z_k *converges to* $sign(X)$ *quadratically fast:*

$$\|Z_{k+1} - sign(X)\| \leq \frac{1}{2}\|Z_k^{-1}\|\|Z_k - sign(X)\|^2 \tag{3.2}$$

See e.g. figure 3.1 for a typical convergence curve (actually, convergence can be made even faster [86] by scaling Algorithm 1 using Fact #3 above, but this won't change the character of our results, so we won't discuss it.) While this proves quadratic convergence, it doesn't give us much intuition about what makes Z_k converge quickly or slowly.

Some insight can be given in the following way. Assume $V\Lambda_k V^{-1}$ is an eigenvalue decomposition (this also works with Jordan decompositions, but is easier to understand with a diagonalization) of Z_k, with Λ_k the diagonal matrix of eigenval-

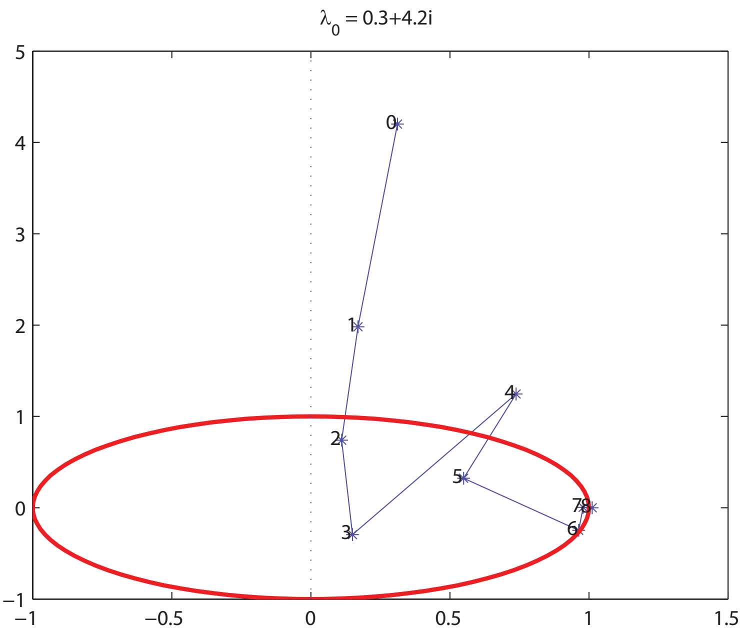


Figure 3.2: A strange trajectory starting at $\lambda_0 = 0.3 + 4.2i$, with unit circle for reference

ues. Then it's easy to see from Algorithm 1 that:

$$\begin{aligned}
\frac{1}{2}(Z_0 + Z_0^{-1}) = \frac{1}{2}(V\Lambda_0 V^{-1} + V\Lambda_0^{-1}V^{-1}) = V\frac{1}{2}(\Lambda_0 + \Lambda_0^{-1})V^{-1} &= V\Lambda_1 V^{-1} = Z_1 \\
\frac{1}{2}(Z_1 + Z_1^{-1}) = \frac{1}{2}(V\Lambda_1 V^{-1} + V\Lambda_1^{-1}V^{-1}) = V\frac{1}{2}(\Lambda_1 + \Lambda_1^{-1})V^{-1} &= V\Lambda_2 V^{-1} = Z_2 \vdots
\end{aligned}$$

Hence V always reduces Z_k to a diagonal matrix, and the iterations on X can equivalently be thought of as iterations on its eigenvalues, each by itself: $\Lambda_{k+1} = \frac{1}{2}(\Lambda_k + \Lambda_k^{-1})$. The convergence of Z_k is thus dominated by the convergence of its eigenvalues, λ to $sign(\lambda)$ through the scalar sign iteration. From Definition 3.1 we know that $\lambda_k \to 1, \forall \lambda_0 \in \mathbb{C}_+$ and $\lambda_k \to -1, \forall \lambda_0 \in \mathbb{C}_-$, but this iterated scalar map can actually have some strange behavior, as shown in figure 3.2. We see that progress is definitely not monotonic, and λ_k comes very close to the imaginary axis before finally going out to $+1$. However, it is possible to analyze the iterative map and figure out some bounds on how many steps it takes λ_k to get within ϵ to $sign(\lambda)$. Using these bounds, one can then bound the convergence of the matrix sign iteration based on its 'worst' eigenvalues, and the conditioning of the generalized eigenvectors V,V^{-1} as follows:

Lemma 3.2 *For a matrix X with no purely imaginary eigenvalues, the number of sign iterations k to reach $\|Z_k - sign(X)\|_2 \le \epsilon$ for $X = PJP^{-1}$ will be $\mathcal{O}(\log_2(\eta)^2 + \log_2(\log_2(\epsilon^{-1} + cond(P))))$, where $\eta = \max_i\{1 + |\Re(\lambda_i)| + |\Re(\lambda_i)|^{-1} + \frac{|\Im(\lambda_i)|}{|\Re(\lambda_i)|}\}$*

Proof: *[28], Lemma 3.5* □

So, using the following assumptions:

- A1: $\exists \beta_1 : \quad \rho(X) < \beta_1 < \infty$
- A2: $\exists \beta_2 : \quad \min_i |\Re(\lambda_i(X))| > \beta_2 > 0$
- A3: $\exists \beta_3 : \quad cond(P) < \beta_3 < \infty$ for $X = PJP^{-1}$

we can use the finite values $\beta_1, \beta_2, \beta_3$ to bound the number of iterations, k, to reach some finite tolerance $\|Z_k - sign(x)\| < \epsilon$. A1 and A2 imply that η in Lemma 3.2 will always be finite, but do not seem very restrictive, as they just imply some analytic continuity around the imaginary axis and a finitely bounded spectrum. A3 just makes up for the difference between the spectral radius and spectral norm, and keep in mind that β_3 can be *very* large without unduly increasing k, for example $\log_2(\log_2(10^{100})) < 9$. From now on the set of matrices that satisfy A1, A2, and A3 will be denoted as $\mathcal{A}$.

Comment 3.2 (When $\lambda(X)$ hits the imaginary axis) *A basic assumption in our definition of the matrix sign is that the spectrum of X does not intersect the imaginary axis. However, it could happen that we unknownly try to perform the sign iteration on a matrix which does not satisfy this condition. In this case, two things can happen: the iteration can go on forever without converging, or one of the Z_k can turn up singular, halting the iteration. The values of λ on the imaginary axis which will lead to a singular Z_k are countable infinite, so we will randomly encounter them with probability zero, but we may hit a Z_k that is numerically singular.*

We also note that as the spectrum of X approaches the imaginary axis, as we saw in Lemma 3.2, the number of iterations necessary for convergence increases, so it can be hard to tell if X has spectrum on the imaginary axis, or just very near it. Anyway, when $\lambda(X)$ has elements very near the imaginary axis, the computational results usually aren't very good(see Chapter 4, section 6), so in practice this difference doesn't really matter.

We've now overviewed the definition of the sign function and some basic facts about it, and gotten some intuition for how the sign iteration converges. We've seen that the convergence is locally quadratic but not monotonic, and that given a few assumptions on our matrices, we can guarantee convergence to some small tolerance ϵ in a finite, and low, number of iterations. Next, we'll show how the sign function can be used in many control problems.

3.2 Applications

In Roberts's original paper [85], he showed how the matrix sign function could be used to solve Lyapunov and Riccati equations, and perform a model order reduction by eliminating the fast modes. In this section we will show some extensions of some of these results, and derive a few new applications.

3.2.1 Matrix Stability and Square Root

The matrix sign can be used to check if the spectrum of a matrix is entirely in one half plane, as follows:

Lemma 3.3 *For some matrix* X, $sign(X) = -I$ *if and only if* $\Re(\lambda(X)) < 0$. $\square$
Proof: *Assume* $sign(X) = -I$, *then* $sign(X) = P^{-1}sign(X)P = sign(P^{-1}XP) = sign(\begin{bmatrix} L & 0 \\ 0 & R \end{bmatrix}) = -I$, *hence* $R \in \mathbb{C}^{0\times 0}$, $L \in \mathbb{C}^{n\times n}$, *and thus* $\Re(\lambda(X)) < 0$. *For necessity, assume* $\Re(\lambda(X)) < 0$, *then* $sign(X) = Psign(\begin{bmatrix} L & 0 \\ 0 & R \end{bmatrix})P^{-1}$, *but* $R \in \mathbb{C}^{0\times 0}$ *and* $L \in \mathbb{C}^{n\times n}$, *so* $sign(X) = P(-I)P^{-1} = -I$ $\square$

Hence to check the stability of a system, $\dot{x} = Ax$, we can simply check if $sign(A) = -I$. In addition, through the Cayley transform $X_c = \mathcal{C}(X_d) \equiv (I + X_d)(-I + X_d)^{-1}$, it is well known that $\rho(X_d) < 1 \Leftrightarrow \Re(\lambda(X_c)) < 0$, so we could also check the spectral radius, or the spectral norm ($\|X_d\| = \rho(X_d X_d^T)$) in a similar way. However, through the bilinear transformation (see comment 3.1) this would be equivalent to the simpler task of repeatedly squaring X_d to see if $X_d^{2^k} \to 0$ (Note: see section 3.3.1 for comments on the numerical viability of such a method, and how to improve on it).

Another computation we'll need, although less often, is the square root:

Lemma 3.4 *Given* A *with no eigenvalues which are both non-positive and real,*

$$sign(\begin{bmatrix} 0 & A \\ I & 0 \end{bmatrix}) = \begin{bmatrix} 0 & A^{1/2} \\ A^{-1/2} & 0 \end{bmatrix} \tag{3.3}$$

Proof: *[89].* $\square$

3.2.2 Lyapunov and Riccati Equations

In most control and analysis applications, we deal with symmetric Lyapunov and Riccati equations, but as we've already seen in the structured matrix inversion methods in Chapter 2, sometimes in this thesis we encounter nonsymmetric problems, and so will state the most general results:

Consider the nonsymmetric Riccati equation:

$$A_2X + XA_1 - XFX + G = 0 \tag{3.4}$$

Lemma 3.5 *([82], Theorem 4) There exists a unique stabilizing solution* $X_p = X$ *to (3.4) if and only if* $In(H) = (n, 0, n)$*, and* X_p *is the unique solution to:*

$$\begin{bmatrix} Z_{12} \\ Z_{22} + I \end{bmatrix} X_p = -\begin{bmatrix} Z_{11} + I \\ Z_{21} \end{bmatrix} \tag{3.5}$$

where $In()$ *is the matrix inertia,* $H = \begin{bmatrix} A_1 & -F \\ -G & -A_2 \end{bmatrix}$*, and* $sign(H) = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}$

The gist of this Lemma is that any time the Riccati equation (3.4) has a stabilizing solution, we can calculate it using the matrix sign function. When $A_1 = A_2^T$ and F and G are symmetric, then H is a Hamiltonian matrix, and the inertia condition reduces to the familiar 'H has no eigenvalues on the imaginary axis'. In this symmetric case, we will refer to (3.5) as the 'matrix sign equations' for the Hamiltonian matrix H and Riccati equation (3.4).

Also, when $F = 0$ and A_1 and A_2 are both stable, the Riccati equation reduces to a Lyapunov equation:

$$A_2X + XA_1 + G = 0 \tag{3.6}$$

which can be solved even easier:

Lemma 3.6 *There exists a unique solution to (3.6):*

$$sign(\begin{bmatrix} A_1 & 0 \\ -G & -A_2 \end{bmatrix}) = \begin{bmatrix} -I & 0 \\ -2X & I \end{bmatrix} \tag{3.7}$$

Proof: *follows from Lemmas 3.5 and 3.3.* □

As for the discrete versions, the nonsymmetric Riccati equation:

$$X = RXW + (Q - RXV)(D - PXV)^{-1}(U - PXW) + Z \tag{3.8}$$

can be solved for a stabilizing X using the doubling algorithm (see comment 3.1) in [90], which for $D, Q, V, U, P = 0$ and $\rho(R), \rho(W) < 1$, reduces to using the 'Smith Squared' iteration [91] to solve the Sylvester equation:

$$RXW - X + Z = 0 \tag{3.9}$$

As discussed above in Comment 3.1, these iterations are just a bilinear transform away from the sign iteration, and share its quadratic rate of convergence.

3.2.3 Block Diagonalization

We can also use the matrix sign function for block diagonalization, which will in turn be used for model order reduction in the next section.

Lemma 3.7 *Given some constant matrix $X \in \mathbb{C}^{c\times c}$ with positive real eigenvalues, where $c \in \mathbb{N}$, and some scalar α such that $X - \alpha I$ has Jordan decomposition*

$$X - \alpha I = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} R & 0 \\ 0 & L \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}^{-1} \tag{3.10}$$

with $L \in \mathbb{C}^{a\times a}$, $R \in \mathbb{C}^{b\times b}$, $\lambda(L) < 0$, $\lambda(R) > 0$. If $0 \notin \lambda(P_{11}) \bigcup \lambda(P_{22})$, then

$$V = sign(X - \alpha I) + \underbrace{\begin{bmatrix} I_R & 0 \\ 0 & -I_L \end{bmatrix}}_{K} \tag{3.11}$$

block diagonalizes X: $VXV^{-1} = \begin{bmatrix} Y_1 & 0 \\ 0 & Y_2 \end{bmatrix}$, where $Y_1 \in \mathbb{C}^{a\times a}$, $Y_2 \in \mathbb{C}^{b\times b}$, $\lambda(Y_1) \bigcup \lambda(Y_2) = \lambda(X)$ with $\lambda(Y_1) > \alpha$ and $\lambda(Y_2) < \alpha$. ***Proof:***

$$\begin{aligned} V &= sign(X - \alpha I) + K = PKP^{-1} + K \\ &= (PK + KP)P^{-1} = 2\begin{bmatrix} P_{11} & 0 \\ 0 & -P_{22} \end{bmatrix} P^{-1} \\ VXV^{-1} &= 2\begin{bmatrix} P_{11} & 0 \\ 0 & -P_{22} \end{bmatrix} P^{-1} X \frac{1}{2} P \begin{bmatrix} P_{11}^{-1} & 0 \\ 0 & -P_{22}^{-1} \end{bmatrix} \\ &= \begin{bmatrix} P_{11}(R + \alpha I)P_{11}^{-1} & 0 \\ 0 & P_{22}(L + \alpha I)P_{22}^{-1} \end{bmatrix} \end{aligned}$$

This result is a simplified but extended(to $X \in \mathbb{C}$) version of that in [92]. □

The interesting and useful part of this result is that we need only compute $sign(X)$ and K, but not the Jordan Decomposition, and while we do not know *a priori* the sizes of I_R and I_L, after computing $sign(X)$ it is easy to find the dimensions of K using fact #7. Hence all of this can be done with the matrix sign.

Actually, even more complicated spectral splitting computations are also possible, e.g. a generalized block Schur decomposition, and an optimal low rank approximation, by employing QR and Cholesky factorizations and multiple computations of the above block diagonalization.

3.2.4 Balanced Model Order Reduction

Roberts, in his original [85], also had model order reduction in mind as an application of the matrix sign function, but 'balanced truncation' hadn't been invented yet, so his method didn't have error bounds, and isn't very useful for us. In the following, we will show how to use the sign iteration for model order reduction with an H_∞ error bound.

Given some system $G = \left[\begin{array}{c|c} A & B \\ \hline C & 0 \end{array}\right]$, we compute $P, Q \succeq 0$ such that (for continuous time, but discrete time works analogously):

$$
\begin{aligned}
AP + PA^* + BB^* &= 0 \qquad (3.12)\\
A^*Q + QA + C^*C &= 0 \qquad (3.13)
\end{aligned}
$$

Since $P, Q \succeq 0$, $R = PQ$ will have real non-negative spectrum $\lambda(R) \in \mathbb{R}_0^+$, and the square roots of $\lambda(R)$ will be the Hankel singular values of the system. If we then pick some $\alpha > 0$ and apply the partial diagonalization procedure described above to calculate V such that

$$
V^{(-1)}RV = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \qquad (3.14)
$$

with $\lambda(\Sigma_1) = \{\lambda(R) > \alpha\}$, $\lambda(\Sigma_2) = \{\lambda(R) < \alpha\}$, and perform the corresponding 'partial balancing' state transformation: $\left[\begin{array}{c|c} V^{-1}AV & V^{-1}B \\ \hline CV & 0 \end{array}\right] = \left[\begin{array}{cc|c} \hat{A}_{11} & \hat{A}_{12} & \hat{B}_1 \\ \hat{A}_{21} & \hat{A}_{22} & \hat{B}_2 \\ \hline \hat{C}_1 & \hat{C}_2 & 0 \end{array}\right]$ then the truncated system $\hat{G} = \left[\begin{array}{c|c} \hat{A}_{11} & \hat{B}_1 \\ \hline \hat{C}_1 & 0 \end{array}\right]$ can be shown to only be a state transformation away from the fully balanced truncated system[93], and thus has the same stability properties and error bound: $\hat{A}_{11}$ is guaranteed stable and $\|G - \hat{G}\|_\infty < 2\sum(\lambda_i(\Sigma_2))^{1/2}$. We note that this error upper bound can be efficiently calculated as $2Tr((\Sigma_2)^{1/2})$ interpreted as the matrix square root, which can also be computed using the matrix sign- see Lemma 3.4. Hence, using the above methods for Lyapunov equations, block diagonalizations, and matrix square roots, we now have a procedure for balanced truncation state space model order reduction, only requiring matrix arithmetic and multiple calculations of the matrix sign. For comparison with the usual balanced truncation method, which uses a full eigenvalue decomposition, see Lemma 2.5.

3.2.5 H_2 and H_∞ Analysis and Controller Synthesis

After showing how to solve Riccati equations and check stability, it isn't much of a stretch to assume that we can solve H_2 and H_∞ control problems, but it's still a bit of a trick to do it without ever using an SVD or rank revealing factorization, so we'll show the formulas here. We'll just show continuous time, although discrete time can also be derived similarly using doubling algorithms.

Analysis of H_2 and H_∞ Performance.

Given some stable $G = \left[\begin{array}{c|c} A & B \\ \hline C & 0 \end{array}\right]$, the H_2 norm can be calculated as follows: $\|G\|_2^2 = Tr(X)$ where $X = C^T W_c C$ and $AW_C + W_C A^T + BB^T = 0$ or $X = BW_o B^T$ and $A^T W_o + W_o A + C^T C = 0$. Since either Lyapunov equation can be solved using the matrix sign as in section 3.2.2, this is no problem. The H_∞ case is a bit trickier:

Lemma 3.8 *For some* $\gamma > 0$, $\lambda(A) \in \mathbb{C}_-$, *we have* $\|D + C(\sigma I - A)^{-1}B\|_\infty < \gamma$ *if and only if* $\|D\|_2 < \gamma$, *and the Riccati equation(see Lemma 3.5 and comments thereafter) corresponding to the Hamiltonian matrix:*

$$H = \begin{bmatrix} A + BR_\gamma D^T C & BR_\gamma B^T \\ -C^T(I + DR_\gamma D^T)C & -(A + BR_\gamma D^T C)^T \end{bmatrix} \tag{3.15}$$

(where $R_\gamma = (\gamma^2 I - D^T D)^{-1}$*) has a stabilizing solution.* ***Proof:*** *It is well known that* $\|D + C(\sigma I - A)^{-1}B\|_\infty < \gamma$ *if and only if* $\|D\|_2 < \gamma$ *and* H *has no eigenvalues on the imaginary axis, which in turn implies that the Riccati equation has a stabilizing solution [60]. Conversely, if* $\|D\|_2 < \gamma$ *then* $BR_\gamma B^T \succeq 0$, *and then the existence of a stabilizing Riccati solution* X *implies that* H *has no eigenvalues on the imaginary axis [94].* ☐

To bound the H_∞ norm of a transfer function, one need not actually compute the solution to the Riccati equation, but just verify that H has no eigenvalues on the imaginary axis. As is suggested in [95] we can use this condition for narrowing bounds on the H_∞ norm of transfer functions by bisection, but we should note that using the sign iteration to find the norm exactly will not work, since as γ approaches $\|D + C(\sigma I - A)^{-1}B\|_\infty$ from above, the eigenvalues of (3.15) will approach the imaginary axis, thus breaking assumption A2, but we may compute upper and lower bounds to within some fine tolerance.

H_2 and H_∞ Synthesis

Setting $\gamma = \infty$ in H_∞ synthesis recovers the H_2 case, so we'll omit this and just focus on the H_∞ case. We will first restate versions of some well known results in H_∞ synthesis, specialized to suit our situation. For a state space system:

$$\Sigma : \begin{bmatrix} \dot{x} \\ z \\ y \end{bmatrix} = \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} x \\ w \\ u \end{bmatrix} \tag{3.16}$$

if we define $\mathcal{K}$ as the set of controllers K such that $\mathcal{F}_l(\Sigma, K)$ is internally stable, then specifically, we would like to find some sub-optimal controller $K \in \mathcal{K}$ such that $\|\mathcal{F}_l(\Sigma, K)\|_\infty < \gamma$ where $\gamma_{opt} + \gamma_{tol} \geq \gamma > \gamma_{opt} = \inf_{K \in \mathcal{K}} \|\mathcal{F}_l(\Sigma, K)\|_\infty$. For some γ and under some general assumptions on Σ, we have the existence result:

Lemma 3.9 *[96] For the system* Σ, *with the assumptions of the 'general problem' [96] of* (A_1, B_2, C_2) *stabilizable and detectable,* D_{12} *full column rank and* D_{21} *full row rank (all of the measurements y are corrupted by noise, and all of the inputs u are present in the cost term, z), and the transfer function matrices of* $y = \Sigma_{(21)}w$ *and* $z = \Sigma_{(12)}u$ *have no invariant zeros on the imaginary axis, there exists a controller* $K \in \mathcal{K}$ *such that* $\|\mathcal{F}_l(\Sigma, K)\|_\infty < \gamma$ *if and only if* $\gamma > \max(\|\hat{D}_{11}\|_2, \|\tilde{D}_{11}\|_2)$, *and*

1. *The Hamiltonian matrices:*

$$H_X = \begin{bmatrix} \hat{A} + \hat{B}_1 \hat{D}_{11}^T \hat{D}_\gamma^{-1} \hat{C}_1 & (\hat{B}_1 \hat{D}_{\gamma T}^{-1} \hat{B}_1^T - \hat{B}_2 \hat{B}_2^T) \\ -(\gamma^2 \hat{C}_1^T \hat{D}_\gamma^{-1} \hat{C}_1) & -(\hat{A} + \hat{B}_1 \hat{D}_{11}^T \hat{D}_\gamma^{-1} \hat{C}_1)^T \end{bmatrix}$$

$$H_Y = \begin{bmatrix} (\tilde{A} + \tilde{B}_1 \tilde{D}_{11}^T \tilde{D}_\gamma^{-1} \tilde{C}_1)^T & \tilde{C}_1^T \tilde{D}_\gamma^{-1} \tilde{C}_1 - \tilde{C}_2^T \tilde{C}_2 \\ -(\gamma^2 \tilde{B}_1 \tilde{D}_{\gamma T}^{-1} \tilde{B}_1^T) & -\tilde{A} - \tilde{B}_1 \tilde{D}_{11}^T \tilde{D}_\gamma^{-1} \tilde{C}_1 \end{bmatrix} \qquad (3.17)$$

   have no eigenvalues on the imaginary axis.

2. *The matrix sign equations ((see Lemma 3.5 and comments thereafter)) of the Hamiltonian matrices H_X and H_Y have unique solutions, $X \succeq 0$, $Y \succeq 0$*

3. $\gamma^{-2}\rho(XY) < 1$.

where the˜ andˆ matrices are defined as:

$$\begin{gathered} \hat{B}_2 = B_2 D_{12}^+, \qquad \hat{A} = A - \hat{B}_2 C_1; \qquad \hat{B}_1 = B_1 - \hat{B}_2 D_{11} \\ \tilde{C}_2 = D_{21}^+ C_2, \qquad \tilde{A} = A - B_1 \tilde{C}_2; \qquad \tilde{C}_1 = C_1 - D_{11} \tilde{C}_2 \\ \hat{C}_1 = (I - D_{12} D_{12}^+) C_1 \qquad \hat{D}_{11} = (I - D_{12} D_{12}^+) D_{11} \\ \tilde{B}_1 = B_1 (I - D_{21}^+ D_{21}) \qquad \tilde{D}_{11} = D_{11} (I - D_{21}^+ D_{21}) \\ \hat{D}_\gamma = (\gamma^2 I - \hat{D}_{11} \hat{D}_{11}^T), \quad \hat{D}_{\gamma T} = (\gamma^2 I - \hat{D}_{11}^T \hat{D}_{11}) \\ \tilde{D}_\gamma = (\gamma^2 I - \tilde{D}_{11} \tilde{D}_{11}^T), \quad \tilde{D}_{\gamma T} = (\gamma^2 I - \tilde{D}_{11}^T \tilde{D}_{11}) \end{gathered}$$

Proof: *This follows from the Riccati equations formulation in [96], and the proof of Lemma 3.5 in [82].* □

As suggested in [97], for some $\gamma > \gamma_{opt}$ we will perturb the Hamiltonians (3.17) by subtracting $\varepsilon I \succ 0$ from the $(2,1)$ blocks, or equivalently, adding εI to the G matrix of the corresponding Riccati equations (3.4), where ε is small enough such that there exist stabilizing solutions: $X_\varepsilon \succ 0$ and $Y_\varepsilon \succ 0$. The scaled inverses of these perturbed solutions will satisfy the H_∞ Riccati Inequalities ([97]), allowing the computation of explicit controllers using LMI results:

Lemma 3.10 *Given some sub-optimal γ and positive definite stabilizing solutions to the perturbed Riccati equations, a stabilizing controller: $K = \left[\begin{array}{c|c} A_K & B_K \\ \hline C_K & D_K \end{array}\right]$ such that $\|\mathcal{F}_l(\Sigma, K)\|_\infty < \gamma$ may be constructed as follows:*

Set $Z = (I - \gamma^{-2} Y_\varepsilon X_\varepsilon)^{-1}$ and define:

$$\begin{aligned} D'_K &= -D_{12}^+ D_{11} D_{21}^+ \\ B'_K &= Z \left[B_2 D'_K + B_1 D_{21}^+ \right] + Z \Omega_B^T \\ C'_K &= -\left[D'_K C_2 + D_{12}^+ C_1 \right] + \Omega_C \qquad (3.18) \\ A'_K &= Z(\hat{A} + \tilde{A} - A + \gamma(B_2 \Omega_C - \Omega_B^T C_2) + \gamma^{-2} Y_\varepsilon (A + B_2 D'_K C_2)^T X_\varepsilon - B_2 D'_K C_2) \\ &\quad - \begin{bmatrix} \tilde{B}_1^T Z^T - D_{21}^T \Omega_B Z^T \\ \gamma^{-1}(C_1 + D_{12} D'_K C_2) Y_\varepsilon Z^T \end{bmatrix}^T \begin{bmatrix} -\gamma I & D_{cl}^T \\ D_{cl} & -\gamma I \end{bmatrix}^{-1} \begin{bmatrix} \gamma^{-1}(B_1 + B_2 D'_K D_{21})^T X_\varepsilon \\ \hat{C}_1 - D_{12} \Omega_C \end{bmatrix} \end{aligned}$$

where

$$\begin{aligned}\Omega_B &= \gamma^{-2}(D_{21}^{+})^T(D_{cl}^T C_1 + (\gamma^2 - D_{cl}^T D_{11})\tilde{C}_2)Y_\varepsilon \\ \Omega_C &= \gamma^{-2}D_{12}^{+}(D_{cl}B_1^T + (D_{cl}D_{11}^T - \gamma^2)\hat{B}_2^T)X_\varepsilon \\ D_{cl} &= D_{11} + D_{12}D'_K D_{21}\end{aligned}$$

and $^+$ *indicates the MP pseudo-inverse. Then the state space matrices of* K *can be computed as:*

$$A_K = A'_K - B'_K D_{22} P' C'_K, \quad B_K = B'_K - B'_K D_{22} P' D'_K$$

$$C_K = P' C'_K, \quad D_K = P' D'_K \tag{3.19}$$

where $P' = (I + D'_K D_{22})^{-1}$ *is assumed to exist.*

Proof: *The controller* K' *formulas follow from the LMI methods in [83] for the central controller and under our regularity assumptions, with a slight change of notation, and picking the free parameters such that* X_ε *and* Y_ε *need not be inverted. The conversion from* K' *to* K *is to reparameterize the controller to the* $D_{22} \neq 0$ *case.*

Comment 3.3 *Note that there is a typographical error in [83] on page 1011, equation (41). The second time that* Θ_B *appears in this equation, it should be transposed.*

□

Note that in presenting our H_∞ controller synthesis routine, we've only employed Riccati solutions, and checked positive definiteness, spectral radius, and spectral norm, all of which can be done using the matrix sign, and basic arithmetic.

It is now apparent why we do not attempt to find an H_∞ *optimal* controller. As γ approaches γ_{opt} from above, it has been shown [96] that the eigenvalues of H_{X_ε}, H_{Y_ε},X_ε, Y_ε, or $\mathcal{C}(\gamma^{-2}X_\varepsilon Y_\varepsilon)$ could asymptotically approach the imaginary axis, thus necessarily breaking assumption A2, and we will no longer be able to use the sign iterations efficiently. However, we can still calculate a controller that is arbitrarily close to achieving the optimal solution.

3.3 Numerical Problems and What to Do about Them

In the previous section we gave routines for checking stability, solving Lyapunov and Riccati equations, performing balanced truncations and H_2 and H_∞ synthesis etc. using only basic arithmetic and the matrix sign function. The obvious goal of this development is to use these analysis and synthesis results on the structured matrices of Chapter 2 (as we will see in section 3.4 of this chapter).

However, everything is not as simple as we have made it out to be, due to numerical problems. While the iteration in Algorithm 1 and its convergence analysis *looks* simple, each step performed in floating point arithmetic introduces rounding errors, and when used with one of the structure preserving arithmetics in Chapter 2, order reductions introduce additional errors.

3.3.1 Numerical Problems

With respect to rounding errors, the sign function has been found to often work just as well as invariant subspace techniques [98], but as expected, larger intermediate approximations often cause larger residual errors.

Such errors can have disastrous consequences on ill-conditioned calculations. For example, for non-Hermitian X with complex $\lambda_0 \in \lambda(X)$ the iterated map $\lambda_{k+1} = \frac{1}{2}(\lambda_k + \lambda_k^{-1})$ that occurs during the sign iteration of X may lead to some very erratic behavior of $\lambda_k \in \lambda(Z_k)$. We know that when λ_0 is not on the imaginary axis, then $\lim_{k\to\infty} \lambda_k \in \{-1, 1\}$, but λ_k can come arbitrarily close to the imaginary axis during its journey to -1 or 1 (remember figure 3.2!). This is unfortunate, since if λ_k jumps over the imaginary axis due to an approximation induced error, the resulting computed $sign(X)$ might be very inaccurate indeed.

For checking matrix stability, an unsuitable low-order approximation could bump an unstable eigenvalue into the left half plane, or vice-versa, falsifying the result. For stable matrices, $\Re(\lambda(X)) < 0$, $X \in \mathcal{A}$, due to the erratic behavior of the spectrum $\lambda(Z_k)$ during the iterations, any approximation, $\tilde{Z}_k$, must be very accurate: $\|Z_k - \tilde{Z}_k\|_2 < \frac{\beta_2}{\beta_1\beta_3}$ to guarantee that no eigenvalues cross the imaginary axis.

A concrete example of this problem can occur when using our suggestion in subsection 3.2.1 to use the Cayley transform on some X_d to use the sign iteration to check if $sign(\mathcal{C}(X_d)) = -I$ and hence $\rho(X_d) < 1$. By using the Cayley transform on the sign iteration as mentioned in comment 3.1, this can be shown to be equivalent to repeatedly squaring $\bar{X}_d$ to check if $\lim_{k\to\infty} \|X_d^k\| = 0$ and hence $\rho(X_d) < 1$, which is well known to be numerically problematic [99].

Hence we cannot ‘solve’ the Lyapunov and Riccati equations; there will always be non-zero residuals, $\mathcal{L}(\tilde{X})$, $\mathcal{R}(\tilde{X})$, the norms of which are not necessarily a good measure of the backwards error[100][101], and the criteria suggested for which are not easily calculable. Furthermore, due to the potential fragility of optimal controllers[102] it is not acceptable to blindly use an approximated controller and assume that it will have the expected closed loop stability and performance; some kind of satisfactory closed loop test is needed, which we will next describe.

3.3.2 What to Do about Them

The solution is to deal with *symmetric* matrices, for which it's much easier to check stability; it is equivalent to negative definiteness. For all of our matrix structures in the previous chapter, this doesn't even require any iterations, just a single check consisting of a low dimensional Riccati equation or recursion corresponding to the Positive Real Lemma (Chapter 2 subsections 2.2.7, 2.3.6, 2.5.3) (Also, for general matrices, the matrix sign iteration is numerically much nicer for symmetric problems, see [103]) Fortunately, we can turn the verification problems of stability and performance analysis into symmetric stability checks.

It is well known[104] for some $E, P \in \mathbb{R}^{N\times N}$, $P = P^T$ with $EP+PE^T \prec 0$, that $\Re(\lambda(E)) < 0$ if and only if $P \succ 0$. Hence one can use the matrix sign function to solve $AP + PA^T + I = 0$ for P using iterative approximations, and if the resulting $\tilde{P} \succ 0$ and $A\tilde{P} + \tilde{P}A^T \prec 0$, which are both symmetric stability problems, then it is guaranteed that $\Re(\lambda(A)) < 0$.

As for the closed loop H_2 performance, after stability has been verified, the Lyapunov equation: $AS + SA^T + BB^T = 0$, may be relaxed to a strict inequality: $\|C(sI-A)^{-1}B\|_2 < \gamma$ if and only if $\exists\Pi$: $Tr(C\Pi C^T) < \gamma^2$ where $A\Pi+\Pi A^T+BB^T \prec 0$. For any such γ, there exists some small $\epsilon > 0$ such that, $Tr(CS_\epsilon C^T) < \gamma^2$, where S_ϵ is the solution to the perturbed Lyapunov equation[105]: $\mathcal{L}(S_\epsilon) = AS_\epsilon + S_\epsilon A^T + BB^T + \epsilon I = 0$. In practice, this means that we can use the matrix sign function to solve such a perturbed Lyapunov equation for some $\tilde{S}_\epsilon$ using a small ϵ, and $(Tr(C\tilde{S}_\epsilon C^T))^{1/2} < \gamma_u$ is an upper bound on the H_2 norm, as long as the residual satisfies $\mathcal{L}(\tilde{S}_\epsilon) \prec \epsilon I$, which is just a symmetric stability problem. Since $S_\epsilon = S+\epsilon P$, then given the residuals of the closed loop stability performance equations, any ϵ such that $\left(A\tilde{S} + \tilde{S}A^T + BB^T\right) + \epsilon\left(A\tilde{P} + \tilde{P}A^T\right) \prec 0$, with corresponding $\tilde{S}_\epsilon = \tilde{S} + \epsilon\tilde{P}$ will work.

All of these arguments and techniques very similarly go through for discrete time (see [106]) and H_∞ performance (see [107]) using Riccati equations. So we will not repeat them here.

In summary, the iterative techniques presented in this chapter *can* cause numerical problems in the analysis, controller synthesis, and model order reduction schemes, but the *a posteriori* stability and performance analysis can be converted to symmetric stability problems, and thus checked reliably.

3.4 Application to this Thesis

The reason we have spent this whole chapter so far investigating the details and applications of the matrix sign function is because it can be performed using the structure preserving arithmetics we built in the previous chapter. To be clearer, for some $\bar{Z}_0 = \bar{X}$ with SSS, $\mathcal{S}$, ATSSS, or $\mathcal{S}_c$, structure, each step of the sign iteration:

$$\begin{aligned}\bar{Z}_1 &= \frac{1}{2}(\bar{Z}_0 + \bar{Z}_0^{-1})\\ \bar{Z}_2 &= \frac{1}{2}(\bar{Z}_1 + \bar{Z}_1^{-1})\\ &\vdots\end{aligned}$$

only uses scalar multiplication, operator addition, and operator inversion. Each of these calculations can be performed in a structure preserving way as developed in Chapter 2; so if $\bar{X} = \bar{Z}_0$ has the structure, so will $\bar{Z}_1$ and $\bar{Z}_{10}$. Furthermore, these steps can be performed using the explicit arithmetic we developed in these chapters for each of these structures. So if $\bar{X}$ has SSS structure, each step of the sign iteration will be $\mathcal{O}(N)$, likewise $\bar{Z}_{10}$ or $\bar{Z}_{20}$ can be computed in $\mathcal{O}(N)$, and due to the fast convergence of the sign iteration, we can thus find a very close approximation to $sign(\bar{X})$ in $\mathcal{O}(N)$. The same idea holds for the other structures.

Hence, we can do all of the applications (stability check, model order reduction, H_∞-synthesis, etc) discussed in this section, very quickly, and preserving the special matrix structure. In the next subsection we will make these results precise.

3.4.1 Structure Preserving Property and Convergence Results

For the Laurent operators with rational symbols of Chapter 2, section 2, we want to prove ‘uniform convergence’, or ‘convergence in the operator norm’ of the sign iteration. This implies that for any $\epsilon > 0$, there is some m such that $\|\bar{Z}_k - sign(\bar{X})\| < \epsilon, \quad \forall k > m$. We have the following results:

Theorem 3.1 *For some Laurent matrix $\bar{X} \in \mathcal{S}$, assume that $\lambda(\bar{X}) \in \mathbb{C}_- \bigcup \mathbb{C}_+$. Then $Z_k(z)$ converges uniformly, $sign(X(z)) \in \mathcal{L}_\infty(\mathbb{T})$ is continuous in $z \in \mathbb{T}$, and $\bar{Z}_\infty = sign(\bar{X})$ is a Laurent matrix.* ***Proof:*** *Since the scalar sign function, $sign(x)$, is infinitely continuously differentiable on $x \in \mathcal{D}$, Theorem 1.19 of [87] states that the matrix sign function, $sign(X)$, is a continuous function on the set of matrices $X \in \mathbb{C}^{n\times n}$ with spectrum in $\mathcal{D}$. Since $\bar{X} \in \mathcal{S}$, hence $X(z)$ is continuous on $z \in \mathbb{T}$, and since $\lambda(\bar{X}) \in \mathcal{D}$, thus $X(\mathbb{T})$ is in the set of matrices with spectrum in $\mathcal{D}$, i.e. the set of matrices for which $sign()$ is continuous. So $sign(X(z))$ is a continuous function of a continuous function of $z \in \mathbb{T}$, and hence is itself continuous on $z \in \mathbb{T}$.*

Also, we need a general result:

Lemma 3.11 *Assume $\bar{Y}$ is a bounded operator and $\rho(\bar{Y}) < 1$. Then $\lim_{n\to\infty} \|\bar{Y}^n\| = 0$* ***Proof:*** *Follows from Gelfand's formula (see e.g. [108]): $\rho(\bar{Y}) = \lim_{n\to\infty} \|\bar{Y}^n\|^{\frac{1}{n}}$*
□

Now we make a change of variables and iteration. For all $k \in \mathbb{N}$, define [87]:

$$G_k(z) = (Z_k(z) - sign(X(z)))(Z_k(z) + sign(X(z)))^{-1} \tag{3.20}$$

Since both $Z_0(z)$ and $sign(X(z))$ are continuous thus bounded over $z \in \mathbb{T}$, $G_0(z)$ is also continuous and bounded over $z \in \mathbb{T}$. Since the spectrum of $Z_0(z)$ doesn't touch the imaginary axis, the spectrum of $G_0(z)$ is strictly inside the unit circle. Furthermore, it can be shown that $G_{k+1}(z) = (G_k(z))^2$.

Lemma 3.12 $\lim_{k\to\infty}\|G_k(z)\|_\infty = 0$. ***Proof:*** *Since $G_0(z)$ is continuous, its spectrum is compact and $\rho(G_0(z)) < 1$. Since $G_0(z)$ is bounded, $\lim_{p\to\infty}\|(G_0(z))^p\|_\infty = 0$ by Lemma 3.11, and $G_k(z)$, $k \in \mathbb{N}$ is a subsequence of $(G_0(z))^p$, $p \in \mathbb{N}$ and thus also converges uniformly to 0.* □

Lemma 3.13 $Z_k(z)$ *converges uniformly to $sign(X(z))$.* ***Proof:*** *From (3.20) it directly follows that $Z_k(z) - sign(X(z)) = 2(I - G_k(z))^{-1}G_k(z)sign(X(z))$. If we have gotten to some k such that $\|G_k(z)\| < 1$ (no problem, by Lemma 3.12), then:*

$$\|Z_k(z) - sign(X(z))\| \quad \leq \quad 2(1 - \|G_k(z)\|)^{-1}\|G_k(z)\|\|sign(X(z))\|$$

So if I want some m such that $\|Z_k(z) - Z_\infty(z)\| < \epsilon, \forall k > m$, it is good enough to just find some m such that

$$\forall k > m: \qquad \|G_k(z)\| < \frac{\epsilon}{2\|sign(X(z))\| + \epsilon} \tag{3.21}$$

$sign(X(z))$ is continuous on the compact set $z \in \mathbb{T}$ and hence bounded, so $\|sign(X(z))\|$ is finite, hence the righthand side of (3.21) is positive. By Lemma 3.12, there exists such an m, completing the proof. □

Now to get back to our Laurent matrices, Since $\mathbb{T}$ is compact, then $sign(X(z)) \in \mathcal{L}_\infty(\mathbb{T})$ (a continuous function is always bounded on a compact set), and also $sign(X(\mathbb{T}))$ is compact, and thus the symbol of a Laurent matrix [51]. So $sign(\bar{X})$ is a bounded Laurent matrix. □

So for some $\bar{X} \in \mathcal{S}$ with no spectrum on the imaginary axis, $sign(\bar{X})$ will be a Laurent operator, but we note that it will **not** always have a rational symbol, since the space of rational functions is not complete. However, $\bar{Z}_k \in \mathcal{S}, \quad \forall k < \infty$ and thus we can approximate $sign(\bar{X})$ arbitrarily close in $\mathcal{S}$, and the approximation generated by the halted sign iteration converges uniformly, and locally quadratically fast to $sign(\bar{X})$.

Since the sign iteration can be used to solve Riccati and Lyapunov equations etc., as shown in section 3.2, this essentially means that we can solve $\mathcal{S}$ realization Lyapunov and Riccati equations, order reductions, H_2 and H_∞ synthesis problems, etc. to arbitrary accuracy, and our result will also have the $\mathcal{S}$ realization structure. See appendix 3.7 for details.

However, matrices with spectrums very near the imaginary axis will take longer to converge than those with spectrums very near ±1. Hence, one could construct a sequence of SSS or ATSSS matrices growing in size N with spectrums that asymptotically approach the imaginary axis. Calculating the matrix sign to some accuracy ϵ thus would not be $\mathcal{O}(N)$ since the number of sign iterations necessary would increase with N. We will thus assume that the set of matrices that we will work with are all in $\mathcal{A}$ for some specific β_1, β_2, and β_3 (although they may be arbitrarily large or arbitrarily small). This assumption basically guarantees that all matrices in this class can have their matrix sign computed to ϵ in less than some fixed $k_c < \infty$ iterations (We will see some physical examples of this in Chapters 4 and 9). When this is the case, we can say:

Lemma 3.14 *For the set of SSS matrices* $\overline{X} \in \mathcal{A} \quad \forall N \in \mathbb{N}$*, an SSS approximation,* $\tilde{S}$ *to* $sign(\overline{X})$ *can be calculated to within some prespecified positive tolerance* $\epsilon > \|\tilde{S} - sign(\overline{X})\|_2$ *in* $\mathcal{O}(N)$.□

Since for SSS matrices we can calculate $sign(\bar{X})$ to arbitrary accuracy in $\mathcal{O}(N)$, we can hence perform all of the other applications in section 3.2 in $\mathcal{O}(N)$ also, from model order reduction to H_∞ synthesis (assuming relevant matrices are in $\mathcal{A}$). As for ATSSS matrices, since the number of k_c iterations necessary to get within ϵ for some set satisfying $\mathcal{A}$ is independent of how each iteration is calculated, we have the corresponding result:

Lemma 3.15 *For the set of ATSSS matrices* $\overline{X} \in \mathcal{A} \quad \forall N \in \mathbb{N}$*, with* $N >> N_T + N_B \in \mathcal{O}(1)$*, an ATSSS approximation,* $\tilde{S}$*, to* $sign(\overline{X})$ *can be calculated to within some prespecified positive tolerance* $\epsilon > \|\tilde{S} - sign(\overline{X})\|_2$ *in* $\mathcal{O}(1)$. ***Proof:*** *Obviously, for this set of matrices, less than* k_c *iterations are needed to converge to within* ϵ*. Even though the* N_T *and* N_B *may grow during the sign iteration, there are only a finite number of iterations, so they can only grow finitely, and there will always be an* $N \in \mathbb{N}$ *above which their largest values are* $N_T, N_B \ll N$*, and hence independent of* N*, for all iterations* k*.* □

Since for ATSSS matrices we can calculate $sign(\bar{X})$ to arbitrary accuracy in $\mathcal{O}(1)$, we can hence perform all of the other applications in sections 3.2 in $\mathcal{O}(1)$ (assuming again that the relevant matrices are in $\mathcal{A}$).

As for the transfer functions over the extended imaginary axis:

Lemma 3.16 *For some transfer function on the imaginary axis,* $\bar{X} \in \mathcal{S}_c$*, assume that* $\lambda(\bar{X})$ *does not touch the imaginary axis. Then* $Z_k(s)$ *converges uniformly on the extended imaginary axis, and* $sign(X(s))$ *is continuous and bounded* ***Proof:*** $X(s)$ *is continuous on the extended imaginary axis, which is compact, and hence the proof is formally identical to that for the sign iteration on transfer functions on the unit circle above.* □

Just as for the Laurent matrices, we note that $Z_\infty(s) = sign(X(s))$ will **not** always be rational, since the space of rational functions is not complete, but we can approximate $sign(\bar{X})$ arbitrarily close in $\mathcal{S}_c$, and the approximation generated by the halted sign iteration converges locally quadratically fast to $sign(\bar{X})$ in operator norm, so in practice this is not a problem. Also as with the L_r matrices (see appendix 3.7, the proofs for $\mathcal{S}_c$ realizations are similar), this result essentially shows that we can solve $\mathcal{S}_c$ realization Riccati equations etc. to arbitrary accuracy, and our result will also have the $\mathcal{S}_c$ realization structure.

3.4.2 Summary

So, we've seen that for a broad set of SSS and ATSSS matrices, the matrix sign can be calculated to some ϵ precision in less than an *a priori* bounded number of iterations, k_c, resulting in $\mathcal{O}(N)$ and $\mathcal{O}(1)$ computational complexities, respectively.

As for the Laurent matrices with rational symbols and rational transfer functions on the imaginary axis, if the matrix sign exists, then the iteration converges uniformly, meaning also that we can get to within some ϵ precision in less than an *a priori* bounded number of iterations, k_c.

Hence, using the sign iteration and doubling algorithms, we can solve Lyapunov and Riccati equations, order reductions, H_2 and H_∞ synthesis problems, etc. for systems with realizations of SSS, ATSSS, $\mathcal{S}$ and $\mathcal{S}_c$ realizations. The SSS and ATSSS computations are $\mathcal{O}(N)$ and $\mathcal{O}(1)$ computational complexities, respectively. Furthermore, the results will preserve the structure, whichever it may be.

3.5 Multi-level Structured Operators

While the above results are nice, in many applications in nature, as we will discuss in the second part of the thesis, problems will not be on one level, but in two or more(see e.g. the school of fish in Chapter 1). Fortunately, our results generalize to this case as follows. We will just discuss an example of multi-level Laurent matrices, but the same could be done for operators on direct products of all kinds of different spaces; e.g. a Laurent matrix with blocks that have SSS structure, whose generators are $\mathcal{S}_c$ operators.

3.5.1 From 1-D to 2-D to n-D

As an example we will consider 2-level Laurent operators with rational symbols. Such matrices are block Laurent, where the blocks have Laurent structure too. See figure 3.3 for an example of a truncation of such a matrix. Such operators have symbols:

$$\mathcal{F}_z \bar{\bar{F}} \mathcal{F}_z^{-1} = \bar{F}(z) = \bar{B}(z\bar{I} - \bar{W})^{-1}\bar{C} + \bar{A} + \bar{E}(z^{-1}\bar{I} - \bar{R})^{-1}\bar{G}$$

bounded on $z \in \mathbb{T}$ that also have the L_r structure, and which, when Fourier transformed again, in a different variable, have symbols that take the form of multivariable transfer functions:

$$\mathcal{F}_\zeta \mathcal{F}_z \bar{\bar{F}} \mathcal{F}_z^{-1} \mathcal{F}_\zeta^{-1} = F(z,\zeta) = B(\zeta)(zI - W(\zeta))^{-1}C(\zeta) + A(\zeta) + E(\zeta)(z^{-1}I - R(\zeta))^{-1}G(\zeta) \tag{3.22}$$

Where $B(\zeta), W(\zeta), C(\zeta), A(\zeta), E(\zeta), R(\zeta), G(\zeta) \in \mathcal{RL}_\infty(\mathbb{T})$ are the symbols of $\bar{B}, \bar{W}$, $\bar{C}, \bar{A}, \bar{E}, \bar{R}, \bar{G}$ respectively.

For multilevel L_r matrices, we will use $\mathcal{S}^n$ to indicate a stable realization in multiple levels, such as in (3.22) we will use $\bar{\bar{F}} \in \mathcal{S}^2$ to indicate that $\bar{\bar{F}} = \mathcal{S}(\bar{B}, \bar{W}, \bar{C}, \bar{A}, \bar{E}, \bar{R}, \bar{G})$ with each $\bar{B}, \bar{W}, \bar{C}, \bar{A}, \bar{E}, \bar{R}, \bar{G} \in \mathcal{S}$ and $\rho(\bar{R}), \rho(\bar{W}) < 1$.

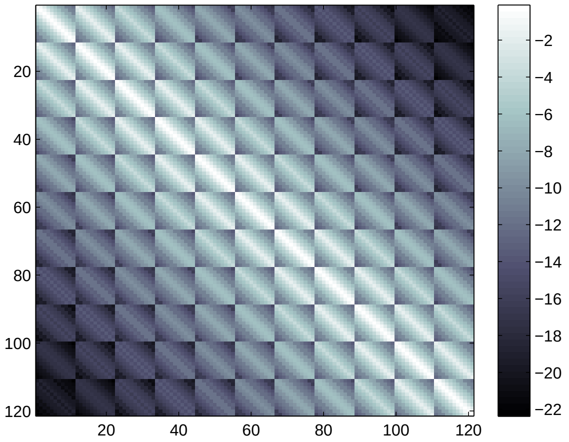


Figure 3.3: Note that this 2-level Laurent matrix is truncated twice, once in the overall Laurent structure, and once to make each block finite!

Lemma 3.17 *All L_r operators in $\mathcal{S}$, $\mathcal{S}^2$, $\mathcal{S}^3$ etc, with finitely many levels, are in the Wiener Algebra, and hence are bounded on l_∞.* ***Proof:*** *Any Laurent matrix in $\mathcal{S}^n$ will have a multivariable rational transfer function, holomorphic with respect to each variable independently, everywhere on the unit circle. The main result of [109] then states that the row of such a Laurent matrix is absolutely summable, hence the Laurent matrix is in the Wiener Algebra and is bounded on l_∞.* $\square$

Comment 3.4 *As an interesting aside, if we remember that the spectrums of L_r matrices with $\mathcal{S}$ realizations were continuous mappings of the unit circle onto the complex plane, generating weird closed curves, it makes sense that the spectrums of 2-level L_r matrices are weird warpings of a toroid, or rather the planar projection thereof. See figure 3.4 for an example. Note that since the spectrums of such operators are not points or lines, but* areas, *in practice it is not uncommon for a randomly generated $\mathcal{S}^2$ realization to be singular.*

We now have all of the computational tools in 1-D to build an arithmetic in 2-D, and by induction n-D. We first point out that all of the operator sign, Lyapunov, Riccati, and model-order reduction computations described in section 3.2 just consist of multiple applications of the basic arithmetic in Chapter 2, so we will only describe how to perform these +, ×, order reduction, $^{-1}$, norm, and permutation computations in 2-D, and the rest follows by induction.

Addition and multiplication of 2-D operators $\bar{\bar{X}}$, $\bar{\bar{Y}} \in \mathcal{S}^2$ can be performed almost exactly as in the 1-D versions in Chapter 2, just adding an extra $\overline{bar}$ to each term. They will require the addition and multiplication of 1-D operators, and the solution of Sylvester equations in 1-D operators (subsection 3.2.2). As a concrete example, we will show the explicit form for multiplication of 2-Level Laurent matrices:

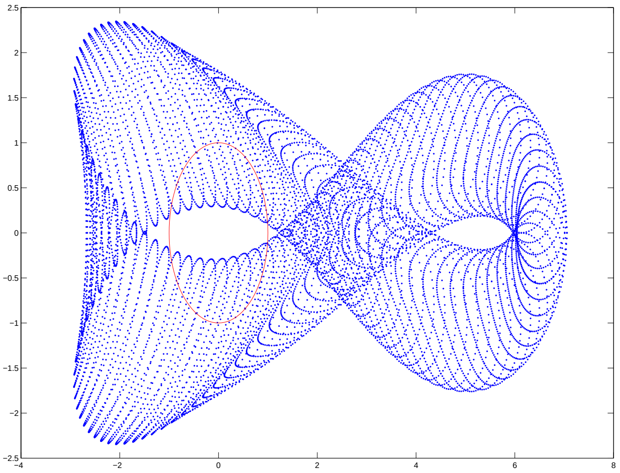


Figure 3.4: sampled spectrum of an $\mathcal{S}^2$ matrix, with unit circle for reference

Lemma 3.18 *Given:* $\bar{\bar{X}} = \mathcal{S}\{\bar P_X, \bar R_X, \bar Q_X, \bar D_X, \bar U_X, \bar W_X, \bar V_X\}$ *and* $\bar{\bar{Y}} = \mathcal{S}\{\bar P_Y, \bar R_Y, \bar Q_Y, \bar D_Y, \bar U_Y, \bar W_Y, \bar V_Y\}$*, then* $\bar{\bar{X}}\bar{\bar{Y}} = \bar{\bar{Z}} = \mathcal{S}\{\bar P_Z, \bar R_Z, \bar Q_Z, \bar D_Z, \bar U_Z, \bar W_Z, \bar V_Z\}$ *where*

$$\begin{aligned}
\bar D_Z &= \bar D_X \bar D_Y + \bar P_X \bar S \bar V_Y + \bar U_X \bar T \bar Q_Y \\
\bar P_Z &= \begin{bmatrix} \bar D_X \bar P_Y + \bar U_X \bar T \bar R_Y & \bar P_X \end{bmatrix} \Pi_R, \qquad \bar U_Z = \begin{bmatrix} \bar D_X \bar U_Y + \bar P_X \bar S \bar W_Y & \bar U_X \end{bmatrix} \Pi_R, \\
\bar R_Z &= \Pi_L \begin{bmatrix} \bar R_Y & 0 \\ \bar Q_X \bar P_Y & \bar R_X \end{bmatrix} \Pi_R, \quad \bar W_Z = \Pi_L \begin{bmatrix} \bar W_Y & 0 \\ \bar V_X \bar U_Y & \bar W_X \end{bmatrix} \Pi_R \\
\bar Q_Z &= \Pi_L \begin{bmatrix} \bar Q_Y \\ \bar Q_X \bar D_Y + \bar R_X \bar S \bar V_Y \end{bmatrix}, \bar V_Z = \Pi_L \begin{bmatrix} \bar V_Y \\ \bar V_X \bar D_Y + \bar W_X \bar T \bar Q_Y \end{bmatrix}
\end{aligned}$$

where $\bar S$ *and* $\bar T$ *are the unique solutions to the Sylvester equations:*

$$\bar S = \bar R_X \bar S \bar W_Y + \bar Q_X \bar U_Y, \quad \bar T = \bar W_X \bar T \bar R_Y + \bar V_X \bar P_Y$$

which can be solved using an $\mathcal{S}$ *structure preserving doubling algorithm, as discussed in subsection 3.2.2, and* Π_L *and* Π_R *are the permutation operators given in Lemma 2.6 for turning Laurent block matrices into block Laurent matrices.*

Note how closely this Lemma follows the form of Lemma 4 in Chapter 2; the only difference is the additional permutations Π_L and Π_R. The other arithmetic operations can similarly be extended, as follows. Order reduction of 2-D L_r operators with $\mathcal{S}^2$ realizations can be performed using the order reduction for 1-D L_r operators as just described in section 3.2.4. Calculating the inverse and norm of 2-D Laurent operators also follows the same pattern as in Lemma 2.8 and subsection 2.2.8 respectively; they require the same operations as in addition and multiplication, with the supplement of solving discrete domain 1-D Laurent operator Riccati equations, which is possible using the techniques in subsection 3.2.2 Finally, the permutation of a block of 2-D Laurent operators into a 2-D block Laurent operator, and vice versa, follows the 1-D version in subsection 2.6 of Chapter 2 exactly, with the addition of extra 1-D permutations on the lower level.

To move on to 3-D $\mathcal{S}^3$ arithmetic, we simply perform the same induction using the 2-D arithmetic; the arithmetic operations being performed in a hierarchical manner, down the spatial dimensions. Hence, in theory at least, we can now efficiently perform arithmetic computations on n-D Laurent operators, and by using the tools described in sections 3.2 extended to this dimension, synthesize controllers with arbitrarily small loss of optimality, and calculate closed loop stability and performance.

In the case of multilevel SSS and ATSSS matrices we also accomplish the pragmatically more important result of extending the $\mathcal{O}(N)$ and $\mathcal{O}(1)$ computational complexity to multiple dimensions in similar complexity; e.g. $\mathcal{O}(NM)$ complexity for a 2-level SSS matrix with N and M blocks and $\mathcal{O}(1)$ if the matrix is additionally 'almost Toeplitz' on each level. For large problems e.g. N,M$\sim$ 1000, This provides an important boost in computational efficiency over traditional methods for such matrices, which would be $\mathcal{O}(N^3M^3)$ complexity.

3.5.2 Dealing with Approximation Errors on Multiple Levels

However, in practice there are additional problems caused by the spatial order reductions. In 1-D, we repeatedly perform spatial order reductions on the Laurent operator symbols to keep the complexity low and the eventual implementation of the controller simple, but this introduces small errors in the iterative computations. For multiple dimensional operators and calculations, the situation is even more complicated, as the order reductions are being performed on more than one level, and errors on the lower level are being introduced even when evaluating the accuracy on the upper level, e.g. in a norm calculation.

In addition, for multiple spatial dimensions we can no longer hope to find exact rational solutions using our $\mathcal{S}$ arithmetic. For example, for some invertible $\bar{X} \in \mathcal{S}$, we know that we can always find a $\bar{X}^{-1} = \bar{F} \in \mathcal{S}$ where the rational order of $F(z)$ is no greater than three times that of $X(z)$. However, for multidimensional operators, for some rational $X(z, \zeta)$ as in (3.22), our procedure as outlined above will calculate a rational $\tilde{X}^{-1}(z, \zeta)$, but it will be an approximation of some irrational $X^{-1}(z, \zeta)$, since a 1-D Riccati equation with an irrational solution is solved in the process. This will often necessitate higher order approximations in these multiple dimensional problems (as we will see in Chapter 8) and the spatial order reductions will correspondingly cause larger errors.

However, as in section 3.3.2, symmetry saves the day *a posteriori*. The verification of closed loop stability and ϵ sub optimal performance in n-D can be reduced to checking the strict positive definiteness of n-D Hermitian L_r operators. This in turn can be performed by similarly finding strictly positive definite (n-1)-D L_r operator certificates (a solution to a Riccati equation and the Riccati residual) using the Positive Real Lemma (as in Lemma 2.9). Checking the positive definiteness of these (n-1)-D L_r operator certificates can then in turn be performed by finding (n-2)-D L_r operator certificates using the Positive Real Lemma again, and hence on down to the 0 spatial dimensional problem, where the Positive Real Lemma uses a Riccati matrix equation in finite dimensions, which can be solved using Matlab.

3.6 Conclusion

So, in this chapter we overviewed the matrix sign function and the matrix sign iteration for calculating it, and showed that it can be used for many things, from model order reduction to H_∞ synthesis. We also showed that when applying the matrix sign iteration to our different arithmetics from Chapter 2, it preserves the structure, and allows for very fast computations; $\mathcal{O}(N)$ and $\mathcal{O}(1)$ for SSS and ATSSS matrices respectively, hence allowing extremely efficient controller synthesis for large systems with this structure. We then showed how these techniques allowed us to 'lift' these results to higher dimensions, by using the solutions to 1-level Lyapunov and Riccati equations to generate an arithmetic of 2-level operators, and so on.

The rest of this thesis will be devoted to simply applying these results to a variety of different examples in different ways, to show how they work, and how diverse the potential applications are.

3.7 Appendix: Special Control Related Results for Laurent Matrices

Unlike for the SSS and ATSSS matrices, as we go to infinite dimensions, things can get a little bit weird (see examples in Chapter 2), so it's necessary that we first prove some things about closedness, uniqueness, and error bounds regarding the use of the sign iteration for solving L_r operator control problems. We will only show the results which are a little bit tricky or non-obvious: stability analysis, Riccati and Lyapunov solutions, H_2 performance, and balanced truncation model order reduction. All matrices with a $\bar{bar}$ in this appendix will be assumed to be in $\mathcal{S}$.

Lemma 3.19 *For some* $\overline{X} \in \mathcal{S}$, *sign*$(\overline{X}) = -\bar{I}$ *if and only if* $\Re(\lambda(\overline{X})) < 0$. □
Proof: *Since the Fourier transform is an isomorphism, we can say that* $sign(\overline{X}) = -\bar{I}$ *if and only if* $\mathcal{F}sign(\overline{X})\mathcal{F}^{-1} = \mathcal{F}(-\bar{I})\mathcal{F}^{-1}$, *but obviously* $\mathcal{F}(-\bar{I})\mathcal{F}^{-1} = -I$ *and*

$$\mathcal{F}sign(\overline{X})\mathcal{F}^{-1} = sign(\mathcal{F}\overline{X}\mathcal{F}^{-1}) = sign(X(z)), z \in \mathbb{T} \tag{3.23}$$

from Lemma 3.3 we know that in finite dimensions, at every $z_0 \in \mathbb{T}$, $sign(X(z_0)) = -I \Leftrightarrow \Re(\lambda(X(z_0))) < 0$. *Since* $\lambda(\overline{X}) = \lambda(X(\mathbb{T}))$ *this completes the proof.* □

Now for the Riccati equation:

$$\bar{X}\bar{A} + \bar{A}^*\bar{X} + \bar{Q} - \bar{X}\bar{R}\bar{X} = 0 \tag{3.24}$$

where $\bar{Q} = \bar{Q}^*$, $\bar{R} = \bar{R}^*$, which can be solved, just as in finite dimensions, by calculating the sign of a Hamiltonian operator:

$$sign(H) = sign(\begin{bmatrix} \bar{A} & -\bar{R} \\ -\bar{Q} & -\bar{A}^* \end{bmatrix}) = \begin{bmatrix} \bar{S}_{11} & \bar{S}_{12} \\ \bar{S}_{21} & \bar{S}_{22} \end{bmatrix} \tag{3.25}$$

and solving the linear system of equations:

$$\begin{bmatrix} \bar{S}_{12} \\ \bar{S}_{22} + I \end{bmatrix} \bar{X} = - \begin{bmatrix} \bar{S}_{11} + I \\ \bar{S}_{21} \end{bmatrix} \tag{3.26}$$

Note that in calculating $sign(\begin{bmatrix} \bar{A} & -\bar{R} \\ -\bar{Q} & -\bar{A}^* \end{bmatrix})$ here, in practice we will avoid block arithmetic, instead using Lemma 2.6 to permute H into $\bar{H}$, then calculating $sign(\bar{H})$ using algorithm 1, then using 2.7 to permute $sign(\bar{H})$ back into block form; $\begin{bmatrix} \bar{S}_{11} & \bar{S}_{12} \\ \bar{S}_{21} & \bar{S}_{22} \end{bmatrix}$.

In fact, as in the finite constant matrix case, this procedure will always work if (3.24) has a stabilizing solution:

Lemma 3.20 *The Riccati equation (3.24) has a unique bounded exponentially stabilizing solution, $\bar{X}$, if and only if $\begin{bmatrix} \bar{A} & -\bar{R} \\ -\bar{Q} & -\bar{A}^* \end{bmatrix}$ has no spectrum on the imaginary axis and $\begin{bmatrix} \bar{S}_{12} \\ \bar{S}_{22} + I \end{bmatrix}$ has a left inverse.* ***Proof:*** *This proof works basically in the same way as in Lemma 3.19, by using the Fourier transform to reduce everything to finite dimensions, parametrized on the unit circle, and then using the corresponding finite dimensional result [82] at each $z \in \mathbb{T}$. The boundedness of the solution comes from Theorem 3.1.* □

For $\overline{R} = 0$, the above Lemma specializes to:

Lemma 3.21 *The Lyapunov equation $\bar{X}\bar{A} + \bar{A}^*\bar{X} + \bar{Q} = 0$ has a unique solution if and only if $\Re(\lambda(\bar{A})) < 0$, in which case*

$$sign(\begin{bmatrix} \bar{A} & 0 \\ -\bar{Q} & -\bar{A}^* \end{bmatrix}) = \begin{bmatrix} -I & 0 \\ 2\bar{X} & I \end{bmatrix} \tag{3.27}$$

Furthermore, $\bar{X} \in \mathcal{S}$. ***Proof:*** *This follows from Lemmas 3.20 and 3.19. The fact that $\bar{X} \in \mathcal{S}$, unlike for Riccati solutions, follows from the other, computationally unattractive, method for solving rational Lyapunov equations; vectorizing $X(z)A(z) + A^*(z)X(z) + Q(z) = 0$ and solving the resulting linear system for $X(z)$, which will be of high but finite rational order.* □

The H_2 performance calculation of a system $\overline{G}(s) = \left[\begin{array}{c|c} \overline{A} & \overline{B} \\ \hline \overline{C} & 0 \end{array}\right]$ is also tricky, since the matrix trace of an L_r matrix will be either 0 or infinite, hence we need a slightly different definition. The H_2 spatiotemporal norm (in continuous time and discrete space) is defined as [26]

$$\|G\|_2^2 = (\frac{1}{2\pi})^2 \int_0^{2\pi} \int_{-\infty}^{\infty} Tr[G(e^{j\theta}, j\omega)G(e^{j\theta}, j\omega)^*]d\omega d\theta \tag{3.28}$$

We can calculate the integral on the imaginary axis using the traditional finite dimensional result (see e.g. [105]), at each θ, to get:

$$= \frac{1}{2\pi} \int_0^{2\pi} Tr[X(e^{j\theta})]d\theta \tag{3.29}$$

where either $\bar{X} = \bar{C}^*\bar{W}_c\bar{C}$ with $\bar{A}\bar{W}_c + \bar{W}_c\bar{A}^* = -\bar{B}\bar{B}^*$ or $\bar{X} = \bar{B}\bar{W}_o\bar{B}^*$ with $\bar{A}^*\bar{W}_o + \bar{W}_o\bar{A} = -\bar{C}^*\bar{C}$, which can be efficiently calculated to arbitrary accuracy using the L_r operator sign function as in Lemma 3.21. As for the remaining integral on θ, it reduces simply in Lemma 2.12.

The last component in need of special consideration is structure preserving model order reduction. Given some system of $\mathcal{S}$ realizations: $\overline{G}(s) = \left[\begin{array}{c|c} \overline{A} & \overline{B} \\ \hline \overline{C} & 0 \end{array}\right]$, we would like to calculate an $\mathcal{S}$ realization approximation, $\hat{\overline{G}}(s) = \left[\begin{array}{c|c} \hat{\overline{A}} & \hat{\overline{B}} \\ \hline \hat{\overline{C}} & 0 \end{array}\right]$ which is 'near' to $\overline{G}$: $\|\overline{G} - \hat{\overline{G}}\| < \gamma$ and is guaranteed stable, but has a smaller state dimension: $dim(A(z)) > dim(\hat{A}(z))$. As in the previous few subsections, we can use an extension of finite dimensional results.

Lemma 3.22 *Suppose that for stable* $\overline{G}(s) = \left[\begin{array}{c|c} \overline{A} & \overline{B} \\ \hline \overline{C} & 0 \end{array}\right]$ *we have some solutions* $\overline{P} \succeq 0$, $\overline{Q} \succeq 0$ *to the operator Lyapunov equations:*

$$\bar{A}\bar{P}\bar{A}^* - \bar{P} + \bar{B}\bar{B}^* = 0, \qquad \bar{A}^*\bar{Q}\bar{A} - \bar{Q} + \bar{C}^*\bar{C} = 0$$

such that there is a 'shuffle' permutation operator Π *as in Chapter 2 Lemma 7 such that*

$$\Pi\bar{P}\bar{Q}\Pi^* = \begin{bmatrix} \bar{\Sigma}_1 & 0 \\ 0 & \bar{\Sigma}_2 \end{bmatrix} \tag{3.30}$$

where $\lambda(\bar{\Sigma}_1) > \lambda(\bar{\Sigma}_2)$. *Then, if we similarly permute the system:* $\left[\begin{array}{c|c} \Pi^*\bar{A}\Pi & \Pi^*\bar{B} \\ \hline C\Pi & 0 \end{array}\right] = \left[\begin{array}{cc|c} \hat{\bar{A}}_{11} & \hat{\bar{A}}_{12} & \hat{\bar{B}}_1 \\ \hat{\bar{A}}_{21} & \hat{\bar{A}}_{22} & \hat{\bar{B}}_2 \\ \hline \hat{\bar{C}}_1 & \hat{\bar{C}}_2 & 0 \end{array}\right]$ *then the truncated system* $\hat{\bar{G}} = \left[\begin{array}{c|c} \hat{\bar{A}}_{11} & \hat{\bar{B}}_1 \\ \hline \hat{\bar{C}}_1 & 0 \end{array}\right]$ *will also be stable, and will satisfy the error bound*

$$\|\overline{G}(s) - \hat{\overline{G}}(s)\|_\infty \leq \sup_{z_0 \in \mathbb{T}} 2 \sum_{i=1}^{k} (\lambda_i(\Sigma_2(z_0)))^{1/2} \tag{3.31}$$

where $\Sigma_2(z_0) \in \mathbb{C}^{k\times k}$*.* ***Proof:*** *Using the norm equivalence between* L_r *operators and their symbols:* $\|\overline{G}(s) - \overline{\hat{G}}(s)\|_\infty = \sup_{z\in\mathbb{T}} \|G(s,z) - \hat{G}(s,z)\|_\infty$ *and if we just consider the problem of order reduction of* $G(s,z_0) = \left[\begin{array}{c|c} A(z_0) & B(z_0) \\ \hline C(z_0) & 0 \end{array}\right]$ *at each point* $z_0 \in \mathbb{T}$*, then*

$$\|G(s,z_0) - \hat{G}(s,z_0)\|_\infty \leq 2\sum_{i=1}^{k}(\lambda_i(\Sigma_2(z_0)))^{1/2} \tag{3.32}$$

using the 'partial balancing' technique in finite dimensions of [93], thus giving us equation (3.31) for the operator version. □

Note that we are performing a model order reduction of a 'non-nuclear' system, e.g. we are cutting off an infinite number of non-zero Hankel singular values, but still have a finite error bound, so this result might be a bit surprising.

4

Heterogeneous Distributed Systems

As was discussed in Chapter 1, distributed systems are common and difficult to deal with, especially if they're spatially heterogeneous. The system matrix describing the input-state-output behavior of N interconnected subsystems (ODE's) in a line (as in figure 4.1), each of size(order) n, will be $nN \times nN$, and thus most matrix operations will be $\mathcal{O}(n^3N^3)$ floating point operations, making traditional robust or optimal controller design prohibitively expensive for fine discretizations or large numbers of discrete subsystems. Much research has been dedicated to surmounting this computational obstacle. In [27] and [28], multilevel techniques and the special matrix structure($\mathcal{H}$-matrix) have been exploited in iterative methods for finding fast ($\mathcal{O}(N^2)$, $\mathcal{O}(N\log(N))$) approximate solutions to Lyapunov and Riccati equations for systems governed by discretized PDE's. In [32][33] an efficient LMI method for distributed controller synthesis for finitely many heterogeneous subsystems in an array with boundary conditions was developed with $\mathcal{O}(n^{2\alpha}N^{\alpha})$(where $3.5 < \alpha < 5$) complexity. For model order reduction, there have been similar results [110][111]. There has also been a conservative extension of the results of [4] to control of heterogeneous systems through robust synthesis and by treating the heterogeneity as norm bounded uncertainty [34]. There are of course also many other approaches to controlling distributed systems, e.g. distributed model predictive control [37][112], for a more thorough overview of distributed and decentralized control research, see [113], [114], and the introduction to [4].

In this Chapter we will show that the methods developed in Chapters 2 and 3 can be used to attack these problems with $\mathcal{O}(N)$ complexity, non-conservative distributed controller synthesis routines. We will also show how to use the results of Chapter 2 for efficient identification of such distributed systems, the first step necessary in practice before applying any of these control methods.

4.1 Heterogeneous String Interconnected Systems ⇔ SSS Matrices

In this chapter we consider arbitrarily spatially varying distributed systems: we will generally allow each subsystem Σ_s, in figure 4.1 to be arbitrarily different from every other subsystem, even having different state, input, and output dimensions,

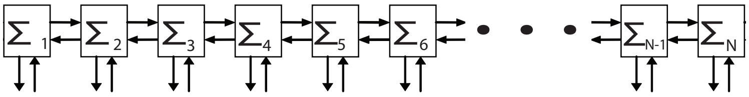


Figure 4.1: String interconnection of fully heterogeneous subsystems

as long as the interconnections are of correct size. We assume that the subsystems, Σ_s, have the following structure:

$$\Sigma_s: \quad \begin{bmatrix} \dot{x}_s \\ v^p_{s-1} \\ v^m_{s+1} \\ z_s \\ y_s \end{bmatrix} = \begin{bmatrix} A_s & B^p_s & B^m_s & B^1_s & B^2_s \\ C^p_s & W^p_s & Z^m_s & L^p_s & V^p_s \\ C^m_s & Z^p_s & W^m_s & L^m_s & V^m_s \\ C^1_s & J^p_s & J^m_s & D^{11}_s & D^{12}_s \\ C^2_s & H^p_s & H^m_s & D^{21}_s & D^{22}_s \end{bmatrix} \begin{bmatrix} x_s \\ v^p_s \\ v^m_s \\ w_s \\ u_s \end{bmatrix} \tag{4.1}$$

where x_s are the local states, v^m_s and v^p_s are interconnections to other subsystems, with performance channels and disturbance inputs (z_s and w_s), and measured outputs and controlled inputs (y_s and u_s). The $W^\bullet_s$ terms represent information feedthrough between subsystems Σ_{s+1} and Σ_{s-1}. This type of subsystem has appeared in [4], [32] and associated papers, and is also similar to subsystems considered earlier in [7] and [12]. As in these works, we will assume that the subsystem interconnections v^m and v^p are ideal, without any delay.

Six examples of such subsystem models will be shown in the following sections, and others are available in the literature, such as multiple vehicle systems [34], flight formations [9], offshore bases [115], and discretizations of various PDE's [116], [4], [3] etc.

Due to the non-zero Z^p_s and Z^m_s terms, the interconnection between subsystems (4.1) might not be well-posed (see [4] for a discussion). Fortunately, there exist sufficient conditions for well-posedness of N interconnected subsystems that can be verified in $\mathcal{O}(N)$, (see for example, [107])For structural reasons to be revealed shortly, we will always assume that the Z^m and Z^p terms are 0, assuming that the system is inherently in this form (as in the case of discretizations of PDE's), or has been converted to this form using a method like that in [107].

4.1.1 Interconnected System $\Longrightarrow$ SSS Matrices

If N of these subsystems (4.1) are connected together in a string (see Figure 4.1) with zero boundary inputs ($v^m_1 = 0$,$v^p_N = 0$) and the interconnection variables are resolved, we obtain the interconnected system:

$$\overline{\Sigma}: \quad \begin{bmatrix} \dot{\overline{x}} \\ \overline{z} \\ \overline{y} \end{bmatrix} = \begin{bmatrix} \overline{A} & \overline{B}_1 & \overline{B}_2 \\ \overline{C}_1 & \overline{D}_{11} & \overline{D}_{12} \\ \overline{C}_2 & \overline{D}_{21} & \overline{D}_{22} \end{bmatrix} \begin{bmatrix} \overline{x} \\ \overline{w} \\ \overline{u} \end{bmatrix} \tag{4.2}$$

where the $\overline{\text{overline}}$ indicates a 'lifted' variable; for vectors: $\overline{x} = \begin{bmatrix} x_1^T & x_2^T & \dots & x_N^T \end{bmatrix}^T$, and the system matrices have the SSS structure, as discussed in Chapter 2:

$$\begin{aligned}
\overline{A} &= \mathcal{SSS}(B_s^m, W_s^m, C_s^m, A_s, B_s^p, W_s^p, C_s^p) \\
\overline{B}_1 &= \mathcal{SSS}(B^m, W^m, L^m, B^1, B^p, W^p, L^p) \\
\overline{B}_2 &= \mathcal{SSS}(B^m, W^m, V^m, B^2, B^p, W^p, V^p) \\
\overline{C}_1 &= \mathcal{SSS}(J^m, W^m, C^m, C^1, J^p, W^p, C^p) \\
\overline{D}_{11} &= \mathcal{SSS}(J^m, W^m, L^m, D^{11}, J^p, W^p, L^p) \\
\overline{D}_{12} &= \mathcal{SSS}(J^m, W^m, V^m, D^{12}, J^p, W^p, V^p) \\
\overline{C}_2 &= \mathcal{SSS}(H^m, W^m, C^m, C^2, H^p, W^p, C^p) \\
\overline{D}_{21} &= \mathcal{SSS}(H^m, W^m, L^m, D^{21}, H^p, W^p, L^p) \\
\overline{D}_{22} &= \mathcal{SSS}(H^m, W^m, V^m, D^{22}, H^p, W^p, V^p)
\end{aligned}$$

Since all of the matrices in the realization have this structure, structure preserving iterations can be used to compute controllers (e.g. H_2 or H_∞) in $\mathcal{O}(N)$, as discussed in Chapter 3. Since the iterations preserve the structure, the controller itself will have a realization of SSS matrices, which is very convenient for implementation purposes, as follows.

4.1.2 SSS Matrices $\Longrightarrow$ Interconnected System

Suppose the methods in Chapters 2 and 3 have been used to design such a controller $\overline{K} : \begin{bmatrix} \dot{\overline{\xi}} \\ \overline{u} \end{bmatrix} = \begin{bmatrix} \overline{A}_K & \overline{B}_K \\ \overline{C}_K & \overline{D}_K \end{bmatrix} \begin{bmatrix} \overline{\xi} \\ \overline{y} \end{bmatrix}$, for the distributed system (4.2), where the SSS generators of each matrix are:

$$\begin{aligned}
\overline{A}_K &= \mathcal{SSS}(G_s^m, M_s^m, N_s^m, F_s, G_s^p, M_s^p, N_s^p) \\
\overline{B}_K &= \mathcal{SSS}(K_s^m, R_s^m, Q_s^m, G_s, K_s^p, R_s^p, Q_s^p) \\
\overline{C}_K &= \mathcal{SSS}(S_s^m, T_s^m, U_s^m, N_s, S_s^p, T_s^p, U_s^p) \\
\overline{D}_K &= \mathcal{SSS}(X_s^m, Z_s^m, P_s^m, Y_s, X_s^p, Z_s^p, P_s^p)
\end{aligned}$$

then it can be verified that such a controller can be directly distributed into sub-controllers:

$$K_s : \quad \begin{bmatrix} \dot{\xi}_s \\ f_{s-1}^p \\ f_{s+1}^m \\ u_s \end{bmatrix} = \begin{bmatrix} F_s & \hat{G}_s^p & \hat{G}_s^m & G_s \\ \hat{N}_s^p & \hat{R}_s^p & 0 & \hat{Q}_s^p \\ \hat{N}_s^m & 0 & \hat{R}_s^m & \hat{Q}_s^m \\ N_s & \hat{S}_s^p & \hat{S}_s^m & Y_s \end{bmatrix} \begin{bmatrix} \xi_s \\ f_s^p \\ f_s^m \\ y_s \end{bmatrix} \tag{4.3}$$

where

$$\begin{aligned}
\hat{G}_s^\bullet &= \begin{bmatrix} G_s^\bullet & K_s^\bullet & 0 & 0 \end{bmatrix}, \quad \hat{S}_s^\bullet = \begin{bmatrix} 0 & 0 & S_s^\bullet & X_s^\bullet \end{bmatrix}, \hat{R}_s^\bullet = diag(M_s^\bullet, R_s^\bullet, T_s^\bullet, Z_s^\bullet) \\
\hat{N}_s^\bullet &= \begin{bmatrix} (N_s^\bullet)^T & 0 & (U_s^\bullet)^T & 0 \end{bmatrix}^T, \quad , \hat{Q}_s^\bullet = \begin{bmatrix} 0 & (Q_s^\bullet)^T & 0 & (P_s^\bullet)^T \end{bmatrix}^T
\end{aligned}$$

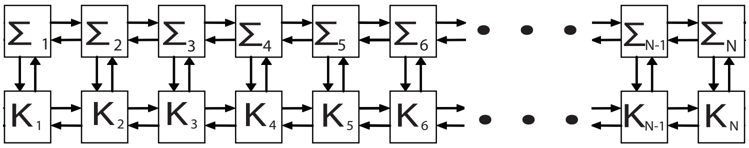


Figure 4.2: Distributed Controllers K_s are equivalent to $\bar{K}$

where $^\bullet$ is held constant as either m or p in each term. This is obviously the same structure as the subsystems, as shown in figure 4.2, where the $f_s^\bullet$ channels represent the communications between each subcontroller. This illustrates a key advantage of SSS over $\mathcal{H}$ matrix or frequency domain controller design methods for distributed systems: SSS structured controllers admit a simple distributed controller implementation, similar in structure to those sought in [4] and [32], without any additional computation.

Another nice feature of such SSS structured controllers is that their communication dimensions can be tuned after the fact in an easy and efficient way. One consequence of using high SSS orders in the computation of the distributed controller is that it increases the resulting dimension of the communication links ($f^\bullet$ in Figure 4.2) between the subcontrollers. In fact, the dimension of $f^\bullet$ is the sum of the SSS orders of $\bar{A}_k$, $\bar{B}_k\bar{C}_k$, and $\bar{D}_k$, making this a serious matter, since such a large interconnection dimension could over-tax the inter-controller communication links. However, the SSS order reductions of Chapter 2 can trivially be used (along with the shuffle permutation) to decrease the dimension of $f^\bullet$, in the following way:

Through the SSS versions of the $\mathcal{S}$ realization Lemmas 2.6 and 2.7, the input-output and state dynamics of the controller ($\bar{K}$) may be expressed as a single block SSS matrix: $\overline{\begin{bmatrix} \dot{\xi}_s \\ u_s \end{bmatrix}} = \overline{P}\overline{\begin{bmatrix} \xi_s \\ y_s \end{bmatrix}}$ with

$$\overline{P} \quad = \quad \mathcal{SSS}(\begin{bmatrix} \hat{G}_s^m \\ \hat{S}_s^m \end{bmatrix}, \hat{R}_s^m, \begin{bmatrix} \hat{N}_s^m & \hat{Q}_s^m \end{bmatrix}, \begin{bmatrix} F_s & G_s \\ N_s & Y_s \end{bmatrix}, \begin{bmatrix} \hat{G}_s^p \\ \hat{S}_s^p \end{bmatrix}, \hat{R}_s^p, \begin{bmatrix} \hat{N}_s^p & \hat{Q}_s^p \end{bmatrix}) \quad (4.4)$$

And by doing either a spatial truncation, or an SSS model order reduction (as in section 3.4 of Chapter 2) on this matrix, the size of the $\hat{R}_s^\bullet$ terms, and hence the $f_s^\bullet$ terms, may be reduced without destroying the distributed structure.

This also illustrates a useful feature of the controllers produced within the SSS framework: the communication links (f_s^m,f_s^p) may by large if desired, corresponding to a more centralized and higher performance controller. In fact, it can be shown that *any* centralized controller can be distributed in the form of (4.3) using large enough communication channels: $n_{f_s^m}, n_{f_s^p} \geq \sum_{s=1}^{N} (n_{x_s} + \max(n_{x_s}, n_{y_s}) + \max(n_{x_s}, n_{u_s}) + \max(n_{u_s}, n_{y_s}))$. Of course, for implementation, smaller is better, and as we will see in the examples, the freedom to pick the size of the communication links may be very useful in terms of design trade-offs.

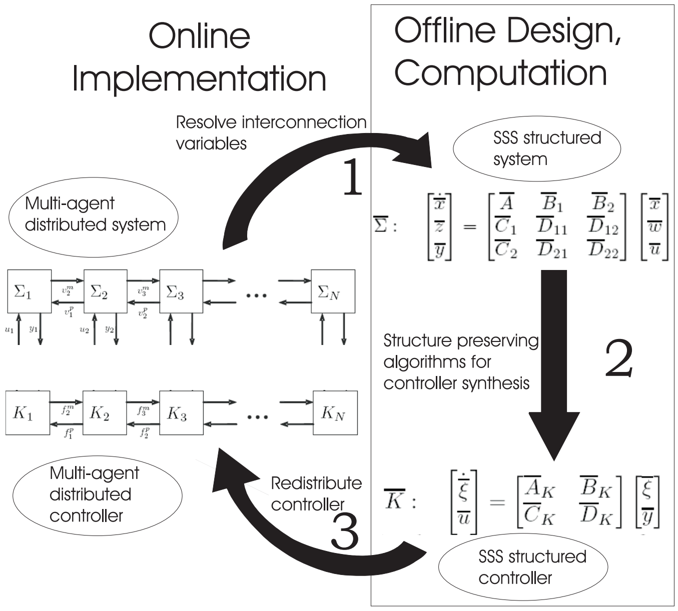

Online Implementation
Offline Design, Computation
Resolve interconnection variables
1
SSS structured system
Multi-agent distributed system
$\overline{\Sigma}: \begin{bmatrix}\dot{\overline{x}}\\ \overline{z}\\ \overline{y}\end{bmatrix} = \begin{bmatrix}\overline{A} & \overline{B}_1 & \overline{B}_2\\ \overline{C}_1 & \overline{D}_{11} & \overline{D}_{12}\\ \overline{C}_2 & \overline{D}_{21} & \overline{D}_{22}\end{bmatrix}\begin{bmatrix}\overline{x}\\ \overline{w}\\ \overline{u}\end{bmatrix}$
Σ_1 Σ_2 Σ_3 … Σ_N
v_2^m v_1^p v_3^m v_2^p
u_1 y_1 u_2 y_2
Structure preserving algorithms for controller synthesis
2
K_1 K_2 K_3 … K_N
f_2^m f_1^p f_3^m f_2^p
Multi-agent distributed controller
Redistribute controller
3
$\overline{K}: \begin{bmatrix}\dot{\xi}\\ \overline{u}\end{bmatrix} = \begin{bmatrix}\overline{A}_K & \overline{B}_K\\ \overline{C}_K & \overline{D}_K\end{bmatrix}\begin{bmatrix}\xi\\ \overline{y}\end{bmatrix}$
SSS structured controller

Figure 4.3: Conceptual diagram of SSS distributed control synthesis

4.1.3 Distributed Computation of SSS Algorithms

The SSS distributed controller synthesis method as we have described it can be summarized in Figure 4.3. In the first step, the model of the distributed system (4.1) is 'lifted' into an SSS structured system (4.2). In the second step, conducted offline on a desktop or mainframe computer, the special SSS structure is exploited in iterative computations for $\mathcal{O}(N)$ controller synthesis, producing a controller $\overline{K}$ with SSS structure, which can in step 3 be 'redistributed' into subcontrollers K_s, as in Fig 4.2, for online distributed implementation.

It's also interesting to note that SSS matrices can alternatively be thought of as computational interconnected networks [117], and due to the specific forms taken by the SSS arithmetic routines, by pursuing this idea [118], step 2 above may actually be performed in a distributed way by the micro-controllers themselves; steps 1 and 3 are unnecessary, and all computations may be performed using distributed computing.

For this method to work, we assume that each of the N micro-controllers (called 'agents') can perform small dense matrix computations; addition, multiplication, inversion, SVD, etc, each of size $\mathcal{O}(n)$, and has a local cache of size $\mathcal{O}(n^2)$ and local RAM memory of size $\mathcal{O}(n^2)$. The agents have message-passing capabilities only to nearest neighbors, that is, they are connected in a 'linear processor array'(see Fig 4.2) in distributed computing terminology [119].

While this is a nice result for practical applications, the specific calculation and communication schemes are rather boring and unenlightening, so for specific formulas, please see [118]. The distributed memory and linear communication properties of this method will be demonstrated on an example in section 4.2.

4.1.4 Summary

In this section, we have shown how interconnections of heterogeneous subsystems on a line lead to interconnected systems with realizations of Sequentially Semi Separable matrices (SSS matrices), and likewise how the reverse is also true: systems with realizations of SSS matrices can be distributed into interconnections of heterogeneous subsystems on a line. In combination with the $\mathcal{O}(N)$ SSS arithmetic in Chapter 2 and the structure preserving sign iteration techniques for system analysis and controller synthesis in Chapter 3, these results lead to relatively very fast ($\mathcal{O}(N)$, compared to $\mathcal{O}(N^3)$) distributed system analysis and distributed controller synthesis methods. Furthermore, as discussed in subsection 4.1.3, due to the special character of the SSS arithmetic operations, such synthesis and analysis computations can actually be performed online, on a distributed processing and memory array.

In the following sections, we will demonstrate these results using six numerical examples (five of them physically motivated). We will thereby see that the method described can produce low communication order controllers with closed-loop performance nearly equal to that of the centralized optimal controller, computationally much more efficiently. The first example in section 4.2 will be the simplest, and will just demonstrate a stability check on a mass-spring-damper-actuator array. In this section (and only this section) the distributed computing capabilities will also be demonstrated. The second example in section 4.3 will demonstrate distributed H_2 synthesis on a more complicated car platooning system, and the third example in section 4.4 will demonstrate distributed H_∞ synthesis on a discretization of the wave PDE. The fourth example in section 4.5 demonstrates structure preserving model order reduction, and is purely numerical, to better show the features of such an approach. The fifth example, in section 4.6 demonstrates SSS structure parametric system identification on a heat conduction problem. The sixth example, in section 4.7.3, is actually a counterexample. In it, we show what can go wrong in SSS controller synthesis when one of the assumptions A1,A2, or A3 for sign iteration convergence in chapter 3 is not satisfied.

4.2 Example: Stability Check

As in the example of [120] we consider point masses m_i on a line, connected by springs and dampers k_{ij}, c_{ij}, with surface friction $\lambda = 200$, except in our model, each mass is connected directly by spring and damper to its *two* nearest neighbors on each side(note also that the first mass is anchored to a static reference to avoid drift), and the parameters are independently generated randomly from a uniform

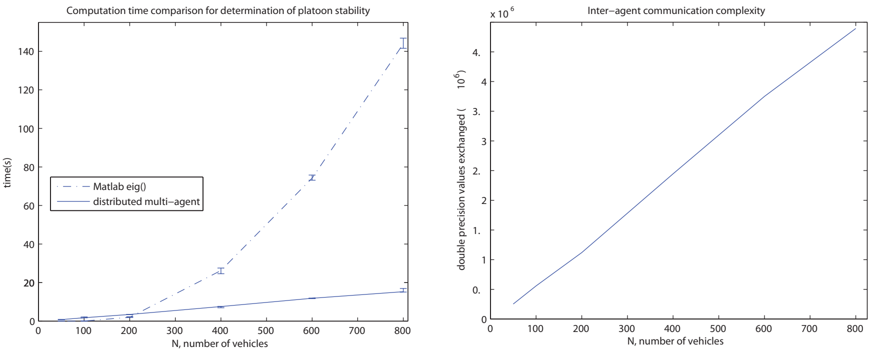


Figure 4.4: On the left we see the computational times comparison of multi-agent/SSS vs Matlab stability check. 'Error bars' indicate maximum and minimum times.On the right we see the communication complexity of multi-agent stability check

distribution, $\mathcal{U}$:

$$\begin{aligned} m_i &= 500\mathcal{U}(0.5,1.5), \qquad \forall i \in 1,2,...N \\ c_{ij} &= 10\mathcal{U}(0,1), \qquad \forall i,j \in 1,2,...N, |i-j| = 1 \\ c_{ij} &= 0.1\mathcal{U}(-0.5,0.5), \qquad \forall i,j \in 1,2,...N, |i-j| = 2 \\ k_{ij} &= 400\mathcal{U}(0,1), \qquad \forall i,j \in 1,2,...N, |i-j| = 1 \\ k_{ij} &= 4\mathcal{U}(-0.5,0.5), \qquad \forall i,j \in 1,2,...N, |i-j| = 2 \end{aligned}$$

Since some of the damping and spring coefficients may take *negative* values(note that this is not suggested for an actual system, just as a numerical example), and thus the subsystems are not strictly passive(as they are in [120]), the overall stability of the system is nontrivial, and must be numerically checked. We can write this in subsystem form (4.1)(where z_s, w_s, y_s, u_s are nonexistent for this case), implicitly generating the SSS matrices as listed in (4.2). Checking the overall system stability can then be done by either using commercially available eigenvalue methods to check that all of the eigenvalues of $\overline{A}$ in (4.2) have negative real parts, or by using the iterative SSS structure preserving sign function iterations, as in Chapter 3.

For the distributed calculation of the system stability, as discussed in section 4.1.3, we created a simple object-oriented based multi-agent test environment in Matlab to simulate the necessary message passing and local distributed memory and computing. For 25 random realizations of the above platoon example for sizes $N = [50, 100, 200, 400, 600, 800]$, the overall computational time, the local memory space utilized, and the amount of data communicated in agent-to-agent messages was recorded during a matrix sign function based stability check(see [121] for a discussion of notions of complexity in distributed computations).

In Fig. 4.4 we see the average computational time for the distributed SSS based method, confirming our estimate of $\mathcal{O}(N)$ complexity, compared to centralized eigenvalue methods as used by MATLAB(using function `eig()`), where the SSS computations are faster than those of MATLAB for $N \gtrsim 250$. Of course, this only represents the actual *calculations* performed, and for a real distributed computation, the inter-agent communications would add significantly to this time. However, as we see in Fig. 4.4, the average total number of double precision values communicated between neighboring agents in the same computations also increases only linearly with the number of agents, and hence the total processing time for a real distributed system, while slower than that in Fig. 4.4, will still scale linearly for large N. Also, in each of the above computations, the maximum amount of each agent's local memory used at any given time was less than 800 double precision values($\sim$ 6 kilobytes). Hence the distributed computation scheme we have described should be implementable on commercially available microcontrollers with only local memory.

4.3 Example: H_2 Control of a Vehicle Platoon

For our next demonstration, we will compute an H_2 controller for a car platoon. The dynamics of each car is modeled as a simple point mass with an actuator gain(g_s) and lag(τ_s):

$$\begin{bmatrix} \dot{x}_s^1 \\ \dot{x}_s^2 \\ \dot{x}_s^3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & \frac{-1}{\tau_s} \end{bmatrix} \begin{bmatrix} x_s^1 \\ x_s^2 \\ x_s^3 \end{bmatrix} + \begin{bmatrix} 0 \\ q_s^1 \\ 0 \end{bmatrix} v_s + \begin{bmatrix} 0 \\ 0 \\ g_s \end{bmatrix} u_s$$

which is similar to models previously considered in the literature[122][123][124], with a force disturbance input $v_s(t)$ representing wind gusts. Each car measures its own velocity, and the relative position between itself and the car in front of it:

$$y_s = \begin{bmatrix} -c_s^1 & 0 \\ 0 & c_s^2 \end{bmatrix} \begin{bmatrix} x_s^1 \\ x_s^2 \end{bmatrix} + \begin{bmatrix} c_s^1 \\ 0 \end{bmatrix} x_{s-1}^1 + \begin{bmatrix} q_s^3 & 0 \\ 0 & q_s^4 \end{bmatrix} n_s$$

except for the front car, which measures its own position. The cost function will be based on each car's input, $z_s^1 = f_s^1 u_s$, and the difference between its following distance and a reference, $z_s^2 = (x_{s-1}^1 - x_s^1 - \hat{r}_s)$, where the reference: $\dot{\hat{r}} = \frac{-1}{\kappa_s}\hat{r} + r_s$, is treated as a disturbance, but filtered through a lag to keep the H_2 norm finite and better represent a real situation. Note that the dynamics are uncoupled in this example(although linear draft dynamics could easily be added), but the vehicles are coupled through their measurements and cost functions. Such an interconnected

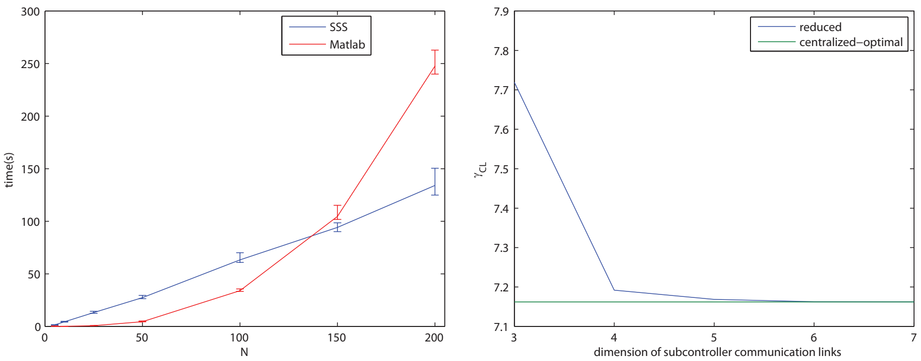


Figure 4.5: On the left we see an average computational time comparison of SSS vs Matlab H_2 synthesis routines for the platooning example. Error bars indicate maximum and minimum times. Note the linear trend for the SSS based solver compared to the cubic trend of Matlab's unstructured solver. On the right we see the closed loop performance of communication-reduced controllers.

system can be put in subsystem form(4.1) simply as:

$$\Sigma_s : \left[\begin{array}{cccc|c|c|cccc|c}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & q_s^1 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{-1}{\tau_s} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & g_s \\
0 & 0 & 0 & \frac{-1}{\kappa_s} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & f_s^1 \\
-f_s^2 & 0 & 0 & -f_s^2 & 0 & f_s^2 & 0 & 0 & 0 & 0 & 0 \\
\hline
-c_s^1 & 0 & 0 & 0 & 0 & c_s^1 & 0 & 0 & q_s^3 & 0 & 0 \\
0 & c_s^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & q_s^4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}\right]$$

where the disturbance input is partitioned as: $w_s = \left[\begin{array}{ccc} v_s & r_s & n_s^T \end{array}\right]^T$, and each car is allowed to measure its own unfiltered reference: $y_s^3 = r_s$.

This subsystem model can then be used to form the lifted system of SSS matrices in (4.2), allowing the use of the computational tools described in Chapter 3 to perform H_2 controller synthesis.

4.3.1 $\mathcal{O}(N)$ Demonstration

For this example, we will allow the coefficients to vary randomly in space. Each coefficient will have the following mean values: $\{\overline{\tau}, \overline{\kappa}, \overline{q^1}, \overline{g}, \overline{f^1}, \overline{f^2}, \overline{c^1}, \overline{c^2}, \overline{q^3}, \overline{q^4}\} = \{0.1, 0.5, 0.1, 1, 1, 1, 1, 1, 0.1, 0.1\}$, but at each point $s \in \{1, 2, ...N\}$ they will be allowed to randomly vary within 20% of this value(except for κ which is held constant). For example, at each $s \in \{1, 2, ...N\}$, $g_s = (1 + \frac{1}{5}\mathcal{U}(-1, 1))\overline{g}$, where $\mathcal{U}$

is a uniform distribution. For problems of size $N = \{5, 10, 25, 50, 100, 150, 200\}$ we did this 50 times each using our SSS solver and MATLAB's `h2syn`. The closed loop $\| \cdot \|_2$ norms varied between about 2 and 15, and the H_2 performance of the SSS controllers was always within 10^{-5} of that of the MATLAB centrally optimal controllers.

In figure 4.5 we see a comparison of the synthesis computation times, where the bars show the maximum and minimum time for each value of N, and the SSS approach becomes an advantage after about $N \approx 150$. For reference to the algorithmic discussions, SSS orders of $w_u = w_l = 16$ were used in all iterative schemes, and the sign iterations took about 11 iterations to converge.

Comment 4.1 *This example was published in [103], mostly as we see it above. However, since then it has come to our attention that the platooning cars Σ_s as we have formulated them, lead to a distributed system $\bar{\Sigma}$ with an H_2 problem and corresponding Riccati equation Hamiltonions that* do not *actually satisfy assumptions A1,A2,A3 in Chapter 3. This was brought to our attention by [125], where it is pointed out that the vehicles should also each have absolute position tracking in order for the infinite system to be well posed. It's interesting to note though that this doesn't* appear *to ruin the $\mathcal{O}(N)$ complexity, looking at figure 4.5, but actually it does, it's just hard to notice since the assumptions are violated in an almost benign way, so that the number of steps necessary for the sign iteration to converge only increases extremely slowly as $N \to \infty$. So if figure 4.5 were expanded out to $N = 10^4$ and beyond, we would probably see something more like $\mathcal{O}(N \log(N))$ complexity. Unfortunately, as we'll discuss in the example of section 4.7.3, such violations of A1:A3 do not always have such benign results.*

4.3.2 Distributedness

We'll now 'tune' the dimension of the subcontroller K_s interconnection links $f_s^\bullet$ using the method mentioned in section 4.1.2. For a typical example with $N = 50$, figure 4.5 shows the decrease in closed loop H_2 norm with the increase in communication order. All reduced controllers with communication links of size at least 3 were stable, and as we see, there is an exponential-like decrease, with high performance controllers even for very small communication links.

4.4 Example: H_∞ control of the Wave Equation

For our third heterogeneous systems example, we will demonstrate the more difficult task of H_∞ synthesis. As discussed in Chapter 3, it's not practical (and probably not desirable) to seek the absolute H_∞ optimal controller using our methods, so we will find ϵ sub-optimal controllers, but much faster and less conservatively than other distributed control methods.

4.4.1 Introduction/Discretization

Consider the 1-dimensional spatially heterogeneous wave equation with position-fixed boundary conditions:

$$\frac{\partial^2 x}{\partial t^2} = k(s)\frac{\partial^2 x}{\partial s^2} + b(s)u + q^1(s)w^1, \qquad x(0) = x(L) = 0$$

with some performance and measurement outputs:

$$z^1 = f^1(s)x, \qquad z^2 = f^2(s)u, \qquad y = c(s)x + q^2(s)w^2$$

where $x(s)$, $u(s)$, $y(s)$, $z^1(s)$, $z^2(s)$, $w^1(s)$, and $w^2(s)$ are in Hilbert spaces defined on $s \in [0, L]$. Using a finite difference approximation, $x, y, z^1, z^2, w^1, w^2, u$ can be restricted to be in some finite dimensional Euclidean space $\mathbb{R}^N$, where we approximate the spatial derivative to 4th order accuracy [126]:

$$\frac{\partial^2 x(s)}{\partial s^2} \quad \approx \quad \frac{1}{12\Delta_s^2}(-x_{s-2} + 16x_{s-1} - 30x_s + 16x_{s+1} - x_{s+2}) + \mathcal{O}(\Delta_s^4)$$

where $\Delta_s = \frac{L}{N}$ and with new discrete indices $s \in \{0, 1, 2, ..., N\}$. The approximated system can be written in the form of (4.1) where

$$\begin{aligned}
A_s &= \begin{bmatrix} 0 & 1 \\ -k_s \frac{5}{2\Delta_s^2} & 0 \end{bmatrix}, \quad B_s^m = B_s^p = \begin{bmatrix} 0 & 0 \\ \frac{2k_s}{3\Delta_s^2} & \frac{2k_s}{3\Delta_s^2} \end{bmatrix}, \quad W_s^p = W_s^m = \begin{bmatrix} 0 & -\frac{1}{8} \\ 0 & 0 \end{bmatrix}, \\
B_s^1 &= \begin{bmatrix} 0 & 0 \\ q_s^1 & 0 \end{bmatrix}, \quad B_s^2 = \begin{bmatrix} 0 \\ b_s \end{bmatrix}, \quad D_s^{12} = \begin{bmatrix} 0 \\ f_s^2 \end{bmatrix} \\
C_s^p &= C_s^m = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \quad C_s^1 = \begin{bmatrix} f_s^1 & 0 \\ 0 & 0 \end{bmatrix}, \quad C_s^2 = \begin{bmatrix} c_s & 0 \end{bmatrix}, \quad D_s^{21} = \begin{bmatrix} 0 & q_s^2 \end{bmatrix}
\end{aligned}$$

and the other terms are 0. These parameters can then be used to form the lifted system of SSS matrices in (4.2), allowing the use of the computational tools described in Chapter 3 to perform H_∞ controller synthesis. The first computational subsection will demonstrate the $\mathcal{O}(N)$ computational complexity for nearly centrally optimal controllers, and the second will show how the communication-order reduction described in section 4.1.2 can be used to approximate these controllers with very small sized communication channels.

4.4.2 $\mathcal{O}(N)$ H_∞ Synthesis

To demonstrate the application to heterogeneous systems, the system parameters will be chosen to vary randomly in space. This is not meant to represent systems actually encountered in practice, but instead to demonstrate that there is no loss of algorithmic performance or gain in conservatism for the very heterogeneous case. Each coefficient $(k_s, q_s^1, b_s, f_s^2, f_s^1, c_s, q_s^2)$ was taken independently to be 1, plus a value picked from a uniform random distribution at each $s \in \{1, 2...N\}$; for example, $k_s = 1 + \frac{1}{2}\mathcal{U}(-1, 1)$. To show the consistency of the iterative methods, 25 ran-

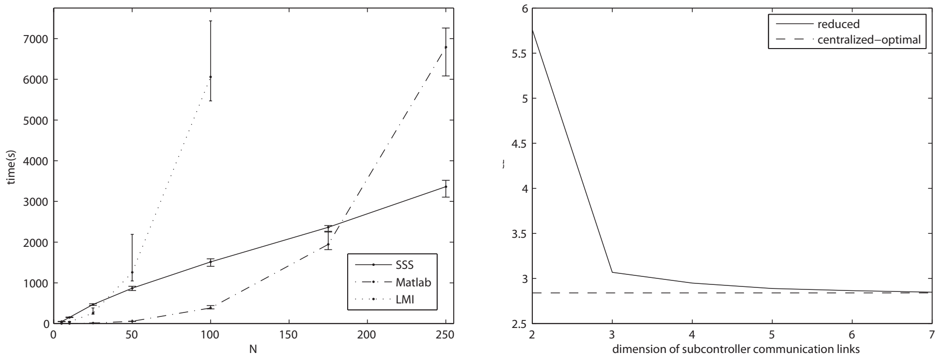


Figure 4.6: On the left we see the average computational time comparison of SSS vs Matlab H_∞ vs distributed LMI synthesis routines for the wave example. Error bars indicate maximum and minimum times. On the right we see the closed loop H_∞ performance of communication-reduced controllers

dom systems were generated for each problem size $N \in \{5, 10, 25, 50, 100, 175, 250\}$, and H_∞ sub-optimal controllers produced using MATLAB's `hinfsyn` (using the Riccati solver option) and the SSS based solver with $\gamma_{tol} = 10^{-2}$. The γ_{opt} values ranged between ≈ 2 and ≈ 3, and both the MATLAB and SSS based closed loop H_∞ norm values, γ_M and γ_{SSS}, were within the tolerance for each trial of each problem size. In figure 4.6 we see a comparison of the synthesis computation times, where the bars show the maximum and minimum time for each value of N, and the linear complexity of the SSS approach becomes an advantage after about $N \approx 200$.

The LMI based distributed control method as implemented in the Graph Control Toolbox [33] was also used to synthesize controllers for the reduced set of problem sizes $N \in \{5, 10, 25, 50, 100\}$, with the same γ tolerance as before. The results in figure 4.6 roughly confirm the polynomial computational complexity estimated in [32] (The number of LMI variables in [32] and [33] are about the same for this type of problem).

For reference to the algorithm discussions, SSS orders of $w_u = w_l = 18$ were used in all iterative schemes, the sign iterations took about 10 iterations to converge, ε was taken to be 10^{-3}, and $\sqrt{\rho(\overline{X}_\varepsilon \overline{Y}_\varepsilon)}$ was the most prominent constraint for the greatest lower bound of γ_{opt}. To produce distributed controllers with nearly centrally-optimal performance, a small γ_{tol} was chosen, but if this constraint were made less severe, lower order (w_u, w_l) approximations could probably be used, with a considerable decrease in computation time.

4.4.3 Distributed Implementation

For a typical example with $N = 50$, figure 4.6 shows the decrease in closed loop H_∞ norm with the increase in communication order, using a simple suboptimal

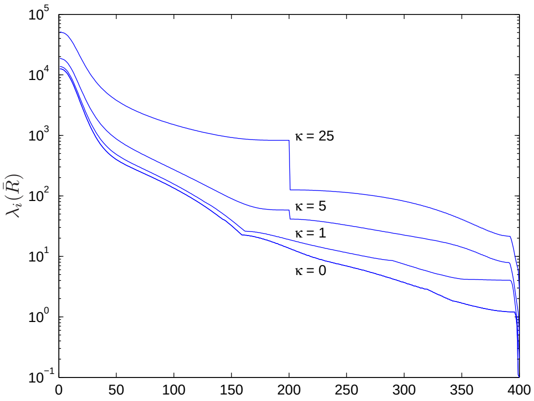


Figure 4.7: ordered eigenvalues of $\bar{R}$ for different values of κ

SSS order reduction algorithm (see [48]) similar in result to the balanced truncation method discussed in Chapter 2. Putting the system matrix into the appropriate 'proper' ([48]) form for reduction took 0.45 seconds (the process is $\mathcal{O}(N)$), and thereafter each order reduction consisted of simply truncating the appropriate matrices to the reduced dimension, taking negligible time. All reduced controllers with communication links of size at least 2 were stable, and as we see, there is an exponential-like decrease, with high performance controllers even for very small communication links. For comparison, using the truncation-based distribution method, it was necessary to truncate out to 17 spatial indices for a stabilizing controller, and 19 spatial indices to get decent performance ($\gamma_T = 3.24$).

On this same example we used the LMI method as implemented in the Graph Control Toolbox [33], and obtained a closed loop performance of $\gamma_{LMI} = 9.260$, with communication size 6 (although it should be possible to reduce this to 2 [33]). This is somewhat surprising: while the original high SSS-order controller $\overline{K}$ should be non-conservative with respect to the centralized methods, it is unexpected that the communication-reduced controllers would still have superior performance to the dedicated LMI technique, and this is likely not a general result (although it occurred for all examples tested by the authors).

4.5 Example: SSS Structure Preserving Model Order Reduction

Having now demonstrated H_2 and H_∞ synthesis, and thus proven the effectiveness of our Riccati solvers from Chapter 3, we will now switch to model order reduction, to demonstrate the Lyapunov solvers and the block diagonalization technique. We use a contrived computational example with no physical meaning in order to better exhibit some of the subtle numerical issues discussed in Chapter 3.

Consider the discrete time system $\bar{G}$: $\begin{bmatrix} \bar{x}_{k+1} \\ \bar{y}_k \end{bmatrix} = \begin{bmatrix} \bar{A} & \bar{B} \\ \bar{C} & 0 \end{bmatrix} \begin{bmatrix} \bar{x}_k \\ \bar{u}_k \end{bmatrix}$ where we have

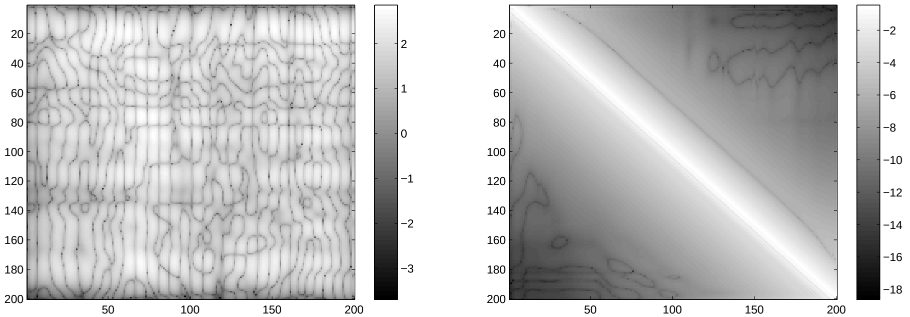


Figure 4.8: Entrywise $\log_{10}(|\hat{\hat{A}}_{11}|)$ for $\kappa = 0$ and $\kappa = 5$

picked

$$\begin{aligned}
\bar{A} &= -\frac{1}{10}\mathcal{SSS}(\begin{bmatrix}-2\\1\end{bmatrix}, -.1, \begin{bmatrix}1 & 2.5\end{bmatrix}, \begin{bmatrix}1 & 1\\2 & .5\end{bmatrix}, \begin{bmatrix}.3\\-.01\end{bmatrix}, .7, \begin{bmatrix}9 & 5\end{bmatrix}),\\
\bar{B} &= \mathcal{SSS}(\begin{bmatrix}1.6\\-.7\end{bmatrix}, .4, -1.5, \begin{bmatrix}1.3\\.2\end{bmatrix}, \begin{bmatrix}1.7\\-.2\end{bmatrix}, .7, 1.7)\\
\bar{C} &= \mathcal{SSS}(2.6, -.5, \begin{bmatrix}0.2 & 1.7\end{bmatrix}, \begin{bmatrix}.5, .3\end{bmatrix}, 3.5, .4, \begin{bmatrix}.5 & -.5\end{bmatrix})
\end{aligned} \tag{4.5}$$

to be spatially invariant for $s = 1 : 200$ for simplicity, although as in the H_2 and H_∞ cases, the computational methods and complexity hold also for heterogeneous systems. The original system with state dimension 400 has l_2 induced norm $\|\bar{G}\|_\infty \approx 197.2$ and with MATLAB's `reduce()` using balanced truncation, a non-structured reduced order system $\hat{G}$ with state dimension 200 was found with $\|\bar{G} - \hat{G}\|_\infty \approx 7.4565$. Using the SSS structured methods (with SSS order of 10 for all calculations) of Chapters 2 and 3, an SSS structured reduced order system $\hat{\hat{G}}$ with state dimension 200 was found with $\|\bar{G} - \hat{\hat{G}}\|_\infty \approx 6.3795$. In this case, only 3 α-iterations were needed to find the correct value for splitting the eigenvalues, and by extending N to larger values, it was found that the estimate of linear computational complexity($\mathcal{O}(N)$) holds, and the SSS model order reduction routines become faster than the MATLAB routines at $N \approx 350$ and ≈ 110 seconds.

In figure 4.7 we see how solving Lyapunov *inequalities* instead of equations in the balanced truncation method, using offset $\kappa\Pi_L \begin{bmatrix}I & 0\\0 & 0\end{bmatrix}\Pi_R$, affects the eigenvalues of $\bar{R}$ for this problem. For $\kappa = 0$, there is not much gap (leading to more bisection α-iterations), the resulting $\bar{V}$ matrix is ill-conditioned, and $\hat{\hat{A}}_{11}$ has large off diagonal elements (see figure 4.8, left hand side) making it difficult to implement. However, for $\kappa = 5$ the resulting gap in $\lambda(R)$ is noticeable, leading to faster α-bisection convergence and a nicer realization of $\hat{\hat{A}}_{11}$ (see figure 4.8 right hand side). Note that the figures show the entrywise $\log_{10}$ of the absolute values of the matrices, and we thus see that $\hat{\hat{A}}_{11}$ clculated with $\kappa = 5$ has an exponential spatial decay away from the diagonal, making it much easier to implement.

However, unnecessarily large κ values lead to decreased accuracy in the approximation, for example, with $\kappa = 25$ we have $\|\bar{G} - \hat{\bar{G}}\|_\infty \approx 8.0516$ and for $\kappa = 100$ we have $\|\bar{G} - \hat{\bar{G}}\|_\infty \approx 10.2025$, but an appropriate κ value can be iteratively found by bisection and checking that the off diagonal corners of $\bar{V}^{(-1)}$ approach 0. It would be interesting to research how to optimally pick the offset scalar κ, or some other more general offset, so as to end up with a nice SSS structure *and* a small error (see Chapter 11 for further discussion).

4.6 Example: Fast Structured System ID

In this section, we will use our SSS results for System Identification (SysID), by exploiting the SSS structure in the partial derivatives present in an Extended Kalman Filter(EKF) employed for parameter estimation. The result will be an $\mathcal{O}(n^5NT)$ complexity algorithm for system identification of this type of structured system, where N is the number of subsystems, n is their much smaller size($n << N$), and T is the number of EKF steps until convergence. Furthermore, since the method preserves the SSS structure at each point, the resulting identified system will have a realization of SSS matrices, allowing the use of the aforementioned $\mathcal{O}(N)$ SSS analysis and synthesis routines and distributed implementation. For the same reason, this computation could actually be *performed* on a distributed processor array, as discussed in section 4.1.2.

4.6.1 EKF for Parametric ID

The SysID method we will use is a rather old one, which employs the Extended Kalman Filter as a parameter estimator, but it is still popular, and we will overview it for ease of reference in later discussions. This method has been much discussed in the literature, and we will take the following form of the formulas from [127] (see this reference for further discussion and other relevant literature). Basically the idea is that given some LTI system:

$$\begin{aligned} x_{t+1} &= A(\theta)x_t + B(\theta)u_t + w_t \\ y_t &= C(\theta)x_t + v_t \end{aligned} \tag{4.6}$$

where

$$\begin{aligned} E[w_j w_k^T] &= Q\delta_{jk}, \quad E[v_j v_k^T] = R\delta_{jk}, \\ E[w_j v_k^T] &= S\delta_{jk}, \quad E[x_0] = 0, \quad E[x_0 x_0^T] = \Pi_x \end{aligned} \tag{4.7}$$

and θ is some vector of unknown parameters with $E[\theta\theta^T] = \Pi_\theta$, given input output data u_t, y_t, $t = 1 : N$, we can use Kalman filtering, but additionally append a parameter estimate $\hat{\theta}_t$ to the unknown state $\hat{x}_t$, to simultaneously perform state estimation and SysID. This is a nonlinear estimation problem, for which we can use the ‘extended Kalman Filter’ (EKF), which takes the form:

$$
\begin{aligned}
T_t &= \begin{bmatrix} C_t & N_t \end{bmatrix} \Pi_t \begin{bmatrix} C_t & N_t \end{bmatrix}^T + R \\
\begin{bmatrix} K_t \\ L_t \end{bmatrix} &= (\begin{bmatrix} A_t & M_t \\ 0 & I \end{bmatrix} \Pi_t \begin{bmatrix} C_t & N_t \end{bmatrix}^T + \begin{bmatrix} S \\ 0 \end{bmatrix}) T_t^{-1} \\
\hat{x}_{t+1} &= A_t \hat{x}_t + B_t u_t + K_t (y_t - C_t \hat{x}_t) \\
\hat{\theta}_{t+1} &= \hat{\theta}_t + L_t (y_t - C_t \hat{x}_t) \\
\Pi_{t+1} &= \begin{bmatrix} A_t & M_t \\ 0 & I \end{bmatrix} \Pi_t \begin{bmatrix} A_t & M_t \\ 0 & I \end{bmatrix}^T - \begin{bmatrix} K_t \\ L_t \end{bmatrix} T_t \begin{bmatrix} K_t \\ L_t \end{bmatrix}^T + \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix}
\end{aligned}
$$

where

$$
\begin{aligned}
\Pi_0 &= \begin{bmatrix} \Pi_x & 0 \\ 0 & \Pi_\theta \end{bmatrix}, \quad N_t = \frac{\partial}{\partial \theta} [C(\theta) \hat{x}_t]|_{\hat{\theta}_t} \\
M_t &= \frac{\partial}{\partial \theta} [A(\theta) \hat{x}_t + B(\theta) u_t]|_{\hat{\theta}_t} \\
A_t &= A(\hat{\theta}_t), \quad B_t = B(\hat{\theta}_t), \quad C_t = C(\hat{\theta}_t)
\end{aligned}
\tag{4.8}
$$

(note that there is a typographical error in [127] in equation (3.16a), which we have corrected in the formula above). As is know well known, this method is not always guaranteed to converge, has problems with bias, etc, but often works well in practice given a decent initial guess $\hat{x}_0$ and $\hat{\theta}_0$. It can also be modified to have guaranteed convergence [127] and this method will be used here as an exposition of our computational method, which hopefully can be applied to other SysID techniques as well.

Our approach will be to now assume that the matrices $\bar{A}(\theta), \bar{B}(\theta), \bar{C}(\theta), \bar{R}, \bar{S}, \bar{Q}$ in (4.6) and (4.7) come from a heterogeneous distributed system composed of a linear interconnection of subsystems as in (4.1), and hence each have the SSS structure as previously discussed. Under such circumstances, all of the EKF calculations above, consisting of simple matrix arithmetic and partial derivatives (see section 2.3.7), could be performed in $\mathcal{O}(N)$, leading to $\mathcal{O}(N)$ complexity steps in the EKF.

Note that we are parameterizing each of $\bar{A}$, $\bar{B}$, $\bar{C}$ by roughly $\mathcal{O}(Nn^2)$ values, far less than the $N^2(2 + n_x)(n_y + n_u)$ parameters that are considered 'minimal' for characterizing a state space system of input, state, output dimensions of Nn_u, Nn_x, Nn_y in companion matrix form[128]. This is of course only possible because we are assuming a specific type of factorization of the matrices $\bar{A}$, $\bar{B}$, $\bar{C}$, but we should also note that our above parameterization via $\bar{\theta}$ may still not be the most minimal possible for a particular problem, but this shouldn't concern us too much, since in the end the $\mathcal{O}(N)$ complexity of this routine overcomes any minor inefficiencies.

4.6.2 Numerical Demonstration

We will only demonstrate a very simple example here, where each subsystem Σ_s in figure 4.2 has only one input, one output, and one state variable, but the technique and formulas as shown above work generally on any type of subsystems with vector inputs, outputs, and states, in the form of (4.1).

Introduction

We consider a heat conduction (diffusion) problem with a negative offset to make it nicely stable:

$$\dot{T}(r) = k(r)\frac{\partial^2 T(r)}{\partial r^2} - \frac{1}{2}T(r) \tag{4.9}$$

with boundary conditions $T(R) = T(0) = 0$. Using a second order finite difference approximation ($\frac{\partial^2 T(r)}{\partial r^2} \approx T_{s-1} - 2T_s + T_{s+1}$ for $s = 1 : N$), we can write this down as a coupled set of ODE's in a tridiagonal matrix $\bar{A}_c$, which we can convert to discrete time using a bilinear(Cayley) transform: $\bar{A}_d = (I + \bar{A}_c)(I - \bar{A}_c)^{-1}$. We then add an input, $u(t)$ and output $y(t)$ to to make a distributed system fitting into the description of (4.1) (4.2):

$$\bar{T}_{k+1} = \bar{A}_d \bar{T}_k + diag(b_s)\bar{u}_k, \quad \bar{y}_k = diag(c_s)\bar{T}_k \tag{4.10}$$

over $s = 1 : N$, where k_s, b_s, c_s are spatially varying heat conduction coefficient, and input/output weights.

Demonstration of Convergence, $N = 50$

For our demonstration here we'll sample each 'real' k_s, b_s, d_s from a uniform random distribution at each $s \in [0, 1]$: e.g. $k_s \in 1 + \mathcal{U}[-1, 1]$, but start the estimates in θ as identically 1 for all s. This would correspond to trying to identify a spatially heterogeneous problem, using a homogeneous initial guess. In order to nicely show the convergence, we set $\hat{x}_0 = \mathcal{N}(0, 1)$ and sample v_t and w_t from $10^{-4}\mathcal{N}(0, 1)$.

Running this scenario, the iteration converges to almost steady state and produced system estimates that were many times better than the initial guess, as shown in figure 4.9, where $\hat{\Sigma}_t$ is the estimated system transfer function at time t, Σ_r is the 'real' system, and $\hat{\Sigma}_0$ is the system corresponding to the initial guess θ_0. In this case, with $N = 50$ subsystems, $\|\Sigma_r\|_\infty = 6.0895$, $\|\Sigma_r - \Sigma_0\|_\infty = 4.0363$, and $\|\Sigma_r - \Sigma_{100}\|_\infty = 1.1 \times 10^{-3}$, so the improvement in the system estimate with respect to the infinity norm is enormous. The improvement in 'variance accounted for'(VAF) [44] is also impressive: $VAF_{\hat{\Sigma}_0} = 67.7470$ compared to $VAF_{\hat{\Sigma}_{100}} = 99.5192$.

While the parameter estimates $\bar{\theta}_t$ do not give us k_i directly, since they are parameterized on $\bar{A}_d$ instead of $\bar{A}_c$, we can still perform a 'reverse' Cayley transform to find back $\bar{A}_c(\hat{\theta}) = (I + \bar{A}_d(\hat{\theta}))^{-1}(\bar{A}_d(\hat{\theta}) - I)$ and thus find an estimate $\hat{k}$ for

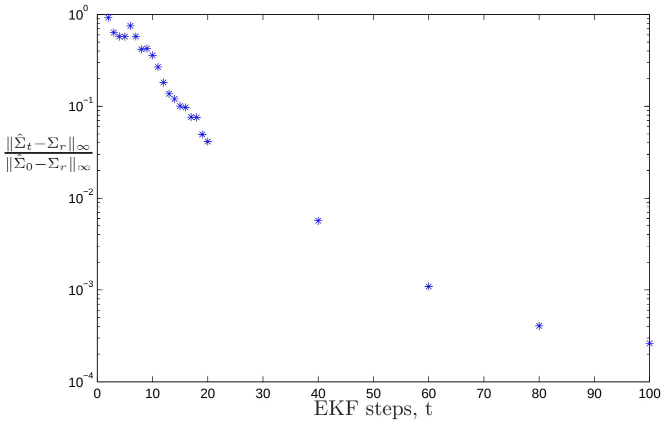


Figure 4.9: Infinity norm relative progress

the 'real' heat conduction coefficient. In figure 4.10 we show these conduction coefficient estimates over a section of space $s \in [26 : 36]$ for various times during the identification, and see that while the initial parameter estimate is bad, the estimates actually converge pretty quickly to near the exact values, leading to a very accurately identified heterogeneous heat conduction system.

Demonstration of $\mathcal{O}(N)$ Complexity

For this part of the example, we'll increase the size of the problem by slowly increasing N, the number of subsystems in figure 4.2, which in this case is the size of the heat conduction problem, and investigate the time necessary for the computations in section 4.6.1 to converge. For this part, we'll start out with a closer initial guess of $\hat{x}(0) = x(0) + \frac{1}{10}\mathcal{N}(0,1)$ and sample the 'real' parameters k_s, b_s,c_s from $1 + \frac{1}{5}\mathcal{U}[-1,1]$ to decrease the randomness in the number of steps to convergence. We then set the initial estimates to be identically 1 just as before, and randomly generate 10 systems for each size $N \in \{25, 50, 75, 100, 150, 200, 300\}$, and run the EKF until convergence, defined here as when $\frac{\|\hat{\hat{\theta}}_t - \hat{\hat{\theta}}_{t-1}\|_2}{\sqrt{N}} < 5 \times 10^{-4}$ (using this criteria, the values of $\frac{\|\hat{\Sigma}_t-\Sigma_r\|_\infty}{\|\hat{\Sigma}_0-\Sigma_r\|_\infty}$ at convergence were on average about $\frac{1}{30}$, and did not increase with N), measuring the times and final errors for each SysID. The time complexity results are shown in figure 4.11, where we see that not only is the average EKF time clearly increasing linearly with N (just as is guaranteed by the SSS arithmetic), the overall EKF time to convergence also appears to be, on average, linear, since in this case the number of steps to convergence did not seem to increase with N (on average the steps needed for convergence were $\{22.9, 23.4, 22.6, 24.3, 24.5, 23.9, 24.6\}$), or if it did, it was extremely slowly. This is fortuitous, providing truly linear computational complexity distribute system SysID in this case, although this may not always happen; it is unknown if or under what circumstances the number of steps to convergence would grow with N.

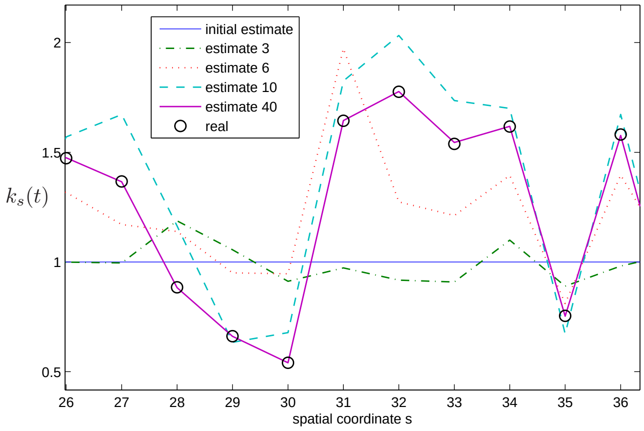


Figure 4.10: heat conduction coefficient progress

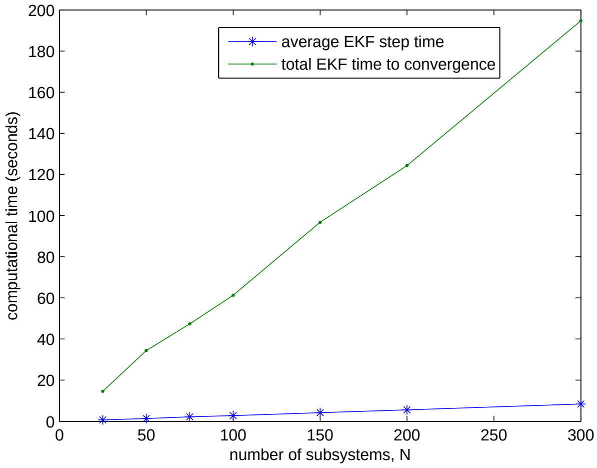


Figure 4.11: Computational Complexity Study, showing linear trends of average EKF step time from t to $t+1$ and total EKF time to convergence for SysID of heat conduction example

4.6.3 Discussion

While we can't absolutely bound the number of steps, T, necessary for a good identification using the simple algorithm above (nor even guarantee that the EKF will converge), we saw in the example that in practice the procedure seems to work very well, and that sometimes T is independent of the size of the distributed system, N leading to truly linear computational complexity structured distributed SysID.

Future work should be devoted to extending these ideas to other types of SysID, e.g. those with guaranteed convergence, and developing a distributed (or parallel) computational method for generating the initial parameter estimate $\hat{\hat{\theta}}_0$ with which to seed this procedure.

While our numerical example was a very simple heat conduction problem, we note that this approach potentially has its greatest advantage in identifying more complicated distributed systems, particularly when the subsystems cannot easily be isolated and analyzed separately from eachother, such as in many systems biology examples (e.g. reaction-diffusion systems in [129][130][131]) for which current SysID methods (called 'reverse engineering' methods) are computationally very expensive (see e.g. [132] at $\mathcal{O}(N^4)$).

4.7 Counterexample: Control of a Smart Blade

We've now seen five examples of distributed systems where our SSS methods work well: they satisfy the necessary assumptions A1:3 in section 3.1.3 and so have $\mathcal{O}(N)$ complexity and produce very good (nearly optimal) results. However, it's very important for the end user of these techniques that it be clear under what conditions, and in what way, these techniques will fail, so we will devote the last section and example of this Chapter to a case where our SSS sign iteration methods don't work well (due to a failure to satisfy assumption A1 of section 3.1.3).

4.7.1 Distributed Model

The model under consideration is that of a 'smart blade' of a wind turbine; e.g. one with distributed sensors and actuators. In this thesis we will only briefly go into the details of smart-blade design considerations, see [133] for details. We will consider a single rotor blade modeled as a Euler-Bernoulli beam with only a flap-wise degree of freedom, and viscous but not strain damping, at a fixed angle of rotation, giving us the partial differential equation:

$$\rho(r)\frac{\partial^2 s(r,t)}{\partial t^2}+\gamma(r)\frac{\partial s(r,t)}{\partial t}+\frac{\partial^2}{\partial r^2}\left(E(r)I(r)\frac{\partial^2 s(r,t)}{\partial r^2}\right)=F_u(r,t)+F_d(r,t) \quad (4.11)$$

with clamped-free boundary conditions:

$$s(0,t) = \frac{\partial s}{\partial r}(0,t) = EI(L)\frac{\partial^2 s}{\partial r^2}(L,t) = \frac{\partial}{\partial r}\left(EI(r)\frac{\partial^2 s}{\partial r^2}(r,t)\right)(L) = 0 \qquad (4.12)$$

where $r \in [0, L]$ is the spanwise position along the blade from the clamped root ($r = 0$) to the free tip ($r = L$), $s(r,t)$ is the displacement in the flapwise direction, ρ is the linear density, γ is the viscous damping constant, EI is the bending stiffness, and F_u and F_d are the control and disturbance linear force intensity inputs, respectively.

We will consider strain sensors, since they are the most widely employed type of sensor in smart blades, although our framework could incorporate accelerometer data and flow data, were such sensors present. Our measurements, y will thus be of the form:

$$y(r,t) \quad = \quad p(r)\frac{\partial^2 s}{\partial r^2} + \eta(r)n(r,t) \qquad (4.13)$$

where $n(r,t)$ is random measurement noise and $p(r)$ and $\eta(r)$ are some sensor proportionality constants, related to the type of sensor used and the position on the blade.

In actual wind turbine design, the engineer must consider power quality, generator loads, gear loads, etc (see e.g. [134]), but even given an accurate wind turbine model including these effects, the task of then finding appropriate importance weights for these factors is a difficult and complicated problem of economics and systems engineering. To simplify it for meaningful analysis we will consider only the fatigue damage caused by vibrations and accompanying stresses in our single blade.

There are techniques for estimating the 'damage intensity' (e.g. [135]) of an open loop transfer function, but there are no direct methods to use control to minimize the damage in closed loop (although there are indirect, optimization based methods [136]), and we will not pursue this. However, from Miner's rule [137], it's apparent that one should try to minimize both the amplitude of the vibration-induced strain oscillations, and the frequency with which they occur. We thus define our cost z to include both strain(z^1) and strain-velocity (z^2) components:

$$\begin{bmatrix} z^1(r) \\ z^2(r) \end{bmatrix} \quad = \quad \begin{bmatrix} p_{z^1}(r)\frac{\partial^2}{\partial r^2} & 0 \\ 0 & p_{z^2}(r)\frac{\partial^2}{\partial r^2} \end{bmatrix} \begin{bmatrix} s(r) \\ \dot{s}(r) \end{bmatrix} \qquad (4.14)$$

Decreasing the 'strain velocity' should damp the high frequency oscillatory modes (intuitively, like a derivative term in a PID controller), and decrease the number of strain oscillations in the closed loop, thus decreasing the damage intensity. The relative weights p_{z^1} and p_{z^2}, and their dependence on the blade location (r), should be picked so as to minimize the damage on the section of the blade with the shortest life (probably heuristically or empirically in practice).

We will model our disturbance as:

$$F_d(r) = \kappa(r)d(r,t) \qquad (4.15)$$

from random wind gusts, turbulence, shadow, etc; To keep our model simple, we will not incorporate any detailed stochastic disturbance model knowledge(although detailed filters can be used to describe the turbulence spectrum [138] or other disturbance sources, as discussed in [139]), and will just assume them to be location dependent and bounded, where $d(r,t)$ is a random bounded vector and $\kappa(r)$ is another proportionality constant, related to the location on the blade.

Probably the most difficult hardware issue for the smart-blade concept is the design of well-functioning actuators reliable enough to operate in a corrosive off-shore environment, through lightening strikes, etc. Whichever type of actuation (or combination of types) is chosen for the smart-blade, each actuator will have its own dynamics, due to the internal mechanical movements involved, and potential aero-elastic coupling. For the notational simplicity of our demonstration, our model will have a gain and a low pass filter to represent the limited bandwidth of the mechanical actuator. We thus have the transfer function:

$$F_u(r) = \frac{g(r)}{\frac{1}{2\pi h}\sigma + 1} u(r) \tag{4.16}$$

where h is the actuator bandwidth in Hertz, g is the gain, and σ is the Laplace variable. Note also that the gain $g(r)$ can vary with r; we might have different actuators with different effects on the lift at different locations on the blade.

Of course, to implement these actuators on a real wind turbine, we should also note that the controller effort, $u(r)$, is not free; it will use energy, and the corresponding movement will cause wear and tear on the actuator itself, and thus should be kept minimal. Hence we introduce an additional term:

$$\overline{z}^3 = \xi(r)\overline{u} \tag{4.17}$$

to our cost function in (4.14), which our feedback control design will attempt to minimize.

Keep in mind that in a real wind turbine, there would be nonlinear aero-elastic coupling, such as Beddoes-Leishman dynamic stall [140], and uncertainty in the actuator lag and gain, but we are leaving these out for this nominal controller synthesis problem. For a model incorporating uncertainty, see [133].

4.7.2 Finite Difference Discretization

While technically correct, equations (4.11), (4.12), (4.13), and (4.14) are not actually very useful for analysis and design. The variables y, s, z are on infinite dimensional Hilbert spaces, and some of the infinite dimensional operators are heterogeneous, making computations quite difficult. We thus divide the function space $s : \mathcal{L}_2[0, L]$ into a finite number (N) of points, $\overline{s} = [s_1, s_2, s_3 ... s_N]^T$ and approximate the spatial partial derivatives using finite difference 2nd order Taylor series expansions [126] (although higher order would also fit into our framework). Such a formulation will also allow us a realistic representation of our actuators, which will likely be discrete in space.

By doing this, we turn our continuous space model into a discrete space model of banded matrices, which can then be put into SSS form, and thus made ready for distributed controller synthesis, as in previous examples. Due to the complexity of the model, we will not do this by hand, but instead use one of the automatic SVD methods for finding SSS realizations of banded matrices (see the appendix in section 2.7), then use the SSS version of the shuffle permutations in Lemma 2.6 to create an SSS state space model, and thus an interconnected model.

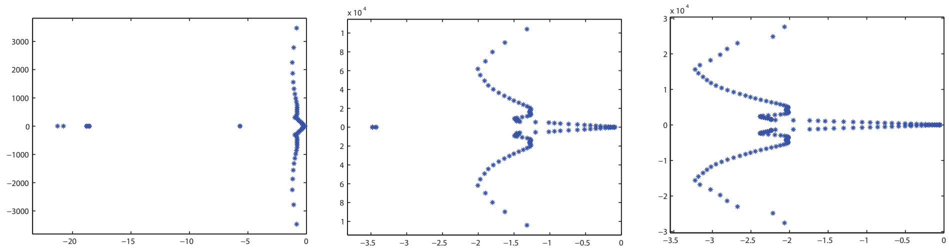
4.7.3 Computational Difficulties

For our computations, we assume a constant apparent wind velocity and angle of attack, etc, and use the stiffness, density, and blade profile from a LM38.8 wind turbine blade [141]. We stretch the blade to be $L = 50$ meters, and try to perform H_∞ synthesis, but unfortunately, our attempts always fail, since as we increase the number of points, N, in the discretization, the spectrums of the Hamiltonion matrices (see Lemma 3.9) diverge from the real axis, while simultaneously approaching the imaginary axis, as we see in figure 4.12. This clearly violates assumption A1 in section 3.1.3, and increases the value of η in that Lemma 3.2, thus increasing the number of iterations necessary for convergence of the sign iteration, and destroying the desired $\mathcal{O}(N)$ complexity. Additionally, due to the placement of these eigenvalues (very small real and very large imaginary components) on the way to convergence, the eigenvalues will pass very close to the imaginary axis (see Figure 3.2), making it likely that an eigenvalue will jump over the axis, causing very large resulting errors. Physically, these eigenvalues correspond to increasingly badly damped vibratory modes in the closed loop blade as we increase N, which is just a physical characteristic of our model and desired control law. This effect can be somewhat mitigated by increasing the control authority, i.e. increasing the available force from the actuators, but in practice this just isn't an option; one can only get so much force from the wind.

H_∞ controllers are shown in [133] to make significant improvements in wind turbine performance by decreasing the damage intensity during blade use, but unfortunately due to the bad damping of the closed loop system, computationally efficient controller synthesis using current SSS based methods may not be an option for such problems. H_2 SSS synthesis fails as well. See the recommendations in Chapter 11 for possible improvements or solutions to these problems.

4.8 Chapter Conclusion

In this chapter we finally provided some motivation for the lengthy derivations of structured matrix arithmetic in Chapter 2 and proofs of sign iteration convergence in Chapter 3. We showed how heterogeneous subsystems interconnected together on a line lead to a system with a Sequentially Semi-Separable matrix realization, and that the opposite is also true, leading to an $\mathcal{O}(N)$ complexity analysis and synthesis method for distributed controllers, and even $\mathcal{O}(N)$ identification of heterogeneous distributed systems. We also discussed how due to the special form of



N=30 N=50 N=80

Figure 4.12: spectrum of Hamiltonians in $\mathbb{C}_-$ for X Riccati with growing N

SSS arithmetic, such computations can also be performed in a distributed manner, on distributed memory machines.

Finally, in a few numerical examples, we demonstrated this $\mathcal{O}(N)$ complexity in comparison with other $\mathcal{O}(N^3)$ distributed techniques, and found that for large problems, it was not only much faster, but also less conservative. For balance, we also showed a distributed control problem where our SSS methods failed, due to a violation of an assumption necessary for the speedy convergence of the sign iteration, caused by certain physical properties of this example.

The following chapters in Part II will follow much the same format as this one, except for different applications and different operator structures.

5

Homogeneous Distributed Systems

In the last chapter, we saw an application of Part I to heterogeneous, finite extent, systems. We presented such systems first because they are the least abstract and hence easiest to understand of our applications, and should provide some intuition for the rest of our work, such as this chapter. Herein, we'll see a much more abstract example of a distributed system, which will induce L_r matrices instead of SSS matrices, but the rest of the results and treatment will be much the same. We will still use the structure preserving iterations of Chapter 3 on one of the arithmetics of Chapter 2, and it will still produce a distributed controller that is arbitrarily close to optimal.

The class of spatially invariant (or 'shift invariant') systems, such as arises in spatially discretized PDE's with spatially constant coefficients and no boundary conditions [26], or doubly infinite arrays of identical interconnected subsystems [4], has enjoyed significant attention lately. Such problems are interesting for their controllability/observability relations to very large homogenous finite dimensional problems [125].

The continued difficulty with this class of system is in finding non-conservative computational methods for checking stability and performance, and synthesizing controllers. Using Fourier transforms to transform a countably infinite dimensional multilevel L-operator Riccati equation into a family of finite dimensional Riccati equations parameterized in multiple variables over the unit circle is not necessarily a simplification: we still have to solve an uncountable number of finite dimensional Riccati equations.

A few advances have been made in this direction; while in [142] and [26] only simple examples are solved by hand, in [143] it is suggested to approach the infinite dimensional solution by solving progressively larger finite dimensional problems, a slowly converging algorithm resulting in high order approximations to a parametric DARE is proposed in [144] (see also [145] for other iterative techniques), and in [4] and [146], LMI relaxations are used to find conservative controllers with a low rational order or 'localized' structure, respectively. However, it is still often desirable to have non-conservative stability and performance analysis and controller synthesis for both continuous and discrete time systems, and a way to calculate them in a relatively efficient manner.

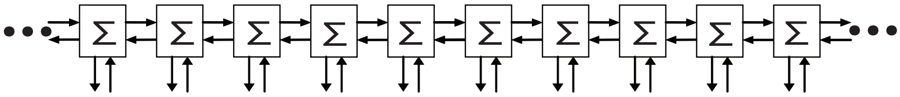


Figure 5.1: String interconnection

To address this problem, we will use the results of Chapters 2 and 3 in Part I of this thesis. In short, will use the structure preserving arithmetic for $\mathcal{S}$-realizations of L_r matrices from Chapter 2 in the sign iteration based controller synthesis techniques of Chapter 3 to compute controllers with realizations of L_r matrices. In the same way as for the SSS matrices of the previous chapter, we can then redistribute such structured controllers into the same interconnection structure as the plant, to achieve a distributed controller implementation.

5.1 Subsystems and Interconnected System

In this Chapter we will study an infinite number of subsystems Σ_s, as in figure 5.1, which are identical to eachother for all $s \in \mathbb{Z}$. We assume that the subsystems, Σ_s, have the following structure:

$$\Sigma_s : \qquad \begin{bmatrix} \dot{x}_s \\ v^p_{s-1} \\ v^m_{s+1} \\ y_s \end{bmatrix} = \begin{bmatrix} A & B^p & B^m & B \\ C^p & W^p & Z^p & V^p \\ C^m & Z^m & W^m & V^m \\ C & H^p & H^m & D \end{bmatrix} \begin{bmatrix} x_s \\ v^p_s \\ v^m_s \\ u_s \end{bmatrix} \tag{5.1}$$

where x_s are the local states, v^m_s and v^p_s are interconnections to other subsystems, and y_s and u_s are measured outputs and controlled inputs. The W_s terms represent information feedthrough between subsystems Σ_{s+1} and Σ_{s-1}.

Lemma 5.1 *[147] The set of interconnected systems (5.1) is well-posed [4] if and only if the matrix pencil* $z\begin{bmatrix} I & -Z^p \\ 0 & W^m \end{bmatrix} - \begin{bmatrix} W^p & 0 \\ -Z^m & I \end{bmatrix}$ *has no generalized eigenvalues on the unit circle. In this case, the interconnected system can always be transformed into 'Kronecker canonical' form, wherein* $Z^m = 0$, $Z^p = 0$. ***Proof:*** *See the Appendix in section 5.6* □

Hence from now on we will make this assumption, $Z^m = 0$, $Z^p = 0$, without loss of generality. In this case the coupling terms, v^m_s and v^p_s, are superfluous, and if we resolve them we get a 'lifted system':

$$\bar{\Sigma} : \quad \begin{bmatrix} \dot{\bar{x}} \\ \bar{y} \end{bmatrix} = \begin{bmatrix} \bar{A} & \bar{B} \\ \bar{C} & \bar{D} \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{u} \end{bmatrix} \tag{5.2}$$

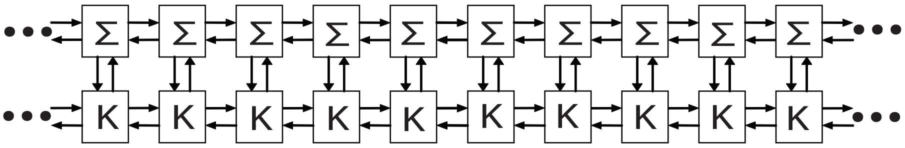


Figure 5.2: String interconnection with similarly connected controller

with states $\bar{x} = \begin{bmatrix} \dots & x_{-1}^T & x_0^T & x_1^T & x_2^T & \dots \end{bmatrix}^T$, and similarly structured inputs $\bar{u}$ and outputs $\bar{y}$ assumed in l_2. $\bar{A}, \bar{B}, \bar{C}, \bar{D}$ are block Laurent operators with rational symbols, as in Chapter 2 section 2; $\bar{A} = \mathcal{S}(B^m, W^m, C^m, A, B^p, W^p, C^p)$. Hence, using the iterative techniques in Chapter 3, we can compute a similarly structured controller $\bar{K}$: $\begin{bmatrix} \overline{\dot{\xi}} \\ \overline{u} \end{bmatrix} = \begin{bmatrix} \overline{A}_K & \overline{B}_K \\ \overline{C}_K & \overline{D}_K \end{bmatrix} \begin{bmatrix} \overline{\xi} \\ \overline{y} \end{bmatrix}$, where $\overline{A}_K, \overline{B}_K, \overline{C}_K, \overline{D}_K \in \mathcal{S}$.

In the same way that the interconnected subsystems Σ_s induce a system of L_r operators, $\bar{\Sigma}$, a controller of L_r operators, $\bar{K}$, can be directly distributed into interconnected subcontrollers K_s of the same structure as the plant, as shown in figure 5.2 (the formulas for doing so are the same as in Chapter 4 for SSS matrices).

In the following sections, we will exhibit a few examples of how well this works. In this chapter we will limit ourselves to continuous time LQR problems and an H_2 problem, since they have been considered as worthy problems in the literature [26], but the techniques from Part I could just as well be applied to H_∞ control or model order reduction, and also in discrete time. (In fact, in Chapter 8, in the course of solving LQR problems for 2-D homogeneous systems, we will have to solve 1-D L_r matrix discrete domain Lyapunov and Riccati equations and perform balanced truncation model order reductions)

5.2 Scalar Example: LQR of a Thin Heated Rod

Consider the heat conduction equation on an infinite bar:

$$\dot{T}(r) = \frac{\partial^2 T(r)}{\partial r^2} + u(r), \qquad y(r) = T(r)$$

for temperature T over continuous domain $-\infty < r < \infty$ with controlled input $u(r)$ and measured output $y(r)$. Discretizing the domain and approximating the derivative using finite difference: $\frac{\partial^2 T(r)}{\partial r^2} \approx T_{s-1} - 2T_s + T_{s+1}$ for $s \in \mathbb{Z}$, and then using the Fourier transform, the LQR problem reduces to solving the parametric Riccati equation: $A(z)^* X(z) + X(z) A(z) - X(z) B B^* X(z) + C^* C = 0, \qquad z \in \mathbb{T}$ where $A = z^{-1} - 2 + z$, $B = 1$, and $C = 1$. Because it is scalar, we can solve for the positive solution by hand: $X(z) = (z^{-1} - 2 + z) + \sqrt{z^{-2} - 4z^{-1} + 7 - 4z + z^2}$, which has an exponential spatial decay([26][57]) but is irrational, and hence does not lead to controllers with discrete subcontroller implementations.

To use our L-operator sign iteration technique, we first must form $\mathcal{S}$ realizations

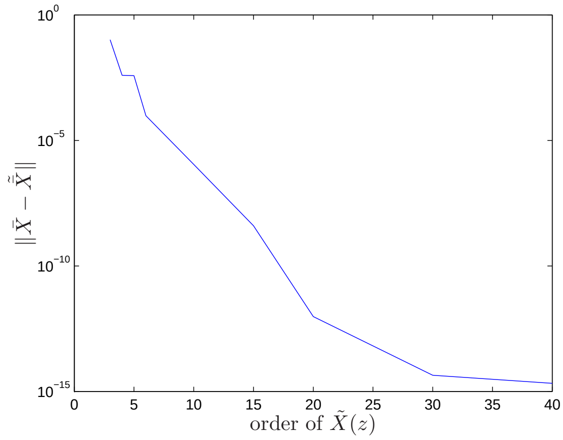


Figure 5.3: Accuracy of Riccati solutions

of $A(z)$, $B(z)$, $C(z)$:

$$\bar{A} = \mathcal{S}\{1,0,1,-2,1,0,1\}, \quad \bar{B} = \mathcal{S}\{0,0,0,1,0,0,0\}, \quad \bar{C} = \mathcal{S}\{0,0,0,1,0,0,0\}$$

Note that this realization step is only necessary because we are working with a discretized PDE. Had our system been provided in subsystem form as in equation (5.1), we could have pulled out the $\mathcal{S}$ realizations, $\bar{A}, \bar{B}, \bar{C}$ directly.

Following the procedures in Chapters 2 and 3, using balanced truncation model reductions after each step, we get convergence of the sign iteration after just $k = 6$ iterations, and in figure 5.3 we see that for larger rational orders, our iterative solution, $\tilde{\bar{X}}$, approaches the exact solution $\bar{X}$ in norm to almost machine precision (note that $\|\bar{X}\| = 1$). Due to the rational symbol of the resulting L-operator controller realizations, they will also admit convenient distributed implementations, as in figure 5.2.

5.3 Matrix Example: LQR for a Square Heated Bar

Of course, the previous example was too easy! Since it was scalar, we could just solve the Riccati equation for its exact solution by hand. So in this section we'll tackle a much more difficult problem, where the parametric Riccati equation will consist of large matrix blocks which cannot be solved by hand.

We consider the system:

$$\dot{T}(r,m,n) = (\frac{\partial^2}{\partial r^2} + \frac{\partial^2}{\partial m^2} + \frac{\partial^2}{\partial n^2})T(r,m,n) + u(r,m,n), \qquad y(r,m,n) = T(r,m,n)$$

of temperature T over continuous 3-dimensional spatial domain $-\infty < r < \infty$, $1 < m < 5$, $1 < n < 5$ with controlled input $u(r,m,n)$ and measured output

$y(r,m,n)$. Note that while our PDE is in three dimensions, we are still considering a 1-D $\mathcal{S}$ operator problem here, in that the system is only infinite in one direction. We will have to wait for Chapter 8 to treat systems which are infinite in multiple directions.

Discretizing the domain and approximating the derivative using finite difference, we get:

$$\begin{aligned} \dot{T}_{r,m,n} &= -6T_{r,m,n} + T_{r-1,m,n} + T_{r+1,m,n} + T_{r,m-1,n} + T_{r,m+1,n} + \\ &\quad T_{r,m,n-1} + T_{r,m,n+1} + u_{r,m,n} \\ y_{r,m,n} &= T_{r,m,n} \end{aligned}$$

over $r \in \mathbb{Z}$, $m \in \{1,2,3,4,5]\}$, $n \in \{1,2,3,4,5]\}$, with zero temperature boundary conditions at $m = 0,6$ and $n = 0,6$. If we then use the Fourier transform in the r direction, the LQR problem reduces to solving the parametric Riccati equation: $A(z)^*X(z) + X(z)A(z) - X(z)BB^*X(z) + C^*C = 0, \qquad z \in \mathbb{T}$ where $B = I$, $C = I$, and $A(z)$ is a 25×25 transfer matrix. Hence it would be prohibitively difficult to solve explicitly by hand, and we must use numerical techniques.

Compared to the thin rod in section 5.2, for this example much higher orders were necessary due to the increased state size, and it was impossible to compare the numerical solutions to the exact solution, as it could not be explicitly computed. So on a 1000 point discretization of $\mathbb{T}$, the finite dimensional complex problem $A(z_0)^*X(z_0)+X(z_0)A(z_0)-X(z_0)BB^*X(z_0)+C^*C = 0$ was solved using Matlab, and the solution was compared to our L_r-operator sign iteration produced solution $\tilde{X}(z)$ at each $z_0 \in \mathbb{T}$. The sign iteration took 7 steps to converge, and produced very close approximations: for a rational order of $\tilde{X}(z)$ of 100, we found $\max_{z_0} \|X(z_0) - \tilde{X}(z_0)\| = 6.7 \times 10^{-12}$, and for order 50, $\max_{z_0} \|X(z_0) - \tilde{X}(z_0)\| = 1.27 \times 10^{-6}$, where $\|X(z)\|_\infty \approx 0.5986$. So, even for problems with large local dimension, this numerical approach works very well.

5.4 Matrix Example: H_2 Control of a Platoon, with Comparison to Other Techniques

For our next example, we consider the problem of controlling the absolute and relative distances in an infinite-dimensional car platoon,assuming a second order model for each vehicle [125]. The dynamics are described by the following:

$$\begin{bmatrix} \dot{x}_i^1 \\ \dot{x}_i^2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_i^1 \\ x_i^2 \end{bmatrix} + \begin{bmatrix} 0 \\ q \end{bmatrix} w_i + \begin{bmatrix} 0 \\ g \end{bmatrix} u_i \tag{5.3}$$

for $i = -\infty \ldots + \infty$; x_i^1 is the position, x_i^2 is the velocity, u_i is the control input, w_i is the disturbance input, and q and g are constants which we assume to be both 1 for ease of presentation. The measured output will be the state itself, while as performance output z_i we choose the following:

$$z_i = \begin{bmatrix} u_i \\ f_1 x_i^1 \\ f_2\left(\frac{1}{2}x_{i-1}^1 - x_i^1 + \frac{1}{2}x_{i+1}^1\right) \end{bmatrix} \tag{5.4}$$

The performance outputs include the control effort, a symmetric measure of the relative position, and absolute position (for well-posedness, as explained in [125]), weighted by f^2 and f^1.

In order to apply our methods to this example, we simply write it as a set of interconnected subsystems as in (5.1):

$$\Sigma_s : \quad \begin{bmatrix} A & B^p & B^m & B^1 & B^2 \\ C^p & W^p & Z^m & L^p & V^p \\ C^m & Z^p & W^m & L^m & V^m \\ C^1 & J^p & J^m & D^{11} & D^{12} \\ C^2 & H^p & H^m & D^{21} & D^{22} \end{bmatrix} = \left[\begin{array}{cc|c|c|c|c} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & q_s^1 & g_s \\ \hline 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 \\ -f_s^2 & 0 & \frac{f_s^2}{2} & \frac{f_s^2}{2} & 0 & 0 \\ f_s^1 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right] \tag{5.5}$$

then resolve the interconnection variables to get the L_r operator system (as in (5.2)), in which we can apply our structure preserving iterative techniques.

5.4.1 Exact H_2 Solution

We consider the $\mathcal{H}_2$ optimal state feedback problem ([60] 14.8.1) and hence only have to solve one Riccati equation, which in our case reduces to:

$$\bar{A}^*\bar{P} + \bar{P}\bar{A} + \bar{C}_1^*\bar{C}_1 - \bar{P}\bar{B}_2^*\bar{B}_2\bar{P} = 0 \tag{5.6}$$

with controller $K = -\bar{B}_2^*\bar{P}$, which minimizes the spatiotemporal $\mathcal{H}_2$ norm (see the appendix of Chapter 3) from the disturbance w to the output z. By using a Fourier transform we convert the infinite dimensional L_r matrix Riccati equation into an infinite set of finite sized Riccati equations, parametrically dependent on $z \in \mathbb{T}$,for which the stabilizing solution is:

$$P(z) = \begin{bmatrix} \sqrt{2k\sqrt{k}} & \sqrt{k} \\ \sqrt{k} & \sqrt{2\sqrt{k}} \end{bmatrix}, \quad \text{with } k = \left(f_1^2 + f_2^2\left(\frac{3}{2} - z^{-1} - z + \frac{1}{4}z^{-2} + \frac{1}{4}z^2\right)\right) \tag{5.7}$$

Unfortunately, this solution is irrational, and hence doesn't have a nice discretely distributed controller implementation as in figure 5.2, and solving such problems by hand isn't an option for more complicated situations, as in the previous example, so we'll use our numerical methods.

5.4.2 Numerical Comparisons

In this example, we'll use our methods from Part I to numerically solve for the H_2 solution, and compare it to the exact optimal solution derived from (5.7). Also, we'll compare our result in terms of H_2 performance to two other methods of numerically computing distributed controllers: the decomposable system approach described in [10] and extended to infinite extent systems in [148] and the distributed LQR method described in [149] and extended to infinite extent systems in [148].

In our technique, we solve for some L_r operator $\bar{\tilde{X}}$ such that $\|\bar{X} - \bar{\tilde{X}}\| < \epsilon$, for some small ϵ and such that $\bar{\tilde{K}} = -\bar{B}_2^* \bar{\tilde{X}}$ will have a discrete distributed implementation as in Figure 5.2. Just as in the previous two examples, we used the $\mathcal{S}$ realization arithmetic of Chapter 2 and the sign iteration techniques of Chapter 3 to solve the Laurent matrix Riccati equation.

For comparison between our sign iteration method and the actual optimal controller, we gridded over 1000 points on the unit circle, solved the H_2 optimal control problem at each problem, then integrated over the unit circle to find the corresponding spatiotemporal H_2 optimal cost. The normalized Riccati error of our solution was less than 10^{-12} in norm (estimated over the same 1000 grid points), and the relative difference between our H_2 cost and the optimal was less than 10^{-15}, indicating that our approximate solution performs very very well.

While our technique gives a solution with almost identical performance to the optimal, this is not necessarily easy, nor true for all techniques. To demonstrate this, we compared the results of our technique with those for the other two methods for different values of the weighting parameter f_2 while keeping $f_1 = 1$. f_2 weights the penalty on the relative positions of the vehicles in the platoon, so for $f_2 \to 0$ the problem turns into decentralized control.

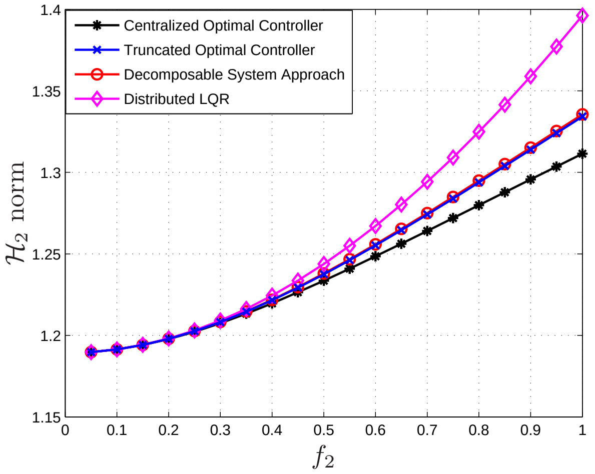


Figure 5.4: Closed-loop performance of the four different controllers.

The results are depicted in Figure 5.4, which shows the spatiotemporal $\mathcal{H}_2$ norm achieved with four different methods: the centralized optimal controller (identical to the performance of our method, to within 10^{-15}), the truncated version of this optimal controller, with only one off-diagonal band on each side (tri-diagonal), the decomposable system approach described in [10], and the distributed LQR controller described in [149].

The centralized optimal controller of course yields the best performance, while its truncated version is slightly better than the controller given by the decomposition approach, followed by the distributed LQR controller. We also see that as the coupling weight becomes small ($f_2 \to 0$) (leading to fully decentralized optimal controllers), the four controllers seem to converge to the same performance.

Thanks to Paolo Massioni and Tamas Keviczky, who collaborated with the author on this numerical example.

5.5 Conclusion

To summarize, in this chapter we've seen how spatially homogeneous distributed systems on a line lead to L_r matrices, and how this fact can be used with the machinery of Chapters 2 and 3 of Part I to efficiently synthesize non-conservative distributed controllers on a line. Both scalar and matrix examples were shown, and the methods were found to be very accurate and high performing, compared both with alternative methods and with the optimal solutions.

5.6 Appendix: Proof of Lemma 5.1

Well-posedness is equivalent to $\det(\begin{bmatrix} z^{-1}I & 0 \\ 0 & zI \end{bmatrix} - \begin{bmatrix} W^p & Z^p \\ Z^m & W^m \end{bmatrix}) \neq 0, \forall z \in \mathbb{T}$. [4]. First we need an intermediate result:

Lemma 5.2 $\det(\begin{bmatrix} z^{-1}I & 0 \\ 0 & zI \end{bmatrix} - \begin{bmatrix} W^p & Z^p \\ Z^m & W^m \end{bmatrix}) \neq 0, \forall z \in \mathbb{T} \Leftrightarrow$ *the matrix pencil:*

$$z \begin{bmatrix} I & -Z^p \\ 0 & W^m \end{bmatrix} - \begin{bmatrix} W^p & 0 \\ -Z^m & I \end{bmatrix} \tag{5.8}$$

has no eigenvalues on the unit circle. ***Proof:*** *Postmultiply* $\Delta - A_{ss}$ *by* $\begin{bmatrix} I & 0 \\ 0 & z^{-1}I \end{bmatrix}$ *then premultiply by* $\begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix}$ *to get (5.8). Since both of these multiplication matrices are invertible for* $z \in \mathbb{T}$*, these are equivalent.* □

Now for the constructive proof; to save space, we will only show the transformations without u_s and y_s. This method was first derived in [147]. **Proof:** Subsystem Σ_s in (5.1) is equivalent to:

$$\begin{bmatrix} I & 0 & -B^m \\ 0 & I & -Z^p \\ 0 & 0 & W^m \end{bmatrix} \begin{bmatrix} \dot{x}_s \\ v^p_{s-1} \\ v^m_s \end{bmatrix} = \begin{bmatrix} A & B^p & 0 \\ C^p & W^p & 0 \\ -C^m & -Z^m & I \end{bmatrix} \begin{bmatrix} x_s \\ v^p_s \\ v^m_{s+1} \end{bmatrix} \tag{5.9}$$

Now, according to Lemma 5.2, if and only if the system is well posed, then the matrix pencil has no eigenvalues on the unit circle, and then there exists Q and P matrices, both unitary, such that

$$Qz \begin{bmatrix} I & -Z^p \\ 0 & W^m \end{bmatrix} P - Q \begin{bmatrix} W^p & 0 \\ -Z^m & I \end{bmatrix} P = z \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} - \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix} \tag{5.10}$$

where $zT_{11} - S_{11}$ has eigenvalues inside the unit circle and $zT_{22} - S_{22}$ has eigenvalues outside the unit circle (this is called a generalized Schur decomposition, see e.g. [46]). Hence S_{22} and T_{11} are both invertible. If we premultiply equation (5.9) by $\begin{bmatrix} I & 0 \\ 0 & Q \end{bmatrix}$, and perform a state transformation $\begin{bmatrix} v^p_s \\ v^m_{s+1} \end{bmatrix} = P \begin{bmatrix} \hat{v}^p_s \\ \hat{v}^m_{s+1} \end{bmatrix}$, then from (5.9) we arrive at:

$$\begin{bmatrix} I & \clubsuit & \clubsuit \\ 0 & T_{11} & T_{12} \\ 0 & 0 & T_{22} \end{bmatrix} \begin{bmatrix} \dot{x}_s \\ \hat{v}^p_{s-1} \\ \hat{v}^m_s \end{bmatrix} = \begin{bmatrix} A & \clubsuit & \clubsuit \\ \clubsuit & S_{11} & S_{12} \\ \clubsuit & 0 & S_{22} \end{bmatrix} \begin{bmatrix} x_s \\ \hat{v}^p_s \\ \hat{v}^m_{s+1} \end{bmatrix} \tag{5.11}$$

Where the $\clubsuit$'s just represent variables who's values don't matter for the proof. Now if we solve the following discrete algebraic Sylvester equations:

$$\begin{aligned} S_{11}T_{11}^{-1}XT_{22}S_{22}^{-1} - X + (S_{11}T_{11}^{-1}T_{12} - S_{12})S_{22}^{-1} &= 0 \qquad (5.12) \\ T_{11}^{-1}S_{11}YS_{22}^{-1}T_{22} - Y + T_{11}^{-1}(S_{12}S_{22}^{-1}T_{22} - T_{12}) &= 0 \qquad (5.13) \end{aligned}$$

the unique solutions X and Y can be shown to satisfy the coupled continuous algebraic Lyapunov equations: $T_{11}Y + XT_{22} + T_{12} = 0$, $S_{11}Y + XS_{22} + S_{12} = 0$, and thus premultiplying the bottom two rows of (5.11) by $\begin{bmatrix} I & X \\ 0 & S_{22}^{-1} \end{bmatrix}$ and performing a state transformation: $\begin{bmatrix} \hat{v}^p_s \\ \hat{v}^m_{s+1} \end{bmatrix} = \begin{bmatrix} T_{11}^{-1} & Y \\ 0 & I \end{bmatrix} \begin{bmatrix} \tilde{v}^p_s \\ \tilde{v}^m_{s+1} \end{bmatrix}$ yields:

$$\begin{bmatrix} I & \clubsuit & \clubsuit \\ 0 & I & 0 \\ 0 & 0 & S_{22}^{-1}T_{22} \end{bmatrix} \begin{bmatrix} \dot{x}_s \\ \tilde{v}^p_{s-1} \\ \tilde{v}^m_s \end{bmatrix} = \begin{bmatrix} A & \clubsuit & \clubsuit \\ \clubsuit & S_{11}T_{11}^{-1} & 0 \\ \clubsuit & 0 & I \end{bmatrix} \begin{bmatrix} x_s \\ \tilde{v}^p_s \\ \tilde{v}^m_{s+1} \end{bmatrix} \tag{5.14}$$

or

$$\begin{bmatrix} \dot{x}_s \\ \tilde{v}^p_{s-1} \\ \tilde{v}^m_{s+1} \end{bmatrix} = \begin{bmatrix} \clubsuit & \clubsuit & \clubsuit \\ \clubsuit & S_{11}T_{11}^{-1} & 0 \\ \clubsuit & 0 & S_{22}^{-1}T_{22} \end{bmatrix} \begin{bmatrix} x_s \\ \tilde{v}^p_s \\ \tilde{v}^m_s \end{bmatrix} \tag{5.15}$$

which is in the correct form, with $\rho(S_{11}T_{11}^{-1})$, $\rho(S_{22}^{-1}T_{22}) < 1$, so it produces Laurent matrices bounded on l_2. $\square$

6

Homogeneous Distributed Systems with Boundary Conditions

In Chapter 5 we investigated doubly infinite extent, perfectly homogeneous systems, and in Chapter 4 we investigated finite, arbitrarily heterogeneous systems. However, there is a gap in these methods; often in real applications, the interconnected systems are mostly homogeneous, but with only heterogeneous boundary conditions. For such systems the class of perfectly homogeneous interconnected systems in Chapter 5 (shift invariant systems which are doubly infinite and have no boundary conditions) is too narrow (all real, non-periodic systems have boundary conditions), but the class of fully heterogeneous systems, as in Chapter 4, is often too broad. Such interconnected subsystems still result in lifted systems of SSS matrices, but they also have the additional structure of being 'almost Toeplitz'. To lump these into the heterogeneous class is to ignore this extra structure, and to miss out on an opportunity for faster computations.

It is well known that, in some ways, finite Toeplitz matrices are good approximations for bi-infinite Laurent matrices, and vice versa, and this fact has been previously used for each to try to calculate Riccati solutions involving the other [29][30]. The difficulty is the boundary conditions; the solution of a Riccati equation of Toeplitz matrices will often be Toeplitz in the middle, but there will be boundary effects at the ends (which of course do not show up in the bi-infinite Laurent operator Riccati solution). It's difficult to know *a priori* how far in these boundary effects will reach, and it's not clear how to nicely calculate them, even if the Toeplitz part in the middle is known, without solving the whole Riccati equation again (it's also ill-advised to simply truncate a Laurent operator controller, as it may be fragile). There also exist suboptimal design methods for systems described by symmetric Toeplitz or circulant systems [24][150], often in $\mathcal{O}(N)$ or $\mathcal{O}(N\log(N))$.

There are many definitions of 'almost Toeplitz' and 'Toeplitz-like' matrices in the literature, see e.g. [151][29] and the references therein. Our definition will be most closely related to that in [29], in which 'almost Toeplitz' matrices are basically defined as matrices in which the interior is Toeplitz, and the boundaries are bounded in their non-Toeplitz-ness by a power decay. But we will instead rely on our own definition of 'almost Toeplitz SSS matrices'(ATSSS) from Chapter 2, which are nicely physically motivated from interconnected systems (as we will see in the next section), and will provide an elegant specialization of the SSS control

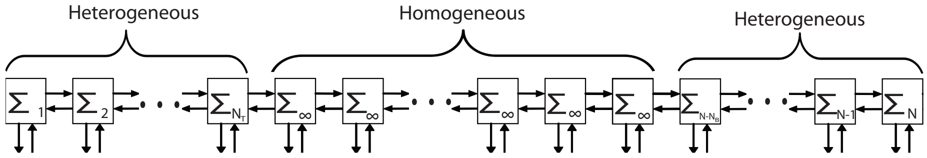


Figure 6.1: String interconnection of Homogeneous system with boundary conditions

methods of Chapter 4. Thereafter, in the same pattern as in Chapters 4 and 5, the structure preserving iterations of Chapter 3 will be used on the $\mathcal{O}(1)$ ATSSS arithmetic to analysize systems and synthesize controllers of the same structure, all in $\mathcal{O}(1)$. In section 6.2 we will see this demonstrated on a heat conduction example, with comparisons to the $\mathcal{O}(N)$ SSS methods of Chapter 4 and the $\mathcal{O}(N^3)$ methods of Matlab's '`h2syn`'.

6.1 Almost Toeplitz Systems

In this section, we will specialize the iterative structure preserving methods of Chapter 4 to a subclass of SSS systems, with additionally 'almost Toeplitz' structure, for even faster computations. Specifically, we assume that the subsystems have the following structure:

$$\Sigma_s : \quad \begin{bmatrix} \dot{x}_s \\ v^p_{s-1} \\ v^m_{s+1} \\ z_s \\ y_s \end{bmatrix} = \begin{bmatrix} A_s & B^p_s & B^m_s & B^1_s & B^2_s \\ C^p_s & W^p_s & 0 & L^p_s & V^p_s \\ C^m_s & 0 & W^m_s & L^m_s & V^m_s \\ C^1_s & J^p_s & J^m_s & D^{11}_s & D^{12}_s \\ C^2_s & H^p_s & H^m_s & D^{21}_s & D^{22}_s \end{bmatrix} \begin{bmatrix} x_s \\ v^p_s \\ v^m_s \\ w_s \\ u_s \end{bmatrix} \tag{6.1}$$

connected together as in figure 6.1. Each subsystem Σ_s has a realization as in (6.1), but we assume that only the small left and right boundaries, for $s \in \{1, 2, ...N_T\}$ and $s \in \{N - N_B, ..., N - 1, N\}$, respectively, are heterogeneous, whereas the interior, which we assume to be much larger ($N \gg N_B + N_T \in \mathcal{O}(1)$), is homogeneous; all Σ_s, $N_T < s < N - N_B$ are identical, and we will denote them as Σ_∞. We will also assume that W^p_s and W^m_s are stable in the interior; $\rho(W^p_\infty), \rho(W^m_\infty) < 1$ (otherwise the resulting lifted system matrices could become unbounded with growing N).

Just as in the previous chapters, assuming zero boundary inputs ($v^m_1 = 0$,$v^p_N = 0$), if the interconnection variables are resolved, we obtain the interconnected system:

$$\overline{\Sigma} : \quad \begin{bmatrix} \dot{\overline{x}} \\ \overline{z} \\ \overline{y} \end{bmatrix} = \begin{bmatrix} \overline{A} & \overline{B}_1 & \overline{B}_2 \\ \overline{C}_1 & \overline{D}_{11} & \overline{D}_{12} \\ \overline{C}_2 & \overline{D}_{21} & \overline{D}_{22} \end{bmatrix} \begin{bmatrix} \overline{x} \\ \overline{w} \\ \overline{u} \end{bmatrix} \tag{6.2}$$

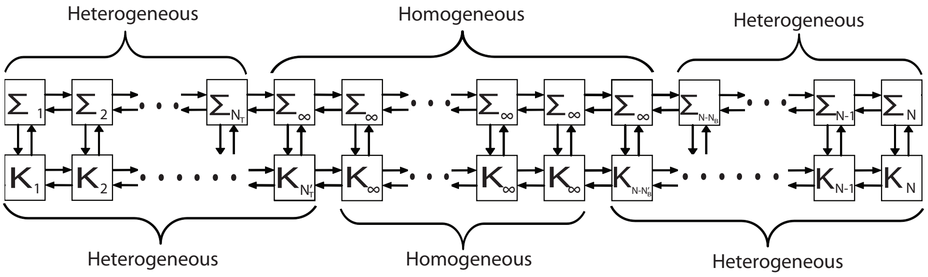


Figure 6.2: String interconnection of Homogeneous system with boundary conditions and ATSSS controller

where the $\overline{\text{overline}}$ indicates a 'lifted' variable; for vectors: $\overline{x} = \begin{bmatrix} x_1^T & x_2^T & \ldots & x_N^T \end{bmatrix}^T$, and the interconnected system matrices, e.g. $\overline{A}$, are structured. They will of course still have the SSS structure, as in Chapter 4, but the generators of the SSS matrices can be separated into three sections; the top, interior, and bottom. For example, for $\bar{A} = \mathcal{SSS}(B_s^m, W_s^m, C_s^m, A_s, B_s^p, W_s^p, C_s^p)$ above, the generators $(B_s^m, W_s^m, C_s^m, A_s, B_s^p, W_s^p, C_s^p)$ in the top, for $0 < s < N_T$, and in the bottom, for $N - N_B < s < N$ can be arbitrarily varying, but in the interior will be constant for all $N_T < s < N - N_B$, which we will denote by $(B_\infty^m, W_\infty^m, C_\infty^m, A_\infty, B_\infty^p, W_\infty^p, C_\infty^p)$. These are our 'almost Toeplitz Sequentially Semi Separable' (ATSSS) matrices of Chapter 2, which have an $\mathcal{O}(1)$ complexity arithmetic. Since all of the matrices in the realization (6.2) have this structure, structure preserving iterations can be used to compute controllers (e.g. H_2 or H_∞) in $\mathcal{O}(1)$, as discussed in Chapter 3 (assuming that the control problem satisfies the growth assumptions therein, e.g. the relevant Hamiltonion matrices $\in \mathcal{A} \forall N \in \mathbb{N}$). Since the iterations preserve the structure, the controller itself will also have a realization of ATSSS matrices, which is very convenient, as follows.

Just as for the L_r and SSS matrices of Chapters 4 and 5, if the above mentioned iterative methods are used to calculate a controller with ATSSS realization $\overline{K}$: $\begin{bmatrix} \dot{\overline{\xi}} \\ \overline{u} \end{bmatrix} = \begin{bmatrix} \overline{A}_K & \overline{B}_K \\ \overline{C}_K & \overline{D}_K \end{bmatrix} \begin{bmatrix} \overline{\xi} \\ \overline{y} \end{bmatrix}$ for the system in figure 6.1, then it can be shown that $\bar{K}$ can be redistributed into a homogeneous distributed system with boundary conditions as shown in figure 6.2 (see Chapter 4 for formulas for the redistribution step). Note that in figure 6.2 we have shown the sizes of the boundaries of the controller (N'_T and N'_B) to be one larger than the sizes of the boundaries of the system (N_T and N_B), but this is just an example for illustrative purposes. In general nothing is known *a priori* about the size of the boundaries of the controller relative to those of the original system, but in practice they are usually slightly larger.

6.2 Example: H_2 control of a heated rod

In Chapter 2 we developed an ATSSS arithmetic that was structure preserving, and which had computational complexity independent of the size of the interior

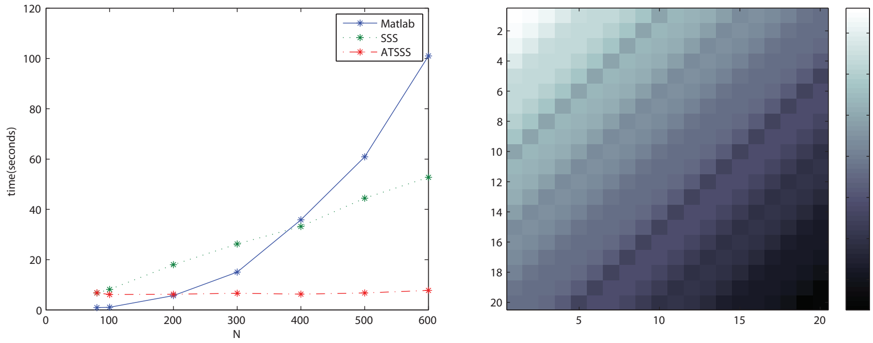


Figure 6.3: On the left is a comparison of the H_2 controller synthesis times. Shown on the right is the non-Toeplitz-ness of $\bar{A}_k$; the difference between the upper left corner of $\bar{A}_k$ and its interior. In MATLAB notation: $\log_{10}|\bar{A}_k(1:20, 1:20) - \bar{A}_k(N_T+1:N_T+20, N_T+1:N_T+20)|$

$N - N_B - N_T$, as long as it was large enough for the iterations to converge to steady state. Formally, using this arithmetic for fast H_2 controller synthesis is straightforward: the ATSSS arithmetic can be used in place of the standard SSS arithmetic to synthesize controllers with ATSSS structure and do ATSSS structure preserving model order reductions. For a computational example we consider a heated rod:

$$\dot{T}(r) = \frac{\partial^2 T(r)}{\partial r^2} + u(r) + w_1(r), \qquad z(r) = \begin{bmatrix} T(r) \\ u(r) \end{bmatrix}, \quad y(r) = T(r) + w_2(r)$$

with temperature T over continuous domain $0 < r < N$ with controlled input $u(r)$, measured output $y(r)$, cost $z(r)$, and disturbances $w_1(r)$, $w_2(r)$. Discretizing the domain and approximating the derivative using finite difference: $\frac{\partial^2 T(r)}{\partial r^2} \approx T_{s-1} - 2T_s + T_{s+1}$ for $s \in \{1...N\}$, gives us an Nth order system:

$$\dot{\bar{T}} = \bar{A}\bar{T} + \bar{u} + \bar{w}_1, \quad \bar{z} = \bar{C}_1\bar{T} + \bar{D}_{12}\bar{u}, \quad \bar{y} = \bar{T} + \bar{w}_2$$

where $\bar{A} = \mathcal{SSS}\{1, 0, 1, -2, 1, 0, 1\}$, $\bar{C}_1$, $\bar{D}_{12}$ have ATSSS structure, with $N_T = N_B = 1$. By varying the size of the problem, N, and computing an H_2 optimal controller using standard MATLAB QZ-based Riccati solvers (where all structure is ignored), the existing SSS structure preserving solver from Chapter 4 (where the 'almost Toeplitz' structure is ignored), and the new ATSSS structure preserving solver, we can show that this new technique is indeed much faster for very large N.

As expected, in figure 6.3 we see the cubic $\mathcal{O}(N^3)$ complexity of the MATLAB unstructured solver, the linear $\mathcal{O}(N)$ complexity of the SSS solver, and the constant complexity of the ATSSS solver. The linear complexity of the SSS approach becomes an advantage after about $N \approx 400$, and the ATSSS approach is advantageous after only $N \approx 200$. The small bit of variation in the computational times for the ATSSS solver were just random variations due to the load on the desktop computer, since the actual calculations performed were exactly the same for each N; the left and right boundary effects and interiors of the resulting controllers were all identical, the only difference was the size of the interior. Hence in practice, the ATSSS synthesis only needs to be performed once, and the resulting ATSSS generators of the controller can be applied to systems of all different interior lengths $N \in \{100, 200, 300, 400, 500, 600\}$ etc, just by correspondingly stretching the lengths of the interiors of the controller matrices.

In figure 6.3 is a comparison of the upper left corner vs. the interior of one of the controller matrices $\bar{A}_K$; as we see, the non-Toeplitzness is mostly limited to the first few rows and columns. The sizes of the non-Toeplitz boundaries, N_T and N_B of $\bar{A}_K$, $\bar{B}_K$, $\bar{C}_K$ were between 20 and 30, and thus did not grow very much in the iterative ATSSS calculations of the Riccati solvers, due to the exponential convergence proven in Chapter 2, but as we see in the figure, these boundaries probably could have been even smaller without sacrificing much accuracy. As for the overall accuracy of the solvers, for all the sizes, N, the closed loop H_2 norm of the controllers produced by the ATSSS and SSS methods were less than 10^{-13} different (relative error) from the closed loop H_2 norm of the controller produced by MATLAB's unstructured solver using 'H2syn'; so the ATSSS results were very close to optimal.

6.3 Conclusion

In this chapter we've shown how homogeneous interconnected subsystems on a line with arbitrary boundary conditions lead to systems with realizations of ATSSS matrices, thus allowing the use of our $\mathcal{O}(1)$ ATSSS arithmetic of Chapter 2 and structure preserving iterations of Chapter 3 to design controllers with ATSSS realizations in only $\mathcal{O}(1)$ complexity. In section 6.2 we demonstrated this result on a heat conduction example, showing that the $\mathcal{O}(1)$ complexity did work in practice, providing controllers arbitrarily close to optimal (within 10^{-13} in the H_2 norm) much faster than the $\mathcal{O}(N^3)$ of Matlab's unstructured solvers, or even the $\mathcal{O}(N)$ of our SSS solvers (which ignore the 'almost Toeplitz' structure).

Note that many of the other results from Chapter 4 could be easily specified to work in $\mathcal{O}(1)$ for the ATSSS case (e.g. H_∞ synthesis), while others might require more work (e.g. for SysID, the measured inputs and outputs would also need to be homogeneous in the interior, a nontrivial assumption).

7

Uncertain Distributed Systems

> “In theory there is no difference between theory and practice. In practice there is.”
>
> - attributed to Yogi Berra

Robustness analysis and synthesis, being considerably more difficult than ordinary control techniques for small scale systems, are even more of a challenge for distributed systems, although some extensions to robustness considerations of the distributed LMI techniques (see the introduction of Chapter 4) have appeared in the literature, e.g.[33][34][152].

In this chapter we will show how to extend some of the previous chapters' results to robustness analysis and controller synthesis using D scalings. For simplicity, our presentation will be for finite heterogeneous systems and SSS matrices, as in Chapter 4, but the methods and computational results could easily be extended to $\mathcal{S}$ realizations of L_r operators and ATSSS matrices, and thus also the types of systems described in Chapters 5 and 6.

7.1 Subsystem model/Interconnection Structure

The subsystem models considered will most generally consist of state space realizations:

$$\Sigma_s : \qquad \begin{bmatrix} \dot{x}_s \\ v^p_{s-1} \\ v^m_{s+1} \\ z^\Delta_s \\ z_s \\ y_s \end{bmatrix} = \begin{bmatrix} A_s & B^p_s & B^m_s & B^\Delta_s & B^1_s & B^2_s \\ C^p_s & W^p_s & 0 & G^p_s & L^p_s & V^p_s \\ C^m_s & 0 & W^m_s & G^m_s & L^m_s & V^m_s \\ C^\Delta_s & F^p_s & F^m_s & D^\Delta_s & L^\Delta_s & V^\Delta_s \\ C^1_s & J^p_s & J^m_s & J^\Delta_s & D^{11}_s & D^{12}_s \\ C^2_s & H^p_s & H^m_s & H^\Delta_s & D^{21}_s & D^{22}_s \end{bmatrix} \begin{bmatrix} x_s \\ v^p_s \\ v^m_s \\ w^\Delta_s \\ w_s \\ u_s \end{bmatrix} \tag{7.1}$$

where v^m_s and v^p_s are interconnections to other subsystems (see Figure 7.1), z_s and w_s are performance channels and disturbance inputs, y_s and u_s are measured outputs and controlled inputs, and z^Δ_s and w^Δ_s are uncertainty interconnections, with block diagonal structured uncertainty $w^\Delta_s = \Delta_s z^\Delta_s$ where $\Delta_s \in \mathbf{\Delta_s}$, which is not coupled to any of the other uncertainties. The W_s terms represent information

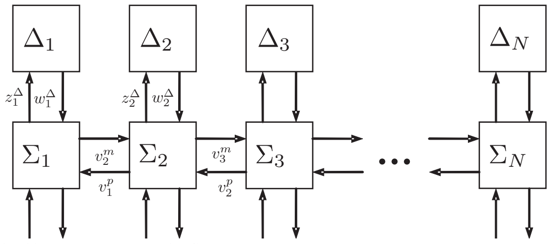


Figure 7.1: String interconnection with uncoupled uncertainties

feedthrough between subsystems Σ_{s+1} and Σ_{s-1}. This type of subsystem has appeared in [4] and associated papers, and also in Chapters 4, 5, and 6.

If N of these subsystems (7.1) are connected together in a string (see Figure 7.1) with zero boundary inputs ($v_1^m = 0$,$v_N^p = 0$) and the interconnection variables are resolved, we obtain the interconnected system:

$$\overline{\Sigma}: \quad \begin{bmatrix} \dot{\overline{x}} \\ \overline{z}_\Delta \\ \overline{z} \\ \overline{y} \end{bmatrix} = \begin{bmatrix} \overline{A} & \overline{B}_\Delta & \overline{B}_1 & \overline{B}_2 \\ \overline{C}_\Delta & \overline{D}_\Delta & \overline{L}_\Delta & \overline{V}_\Delta \\ \overline{C}_1 & \overline{J}_\Delta & \overline{D}_{11} & \overline{D}_{12} \\ \overline{C}_2 & \overline{H}_\Delta & \overline{D}_{21} & \overline{D}_{22} \end{bmatrix} \begin{bmatrix} \overline{x} \\ \overline{w}_\Delta \\ \overline{w} \\ \overline{u} \end{bmatrix} \tag{7.2}$$

with corresponding block diagonal lifted uncertainty: $\overline{w}_\Delta = \overline{\Delta}\overline{z}_\Delta$ where $\overline{\Delta} \in \overline{\boldsymbol{\Delta}} = diag(\boldsymbol{\Delta}_0, \boldsymbol{\Delta}_1, \ldots, \boldsymbol{\Delta}_N)$. The $\overline{\text{overline}}$ indicates a 'lifted' variable; for vectors: $\overline{x} = \begin{bmatrix} x_1^T & x_2^T & \ldots & x_N^T \end{bmatrix}^T$, and the interconnected lifted system matrices have 'Sequentially Semi Separable'(SSS) structure, just as in Chapter 4.

This formulation leads up to one of the fundamental problems in robust control research, extended to SSS distributed systems:

Problem 7.1 (Robust Performance Synthesis) *Given $\overline{\Sigma}$ and $\overline{\boldsymbol{\Delta}}$, find a stabilizing controller $\overline{K}$ with SSS structure, such that:*

$$\|\mathcal{F}_l(\mathcal{F}_u(\overline{\Sigma}, \overline{\Delta}_P), \overline{K})\|_\infty < 1, \quad \forall \overline{\Delta}_P \in \overline{\boldsymbol{\Delta}}_{\mathbf{P}} \tag{7.3}$$

where $\overline{\boldsymbol{\Delta}}_{\mathbf{P}} = \begin{bmatrix} \overline{\boldsymbol{\Delta}} & 0 \\ 0 & \boldsymbol{\Delta}_f \end{bmatrix}$, $\|\overline{\boldsymbol{\Delta}}_P\|_\infty < 1$, *and* $\boldsymbol{\Delta}_f \in \mathbb{C}^{n_{\overline{w}} \times n_{\overline{z}}}$.

We constrain the controller to have a realization of SSS matrices $\overline{K} : \left[\begin{array}{c|c} \overline{A}_K & \overline{B}_K \\ \hline \overline{C}_K & \overline{D}_K \end{array}\right]$ since such controllers can be readily distributed into the same interconnection structure as the system, as in figure 7.2(see Chapter 4 for discussion and formulas).

In the next section we will illustrate how the characteristics of SSS matrices can be exploited to allow for structure preserving $\mathcal{O}(N)$ algorithms for 'D-scalings', building up to $\mathcal{O}(N)$ controller-scaling iteration steps for synthesis of controllers with SSS structure, and thus a distributed implementation.

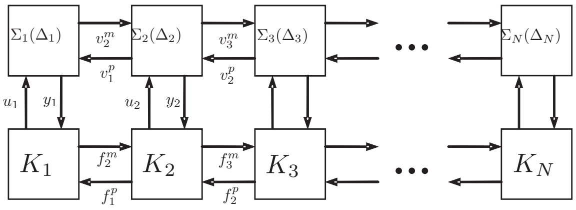


Figure 7.2: Controller implementation

7.2 Computational Methods

In previous chapters, iterative methods based on the structure preserving matrix sign iteration were used to solve Lyapunov and Riccati equations, check matrix stability, and ultimately, to synthesize nominal H_2 and H_∞ controllers with SSS, ATSSS, and rational Laurent structure for interconnected systems such as (7.2). In the following, we will adapt another matrix iteration, Osborne's method [153], to be SSS/ATSSS/Laurent structure preserving, for use with the H_∞ methods of Chapter 3 for SSS/ATSSS/Laurent structure preserving $D-K$ iterations.

7.2.1 Upper bound for the matrix structured singular value

For some complex matrix M and block diagonal class of norm bounded structured uncertainties $\|\mathbf{\Delta}\|_2 < 1$ we would like to solve the minimization problem:

$$\nu = \inf_{D \in \mathcal{D}} \|DMD^{-1}\|_2 \tag{7.4}$$

where $\mathcal{D}$ is some set of complex matrices such that D and $\Delta \in \mathbf{\Delta}$ commute. That is, we would like to compute some upper bound, ν for the matrix structured singular value(SSV), μ defined as $\mu = \max_{\Delta \in \mathbf{\Delta}} \rho(M\Delta)$, which often occurs in problems of robust stability and performance [60]. It is well known that this so called 'D-scalings' method of computing upper bounds for μ can be linearized through a variable change and then written as an LMI, but for the sake of computational efficiency we will use a different, more conservative method, that of Osborne [153], which has a long history of being used for fast SSV upper-bound computations [154]. In [155], Osborne's cyclic method is slightly modified into the so called 'EBE' method for faster MATLAB implementation, and is applied to upper-bound computation for complex and mixed uncertainties. It can further be shown that Algorithm 1 is equivalent to this method,

Algorithm 7.1 (EBE: Matrix Version)

```
$M_0 = M$;
for k = $1, 2, 3, \ldots$
    $R_k^2 = \mathfrak{D}_o([M_k - \mathfrak{D}_o(M_k)]\,[M_k - \mathfrak{D}_o(M_k)]^T)$
    $S_k^2 = \mathfrak{D}_o([M_k - \mathfrak{D}_o(M_k)]^T\,[M_k - \mathfrak{D}_o(M_k)])$
    $D_k = (S_k)^{1/2}(R_k)^{-1/2}$
    $M_{k+1} = D_k M_k D_k^{-1}$
end
```

only in a matrix format, where $\lim_{k\to\infty} \|D_k M D_k^{-1}\|_F = \inf_{D\in\mathcal{D}} \|DMD^{-1}\|_F$ and $\mathfrak{D}_o(X) = diag(X_{(11)}, X_{(22)}, \ldots X_{(NN)})$ for some matrix X where $X_{(ij)}$ represents the element on the i^{th} row and j^{th} column of matrix X. We can also define a modified operator $\mathfrak{D}_o^{\Delta}()$ to be similar to $\mathfrak{D}_o()$, except operating in a way as conformably partitioned with $\mathbf{\Delta}$. For a block diagonal $\mathbf{\Delta}$ with scalar complex blocks, complex repeated blocks, and complex full blocks, for N blocks in total, we define $\mathfrak{D}_o^{\Delta}(Z) = diag(\mathfrak{F}(Z_{(11)}), \mathfrak{F}(Z_{(22)}), \ldots, \mathfrak{F}(Z_{(NN)}))$ where $Z_{(ss)}$ is the block on the diagonal of Z of appropriate size to match the corresponding uncertainty block Δ_j, and

$$\mathfrak{F}(X_{(ii)}) = \begin{cases} \|X_{(ii)}\|_F I^{n\times n} : & \Delta_{(ii)} \in \mathbb{C}^{n\times n} | n > 1 \\ X_{(ii)} : & \Delta_{(ii)} \in pI^{n\times n} | p \in \mathbb{C} \end{cases}$$

Using $\mathfrak{D}_o^{\Delta}()$ in the above algorithm (and with $(S_k)^{1/2}(R_k)^{-1/2}$ interpreted as the matrix square root) extends the EBE method to uncertainties with full and repeated complex blocks(without the guarantee on optimality). In addition, since we have rewritten the EBE iteration only in terms of matrix-matrix multiplications, $\mathfrak{D}_o^{\Delta}()$ operations, and square roots and inverses of block diagonal matrices, it can clearly be implemented in $\mathcal{O}(N)$ for $\overline{M}$ with SSS structure, and further, the optimal D scalings will also have SSS structure(trivially, since they are block diagonal). Note that without exploiting the SSS structure of $\overline{M}$, each iteration of Algorithm 1 would be $\mathcal{O}(N^3)$ complexity, and that the computation of (7.4) using standard LMI solvers is $\mathcal{O}(N^{\sim 6})$.

7.2.2 D-K iteration for Decoupled Uncertainty Structure

In this subsection, we will describe how the structure preserving EBE method may be used with the SSS H_∞ methods of Chapter 4 to perform efficient D-K iterations to attack Problem 7.1.

All of the robust control results that we will use are standard (our innovation will be the $\mathcal{O}(N)$ complexity and preserving the SSS structure through each step), but the reader might want to consult [60] for an explanation of D-K iteration, which can be summarized in 4 steps:

Solve an H_∞ problem for $\overline{\Sigma}$

More precisely, we would like to minimize: $\|\mathcal{F}_l(\overline{\Sigma}, \overline{K})\|_\infty$ over all stabilizing controllers $\overline{K} \in \mathcal{K}$. This step can be efficiently completed using the Riccati based γ-iteration method for SSS matrices as demonstated in Chapter 4. However, in this case, the system(7.2) will have uncertainty channels leading to generalized performance output $\begin{bmatrix}\overline{z}_\Delta^T & \overline{z}^T\end{bmatrix}^T$ and generalized disturbance input: $\begin{bmatrix}\overline{w}_\Delta^T & \overline{w}^T\end{bmatrix}^T$ which need to be 'shuffle permuted', via the SSS version of Lemma 2.6 to avoid block arithmetic. We also note that it can be shown that the stabilizability, detectability, regularity, and invariant zeros assumptions for H_∞ synthesis will hold for the system with uncertainty channels (7.2) if they hold for the nominal system without.

Compute $\overline{D}$-scalings on a frequency grid

At each point on a frequency grid $\omega_k \in [0, \omega_{max}]$: solve:

$$\inf_{\overline{D}_P^{\omega_k} \in \overline{\mathcal{D}}_P} \|(\overline{D}_P^{\omega_k} \mathcal{F}_l(\overline{\Sigma}(i\omega_k), \overline{K}(i\omega_k))(\overline{D}_P^{\omega_k})^{-1}\|_2 \tag{7.5}$$

where $\overline{\mathcal{D}}_P \overline{\boldsymbol{\Delta}}_P = \overline{\boldsymbol{\Delta}}_P \overline{\mathcal{D}}_P$. It can easily be shown that at each frequency $i\omega_k$, $\mathcal{F}_l(\overline{\Sigma}(i\omega_k), \overline{K}(i\omega_k))$ will be a 2×2 block complex matrix, with each block having SSS structure. Therefore a sub-optimal solution to this step can computed efficiently by using Osborne's method(Section 7.2.1) extended for $\overline{\boldsymbol{\Delta}}_P$ at each frequency point ω_k.

Interpolate to form $\overline{D}_P(\omega), \overline{D}_P^{-1}(\omega) \in RH^\infty$

Due to the structure of $\overline{\boldsymbol{\Delta}}_P$, each scaling will have the structure:

$$\overline{D}_P^{\omega_k} = diag(D_1^{\omega_k}, D_2^{\omega_k}, ..., D_N^{\omega_k}, I_{n_{\overline{z}} \times n_{\overline{z}}})$$

where the scalar corresponding to $\boldsymbol{\Delta}_f$ has just been absorbed by the other values at each frequency [60]. Due to this block diagonal structure of each matrix $\overline{D}_P^{\omega_k}$, the corresponding interpolation problem: to create a stable rational proper transfer function $\overline{D}_P(\omega)$ with stable inverse, such that $|\overline{D}_P(i\omega_k)| \approx \overline{D}_P^{\omega_k}$ can just be split up into N small uncoupled interpolation problems: find $D_s(\omega)$ such that $|D_s(i\omega_k)| \approx D_s^{\omega_k}$, for each $s \in \{1, 2, ..., N\}$. Each interpolation problem will be of small size $n_{z_s^\Delta} \times n_{w_s^\Delta}$, and may be computed efficiently with available software [156].

Set $\overline{\Sigma} \to \overline{D}_P \overline{\Sigma} \overline{D}_P^{-1}$ and return to step 1)

Fortunately, the block diagonal structure of $\overline{D}_P$ will allow this step to be done on a local level $\Sigma_s \to D_s \Sigma_s D_s^{-1}$ using minimal realizations of D_s and D_s^{-1}. This step will cause the state dimension of the realization of each subsystem Σ_s to grow by $n_{x_s} \to n_{x_s} + m_{D_s} + m_{D_s^{-1}}$ where m_{D_s} and $m_{D_s^{-1}}$ are the McMillan degrees of each

D_s and D_s^{-1}(See the appendix in section 7.6 for the exact changes made to each Σ_s), but will preserve the SSS structure of $\overline{\Sigma}$, allowing us to return to step 1) and begin another round of upper bound minimization. We also note here that since $\overline{D}_P$ is stable and has a stable inverse, the H_∞ synthesis assumptions about the stabilizability, detectability, regularity and invariant zeros will hold for the newly scaled system $\overline{D}_P\overline{\Sigma}\overline{D}_P^{-1}$ if they held for the old $\overline{\Sigma}$.

The D-K iteration method described above can be continued until the D-scalings approximately converge, the user gives up, or (7.3) is satisfied. After each iteration,

$$\|\overline{D}_P\mathcal{F}_l(\overline{\Sigma},\overline{K})\overline{D}_P^{-1}\|_\infty \tag{7.6}$$

can be efficiently($\mathcal{O}(N)$) upper bounded via SSS arithmetic used with the sign iteration in Lemma 3.8.

Remark 7.1 *We also note that sometimes we would like to find the largest bounded uncertainty set such that we have a certain guaranteed robust performance, or, given a bounded uncertainty set, find the optimal robust performance. Both of these problems can be approached by rescaling $\mathcal{F}_l(\overline{\Sigma},\overline{K})$ and repeating the μ-upper bound analysis[94].*

Remark 7.2 *To make a rigorous claim of $\mathcal{O}(N)$ complexity for $D-K$ iteration steps as we have outlined them, it would also be necessary to assume bounds on the number of EBE iterations, and the fine-ness of the necessary frequency gridding in the interpolation step. Such assumptions would doubtlessly be overly restrictive, and we will only note that in practice, the EBE matrix iteration converges in 2-4 steps, and for reasonably damped systems < 100 frequency points ω_k seem to be sufficient.*

7.3 Other uncertainty structures

As we have seen, the block diagonal uncertainty structure of $\overline{\Delta}$ is absolutely essential for allowing us to perform D-K iterations. The most obvious source of such a decoupled uncertainty structure is a 'local' uncertainty within each subsystem Σ_s, thus it initially appears that we cannot model uncertainties in the interconnections between systems. However, by utilizing some classic methods from robust control, it is often possible to decouple uncertain interconnections.

7.3.1 Parametric Uncertainty in Σ_s

For example, suppose we have k_s real norm-bounded parametric uncertainties $|\delta_s^i| < 1$ for $1 \leq i \leq k_s$ occurring in our subsystem state space model (7.1)(leaving out the uncertainty channels), which we repartition as:

$$\begin{bmatrix} \dot{x}_s \\ \hat{y}_s \end{bmatrix} = \underbrace{\begin{bmatrix} A_s(\delta_s) & \hat{B}_s(\delta_s) \\ \hat{C}_s(\delta_s) & \hat{D}_s(\delta_s) \end{bmatrix}}_{P_s(\delta_s)} \begin{bmatrix} x_s \\ \hat{u}_s \end{bmatrix}$$

where $\hat{y}_s = \begin{bmatrix} v^p_{s-1} \\ v^m_{s+1} \\ z_s \\ y_s \end{bmatrix}$ and $\hat{u}_s = \begin{bmatrix} v^p_s \\ v^m_s \\ w_s \\ u_s \end{bmatrix}$ are generalized outputs and inputs, and the matrices are correspondingly grouped. If we further assume that the uncertainties occur affinely:

$$P_s(\delta_s) = P^0_s + \sum_{i=1}^{k_s} \delta^i_s P^i_s$$

then using a well known method [94] we may pull out the uncertainties and create an efficient model of the form:

$$\begin{bmatrix} \dot{x}_s \\ \hat{y}_s \\ z^\Delta_s \end{bmatrix} = \begin{bmatrix} A^0_s & \hat{B}^0_s & L^1_s \\ \hat{C}^0_s & \hat{D}^0_s & L^2_s \\ R^1_s & R^2_s & 0 \end{bmatrix} \begin{bmatrix} x_s \\ \hat{u}_s \\ w^\Delta_s \end{bmatrix}$$

which is of the same form as (7.1)(to a permutation) and with uncoupled uncertainty connections $w^\Delta_s = \Delta_s z^\Delta_s$, as in Fig (7.1). Each Δ_s is diagonal, with k_s real repeated blocks, each of dimension $rank(P^i_s)$ for $1 \leq i \leq k_s$. This factorization technique for each subsystem will require k_s SVD's of matrices of size $(n_{x_s} + n_{\hat{y}_s}) \times (n_{x_s} + n_{\hat{u}_s})$, and must be performed on each of the N subsystems, and thus will only be $\mathcal{O}(N)$ for the interconnected system.

This procedure is just a trivial extension of a method used for ordinary state space models, but it allows us to model parametric uncertainties in *all* of the subsystem components, including the communication terms between individual subsystems.

7.3.2 Dynamically Coupled Uncertainties

The idea of decoupling can also be used on interconnected dynamic uncertainties, such as the set of interconnected subsystems:

$$\Sigma^c_s : \begin{bmatrix} \dot{x}_s \\ v^p_{s-1} \\ v^m_{s+1} \\ z^\delta_s \end{bmatrix} = \begin{bmatrix} A_s & B^p_s & B^m_s & B^\delta_s \\ C^p_s & W^p_s & 0 & G^m_s \\ C^m_s & 0 & W^m_s & G^p_s \\ C^\delta_s & F^p_s & F^m_s & 0 \end{bmatrix} \begin{bmatrix} x_s \\ v^p_s \\ v^m_s \\ w^\delta_s \end{bmatrix} \qquad (7.7)$$

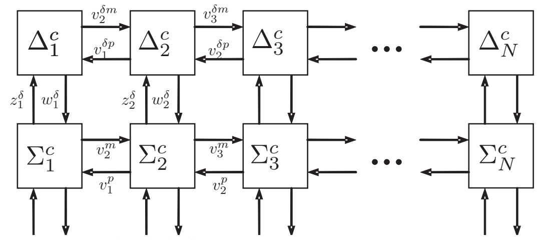


Figure 7.3: Coupled uncertainty

(neglecting the inputs u_s, w_s and outputs z_s, y_s) with coupled dynamic uncertainties:

$$\Delta_s^c : \quad \begin{bmatrix} w_s^\delta \\ v_{s-1}^{\delta p} \\ v_{s+1}^{\delta m} \end{bmatrix} = \begin{bmatrix} \Delta_s^A & \Delta_s^{Bp} & \Delta_s^{Bm} \\ \Delta_s^{Cp} & \Delta_s^{Wp} & 0 \\ \Delta_s^{Cm} & 0 & \Delta_s^{Wm} \end{bmatrix} \begin{bmatrix} z_s^\delta \\ v_s^{\delta p} \\ v_s^{\delta m} \end{bmatrix}$$

interconnected as in figure 7.3. By pulling the uncertainty couplings ($v_s^{\delta p}$, $v_s^{\delta m}$) down into the Σ_s subsystem interconnections using identity feedthrough terms, it is not difficult to decoupled this interconnected system into the form of figure 7.1 with uncertainty structure $\Delta_s = diag(\Delta_s^A, \Delta_s^{Bp}, \Delta_s^{Cp}, \Delta_s^{Wp}, \Delta_s^{Bm}, \Delta_s^{Cm}, \Delta_s^{Wm})$.

This is not the most general uncertainty structure imaginable, but nevertheless any subsystem can be coupled with any other through a series of dynamic uncertainties. This could also be generalized to dynamic weightings on the uncertainties, with some increased complexity in the resulting Σ_s.

Such models could be very useful for designing controllers for systems with dynamic uncertain or nonlinear interconnections, such as the aerodynamic draft effect for formation flight experiments and car platooning. Another possible application could be *a posteriori* closed loop robust stability analysis for uncertainties in the distributed controller, i.e. distortion, lags, or delays in the sub-controller intercommunication channels f_s^m, f_s^p (see [33][157] for such considerations in the LMI framework).

7.4 Example: $\mathcal{O}(N)$ D-K iterations

We consider Example 'A' of [4] restricted to 1 spatial dimension, made spatially heterogeneous, and with parametric uncertainty. To demonstrate the application to heterogeneous systems, the system parameters will be chosen to vary randomly in space. This is not meant to represent systems actually encountered in practice, but instead to demonstrate that there is no loss of algorithmic performance from the spatially invariant case to the very heterogeneous case.

For each subsystem, Σ_s we have:

$$\begin{aligned}
\ddot{x}_s &= \frac{1}{4}(k_s^1 + \delta_s k_s^2)(x_{s-1} + 2x_s + x_{s+1}) + \frac{1}{4}q_s^1(w_{s-1}^1 - 2w_s^1 + w_{s+1}^1) + b_s u_s \\
z_s^1 &= \frac{1}{4}f_s^1(x_{s-1} - 2x_s + x_{s+1}), \qquad z_s^2 = f_s^2 u_s, \qquad y_s = c_s x_s + q_s^2 w_s^2
\end{aligned}$$

which can be rewritten in the form of (7.1) as:

$$\begin{aligned}
A_s &= \begin{bmatrix} 0 & 1 \\ \frac{k_s^1}{2} & 0 \end{bmatrix}, \quad B_s^m = B_s^p = \begin{bmatrix} 0 & 0 \\ \frac{k_s^1}{4} & 0 \end{bmatrix}, \quad B_s^2 = \begin{bmatrix} 0 \\ b_s \end{bmatrix} \\
B_s^\Delta &= \begin{bmatrix} 0 \\ \frac{k_s^2}{4} \end{bmatrix}, \quad B_s^1 = \begin{bmatrix} 0 & 0 \\ \frac{-q_s^1}{2} & 0 \end{bmatrix}, \quad C_s^p = C_s^m = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \quad C_s^1 = \begin{bmatrix} \frac{-f_s^1}{2} & 0 \\ 0 & 0 \end{bmatrix} \\
L_s^m &= \begin{bmatrix} \frac{q_{s-1}^1}{k_{s-1}^1} & 0 \\ 0 & 0 \end{bmatrix}, \quad L_s^p = \begin{bmatrix} \frac{q_{s+1}^1}{k_{s+1}^1} & 0 \\ 0 & 0 \end{bmatrix}, \quad D_s^{12} = \begin{bmatrix} 0 \\ f_s^2 \end{bmatrix}, \quad J^p = J^m = \begin{bmatrix} 0 & \frac{f_s^1}{4} \\ 0 & 0 \end{bmatrix} \\
C^\Delta &= \begin{bmatrix} 2 & 0 \end{bmatrix}, \quad F^P = F^m = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad D_s^{21} = \begin{bmatrix} 0 & q_s^2 \end{bmatrix}, \quad C_s^2 = \begin{bmatrix} c_s & 0 \end{bmatrix} \\
&,
\end{aligned}$$

with the rest of the terms 0 and with $w_s^\Delta = \delta_s z_s^\Delta$. In this example, $N = 50$ for the purposes of demonstration, so $s \in \{1, 2, ..., 50\}$.

7.4.1 Spatially Homogeneous, Nominal

First, all of the coefficients $(k_s^1, b_s, q_s^1, q_s^2, f_s^1, f_s^2, c_s)$ are set to 1 $\forall s \in \{1, 2, ...N\}$, except $k_s^2 = 0$, so this represents a spatially invariant system with no uncertainty. MATLAB's `Hinfsyn`, given a $\gamma_{tol} = 10^{-2}$ computes a centralized controller with closed loop H_∞ norm $\gamma_M = 3.9994$, while the SSS based solver from Chapters 2 and 3, using SSS orders $w_u = w_l = 10$ for all iterative computations, found a distributed controller with $\gamma_{SSS} = 4.0006$ with communication interconnection(f_s^m, f_s^p in Figure 7.2) size of only 6.

7.4.2 Spatially Heterogeneous, Nominal

For this demonstration, the nominal case is still considered: $k_s^2 = 0 \forall s$, but now the coefficients will be randomly spatially varying, as in Chapter 4. At each $s \in \{1, 2, ...N\}$ each coefficient will take its original value plus a sample from a zero-mean normal distribution with a standard deviation one-third that of the original: $\mathcal{N}(0, \frac{1}{3})$. For example, $k_s^1 = 1 + \frac{1}{3}\mathcal{N}(0, 1)$ independently sampled at each $s \in \{1, 2, ...N\}$.

Of course, the resulting closed loop norm will vary depending on the specific values, but we will just give the results for one example deemed representative. For the same tolerance as before, MATLAB produced a centralized controller with closed loop H_∞ norm $\gamma_M = 11.1108$, while the SSS based method, using the same iterative order size as before, found a distributed controller with $\gamma_{SSS} = 11.1178$ and communication size again of only 6. Thus we see that this interconnected system is quite sensitive to heterogeneity, but that the SSS matrix H_∞ technique adds no unnecessary conservatism.

7.4.3 Heterogeneous, Uncertain

In this section, we again consider the spatially varying case, but with uncertainty. All of the coefficients will be held identical from the previous case, except the uncertainty coefficient: $k_s^2 = \frac{1}{100}\mathcal{N}(0,1)$ independently sampled at each $s \in \{1, 2, ...N\}$. After rescaling, it was found that the best upper bound on robust performance from small gain theory alone was $\gamma = 52.4$, but D-scalings computed using the above EBE based SSS method pushed the robust performance upper bound down to $\nu_{SSS} = 13.5$ after 2 iterations, while Matlab's `dkit` achieved $\nu_M = 16.22$ after 9 iterations (note that due to the non-convexity of the problem, we shouldn't necessarily expect $\nu_{SSS} \geq \nu_M$).

These results show us that even a little bit of uncertainty in this system may cause alot of trouble, but that scaled small gain theorem can get rid of much of the conservatism, and the modified EBE method works very well.

For reference to the algorithmic discussions, the Riccati sign iterations usually converged in only 5 steps, the resulting residuals were of order 10^{-7}. The EBE iterations in this example took only 2 or 3 iterations to converge at each frequency point, but since they are in complex arithmetic, the D-scaling at each step took approximately the same amount of time as the H_∞ synthesis, which scales as $\mathcal{O}(N)$(as shown in Chapter 4).

7.5 Conclusions

In summary, in this chapter we have demonstrated how the EBE [155] method of SSV upper bounding can be made SSS structure preserving and used to extend the $\mathcal{O}(N)$ H_∞ synthesis methods of Chapter 4 to robust synthesis for decoupled uncertainty structures using $D-K$ iterations. A few methods of finding models of such distributed systems with decoupled uncertainties were outlined, which suggest that this technique might be applicable to a broad set of distributed systems with uncertain or non-linear interconnections. For example, they could perhaps be used to capture the errors generated by discretizing PDE's for control, thus guaranteeing that the discrete domain controller would stabilize the continuum system, a direction to be investigated in the future.

The computational method was demonstrated on an extended example from the literature, in which it performed comparably to Matlab μ-synthesis routines in terms of performance, but scales very favorably in computational complexity ($\mathcal{O}(N)$).

In the bigger picture, each step of Algorithm 1 could also be easily performed if $\bar{M}$ were an L_r operator or an ATSSS matrix, so these robust analysis and synthesis techniques can also be extended to systems that are infinite and homogenous (Chapter 5), and finite but homogeneous with boundary conditions (Chapter 6).

7.6 Appendix: Absorbing D-scalings

If we create minimal state space realizations of each $D_s(\omega)$ and $D_s^{-1}(\omega)$:

$$D_s : \left[\begin{array}{c|c} A_s^L & B_s^L \\ \hline C_s^L & D_s^L \end{array}\right], \qquad D_s^{-1} : \left[\begin{array}{c|c} A_s^R & B_s^R \\ \hline C_s^R & D_s^R \end{array}\right]$$

then $\overline{D}$ and $\overline{D}^{-1}$ can be absorbed into $\overline{\Sigma}$ with a realization using subsystems of the form of (7.1) where the following variables have been changed:

$$x_s \to \begin{bmatrix} x_s^R \\ x_s \\ x_s^L \end{bmatrix}, \quad A_s \to \begin{bmatrix} A_s^R & 0 & 0 \\ B_s^\Delta C_s^R & A_s & 0 \\ B_s^L D_s^\Delta C_s^R & B_s^L C_s^\Delta & A_s^L \end{bmatrix}, B_s^m \to \begin{bmatrix} 0 \\ B_s^m \\ B_s^L F_s^m \end{bmatrix},$$

$$B_s^p \to \begin{bmatrix} 0 \\ B_s^p \\ B_s^L F_s^p \end{bmatrix}, \quad B_s^2 \to \begin{bmatrix} 0 \\ B_s^2 \\ B_s^L V_s^\Delta \end{bmatrix}, \quad B_s^\Delta \to \begin{bmatrix} B_s^R \\ B_s^\Delta D_s^R \\ B_s^L D_s^\Delta D_s^R \end{bmatrix}, \quad B_s^1 \to \begin{bmatrix} 0 \\ B_s^1 \\ B_s^L L_s^\Delta \end{bmatrix}$$

$$C_s^m \to \begin{bmatrix} G_s^m C_s^R & C_s^m & 0 \end{bmatrix}$$

$$G_s^m \to \begin{bmatrix} G_s^m D_s^R \end{bmatrix}, \quad C_s^p \to \begin{bmatrix} G_s^p C_s^R & C_s^p & 0 \end{bmatrix}$$

$$G_s^p \to \begin{bmatrix} G_s^p D_s^R \end{bmatrix}, \qquad C_s^2 \to \begin{bmatrix} H_s^\Delta C_s^R & C_s^2 & 0 \end{bmatrix}$$

$$F_s^m \to \begin{bmatrix} D_s^L F_s^m \end{bmatrix}, \quad F_s^p \to \begin{bmatrix} D_s^L F_s^p \end{bmatrix}, \quad H_s^\Delta \to \begin{bmatrix} H_s^\Delta D_s^R \end{bmatrix}$$

$$L_s^\Delta \to \begin{bmatrix} D_s^L L_s^\Delta \end{bmatrix}, \quad V_s^\Delta \to \begin{bmatrix} D_s^L V_s^\Delta \end{bmatrix}, \quad J_s^\Delta \to \begin{bmatrix} J_s^\Delta D_s^R \end{bmatrix}$$

$$C_s^1 \to \begin{bmatrix} J_s^\Delta C_s^R & C_s^1 & 0 \end{bmatrix}, \qquad D_s^\Delta \to \begin{bmatrix} D_s^L D_s^\Delta D_s^R \end{bmatrix}$$

$$C_s^\Delta \to \begin{bmatrix} D_s^L D_s^\Delta C_s^R & D_s^L C_s^\Delta & C_s^L \end{bmatrix}$$

8 Multi-D Distributed Systems

So far in this thesis we've come up with very fast, nice, and nearly optimal solutions to control systems distributed in one spatial dimension. However, in nature and engineering, often systems are distributed in 2 or 3 spatial dimensions, such as in irrigation networks [14], the development of tissues from stem-cells [15], multi-cellular gene regulatory networks[158], large adaptive telescope mirrors [11], biochemical reactions [16], wind turbine farms [17], and computer animations [159].

Some distributed control techniques trivially extend to multiple dimensions. The LMI methods of [4][31][32][33][10][34] all work easily in multiple spatial dimensions, or even arbitrary interconnection topologies. The structured matrix methods of [27] and [28] also readily work in multiple spatial dimensions(and perhaps on other interconnection structures to), although it is unknown if they are able to synthesize distributed controllers. The ideas in [26][143] also apply readily to multiple dimensions, even if the computation methods are not yet fully realized.

A drawback of many of these results is the computational complexity (basically the same problem as in 1 dimension). For example, unstructured Riccati based techniques can solve optimal control problems for subsystems of size m linked together on a 2-D grid of N by M points in $\mathcal{O}(m^3N^3M^3)$ complexity, while the structured LMI methods discussed above take something like $\mathcal{O}(m^6N^3M^3)$ complexity. For large problems, e.g. $N, M > 1000$ this is simply too slow.

In this Chapter, we will extend the results of Chapters 4, 5, and 6 to multiple spatial dimensions. The results of this chapter can be applied to any of the SSS, L_r, or ATSSS structures, or even combinations of them; e.g. a 2-D system that is heterogeneous in one direction and homogeneous in another, but we will just discuss the situation for multi-D Laurent operators, as they are simplest. The basic result of this chapter is showing how the multilevel operators discussed in section 3.5 can be used to represent distributed systems, after which the iterative methods of Chapter 3 can be used with the structure preserving arithmetic of Chapter 2 to synthesize similarly structured controllers.

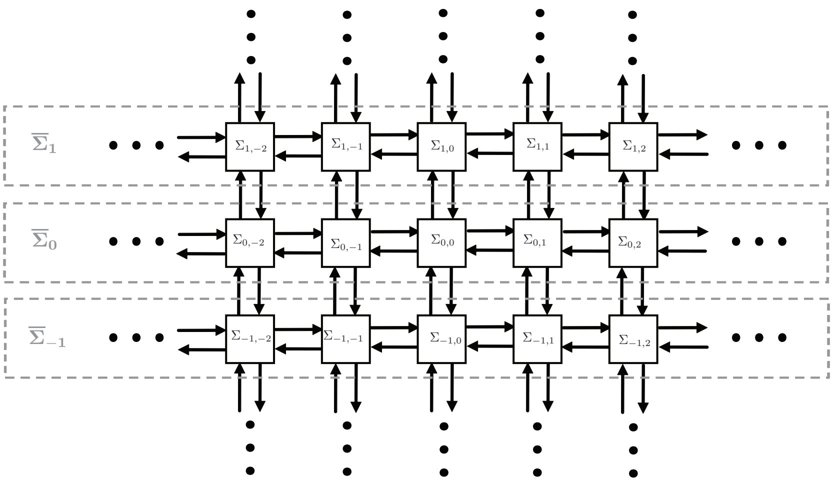


Figure 8.1: 2-D array ⇒ 1-D String of 1-D Strings

8.1 Multi-D structure

Consider the system distributed on a 2-D array in figure 8.1. If we resolve the rows into systems of L_r-operators, $\overline{\Sigma}_{-1}$, $\overline{\Sigma}_0$, $\overline{\Sigma}_1$ as in Chapter 5, we see that we have an infinite vertical string of infinite horizontal string systems, which we may again lift to get the final system $\overline{\overline{\Sigma}}$, which has a realization of 2-level Laurent operators with rational symbols (see Chapter 3, section 5), e.g. $\bar{\bar{F}}$ with symbol:

$$\mathcal{F}_z\bar{\bar{F}}\mathcal{F}_z^{-1} = \bar{F}(z) = \bar{B}(z\bar{I}-\bar{W})^{-1}\bar{C} + \bar{A} + \bar{E}(z^{-1}\bar{I}-\bar{R})^{-1}\bar{G}$$

bounded on $z \in \mathbb{T}$, which also have the L_r structure (where we have used $\bar{I}$ to indicate an infinite identity matrix of appropriate dimension), and which, when Fourier transformed again, in a different variable, have symbols that take the form of multivariable transfer functions, e.g.

$$\mathcal{F}_\zeta\mathcal{F}_z\bar{\bar{F}}\mathcal{F}_z^{-1}\mathcal{F}_\zeta^{-1} = F(z,\zeta) = B(\zeta)(zI-W(\zeta))^{-1}C(\zeta)+A(\zeta)+E(\zeta)(z^{-1}I-R(\zeta))^{-1}G(\zeta) \tag{8.1}$$

Where $B(\zeta), W(\zeta), C(\zeta), A(\zeta), E(\zeta), R(\zeta), G(\zeta) \in \mathcal{RL}_\infty(\mathbb{T})$ are the symbols of $\bar{B}$, $\bar{W}, \bar{C}, \bar{A}, \bar{E}, \bar{R}, \bar{G}$ respectively.

As with the 1-D case, it also holds that any system with a realization of multilevel L_r operators can be exactly written as an interconnected system distributed over multiple dimensions (see the appendix in section 8.4 for the 2-D case). Hence if we compute multilevel L_r structured controllers, they in turn have multidimensional distributed systems implementations, in the same interconnection topology as the original system. So we again have the challenge of finding structured solutions to our distributed control problems, and the motivation is still stronger; if sampling a transfer matrix over one variable on the unit circle and trying to interpolate to draw conclusions about stability, or design a controller, is difficult, then the equivalent calculation for transfer matrices in multiple variables is much worse. Fortunately, the induction process in section 3.5 allows us to form structure preserving arithmetics for such multilevel matrices, allowing the use of the iterative methods in Chapter 3 for efficient structured controller synthesis, just as in Chapters 4, 5, 6, and 7.

In the next section we will show a simple example of using the structure preserving arithmetic of multilevel L_r operators to solve an LQR problem for a heat conduction problem in two spatial dimensions.

8.2 Numerical Example: Heat Flow on a Flat Plate

Consider the system:

$$\dot{T}(m,n) = k_1 \frac{\partial^2 T(m,n)}{\partial^2} + k_2 \frac{\partial^2 T(m,n)}{\partial n^2} + bu(m,n), \qquad y(m,n) = cT(m,n)$$

of temperature T over continuous 2-dimensional spatial domain $-\infty < m < \infty$, $-\infty < n < \infty$ (an infinite flat plate) with controlled input $u(m,n)$ and measured output $y(m,n)$. k_1, k_2, b, and c are constants relating to the sensors, actuators, and the heat conduction. Note that it is probably not physical to have differing heat conduction coefficients $k_1 \neq k_2$ in different directions, but this could be due to e.g. different length scales. Discretizing the domain and approximating the derivative using finite difference, we get:

$$\begin{aligned} \dot{T}_{m,n} &= -2(k_1+k_2)T_{m,n} + k_1(T_{m-1,n} + T_{m+1,n}) + k_2(T_{m,n-1} + T_{m,n+1}) + bu_{m,n} \\ y_{m,n} &= cT_{m,n} \end{aligned}$$

over $m \in \mathbb{Z}$, $n \in \mathbb{Z}$, with zero temperature boundary conditions on top and bottom. If we then Fourier transform the system in the m and n directions, the LQR problem reduces to solving the parametric Riccati equation: $A(z,\zeta)^* X(z,\zeta) + X(z,\zeta)A(z,\zeta) - X(z,\zeta)BB^* X(z,\zeta) + C^*C = 0, \qquad z \in \mathbb{T}, \zeta \in \mathbb{T}$ where $B = bI$, $C = cI$, and $A(z,\zeta) = -2(k_1+k_2) + k_1(z^{-1}+z) + k_2(\zeta^{-1}+\zeta)$.

For values $k_1 = 1$, $k_2 = 5$, $c = 7$, $b = 7$, we used our 2-D Laurent operator arithmetic in the sign iteration methods of chapter 3 to solve this Riccati equation (approximately) for a 2-D Laurent operator $\tilde{\overline{\overline{X}}}$ with rational symbol $\tilde{X}(z,\zeta)$. The sign iteration converged after 4 steps, and an approximate solution $\tilde{X}(z,\zeta)$ with order 15 in z and 30 in ζ was found with $\max_{\zeta_0, z_0} \|\tilde{X}(z_0,\zeta_0) - X(z_0,\zeta_0)\| = 3.41 \times 10^{-7}$, which was calculated by discretizing over 100 points in $z \in \mathbb{T}$, and 100 points in $\zeta \in \mathbb{T}$, solving for the finite solution $X(z_0,\zeta_0)$, and comparing with $\tilde{X}(z_0,\zeta_0)$. Hence the solution was quite accurate, although much higher orders were required than for the 1-D example in Chapter 5.

In figure (8.2) we see the elementwise $\log_{10} |\cdot|$ truncation (to 11 points in m and n) of $\tilde{\overline{\overline{X}}}$, where the exponential spatial decay is clear. We also see the effect of the different heat conduction coefficients; the spatial decay is sharper in the vertical than horizontal direction.

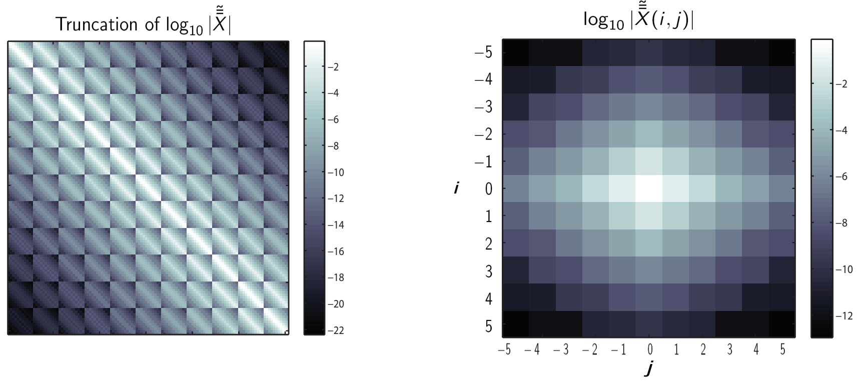


Figure 8.2: On the left, we have the truncation of the multilevel Laurent matrix $\log_{10}|\bar{\bar{\bar{X}}}|$, where the $\log_{10}|\cdot|$ is taken elementwise, and on the right, we show the magnitude of $\log_{10}|\bar{\bar{\bar{X}}}|$ in (m, n) space, where i is an indice for m and j is an indice for n.

8.3 Conclusion

In summary, we have seen that multidimensional infinite extent homogeneous interconnected systems on a cartesian grid can be represented by multilevel L_r operators. We can thus use the structured arithmetic of such operators built up in Chapters 2 and 3 in the structure preserving iterations of Chapter 3 to design multilevel L_r structured controllers which are arbitrarily close to optimal, which in turn have multidimensional distributed systems implementations, in the same interconnection topology as the original system. Due to limitations in our computational resources, we only demonstrated this result on a 2-dimensional homogeneous heat conduction problem, but as discussed in the introduction and Chapter 3, these results could be extended to 3 dimensional problems and beyond, and also to arbitrarily heterogeneous systems(like those in Chapter 4) or homogeneous systems with boundary conditions(like those in Chapter 6). The end result being $\mathcal{O}(NM)$ or $\mathcal{O}(1)$ computational complexity synthesis routines for an $N \times M$ 2-D grid of subsystems, compared to the conventional $\mathcal{O}(N^3M^3)$ discussed in the introduction of this chapter.

8.4 Appendix: 2-level L_r operator ⇒ 2-D interconnected system

Consider the system $\bar{\bar{\Sigma}}$: $\dot{\overline{\overline{x}}} = \overline{\overline{F}}\,\overline{\overline{x}}$ where: $\overline{\overline{F}} = \mathcal{S}(\overline{B}, \overline{W}, \overline{C}, \overline{A}, \overline{E}, \overline{R}, \overline{G})$ and

$$\begin{aligned}
\overline{B} &= \mathcal{S}(H^B, Z^B, J^B, D^B, K^B, Y^B, L^B), & \overline{W} &= \mathcal{S}(H^W, Z^W, J^W, D^W, K^W, Y^W, L^W)\\
\overline{C} &= \mathcal{S}(H^C, Z^C, J^C, D^C, K^C, Y^C, L^C), & \overline{A} &= \mathcal{S}(H^A, Z^A, J^A, D^A, K^A, Y^A, L^A)\\
\overline{E} &= \mathcal{S}(H^E, Z^E, J^E, D^E, K^E, Y^E, L^E), & \overline{R} &= \mathcal{S}(H^R, Z^R, J^R, D^R, K^R, Y^R, L^R)\\
\overline{G} &= \mathcal{S}(H^G, Z^G, J^G, D^G, K^G, Y^G, L^G)
\end{aligned}$$

It is time consuming, but not difficult to show that this 2-D rational Laurent operator system is equivalent to the following set of interconnected subsystems:

$$\left[\begin{array}{c}
\dot{x}_{s,r}\\ \hline
h^p_{s,r-1}\\ \hline
h^m_{s,r+1}\\ \hline
v^p_{s-1,r}\\ \hline
v^m_{s+1,r}
\end{array}\right]
=
\left[\begin{array}{c|ccccccc|ccccccc|c|c}
D^A & K^A & K^E & K^B & 0 & 0 & 0 & 0 & H^A & H^E & H^B & 0 & 0 & 0 & 0 & D^E & D^B\\ \hline
L^A & Y^A & & & & & & & & & & & & & & 0 & 0\\
0 & & Y^E & & & & & & & & & & & & & L^E & 0\\
0 & & & Y^B & & & & & & & & & & & & 0 & L^B\\
L^G & & & & Y^G & & & & & & & & & & & 0 & 0\\
0 & & & & & Y^R & & & & & & & & & & L^R & 0\\
L^C & & & & & & Y^C & & & & & & & & & 0 & 0\\
0 & & & & & & & Y^W & & & & & & & & 0 & L^W\\ \hline
J^A & & & & & & & & Z^A & & & & & & & 0 & 0\\
0 & & & & & & & & & Z^E & & & & & & J^E & 0\\
0 & & & & & & & & & & Z^B & & & & & 0 & J^B\\
J^G & & & & & & & & & & & Z^G & & & & 0 & 0\\
0 & & & & & & & & & & & & Z^R & & & J^R & 0\\
J^C & & & & & & & & & & & & & Z^C & & 0 & 0\\
0 & & & & & & & & & & & & & & Z^W & 0 & J^W\\ \hline
D^G & 0 & 0 & 0 & K^G & K^R & 0 & 0 & 0 & 0 & 0 & H^G & H^R & 0 & 0 & D^R & \\ \hline
D^C & 0 & 0 & 0 & 0 & 0 & K^C & K^W & 0 & 0 & 0 & 0 & 0 & H^C & H^W & & D^W
\end{array}\right]
\left[\begin{array}{c}
x_{s,r}\\ \hline
h^p_{s,r}\\ \hline
h^m_{s,r}\\ \hline
v^p_{s,r}\\ \hline
v^m_{s,r}
\end{array}\right]$$

over $s \in \mathbb{Z}$, $r \in \mathbb{Z}$.

9

Repetitive and Iterative Learning Control

In the previous few chapters we've used the structure preserving iterations of Chapter 3 on the structured matrix and operator arithmetics of Chapter 2 for applications to the analysis and control of distributed systems of various kinds. In this chapter, we'll do something completely different; we use these techniques on Repetitive/Iterative Learning Control, where the matrices are lifted in time instead of space.

9.1 Background/Survey

Often in industrial processes, a certain task, such as the movement of a wafer stage or the cantilever of an atomic force microscope, is executed repetitively, where the final conditions of one trial can be considered the initial conditions of the next. The tracking performance of such a system can be improved from trial to trial by updating a feed-forward control signal based on previous trial knowledge. This is called Repetitive Control(RC), or Iterative Learning Control(ILC), if the initial conditions are reset after each trial. ILC/RC became a popular research topic beginning with [160], and since then has proven to be a powerful method for high performance reference tracking. A recent overview of ILC with an extensive bibliography of applications is available in [161].

The lifted approach ([162]) has been shown to be especially useful for analysis of the stability of ILC/RC controllers and the difficulties for non-minimum phase systems([163],[164]), which are quite common for sampled-continuous time systems ([165]). In [166] it is shown that many of the traditionally non-lifted ILC/RC methods can be written as lifted controllers of a certain general type, yet only recently ([167]) have the tools of optimal control theory been applied to designing lifted type ILC/RC laws. The hesitation to apply linear optimal and robust control synthesis is due to the potentially very large size of the matrices involved in such optimization problems, where the size of the matrices is often proportional to the trial length.

In the literature, for short trial lengths this issue of computational complexity has often been ignored, and it is common to compute the singular value decompositions(SVD) of large lifted matrices for controller design([167],[168]) or use the eigenvalues of a large lifted matrix as a stability criteria([166],[162]). In the BLQG(Batch-LQG) method of [169], and [170] the lifted approach is extended to state space representations, and the ILC/RC problem is formulated as a large LQG problem, requiring solutions of Riccati equations in the lifted variables(see also the unconstrained case in [171]).

However, with modern data acquisition systems, sampling frequencies may be in the kHz or MHz, with trial lengths of $N = 1,000 - 80,000$ or greater([172], [173], [174], [175]), rendering many lifted analysis and design techniques computationally infeasible, since the computation of SVD's, eigenvalue decompositions, and the solutions of Riccati equations is generally of at least $\mathcal{O}(N^3)$ computational complexity, and even the implementation of such a controller is $\mathcal{O}(N^2)$ due to the cost of matrix-vector multiplications. Such computations and resulting controllers may need to be approximated, possibly leading to a loss of performance or even stability.

Fortunately, if derived in a certain way, the relevant matrices can be shown to have Sequentially Semi-Separable structure, and hence the $\mathcal{O}(N)$ analysis, synthesis, and implementation routines built in Chapters 2 and 3 can be used, as we will next show. We note that the link between repetitive/iterative learning control and distributed control has previously been investigated and exploited for analysis/control within the multi-dimensional systems community, see e.g. recent papers [176], [177], [178], however, our techniques and computational methods will be quite different in form.

9.2 Formulation of ILC/RC into an Output Feedback Problem

We consider the LTV system:

$$\Sigma : \begin{cases} x_{t+1} = A_t x_t + B_t u_t + F_t d_t \\ y_t = C_t x_t + D_t u_t + G_t n_t \end{cases} \tag{9.1}$$

where $x \in \mathbb{R}^{\mathrm{n}_x}$ is the state, $y \in \mathbb{R}^{n_y}$ is the output, $u \in \mathbb{R}^{\mathrm{n}_u}$ is the input, $d \in \mathbb{R}^{\mathrm{n}_d}$ is the process disturbance, and $n \in \mathbb{R}^{\mathrm{n}_n}$ is the measurement noise. All matrices are appropriately dimensioned, and the dimensions may change in time ([47])(but for simplicitywe write all matrix dimensions here as if constant). This class of systems is quite broad and includes LTI systems, systems that actually vary in time, LPV systems with known scheduling variables, and nonlinear systems linearized along some trajectory ([179]). In this paper we will not consider the range of validity for such modeling approaches; we will simply assume that Σ is accurate for all x, u, d, n. LTV system models may also, of course, be obtained through dedicated system identification experiments, such as in [180],[181].

For a trial of $N + 1$ time steps: $t \in \{0, 1, 2, ..., N\}$, we can 'lift' Σ to show the full relation between the input and state:

$$\begin{aligned}\underbrace{x_{N+1}}_{x_0^{k+1}} = \underbrace{\Phi_{(0,N)}}_{\hat{A}}\underbrace{x_0}_{x_0^k} \quad &+ \quad \underbrace{\begin{bmatrix} \Phi_{(1,N)}B_0 & \dots & A_N B_{N-1} & B_N \end{bmatrix}}_{\hat{B}} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{bmatrix}}_{\bar{u}^k} \\ &+ \quad \underbrace{\begin{bmatrix} \Phi_{(1,N)}F_0 & \dots & A_N F_{N-1} & F_N \end{bmatrix}}_{\hat{F}} \underbrace{\begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_N \end{bmatrix}}_{\bar{d}^k}\end{aligned} \tag{9.2}$$

and for the output:

$$\begin{aligned}\underbrace{\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_N \end{bmatrix}}_{\bar{y}^k} = \underbrace{\begin{bmatrix} C_0 \\ C_1 A_0 \\ \vdots \\ C_N \Phi_{(0,N-1)} \end{bmatrix}}_{\hat{C}} \underbrace{x_0}_{x_0^k} &+ \underbrace{\begin{bmatrix} G_0 & 0 & \dots & 0 \\ 0 & G_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & G_N \end{bmatrix}}_{\hat{G}} \underbrace{\begin{bmatrix} n_0 \\ n_1 \\ \vdots \\ n_N \end{bmatrix}}_{\bar{n}^k} \\ &+ \underbrace{\begin{bmatrix} D_0 & 0 & \dots & 0 \\ C_1 B_0 & D_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_N \Phi_{(1,N-1)} B_0 & \dots & C_N B_{N-1} & D_N \end{bmatrix}}_{\hat{D}} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{bmatrix}}_{\bar{u}^k} \\ &+ \underbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \\ C_1 F_0 & 0 & \dots & 0 \\ \vdots & C_2 F_1 & \ddots & \vdots \\ C_N \Phi_{(1,N-1)} F_0 & \dots & C_N F_{N-1} & 0 \end{bmatrix}}_{\hat{H}} \underbrace{\begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_N \end{bmatrix}}_{\bar{d}^k}\end{aligned} \tag{9.3}$$

where $\Phi_{(i,j)} = A_j A_{j-1} ... A_i$, $\hat{\ }$ indicates a 'lifted' system matrix, $\bar{\ }$ a lifted vector, k indicates the trial index, $\hat{B} \in \mathbb{R}^{\mathrm{n}_x \times (N+1)\mathrm{n}_u}$ and $\hat{C} \in \mathbb{R}^{(N+1)\mathrm{n}_y \times \mathrm{n}_x}$ are called the extended controllability and extended observability matrices, respectively, and $\hat{D} \in \mathbb{R}^{(N+1)\mathrm{n}_y \times (N+1)\mathrm{n}_u}$ is the extended impulse response matrix. We are also assuming here that the initial state of trial $k+1$, (x_0^{k+1}), is the final state of trial k. This is actually RC, but the problem could also be formulated with 0 or random initial conditions as an ILC problem.

The measurement noise, $\overline{n}^k$, is assumed to be some random trial-uncorrelated bounded signal, but the process disturbance may consist of a trial-uncorrelated component and an integrated 'random walk' component ([182],[169]):

$$\overline{d}^k = \overline{v}^k + \overline{w}^k, \qquad \text{where} \qquad \overline{v}^k = \overline{v}^{k-1} + \overline{\zeta}^k \tag{9.4}$$

with $\overline{w}^k$ and $\overline{\zeta}^k$ also random trial-uncorrelated bounded. This is a quite versatile noise formulation; the process noise has a stochastic component, but also a random walk nature, so the 'repetitive disturbance' can vary between trials. In addition, the noise and disturbances can be correlated in time.

From (9.3) we define a 'non-stochastic' output:

$$\overline{y}^k_{RC} \quad = \quad \overline{y}^k - \hat{H}\overline{w}^k - \hat{G}\overline{n}^k = \hat{C}x_0^k + \hat{D}\overline{u}^k + \hat{H}\overline{v}^k \tag{9.5}$$

and if we also define the lifted tracking error: $\overline{e}^k = \overline{y}^k - \overline{r}_{ef}$ for some lifted reference signal $\overline{r}_{ef}$, then we have a 'non-stochastic' error: $\overline{e}^k_{RC} = \overline{y}^k_{RC} - \overline{r}_{ef}$, which is independent of the random disturbances from trial k: $\overline{w}^k$, $\overline{n}^k$.

Defining $\Delta z^k = z^k - z^{k-1}$, or, the change between two trials, we find that:

$$\overline{e}^k_{RC} \quad = \quad \overline{e}^{k-1}_{RC} + \hat{C}\Delta x_0^k + \hat{D}\Delta\overline{u}^k + \hat{H}\overline{\zeta}^k \tag{9.6}$$

where the lifted system matrices, $(\hat{A}, \hat{B}, \hat{C}, \hat{D}, \hat{H}, \hat{G})$ and the desired reference signal $\overline{r}_{ef}$ are considered trial invariant, but the results in this paper could easily be extended to a trial-varying system formulation.

Again using the Δ operator, and from (9.2), (9.6), and (9.4), we can derive an (extremely large dimensional) stochastic linear system, which we call Σ_{RC}:

$$\left[\begin{array}{c}\overline{e}^k_{RC}\\ \overline{w}^k\\ \hline \Delta x_0^{k+1}\end{array}\right] = \underbrace{\left[\begin{array}{cc|c} I & 0 & \hat{C}\\ 0 & 0 & 0\\ \hline 0 & -\hat{F} & \hat{A}\end{array}\right]}_{\mathcal{A}} \left[\begin{array}{c}\overline{e}^{k-1}_{RC}\\ \overline{w}^{k-1}\\ \hline \Delta x_0^k\end{array}\right] + \underbrace{\left[\begin{array}{cc}\hat{H} & 0\\ 0 & I\\ \hline \hat{F} & \hat{F}\end{array}\right]}_{\mathcal{B}_w}\left[\begin{array}{c}\overline{\zeta}^k\\ \overline{w}^k\end{array}\right] + \underbrace{\left[\begin{array}{c}\hat{D}\\ 0\\ \hline \hat{B}\end{array}\right]}_{\mathcal{B}}\Delta\overline{u}^k (9.7)$$

$$\overline{e}^k = \underbrace{\left[\begin{array}{cc|c} I & 0 & \hat{C}\end{array}\right]}_{\mathcal{C}} \left[\begin{array}{c}\overline{e}^{k-1}_{RC}\\ \overline{w}^{k-1}\\ \hline \Delta x_0^k\end{array}\right] + \underbrace{\hat{D}}_{\mathcal{D}}\Delta\overline{u}^k + \underbrace{\left[\begin{array}{ccc}\hat{H} & \hat{H} & \hat{G}\end{array}\right]}_{\mathcal{D}_w}\left[\begin{array}{c}\overline{\zeta}^k\\ \bar{w}^k\\ \overline{n}^k\end{array}\right] \tag{9.8}$$

which we write simply as:

$$\Sigma_{RC} : \left\{ \begin{array}{c} \mathcal{X}^{k+1} = \mathcal{A}\mathcal{X}^k + \mathcal{B}\Delta\overline{u}^k + \nu^k \\ \overline{e}^k = \mathcal{C}\mathcal{X}^k + \mathcal{D}\Delta\overline{u}^k + \mu^k \end{array} \right. \tag{9.9}$$

where the output is the reference error at trial k, and the input is the change in trial input. In the following, we will assume that the measurement noise and process disturbance signals from (9.7) and (9.8) can be modeled as $\overline{n}^k \sim \mathcal{N}(0, R_n)$, $\overline{w}^k \sim \mathcal{N}(0, R_w)$, and $\overline{\zeta}^k \sim \mathcal{N}(0, R_\zeta)$; that is, they are zero-mean stationary Gaussian, and we will assume that they are uncorrelated with each other. Thus the disturbance joint covariance matrix of μ^k and ν^k from (9.9) is calculated as (using the stochastic expectation($\mathbf{E}[\cdot]$) [44]):

$$\begin{aligned}
\mathbf{E}\left[\begin{bmatrix}\mu^k \\ \nu^k\end{bmatrix}\begin{bmatrix}\mu^{jT} & \nu^{jT}\end{bmatrix}\right] &= \begin{bmatrix}\mathcal{R}_w & \mathcal{S}_w^T \\ \mathcal{S}_w & \mathcal{Q}_w\end{bmatrix}\delta(k-j) \qquad (9.10)\\
&= \begin{bmatrix}\mathcal{D}_w\begin{bmatrix}R_\zeta & 0 & 0\\ 0 & R_w & 0\\ 0 & 0 & R_n\end{bmatrix}\mathcal{D}_w^T & * \\ \mathcal{B}_w\begin{bmatrix}R_\zeta & 0 & 0\\ 0 & R_w & 0\end{bmatrix}\mathcal{D}_w^T & \mathcal{B}_w\begin{bmatrix}R_\zeta & 0\\ 0 & R_w\end{bmatrix}\mathcal{B}_w^T\end{bmatrix}\delta(k-j)
\end{aligned}$$

where δ is the Kronecker delta function.

Having now clearly defined the system we are working on, define the RC output feedback problem:

RC Problem: Given Σ_{RC} in (9.7),(9.8), design a dynamic feedback controller: $\Upsilon_{RC} : \left[\begin{array}{c|c}\mathcal{A}_c & \mathcal{B}_c \\ \hline \mathcal{C}_c & 0\end{array}\right]$ such that Υ_{RC} stabilizes Σ_{RC}.

The controller is assumed strictly proper; $\mathcal{D}_c$ is assumed 0, because current trial measurements are assumed unavailable for feedback(However, a real-time feedback controller may be incorporated into this structure, the formulas for which we exclude for brevity. See section 9.3 for a brief discussion).

This formulation is advantageous in the sense that resulting control laws may be 'mixed causal', using all of the available information from the previous trial to compute the input of trial k, u_t^k, at each time step t. By requiring that Υ_{RC} stabilizes Σ_{RC}, we guarantee exponential convergence of the RC algorithm. While this does not guarantee the monotonic convergence recommended in ([183]), it should avoid the terrible learning transients experienced with some 'universal' laws, as described in the paper. Also, since the proposed controller is dynamic, the RC input updating laws will be generally higher order(see [184] for a discussion of the advantages of higher order ILC/RC laws).

However, calculating a stabilizing repetitive controller using optimal control techniques, or given a prospective controller, even verifying closed loop stability, will generally be $\mathcal{O}(N^3)$ computational complexity, and thus infeasible for long trial lengths(N). Hence in the following, we will show how to rewrite Σ_{RC} in such a way as to make possible fast, numerically robust, structure preserving calculation techniques for optimal control synthesis and analysis.

9.2.1 SSS structure of RC problem

It is now possible to show how our system matrices $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{R}_w, \mathcal{Q}_w, \mathcal{S}_w$ in (9.7), (9.8), and (9.10) can be permuted to fit well into the SSS structure. For convenience, we rewrite the system (9.7) and (9.8) as:

$$\begin{aligned}\left[\begin{array}{c}\overline{e}^k_{RC}\\ \overline{w}^k\\ \hline \Delta x_0^{k+1}\end{array}\right] &= \underbrace{\left[\begin{array}{c|c}\mathcal{A}_{11} & \mathcal{A}_{12}\\ \hline \mathcal{A}_{21} & \hat{A}\end{array}\right]}_{\mathcal{A}}\left[\begin{array}{c}\overline{e}^{k-1}_{RC}\\ \overline{w}^{k-1}\\ \hline \Delta x_0^{k}\end{array}\right] + \underbrace{\left[\begin{array}{c}\mathcal{B}_1^w\\ \hline \mathcal{B}_2^w\end{array}\right]}_{\mathcal{B}_w}\left[\begin{array}{c}\overline{\zeta}^k\\ \overline{w}^k\end{array}\right] + \underbrace{\left[\begin{array}{c|c}\mathcal{B}_1^T & \hat{B}^T\end{array}\right]^T}_{\mathcal{B}}\Delta\overline{u}^k\\ \overline{e}_k &= \underbrace{\left[\begin{array}{c|c}\mathcal{C}_1 & \hat{C}\end{array}\right]}_{\mathcal{C}}\left[\begin{array}{c}\overline{e}^{k-1}_{RC}\\ \overline{w}^{k-1}\\ \hline \Delta x_0^{k}\end{array}\right] + \mathcal{D}\Delta\overline{u}^k + \mathcal{D}_w\left[\begin{array}{c}\overline{\zeta}^k\\ \overline{w}^k\\ \overline{n}^k\end{array}\right] \qquad (9.11)\end{aligned}$$

where we have just partioned the matrices in a different way and renamed them for the following discussion. Unfortunately, to make this clear we will need a little bit of additional notation (which doesn't appear in any other chapters).

From Chapter 2 the reader should be familiar with the generator notation of SSS matrices, and of their LTV input-output map interpretation. Hence we will next speak about the 'order' of an SSS matrix, which would be the 'order' of its corresponding LTV system, defined precisely in the following:

Definition 9.1 *The maximum lower and upper order of an SSS matrix is the largest size of its lower and upper multiplier terms(R_s and W_s in $\overline{X} = \mathcal{SSS}(P, R, Q, D, U, W, V)$), respectively. The class of SSS matrices of maximum lower and upper orders w_l and w_u with N diagonal terms is denoted as $\mathcal{SSS}^{w_l,w_u,N}$.*

Hence the growing order of SSS matrices under arithmetic can be related as:

Lemma 9.1 *For conformably partitioned matrices $\overline{A} \in \mathcal{SSS}^{a_l,a_u,N}$ and $\overline{B} \in \mathcal{SSS}^{b_l,b_u,N}$, then $\overline{A} + \overline{B} = \overline{C} \in \mathcal{SSS}^{c_l,c_u,N}$ where $c_l \leq a_l + b_l$, $c_u \leq a_u + b_u$, and $\overline{AB} = \overline{D} \in \mathcal{SSS}^{d_l,d_u,N}$ where $d_l \leq a_l + b_l$, $d_u \leq a_u + b_u$.* ***Proof:*** *This can be seen from the addition and multiplication algorithms (Chapter 2, section 3) for SSS matrices, and makes sense under the LTV interpretation of SSS addition as LTV systems in parallel, and SSS multiplication as LTV systems in series.* □

We will now use this new notation to show how all of the matrices in (9.7) and (9.8) can be permuted to fit well into the SSS structure.

From (9.3), we can clearly see that $\mathcal{D} \in \mathcal{SSS}^{n_A,0,N+1}$ but it is not obvious how $\mathcal{A}$, for example, can be put into the SSS structure. However, it is clear that $\mathcal{A}_{11}$ consists of four matrices each having SSS structure($I \in \mathcal{SSS}^{0,0,N+1}$, $0 \in \mathcal{SSS}^{0,0,N+1}$). If we perform a 'shuffle' permutation on the top part of the state vector in (9.11): $\begin{bmatrix}\overline{e}^{k-1}_{RC}\\ \overline{w}^{k-1}\end{bmatrix} \rightarrow \overline{\begin{bmatrix}e^{k-1}_{RC}\\ w^{k-1}\end{bmatrix}}$, in which the elements of the lifted vectors $\overline{e}^{k-1}_{RC}$ and $\overline{w}^{k-1}$ are interleaved together to form the lifted vector $\overline{\begin{bmatrix}e^{k-1}_{RC}\\ w^{k-1}\end{bmatrix}}$, then by the SSS version of Lemma 2.6, $\mathcal{A}_{11}$ will be permuted into $\mathcal{A}^p_{11} \in \mathcal{SSS}^{0,0,N+1}$ and $\mathcal{B}_1$ and $\mathcal{C}_1$ will be permuted into $\mathcal{B}_1 \rightarrow \mathcal{B}_1^p \in \mathcal{SSS}^{n_A,0,N+1}$, $\mathcal{C}_1 \rightarrow \mathcal{C}_1^p \in \mathcal{SSS}^{0,n_A,N+1}$(where we use the superscript ^{p} to indicate the matrices after permutations). If we additionally permute $\begin{bmatrix} \overline{\zeta}^k \\ \bar{w}^k \\ \overline{n}^k \end{bmatrix} \rightarrow \overline{\begin{bmatrix} \zeta^k \\ w^k \\ n^k \end{bmatrix}}$ and $\begin{bmatrix} \overline{\zeta}^k \\ \bar{w}^k \end{bmatrix} \rightarrow \overline{\begin{bmatrix} \zeta^k \\ w^k \end{bmatrix}}$ then $\mathcal{D}_w$ and $\mathcal{B}_1^w$ will also be permuted into low order SSS matrices.

In addition, it can be shown that $\left[\begin{array}{c|c} 0 & \mathcal{A}_{12}^p \\ \hline \mathcal{A}_{21}^p & \hat{A} \end{array}\right] \in \mathcal{SSS}^{n_a,n_a,N+2}$, and since $\mathcal{A}^p = \left[\begin{array}{c|c} 0 & \mathcal{A}_{12}^p \\ \hline \mathcal{A}_{21}^p & \hat{A} \end{array}\right] + \left[\begin{array}{c|c} \mathcal{A}_{11}^p & 0 \\ \hline 0 & 0 \end{array}\right]$ where $\left[\begin{array}{c|c} \mathcal{A}_{11}^p & 0 \\ \hline 0 & 0 \end{array}\right] \in \mathcal{SSS}^{0,0,N+2}$, then by Lemma 9.1, $\mathcal{A}^p \in \mathcal{SSS}^{n_a,n_a,N+2}$. The same argument can be used to show that $\mathcal{B}^p, \mathcal{B}_w^p \in \mathcal{SSS}^{2n_A,0,N+2}$ and $\mathcal{C}^p \in \mathcal{SSS}^{0,2n_A,N+2}$.

Finally, if the noise covariances R_ζ, R_n, R_w also have SSS structure (for example, if ζ_t, n_t, w_t are the outputs of linear systems, or if they are white noise sequences, corresponding to diagonal R_ζ, R_n, R_w) then, after they are permuted to match the noise signals, $\mathcal{R}_w^p$, $\mathcal{S}_w^p$, $\mathcal{Q}_w^p$ will all have SSS structure due to the closedness property of the structure under multiplication.

So, permutations of the state and disturbance vectors have allowed us to write a linear system equivalent to Σ_{RC} with a realization entirely of matrices with SSS structure. We note that no permutations were performed on the error $\overline{e}^k$ or the input $\Delta\overline{u}^k$. Now that this is estabilished, we can use the SSS arithmetic of Chapter 2 in the structure presering iterations of Chapter 3 for $\mathcal{O}(N)$ control design, as we will show in the next section in an example.

9.3 Illustrative Example

The infinite horizon discrete time LQG solution is well known ([185]). The resulting controller stabilizes Σ_{RC} and the weights can be chosen so as to minimize the expectation of the error over all time:

$$J = \mathbf{E}\left(\lim_{T\to\infty} \frac{1}{T} \sum_{k=0}^{T} \|\overline{e}^k\|_2^2\right) \tag{9.12}$$

Such a minimization is useful for industrial applications, in which 100's or 1,000's of trials may be run. In this paper we will focus on keeping the computational cost of implementation low, so we will use the simplified stationary(in the trial domain) version of the optimal LQG controller. Some of the advantages of framing repetitive control problems in the lifted domain as LQR/LQG problems have already been demonstrated in the literature(e.g. [167], [169], [170]), but without feasible computational methods for long trial lengths.

In the following, we will show the usefulness of optimal repetitive and non-repetitive disturbance rejection in the LQG format, on a test-case for which very long trial lengths are an important consideration. We use a simple but realistic 2nd order SISO example similar to the Digital Light Processing(DLP) projection system considered in [186]. We stress that the chosen example is LTI, but treated as an LTV system using the computational methods outlined in Chapter 2, to

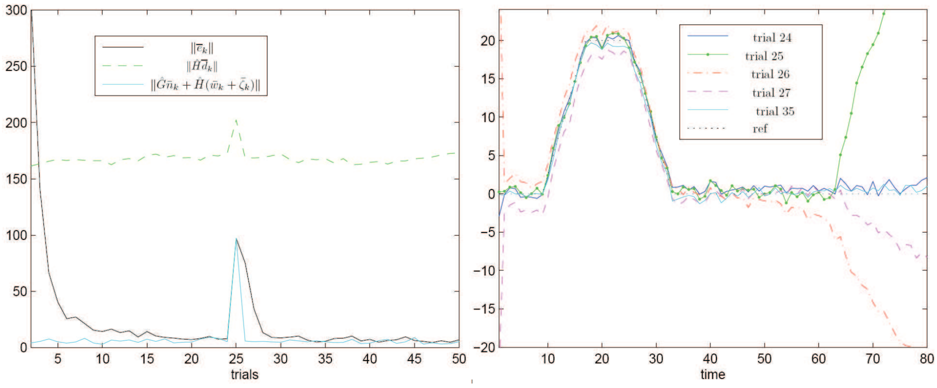


Figure 9.1: These plots are for the temporary disturbance. On the left we see error norms compared to noise terms in the trial domain, while on the right we see the output compared to reference over a number of trials.

simplify the demonstration only. Our techniques work the same, and have the same $\mathcal{O}(N)$ computational complexity, irrespective of the LTI or LTV nature of the problem. We also note that while our example is minimum-phase SISO, the authors have observed these techniques to work well on a very diverse variety of systems, including intermittently unstable, LTV, MIMO non-minimum phase systems.

DLP systems work at a very high sampling frequency($\geq 3 \times 10^4$Hz), potentially leading to very long trial lengths(N) and making computation and implementation of lifted ILC laws difficult. The input-output behavior is described by the continuous time transfer function:

$$y(s) = \frac{\frac{1}{c}s + \frac{\tau}{\omega^2}}{s^2 + 2\lambda\omega s + \omega^2}(u(s) + d(s)) + n(s) \tag{9.13}$$

With $c = 10^{-4}$, $\tau = 1$, $\lambda = 0.15$, $\omega = 2\pi(7500)$ chosen to closely match frequency responses given in the reference. We then found a discrete time state space representation of this transfer function for the sampling time: $T_s = 1 \times 10^{-6}$ seconds. We assume disturbance and noise covariances of $R_w = R_\zeta = 5I$ and $R_n = \frac{1}{10}I$ (not included in the original reference).

9.3.1 Closed Loop Simulations

For an example trial length of $N = 80$, the RC controllers(as calculated using the methods in Chapter 3) were implemented on the discretized DLP model and simulated. To demonstrate the advantage of formulating the various noise and disturbances (see (9.4) and surrounding discussion) as filtered random noise components in an LQG setting, we consider the closed loop response to both a 'non-repetitive' and 'repetitive' disturbance(ILC/RC laws often have trouble dealing with both si-

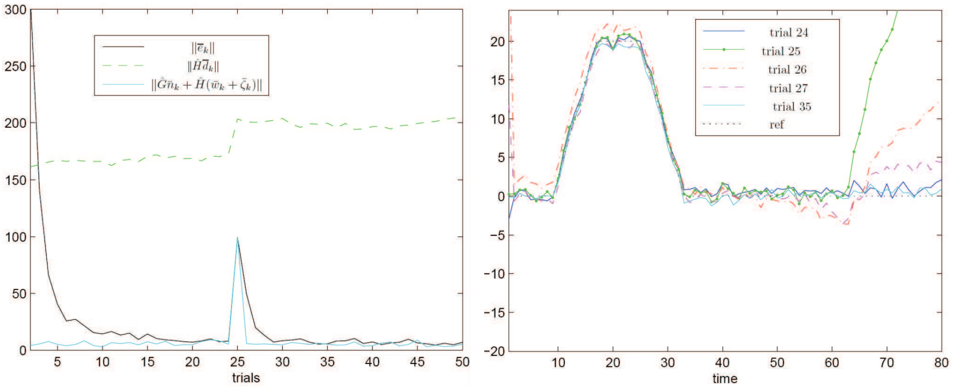


Figure 9.2: These plots are for the permanent disturbance. On the left we see error norms compared to noise terms in the trial domain, while on the right we see the output compared to reference over a number of trials.

multaneously). The same white noise realizations of $\overline{w}_k$, $\overline{n}_k$, $\overline{\zeta}_k$ are used in each simulation, with the exception of a large magnitude disturbance at trial $k = 25$, time $t = \{62, 63, 64, 65, 66\}$ with magnitude 200. For both simulations, the RC controller was initialized at trial $k = 0$ with a 0-vector input $\overline{u}^0 = 0$, and initial estimates of $\overline{e}_{RC}^{-1}$ and $\overline{w}^0$,$\overline{w}^{-1}$ being equal to $\overline{r}_{ef}$ and 0-vectors, respectively. The fast initial convergence to the reference trajectory can be seen(identically) in Figures (9.1), (9.2).

The 'non-repetitive' disturbance is input to the system through the $\overline{w}_k$ term(and thus only lasts for a single trial, as we see in the $\|\hat{H}\overline{d}_k\|$ plot in Figure 9.1) whereas the 'repetitive disturbance' is input to the system through the $\overline{\zeta}_k$ term, and thus continues to have an effect in $\overline{v}$ for all trials thereafter(as we see in the $\|\hat{H}\overline{d}_k\|$ plot in Figure 9.2). However, in both non-repetitive and repetitive cases, we see that after the large disturbances are introduced at $k = 25$, the trial errors $\|\overline{e}_k\|$, quickly converge back to almost $\|\hat{G}\overline{n}_k + \hat{H}(\overline{w}_k + \bar{\zeta}_k)\|$, the completely random component of the error in trial k, which we cannot hope to control using RC. We also see this in the outputs in Figures 9.1 and 9.2 for the non-repetitive and repetitive disturbances, respectively. At trial $k = 25$, time $t = 62$, the large disturbances completely derail the output y_t from the reference, and likewise, trials $k = 26$ and $k = 27$ have fairly bad tracking, as the RC controller tries to compensate, but after 10 trials the output of both simulations has returned to almost perfect tracking(considering the noise and other random disturbances present).

We note that the errors $\|\overline{e}_k\|$ of the trial in which the large disturbance is introduced and those directly after(e.g. $k = 25, 26, 27$), may be much reduced using a real time feedback controller, but a good RC controller would still be necessary to re-achieve perfect tracking after the introduction of a new repetitive disturbance. Within the framework of ([182]), such feedback controllers can be well-incorporated into the lifted system for RC, resulting in a lifted system similar

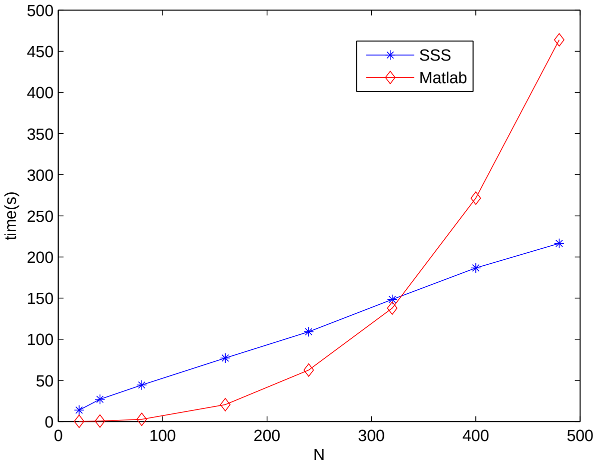


Figure 9.3: A comparison of MATLAB's QZ based Riccati solver ($\mathcal{O}(N^3)$ complexity) and the SSS Riccati solver ($\mathcal{O}(N)$ complexity).

to (9.7) and (9.8), with an additional n_x states, and thus will similarly fit into the SSS structure for RC design.

9.3.2 Computational Complexity and Accuracy of the SSS Structure Preserving Techniques

Given our sampling frequency, reference trajectories of length $2\times10^{-5}, 4\times10^{-5}, 8\times10^{-5}, 1.6\times10^{-4}, 2.4\times10^{-4}, 3.2\times10^{-4}, 4\times10^{-4}$, and 4.8×10^{-4} seconds produce trial lengths of $N = \{20, 40, 80, 160, 240, 320, 400, 480\}$, respectively. We used the algorithms described in Chapters 2 and 3 to compute the LQG controller matrices for each of these trial lengths, and compared these in computational time and closed loop H_2 performance to the results of MATLAB's QZ based DARE solver. The SSS controllers(all of which were closed loop stable) were found to have closed loop performances, γ_{SSS}, of only slightly higher than those produced by MATLAB; for all values of N tested, the normalized difference was $\frac{\gamma_{SSS}-\gamma_M}{\gamma_M} < 2\times10^{-5}$, and in Fig. 9.3, we see the comparison of the MATLAB function `DARE` with the SSS Riccati solver in computation times. The linear complexity of the SSS solver becomes an advantage after about $N = 325$, and following the linear and cubic trends in figure 9.3, for a trial of $N = 3000$(as we would have for the period of 3×10^{-3}s given in [186]), the SSS solver would take approximately 25 minutes, while the MATLAB solver would require about 1.3 days (assuming the computer didn't run out of memory). Hence exploiting the SSS structure for RC optimal controller synthesis, as developed in this paper, is absolutely essential for such large problems. Note that the SSS structured RC controller will also be advantageous for implementation, since SSS structured matrix-vector computations are only $\mathcal{O}(N)$

complexity, as opposed to $\mathcal{O}(N^2)$ complexity for general dense matrices.

For reference to the algorithmic discussions, the SSS orders of the controllers and Riccati solutions were constrained to be $w_u = w_l = 20$ for all computations, and the doubling iterations for solving the LQR and Kalman filter Riccati equations took about 4 and 6 iterations to converge, respectively.

9.4 Conclusions

In this chapter we have formulated the RC problem for stochastic LTV systems as a large linear output feedback problem, including repetitive and non-repetitive noise and disturbances. The special SSS matrix structure of the RC system has been exploited to find LQG solutions with guaranteed exponential convergence in only linear computational complexity ($\mathcal{O}(N)$), allowing this method to be used on trials with a large number of samples.

It should not be difficult to generalize the system (9.1) to include a separate performance output z_t and thus consider the problem within the generalized plant H_2 and H_∞ framework, and it is likely that such problems also have exploitable SSS structure. Also, as mentioned in section 9.2, a trial varying reference, $\overline{r}^k_{ef}$, and system matrices, $\hat{A}^k$ etc. could also be treated in this framework, and the non-stationary Kalman filter could be implemented for the LQG solution with a slight increase in online computation.

We would also like to make clear that while we have considered the particular problem of synthesis and closed loop analysis of a lifted LQG controller, the techniques and algorithms presented herein should be of use in many other areas of lifted ILC/RC research. For example, eigenvalue stability criteria for lifted ILC laws, such as those found in [166], may be computed in $\mathcal{O}(N)$ using the sign iteration stability check in Lemma 3.3, assuming that the ILC matrices themselves fit into the SSS structure(e.g. diagonal, banded, or impulse response matrices), as is often the case.

Within the larger context of this thesis, we should also point out that while this chapter has only used SSS matrices to represent LTV systems lifted to form RC problems, one can also use ATSSS matrices to represent LTI systems lifted to form RC problems, and thus decrease the computational complexity further even from $\mathcal{O}(N)$ to $\mathcal{O}(1)$ in such cases. This should be the subject of future investigation.

10 LPV Analysis and Synthesis

In the previous few chapters we've used structure preserving iterations on structured matrix and operator arithmetics for applications in the analysis and control of distributed systems of various kinds. In this chapter, we'll use these techniques to develop a novel approach to LPV systems.

10.1 Background/Survey

In recent years, system models in many important applications have been shown to have a Linear Parameter Varying(LPV) structure, e.g. wind turbines [187], automated lane keeping systems [188], steam generators for nuclear power plants [189], biomedical applications [190], web servers [191], and bicycles [192]. Correspondingly, there has been a burgeoning field of LPV analysis and synthesis techniques, see e.g. [193][194][195][196][197][198] and the references therein.

An important problem in LPV analysis and control is to find parameter dependent solutions to parameter dependent Lyapunov and Riccati equations and inequalities of both the algebraic and differential type, valid over some set of admissible parameters and parameter rates. Such solutions can be used to guarantee exponential LPV stability or performance [199] or to construct parameter dependent controllers with guaranteed stability and performance(see e.g. [200] and references therein). Unfortunately, finding such parameter dependent solutions (or even verifying that a given solution satisfies a parameter dependent equation or inequality over the entire parameter set) turns out to be quite a difficult problem; even though the inequalities are linear in the unknowns, we must search or optimize over the (often infinite) space of admissible parameters. Such problems can often be cast as verifying or solving a parametric matrix equation (Lyapunov, Riccati) or Linear Matrix Inequality(LMI) for which various techniques have been developed: e.g. [201][144](parametric equations) [202][203] [204][205][206][207][208][209](parametric LMI's). However, these techniques either require some restrictive *a priori* assumption on the form of the solution's parameter dependence (e.g affine, polynomial or rational of order n, etc), the gridding of the parameter space, conservative relaxations, or relaxations which are asymptotically nonconservative but lead to very large (and thus computationally expensive) LMI's.

However, under certain system assumptions, we can use computational techniques based on transfer function arithmetic and the matrix sign function which avoids these problems, and allows us to solve certain parameter dependent algebraic Lyapunov and Riccati equations arbitrarily accurately, thus providing a different approach to such LPV problems. Our idea is to construct a sequence of rationally parametric matrices which approach the exact(and perhaps irrational) solution of the Lyapunov or Riccati equation quadratically fast, using the matrix sign function. Instead of using the computationally expensive operations on transfer matrices, we work only in $\mathcal{S}_c$ realizations.

In section 10.2 we overview the type of LPV systems considered, and show how to convert them into realizations of $\mathcal{S}_c$ operators. In section 10.3 we then show how the tools developed in Chapter 3 can be used for some practical LPV analysis and synthesis problems, with an example in section 10.4.

10.2 Background, notation, and conversion to $\mathcal{S}_c$ realizations

We will consider systems of the sort:

$$\Sigma: \quad \begin{bmatrix} \dot{x} \\ z \\ y \end{bmatrix} = \begin{bmatrix} A(\rho) & B_1(\rho) & B_2(\rho) \\ C_1(\rho) & D_{11}(\rho) & D_{12}(\rho) \\ C_2(\rho) & D_{21}(\rho) & D_{22}(\rho) \end{bmatrix} \begin{bmatrix} x \\ w \\ u \end{bmatrix} \tag{10.1}$$

where $\rho \in \hat{\mathbb{R}}_+$ (the extended positive real line; $\hat{\mathbb{R}}_+ = \mathbb{R}_+ \bigcup +\infty$) and $\alpha < \dot{\rho} < \beta$ are bounds on the measurable parameter $\rho(t)$ and its rate of variation, and the system matrices are assumed to have a proper rational parameter dependence, e.g. $A(\rho) = H + G(\rho I - E)^{-1}F$. At first glance this formulation seems ungainly; parameters of most physical systems won't vary over $\rho \in \hat{\mathbb{R}}_+$, however, some common situations may be transformed into this one:

Example 10.1 (Affine Parameter Dependence on an Interval) *Say we have an affine LPV operator $K(\mu) = \varsigma\mu + \kappa$ valid over $\mu \in [\alpha, \beta]$. We can perform a transformation to parameterize $\mu = \frac{\alpha-\beta}{\rho+1} + \beta$ over $\rho \in \hat{\mathbb{R}}_+$ and equivalently write: $K(\rho) = \varsigma\frac{\alpha-\beta}{\rho+1} + \varsigma\beta + \kappa$ or as a state space realization of an LFT:*

$$K(\rho) = \underbrace{\varsigma}_{G}(\rho I - \underbrace{(-1)}_{E})^{-1}\underbrace{(\alpha-\beta)}_{F} + \underbrace{\varsigma\beta + \kappa}_{H}$$

over $\rho \in \hat{\mathbb{R}}_+$. Note that this construction is well posed and easily generalizable to matrices. Also note that the above parametrization can be extended to any polynomial *in μ, a practical example of which we will exhibit in section 10.4.* ★

For our computational methods, the system in (10.1) is not yet suitable. Given some $K(\rho) = G(\rho I - E)^{-1}F + H$, $\rho \in \hat{\mathbb{R}}_+$, addition, multiplication, and inversion of such operators is trivial (and identical in form to that of LTI transfer functions over the imaginary axis or unit circle), but calculating the induced norm $\|K(\rho)\|$, or finding order reduced approximations $\tilde{K}(\rho) \approx K(\rho) \forall \rho \in \hat{\mathbb{R}}_+$, which will be integral to our iterative procedures, is not. Hence we will further reparametrize such $K(\rho) \quad \forall \rho \in \hat{\mathbb{R}}_+$ into mixed-causal transfer functions over the extended imaginary axis: $K(s), \quad \forall s \in \Im$.

For any well posed $K(\rho)$, this can trivially be accomplished via a change of variables: $K(\rho) = H + G(\rho I - E)^{-1}F = H - G(s^2 I + E)^{-1}F$ where $\rho = -s^2$. This parametrization can in turn be split up into stable and antistable parts, for a stable 'mixed causal' realization:

Lemma 10.1 *Assuming that* $K(-s^2) = H - G(s^2 I + E)^{-1}F$ *is well posed for all* $s \in \Im$*, then equivalently,*

$$K(-s^2) = H - G(sI - (-Z))^{-1}XF - GX(s^*I - (-Z))^{-1}F$$

where $Z := (-E)^{1/2}$ *with* $\Re(\lambda(Z)) > 0$ *and* X *is the unique solution to the Sylvester equation* $ZX + XZ + I = 0$*.* ***Proof:*** *Now since we assume that* $K(-s^2)$ *is well posed for all* $s \in \Im$*, then* E *must have no purely real eigenvalues in* $\mathbb{C}_+$*. Hence* $-E$ *has no purely real eigenvalues in* $\mathbb{C}_-$*, and a unique real* $(-E)^{1/2} := Z$ *exists with* $\Re(\lambda(Z)) > 0$ *[210]. We can hence rewrite:* $K(-s^2) = H - G\left[(sI + Z)(sI - Z)\right]^{-1}F$ *or* $K(-s^2) = H - \begin{bmatrix} G & 0 \end{bmatrix}(sI - \begin{bmatrix} -Z & I \\ 0 & Z \end{bmatrix})^{-1}\begin{bmatrix} 0 \\ F \end{bmatrix}$. *If we then solve the Sylvester equation* $ZX + XZ + I = 0$ *for the unique* X*, and apply the similarity transformation* $\begin{bmatrix} I & -X \\ 0 & I \end{bmatrix}$*, then we get:*

$$K(-s^2) = H - \begin{bmatrix} G & GX \end{bmatrix}(sI - \begin{bmatrix} -Z & 0 \\ 0 & Z \end{bmatrix})^{-1}\begin{bmatrix} XF \\ -F \end{bmatrix}$$

which we can seperate into stable 'causal' and 'anticausal' parts: $K(\rho) = H - G(sI - (-Z))^{-1}XF - GX(s^*I - (-Z))^{-1}F$ *(note that for* s *on the imaginary axis,* $s^* = -s$*), where we remember that* $\Re(\lambda(-Z)) < 0$ *and thus both parts are stable.* □

We have thus successfully rewritten the operator $K(\rho)$ in

$$K(s) = D + P(sI - R)^{-1}Q + U(s^*I - W)^{-1}V \tag{10.2}$$

form, where R and W are both strictly stable, but such stable mixed causal realizations of $K(s)$ are highly non unique, and given some $K(s)$ in transfer function form with high rational orders, finding a minimal realization, while trivial, is extremely computationally inefficient. However, note that $K(s)$ in this form is just an $\mathcal{S}_c$ realization from Chapter 2. Hence we can avoid this step in the following by only dealing with $\mathcal{S}_c$ realizations of $K(s)$ (instead of the elementwise transfer function version), and using the structure preserving arithmetic from Chapter 2 in the iterative analysis and design methods of Chapter 3.

10.3 Use for LPV analysis and synthesis

As we have outlined it above, we can put terms of the form $K(\rho)$ into $\mathcal{S}_c$ form, and thus, using the techniques in Chapters 2 and 3, we can check the positive definiteness of operators (and hence check whether some $X(\rho)$ satisfies a useful LMI or Riccati inequality), and solve Lyapunov and Riccati equations of the sort: $A(\rho)T(\rho)+T(\rho)A(\rho)^T+Q(\rho)=0$ or $A(\rho)S(\rho)+S(\rho)A(\rho)^T+Q(\rho)+S(\rho)R(\rho)S(\rho)=0$ *to arbitrary accuracy* over all $\rho \in \hat{\mathbb{R}}_+$.

However, LPV analysis and control for dynamically varying parameters requires more than this: there is a derivative term in the inequalities. For *verifying* solutions to such LMI's, this provides no added difficulty, for example:

Lemma 10.2

$$A(\rho)S(\rho) + S(\rho)A(\rho)^T + Q(\rho) + \dot{\rho}\frac{\partial S(\rho)}{\partial \rho} \quad \prec \quad 0 \tag{10.3}$$

over all $\rho \in \hat{\mathbb{R}}_+$, $\dot{\rho} \in [\alpha, \beta]$ *if and only if*

$$A(\rho)S(\rho) + S(\rho)A(\rho)^T + Q(\rho) + \gamma\frac{\partial S(\rho)}{\partial \rho} \quad \prec \quad 0$$

$\forall \rho \in \hat{\mathbb{R}}_+$ *for both* $\gamma \in \{\alpha, \beta\}$. ***Proof:*** *(10.3) is convex in* $\dot{\rho}$. ☐

where the derivative with respect to ρ can be calculated as:

Lemma 10.3 *Given* $X(\rho) = P(\rho I - R)^{-1}Q$, *bounded on* $\rho \in \hat{\mathbb{R}}_+$,

$$\frac{\partial X(\rho)}{\partial \rho} = \begin{bmatrix} P & 0 \end{bmatrix} \left(\rho I - \begin{bmatrix} R & -I \\ 0 & R \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 \\ Q \end{bmatrix} \tag{10.4}$$

Proof: *Chain rule.* ☐

Note that (10.4) is in the usual LFT form in ρ, which we can use the results of Lemma 10.1 to change back into a transfer matrix on s for further computations. Note also that $\frac{\partial X(\rho)}{\partial \rho}$ is bounded on $\rho \in \hat{\mathbb{R}}_+$ if $X(\rho)$ is bounded on $\rho \in \hat{\mathbb{R}}_+$.

So verifying (10.3) is as easy as checking positive definiteness of two parametric matrices over $\rho \in \hat{\mathbb{R}}_+$, which can be efficiently done in the $s \in \Im$ domain using Lemma 19 of Chapter 2.

However, actually finding such an $S(\rho)$ is much more difficult, and we give only a suggestion for solving such problems. We propose the following strategy (for

Lyapunov inequalities): pick some $\wp \in \mathbb{R}$, and solve sequentially for S_0, S_1, ... in:

$$\begin{aligned} A(\rho)S_0(\rho) + S_0(\rho)A(\rho)^T + Q(\rho) &= -\epsilon I \\ A(\rho)S_1(\rho) + S_1(\rho)A(\rho)^T + Q(\rho) + \wp\frac{\partial S_0(\rho)}{\partial\rho}) &= -\epsilon I \\ A(\rho)S_2(\rho) + S_2(\rho)A(\rho)^T + Q(\rho) + \wp\frac{\partial S_1(\rho)}{\partial\rho}) &= -\epsilon I \\ \cdots \end{aligned} \tag{10.5}$$

Since $A(\rho)$ is assumed stable, $S_{i+1}(\rho)$ is continuous in $\wp\frac{\partial S_i(\rho)}{\partial\rho}$, and we showed in Lemma 10.3 that $\frac{\partial S_i(\rho)}{\partial\rho}$ is continuous in $S_i(\rho)$, assuming that $S_i(\rho)$ is bounded on ρ, there thus exists a $\wp$ small enough such that $S_i(\rho)$ will converge to a unique $S_\infty(\rho)$ by the Banach Fixed Point Theorem [108], satisfying

$$A(\rho)S_\infty(\rho) + S_\infty(\rho)A(\rho)^T + Q(\rho) + \wp\frac{\partial S_\infty(\rho)}{\partial\rho}) = -\epsilon I$$

It then remains to check for what bounds $\wp_+ < \dot{\rho} < \wp_-$ our solution S_∞ satisfies the inequality (10.3), a problem like that in Lemma 10.2, which we can easily check. We note that this same idea works for the Riccati inequalities(where the convergence follows using the analyticity results from [211]).

Of course, this will only provide *a posteriori* rate bounds $\wp_+ < \dot{\rho} < \wp_-$ in which our closed loop system has stability or performance, and it is not yet clear how to pick $\wp$ and ϵ(or some other offset matrix) in (10.5) in order to achieve *a priori* desired bounds (see Chapter 11 for ideas).

10.4 Airfoil Flutter Example

For now we will apply our method to a simple static parameter model of a fluttering airfoil. Our model comes from section 4.9 of [212], wherein a 2-D quasi-static flutter model is derived for a 'smart' airfoil; i.e. one with trailing edge flap actuators. The model has four states, one controlled input (flap angle), and two measured outputs (angle of attack and vertical displacement), and is polynomially dependent on the freestream velocity, v, in the following form:

$$\dot{x} = (A_0 + A_1 v + A_2 v^2)x + (B_2 v^2)u, \qquad y = C_0 x$$

We would like to design a controller, $u = -K(v)x$ that stabilizes the system for all static values of $v \in [5, 15]\frac{m}{s}$. We do this via LQR; by solving the parameter dependent Riccati equation:

$$A(v)^T X(v) + X(v)A(v) + C^T C + X(v)B(v)B^T(v)X(v) \quad = \quad 0$$

for $X(v)$, and then using the feedback gain $K(v) = B^T(v)X(v)$. Using our methods outlined above, we found an $X(v)$ that satisfied the Riccati equation with a residual error norm of only $\approx 3.79 \times 10^{-7}$. In figure 10.1 we see a comparison of the open

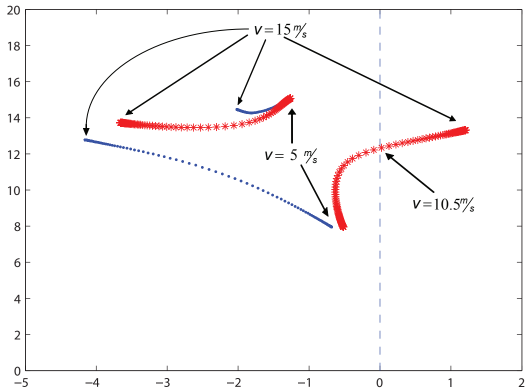


Figure 10.1: Open loop eigenvalues(*) vs closed loop eigenvalues($\cdot$)

loop vs closed loop spectrum: for open loop, the system goes unstable around $v = 10.5\frac{m}{s}$, but for closed loop the system is stable for the entire range $v \in [5, 15]\frac{m}{s}$.

This particular example is actually very ill-suited for our technique. As discussed in Chapter 3, while the ultimate convergence of the sign iteration for calculating $sign(H)$ is quadratic, the initial steps may be very slow, largely dependent on the location of the spectrum of the Hamiltonion, H. If the elements of the spectrum are close to $+1$ or -1, are mostly real, and stay away from the imaginary axis, convergence will be fast, otherwise it will be initially slow and nonmonotonic, requiring high rational orders. In our case of the flutter system above, as we see in figure 10.1, the closed loop spectrum is badly damped, with a relatively very large imaginary component, requiring ~ 17 sign iterations for convergence (5-7 usually suffices in the other examples in this thesis). Hence we expect that on other LPV examples, this technique will perform even better.

10.5 Conclusion

We have demonstrated a new computational approach for efficiently and accurately finding parametric solutions to rationally parametric Lyapunov and Riccati equations, and shown how this might be used in a number of LPV analysis and synthesis problems. We note that this might also be useful in certain robust control applications, and that while we have only developed the framework for a single parameter in this paper, all of the techniques should be extendable to multiple parameters via section 3.5, in the same way that the 1-D distributed system techniques in Chapter 5 can be extended to n-D in Chapter 8.

Future work could be devoted to case-study comparisons of this technique with other control methods for parameter dependent systems in terms of performance, and investigations of under what conditions (e.g. on the closed loop spectrum) this method provides significant improvements in computational complexity.

11

Reflections, Recommendations, and Conclusions

> "Men have become the tools of their tools"
>
> -Henry David Thoreau

11.1 Research Results

This research project was started in order to find robust ways to invert nonlinear systems, by any means necessary. For the practically minded author and promotor, this led to Iterative Learning Control(ILC) and Repetitive Control(RC) in the lifted setting, which led to large structured matrices. This structure and its generalizations were more useful than anticipated, leading to a number of unexpected applications, and provided 3 years worth of ideas for study.

The original structure turned out to be called 'Sequentially Semi-Separable'(SSS), which had already been extensively studied [47][63][48] resulting in $\mathcal{O}(N)$ computational complexity routines for structure preserving addition, multiplication, inversion, Cholesky factorization, and QR factorization. That is to say, for SSS matrices, these arithmetic operations result in SSS matrices, and only take $\mathcal{O}(N)$ flops to compute, compared to $\mathcal{O}(N^3)$ for unstructured dense matrices. We found that it is also possible to compute the spectral norm of SSS matrices in $\mathcal{O}(N)$ and perform structure preserving shuffle, or interlacing, permutations on SSS matrices also in $\mathcal{O}(N)$.

More importantly, we found that the derivation techniques for the SSS arithmetic could be extended and specialized to other related structures of matrices. If SSS matrices additionally have 'almost Toeplitz' structure (together called 'ATSSS' structure), then the SSS results can be specialized to be $\mathcal{O}(1)$ complexity, and preserve the ATSSS structure for all of the SSS arithmetic operations mentioned above.

Further, if the SSS structure is forced to be perfectly Toeplitz and extended to infinite length in both directions, the resulting operators are called 'Laurent operators with rational symbols' (L_r operators), and the SSS arithmetic can also be specialized to this case, including the structure preserving property. Such operators are equivalent, through a Fourier transform, to transfer functions on the unit circle, and this L_r arithmetic is thus also a computationally efficient method for dealing with such objects. The derivation of this arithmetic ties into a number of classic discrete domain systems theory results, such as the Bounded Real Lemma and the Positive Real Lemma, and provides a rare application for nonsymmetric Riccati equations.

In the same way as the arithmetic for the L_r symbols can be derived, which is equivalent to an arithmetic of transfer functions on the unit circle, a similar state-space based arithmetic can be derived for transfer functions on the imaginary axis, using the continuous time BRL and PRL, with the discrete domain nonsymmetric Riccati equations replaced by continuous domain nonsymmetric Riccati equations.

While some of these derivations and results are elegant and generally pleasant by themselves, all of the above matrix structures also have interesting applications to relevant and important control problems. As first stated, SSS matrices can be used to represent the lifted matrices in ILC/RC, and also to represent distributed systems consisting of an interconnection of arbitrarily heterogeneous subsystems on a line. Likewise, ATSSS matrices correspond to homogeneous interconnected systems on a line with boundary conditions, and L_r operators represent fully homogeneous, doubly infinite interconnected systems. Not surprisingly, given the known connections between LPV synthesis and distributed systems, the arithmetic of transfer functions on the imaginary axis can be used to represent certain types of rationally dependent LPV systems. In Figure 11.1 we see the connections between the matrix structures and eachother, and their respective areas of application.

In some of these applications, just the introduction of an efficient arithmetic and norm calculation for their repsective operators is a step forward. However, for all of the four structured arithmetics above, it is also possible to use the 'sign iteration' and other structure preserving iterations to check stability, solve Riccati equations, synthesize H_∞ controllers, compute $D-K$ robustness iterations, and perform model order reductions, all in a structure preserving way. Given some basic assumptions on the systems to be controlled, these routines are very computationally efficient; e.g. $\mathcal{O}(N)$ for systems represented by SSS matrices and $\mathcal{O}(1)$ for systems represented by ATSSS matrices. Furthermore, since this method finds structured approximations arbitrarily close to the centralized solutions, the resulting controllers are arbitrarily close to optimal, a great result for the perfectly homogeneous distributed system and LPV problems.

For the distributed systems, not only do interconnected systems induce our structured operators, but the opposite holds true as well, hence if the structure preserving arithmetic is used to construct a controller that has the same structure as the system, then the controller can be perfectly distributed, in the same interconnection topology as the system. This is illustrated in figure 11.2, where the different types of distributed systems are presented in 3 columns, and the process of lifting, structured controller synthesis, and redistribution is presented from top to bottom. (Note, this is actually my poster from the 2009 CDC. I include the whole thing because I think it presents the results in a very understandable way).

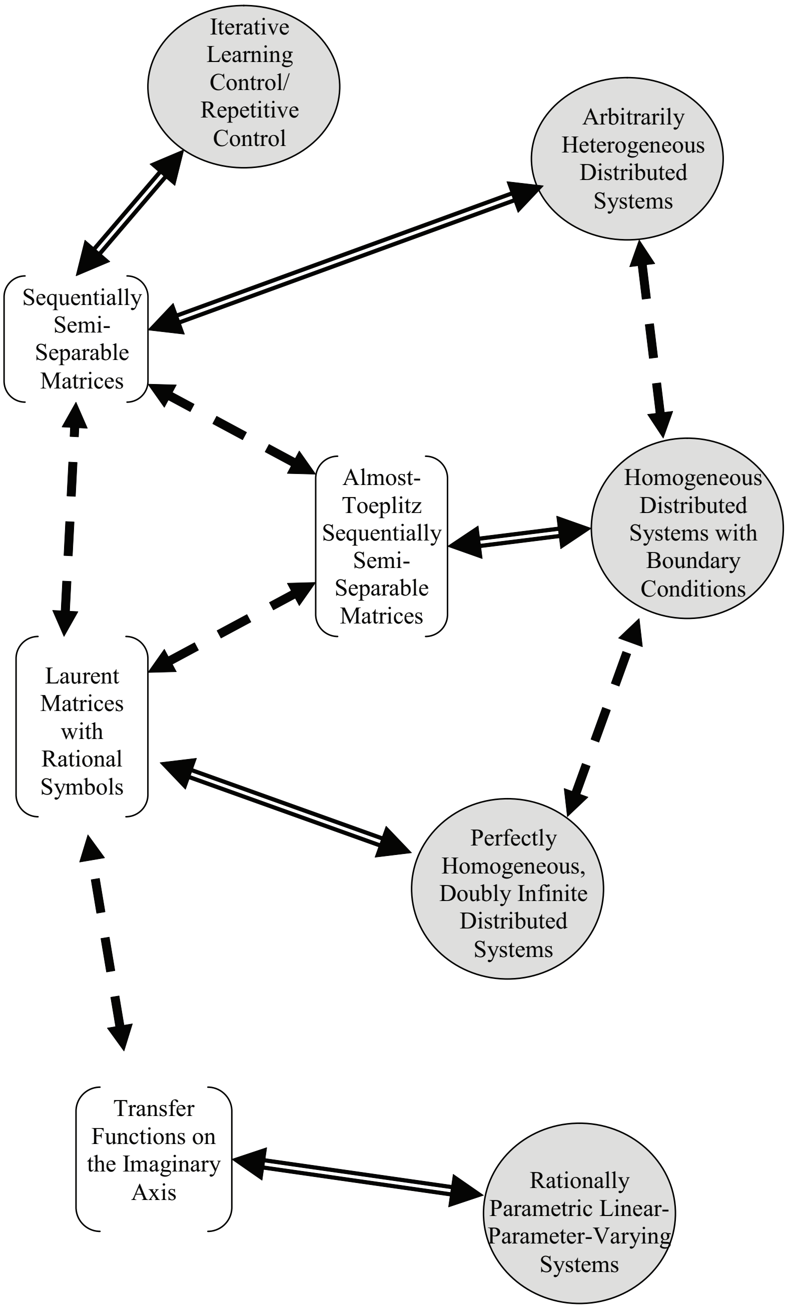

Iterative Learning Control/ Repetitive Control
Arbitrarily Heterogeneous Distributed Systems
Sequentially Semi-Separable Matrices
Almost-Toeplitz Sequentially Semi-Separable Matrices
Homogeneous Distributed Systems with Boundary Conditions
Laurent Matrices with Rational Symbols
Perfectly Homogeneous, Doubly Infinite Distributed Systems
Transfer Functions on the Imaginary Axis
Rationally Parametric Linear-Parameter-Varying Systems

Figure 11.1: Conceptual diagram of matrix structures and their relationships to applications. Double-lines indicate equivalence, while dotted lines indicate structural similarity

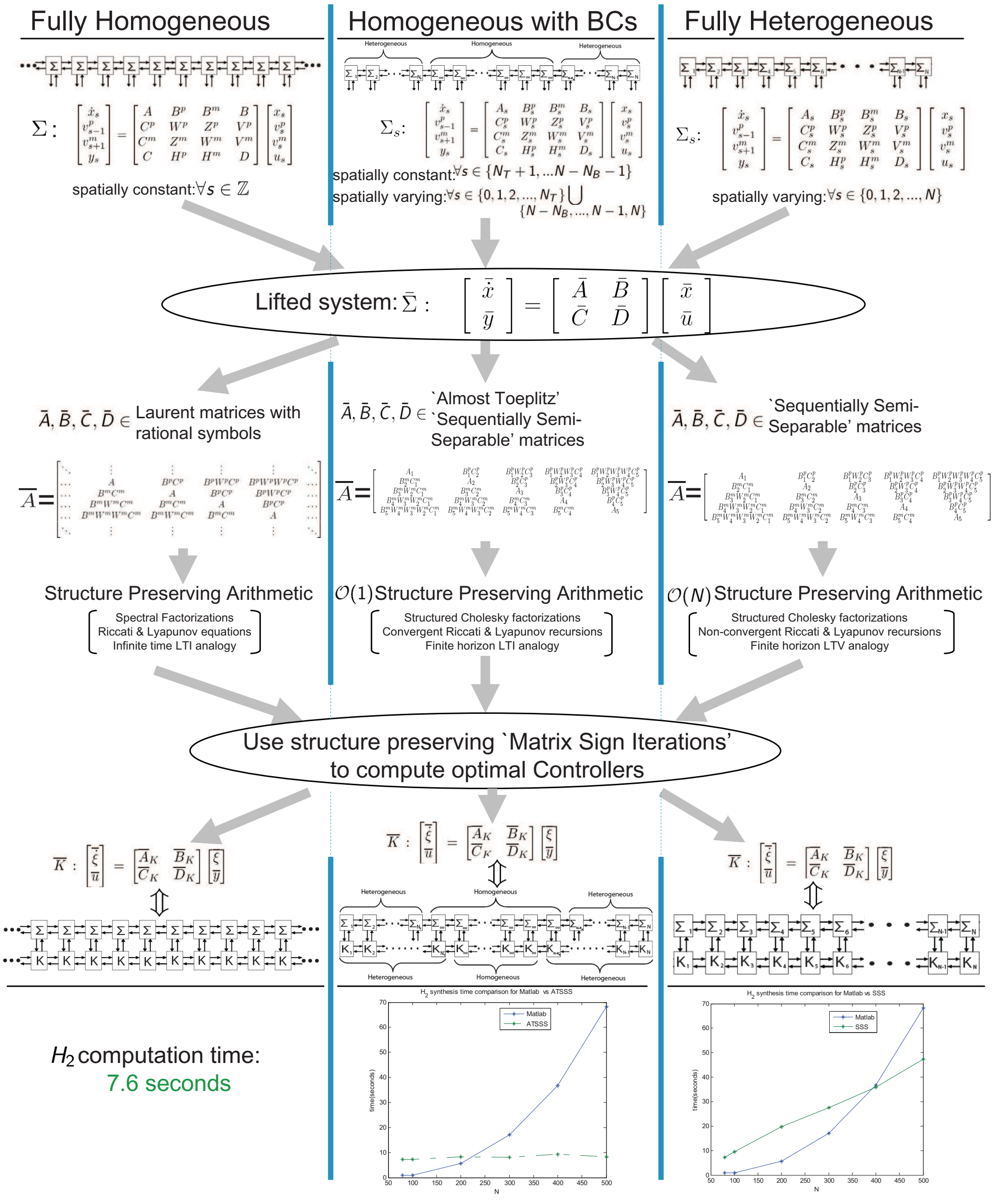

Fully Homogeneous
Homogeneous with BCs
Fully Heterogeneous
Heterogeneous
Homogeneous
Heterogeneous
spatially constant: $\forall s \in \mathbb{Z}$
spatially constant: $\forall s \in \{N_T+1, \dots N-N_B-1\}$
spatially varying: $\forall s \in \{0,1,2,\dots,N_T\} \bigcup \{N-N_B,\dots,N-1,N\}$
spatially varying: $\forall s \in \{0,1,2,\dots,N\}$
Lifted system: $\bar{\Sigma}$:
$\bar{A},\bar{B},\bar{C},\bar{D} \in$ Laurent matrices with rational symbols
$\bar{A},\bar{B},\bar{C},\bar{D} \in$ `Almost Toeplitz' `Sequentially Semi-Separable' matrices
$\bar{A},\bar{B},\bar{C},\bar{D} \in$ `Sequentially Semi-Separable' matrices
Structure Preserving Arithmetic
Spectral Factorizations
Riccati & Lyapunov equations
Infinite time LTI analogy
$\mathcal{O}(1)$ Structure Preserving Arithmetic
Structured Cholesky factorizations
Convergent Riccati & Lyapunov recursions
Finite horizon LTI analogy
$\mathcal{O}(N)$ Structure Preserving Arithmetic
Structured Cholesky factorizations
Non-convergent Riccati & Lyapunov recursions
Finite horizon LTV analogy
Use structure preserving `Matrix Sign Iterations' to compute optimal Controllers
H_2 computation time:
7.6 seconds
H_2 synthesis time comparison for Matlab vs ATSSS
H_2 synthesis time comparison for Matlab vs SSS
Matlab
ATSSS
SSS
time(seconds)
N

Figure 11.2: Conceptual diagram of controller synthesis for distributed systems

Furthermore, the dimension of the interconnection variables of the subcontrollers, K_s, is the sum of the orders of the structured matrices in the realization of the controller $\bar{K}$. Since there are efficient ($\mathcal{O}(N)$ for SSS and $\mathcal{O}(1)$ for ATSSS) order reduction techniques, this allows one to easily make a tradeoff between performance and the required connectivity in the distributed controller. Such order reductions can be performed cheaply after synthesis, making this a nice tuning method.

For fully heterogeneous distributed systems, exploitation of the SSS structure also makes efficient ($\mathcal{O}(N)$) parametric system identification possible, where the resulting system model is in the correct form for analysis and controller design using the above results. Furthermore, for fully heterogeneous problems, due to the special form taken by the SSS arithmetic it's actually possible to perform all of these SysID, analysis and synthesis computations in a distributed manner, on a linear interconnection of microprocessors with distributed memory. Hence, given an unknown heterogeneous distributed system on a cartesian grid, it should now be possible to distribute microprocessors with local memory, sensing, actuation, and communication abilities, and have them first perform a distributed system identification, then using the resulting model, a distributed controller synthesis, and then to analyze the resulting closed loop performance, all in linear computational complexity ($\mathcal{O}(N)$ for N subsystems on a line)[1]. These results should also make adaptive distributed control computationally feasible, resulting in what we could call 'distributed intelligence'.

All of these results have been for 'one-dimensional' problems; the distributed systems are all on a line, the LPV systems are dependent on only one variable. However, by using the structure preserving iterations on the first level to solve structured Lyapunov and Riccati equations and perform model order reductions, we can 'lift ourselves up by our own bootstraps' and create an arithmetic on the second level, and so on, extending all of our above computational results to multilevel operators. Furthermore, distributed systems on cartesian grids can be understood as 'strings of strings' as we see in figure 11.3 (borrowed from Chapter 8), and hence be modeled by multilevel structured operators. The same idea applies to certain types of LPV systems dependent on multiple variables, and could even be used to mix the types of structures together on different levels. For example, one could in theory use these results for repetitive control of an infinite dimensional homogeneous string of interconnected LPV systems, resulting in a model with a realization of SSS matrices with generators of Laurent operators which in turn have realizations consisting of transfer functions on the imaginary axis (3-level operators).

The usefulness of these results has been shown on numerous computational examples, where their computational complexity and accuracy have been demonstrated and favorably compared with other techniques from the literature, where possible.

---

[1] Note that the $\mathcal{O}(N)$ complexity is *proven* in this thesis for all of these computations except for SysID, for which it is only observed in examples, see Chapter 4

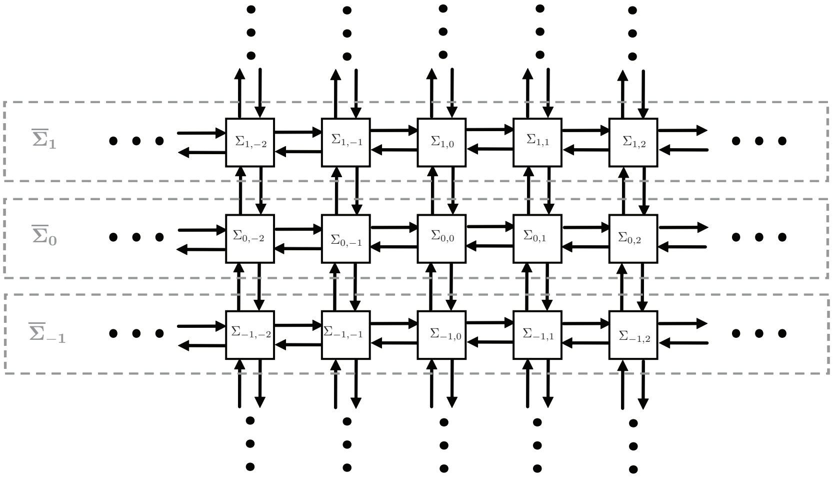


Figure 11.3: 2-D array ⇒ 1-D String of 1-D Strings

11.2 Research Recommendations

The results presented in this thesis are satisfying, with many of the obvious questions answered, but there are still many opportunities for improvement and extension of these techniques.

- The $\mathcal{S}_c$ realizations of transfer functions on the imaginary axis were the last structure to be studied, and the least developed. In addition to LPV applications, such realizations can also be used to represent operators in infinite extent homogeneous distributed continuum systems, e.g. PDE operators without boundary conditions. Further, in the same way that we extended the arithmetic of $\mathcal{S}$ realizations to SSS matrices and ATSSS matrices in Chapter 2, it should be possible to extend the $\mathcal{S}_c$ arithmetic to represent continuum PDE operators that vary arbitrarily in space, or are homogeneous except at the boundaries. In the inversion formula in such arithmetics we would have a differential matrix Riccati equation, instead of the steady state matrix Riccati equation in Lemma 2.19, and it would be necessary to use some numerical integration method (e.g. Runga Kutta) to solve it from one boundary to the other. We conjecture that in the continuum analog of the ATSSS case, the Riccati solution would similarly converge to steady state in the interior, leading to computational savings. Similarly, for the multiplication and model order reduction operations, we would have differential Lyapunov equations, but they would also converge in the interior in the homogeneous-with-BC's case. The computational complexity of such approaches would depend on the stepsize, and this might be a better way to deal with control of PDE systems, instead of first discretizing using finite difference, then designing controllers.
- Also, we once used the Carleman linearization(e.g. [213]) to approximate nonlinear maps in an attempt to use SSS methods for control of nonlinear

systems, but the SSS realizations proved to be unstable, leading to numerical problems. A better idea may be to convert some nonlinear(but rational) function $A(x)$ on $x \in [a, b]$ into a mixed causal stable transfer function $A(s)$ on the imaginary axis using Example 10.1 and Lemma 10.1, then to try to develop some structure preserving arithmetic for use with the sign iteration.

- The ATSSS arithmetic was also developed late in the project, and probably has many more applications. For example, if the LTV or periodic systems used in ILC/RC in Chapter 9 were LTI instead, then the resulting lifted matrices would have ATSSS structure instead of SSS. Hence in this special case (which nonetheless applies to many ILC/RC problems), it should be possible to reduce the $\mathcal{O}(N)$ complexity of SSS controller synthesis to the $\mathcal{O}(1)$ complexity of ATSSS matrices.

- In the application of ATSSS matrices to distributed control of homogeneous systems with boundary conditions in Chapter 6, we mentioned that the resulting controller $\bar{K}$ with ATSSS structure would actually have the same ATSSS 'generators' no matter how large the interior got, hence only one synthesis computation needs to be performed, whether the intended system has size $N = 100$, $N = 1000$, or $N = 10^8$. We note here that this might also have an interesting application in distributed systems that *change* size.

  For example, vehicles might join or leave a homogeneous platoon while it is driving down the highway, and the ATSSS generators produced by one synthesis problem will correspond to ATSSS controllers that are stable, no matter what the length, but stability of employing these controllers while the platoon changes length is another question, and is an interesting switched stability problem. Define the length of the interior as $N_{int} = N - (N_T + N_B)$, and assume we've computed an ATSSS state feedback controller $\bar{K}$ such that $\bar{A}_{cl} = \bar{A} - \bar{B}\bar{K}$ is stable for one N_{int} and hence stable for all N_{int} larger than some threshold and less than infinity. Since $\bar{A}_{cl}$ is stable for any chosen N_{int}, the solution, $\bar{P}$, to the Lyapunov equation $\bar{A}_{cl}\bar{P} + \bar{P}\bar{A}_{cl} + I = 0$ will also have the ATSSS structure, and its generators will also satisify the Lyapunov equation for other interior lengths, N_{int}. But while the ATSSS *generators* of $\bar{P}$ satisfy a Lyapunov inequality for the *generators* of $\bar{A}_{cl}$ for many different lengths N_{int}, these different lengths induce different actual matrices $\bar{P}$, and hence it is not exactly a joint Lyapunov function, and does not prove switch stability. However, in simulation such systems do *seem* to be switch-stable, so it would be interesting to further investigate if and under what conditions this can actually be proven.

- Also, we specified the 'almost Toeplitz' component of the ATSSS definition as Toeplitz in the middle, with heterogeneities at the ends, but this could be generalized to SSS matrices that are Toeplitz almost everywhere, with just a few spots of non-Toeplitzness anywhere in the matrix. The iterations in the ATSSS arithmetic would then proceed from one point to another, converging in each of the long Toeplitz sections, and thus would still be $\mathcal{O}(1)$ for very large interior Toeplitz sections. Such matrices could be used to represent mostly homogeneous systems with boundary conditions and just a few problem spots of heterogeneity. Furthermore, the Toeplitz sections would not even have to be homogeneous with respect to eachother. An application of this might be heat conduction on a very long but finite discretized bar, where the heat conduction coefficient changes at the half-way point.

- Another application that has been investigated for SSS matrices is in fast optical flow computation [214]. Since the matrices involved represent two dimensional Laplacians, they should fit into our multilevel SSS or even ATSSS matrices, making for very fast computations, particularly useful, as mentioned in the reference, for HDTV resolutions of 1920×1080 or higher (an 'ultra high def' TV was developed in 2006 with 7680×4320 pixels).

- While our examples and discussion have focussed on the case where an interconnected system has controllable inputs and measurable outputs everywhere or almost everywhere, it turns out that in some cases, the results of this thesis may be profitably used for boundary control and/or boundary estimation. To explain, if the open loop spectrum of the system is well into the left half plane and nicely damped, no matter what the actuators are, a quadratic optimal control design is unlikely to place the closed-loop poles in a badly damped place. Hence the sign iteration will converge quickly, and a nice, low-order, boundary controller will be produced. A physical example on which this works well is the stabilized and discretized heat conduction problem studied in section 4.6.2 modified to only have dual boundary actuators. We suggest that many boundary control problems are open loop stable with decent damping, and hence this approach may have wide application, and should be further studied.

- Heterogeneous subsystems connected in a 'loop' instead of a string also can be lifted to form an interconnected system with SSS state space matrices (although with much more complicated generators), and thus some of the results herein described (such as stability and performance analysis) should also be readily applicable to these systems. However, for such systems, after a controller, $\overline{K}$, with SSS matrix realization is computed, it is not yet clear how to extract subcontrollers, K_s, that are also interconnected in a *loop*, instead of a string with very large feedthrough terms. This problem could be the subject of future research. We also note that for the matrices induced by such systems, it may be better to develop a new arithmetic based on SSS methods but specialized to the circulant structure(see also the next item).

- While we have focused in this thesis on structures of input-output operators of linear systems lifted over time, it is possible the general idea of structure preserving iterations to quickly calculate structured controllers with special implementations can be extended to other structures also. Indeed, the first structure preserving use of the sign iteration for control design was using Hierarchical matrices ($\mathcal{H}$-matrices) for $\mathcal{O}(Nlog(N))$ computation of LQR for a heat conduction problem [28]. The results for control using $\mathcal{H}$ matrices are not very complete; it is unclear how to make guarantees against controller fragility, or if distributed implementation and distributed computation are possible, but the potential for $\mathcal{H}$ matrices to represent much more complicated tree interconnections is very interesting. We can imagine how an investigation of the similarities and differences between $\mathcal{H}$ and SSS arithmetic could lead to generalizations of both, perhaps leading to arithmetics for any kind of data-sparse matrices representing interconnected subsystems. Hierarchically semi-separable (HSS)[215] matrices seem to be a first step in this direction.

- One of the least satisfying aspects of our results is the sometimes-slow convergence of the sign iteration when solving Riccati equations (as shown in section 4.7). We recognize that this is simply due to badly damped closed loop poles or a 'stiff' system, but as of yet there is nothing we can do about it.

  One idea for approaching this problem is as follows. In almost all of our techniques, we solve a Riccati *equation* to find the *optimal* solution, but there exist relaxations to Riccati inequalities that provide suboptimal solutions. While the sign iteration and doubling algorithms cannot yet be used to solve general LMI problems, there has recently been promising work on solving a sequence of Riccati equations to arrive at an LMI solution[216], and it would be interesting to see if these methods could be used to exploit this freedom to find faster converging solutions. For example, if an SSS structure preserving method can be found for solving certain LMI's, then the closed loop spectrum can be bounded inside an LMI area, perhaps guaranteeing fast convergence of the sign iterations, and hence low SSS ordered controllers. Alternatively, one could try to enforce a banded (or some other) sparsity structure to the suboptimal solutions. Such approaches might also benefit the LPV work in Chapter 10, wherein an LMI could perhaps be solved to find controllers stable for larger parameter variation rates.

- Finally, the most difficult part of distributed control, beyond theory, computation, modeling, or even identification, is implementation, and it would be nice to see how the results of this thesis might fair on real systems. While their are many potential applications for 1-D distributed systems (see the Introduction chapter), there are not many for 2-D systems where it would be feasible to actually build a setup with enough actuators, microcontrollers, and sensors to make our structured techniques worth-while compared to 1-D SSS methods (We estimate perhaps 1000×1000). One potential application is in adaptive optics, which seem favorable since the sensors and actuators are cheap and the funding is plentiful. Furthermore, since such systems are generally modeled as being homogeneous with boundary conditions, 2-level ATSSS matrix methods might be applied to great effect, although some care will need to be taken regarding the changing string length (such adaptive optics sensor arrays are usually hexagonal in shape). However, we suggest that this presents no fundamental difficulty, since the three sets of ATSSS generators *will* be the same from line to line, the size of the interiors will just be different, but this has no affect on the computations, just book-keeping.

- For any real application of these techniques, robustness will be an important factor to consider, given the necessary uncertainty of first principles distributed system models, especially with respect to the sub-controller communication links. Given that the robust analysis and synthesis methods in Chapter 7 seem to work well in numerical examples, further research should be spent on modeling sub-controller communication errors and failures in this format so that distributed controllers can be designed which are non-fragile.

11.3 Reflections

It is very rewarding to arrive at these nice, extremely fast algorithms, however, these methods still have an enormous way to go before being viable for real world applications. As we discussed in the introduction to this thesis, there are fantastic distrubuted controllers out there being implemented on difficult nonlinear time varying and uncertain distributed systems, but they've been designed by either 4.5 billion years of evolutionary optimization, or by God. Either way I think the successful control of simple examples in the literature gives a false sense of what is possible for us as engineers.

Bibliography

[1] D. S. Henningson, M. Hogberg, and M. Chevalier, “Optimal feedback control applied to boundary layer flow,” *Journal of Turbulence*, vol. 6, no. 25, 2005.

[2] T. R. Bewley and S. Liu, “Optimal and robust control and estimation of linear paths to transition,” *Journal of Fluid Mechanics*, vol. 365, pp. 205–349, 1998.

[3] F. Wu and S. E. Yildizoglu, “Distributed parameter-dependent modeling and control of flexible structures,” *Transactions of the ASME*, vol. 127, pp. 230–239, 2005.

[4] R. D’Andrea and G. E. Dullerud, “Distributed control design for spatially interconnected systems,” *IEEE Trans. Autom. Control*, vol. 48, pp. 1478–1495, 2003.

[5] R. Horowitz and P. Varaiya, “Control design of an automated highway system,” *Proceedings of the IEEE*, vol. 88, pp. 913–925, 2000.

[6] D. Swaroop and J. Hedrick, “String stability of interconnected systems,” *IEEE Trans. Autom. Control*, vol. 41, pp. 349–357, 1996.

[7] D. D. Givone and R. P. Roesser, “Multidimensional linear iterative circuits- general properties,” *IEEE Transactions on Computers*, vol. 21, pp. 1067–1073, 1972.

[8] J. P. Lynch and K. H. Law, “Decentralized control techniques for large-scale civil structural systems,” *Proceedings of the IMAC*, 2002.

[9] J. Fowler and R. D’Andrea, “A formation flight experiment,” *IEEE Control Systems Magazine*, vol. 23, pp. 35–43, 2003.

[10] P. Massioni and M. Verhaegen, “Distributed control for identical dynamically coupled systems: A decomposition approach,” *IEEE Trans. Autom. Control*, vol. 54, pp. 124–135, 2009.

[11] S. Jiang, P. G. Voulgaris, L. E. Holloway, and L. A. Thompson, “Distributed control of large segmented telescopes,” in *Proc. Amer. Control Conf.*, 2006.

[12] R. P. Roesser, “A discrete state-space model for linear image processing,” *IEEE Trans. Autom. Control*, vol. 20, pp. 1–10, 1975.

[13] G. E. Stewart, D. M. Gorinevsky, and G. A. Dumont, “Feedback controller design for a spatially distributed system: The paper machine problem,” *IEEE Transactions on Control Systems Technology*, vol. 11, pp. 612–628, 2003.

[14] M. Cantoni, E. Weyer, Y. Li, S. K. Ooi, I. Mareels, and M. Ryan, “Control of large scale irrigation networks,” *Proceedings of the IEEE*, vol. 95, pp. 75–91, 2007.

[15] M. Kishida, A. N. Ford, D. W. Pack, and R. D. Braatz, “Optimal control of cellular uptake in tissue engineering,” *Proc. Amer. Control Conf.*, pp. 2118–2123, 2008.

[16] M. R. Jovanović, M. Arcak, and E. D. Sontag, “Remarks on the stability of spatially distributed systems with a cyclic interconnection structure,” *Proc. Amer. Control Conf.*, pp. 2696–2701, 2007.

[17] A. D. Hansen, P. Sorensen, F. Iov, and F. Blaabjerg, "Centralised power control of wind farm with doubly fed induction generators," *Renewable Energy*, vol. 31, pp. 935–951, 2006.

[18] M. Soleimanzadeh, R. Wisniewski, and S. M. Shakeri, "Wind deficit model in a wind farm using finite volume method," *Proceedings of the American Control Conference*, 2010.

[19] R. Padhi and S. Balakrishnan, "Optimal management of beaver population using a reduced-order distributed parameter model and single network adaptive critics," *IEEE Trans. on Control Syst. Technol.*, vol. 14, pp. 628–640, 2006.

[20] M. Jovanovic and B. Bamieh, "The spatio-temporal impulse response of the linearized Navier-Stokes equations," *proceedings of the ACC*, pp. 1948–1953, 2001.

[21] I. D. Walker, D. M. Dawson, T. Flash, F. W. Grasso, R. T. Hanlon, B. Hochner, W. M. Kier, C. C. Pagano, C. D. Rahn, and Q. M. Zhang, "Continuum robot arms inspired by cephalopods," *Proc. SPIE*, vol. 5804, 2005.

[22] R. T. Hanlon and J. B. Messenger, *Cephalopod Behaviour.* Cambridge University Press, 2008.

[23] W. K. Potts, "The chorus-line hypothesis of manoeuvre coordination in avian flocks," *Nature*, vol. 309, pp. 344–345, 1984.

[24] R. W. Brockett and J. L. Willems, "Discretized partial differential equations: Examples of control systems defined on modules," *Automatica*, vol. 10, pp. 507–515, 1974.

[25] M. Hovd, R. D. Braatz, and S. Skogestad, "SVD controllers for H_2-, H_∞- and μ-optimal control," *Automatica*, vol. 33, pp. 433–439, 1997.

[26] B. Bamieh, F. Paganini, and M. A. Dahleh, "Distributed control of spatially invariant systems," *IEEE Trans. Autom. Control*, vol. 47, pp. 1091–1106, 2002.

[27] I. Rosen and C. Wang, "A multilevel technique for the approximate solution of operator Lyapunov and algebraic Riccati equations," *SIAM Journal of Numerical Analysis*, vol. 32, pp. 514–541, 1995.

[28] L. Grasedyck, W. Hackbusch, and B. Khoromskij, "Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices," *Computing*, vol. 70, pp. 121–165, 2003.

[29] M. Fardad, "The operator algebra of almost Toeplitz matrices and the optimal control of large-scale systems," *Proc. Amer. Control Conf.*, pp. 854–859, 2009.

[30] N. Motee and A. Jadbabaie, "Approximation methods and spatial interpolation in distributed control systems," *Proc. Amer. Control Conf.*, pp. 860–865, 2009.

[31] C. Langbort and R. D'Andrea, "Distributed control of spatially reversible interconnected systems with boundary conditions," *SIAM Journal on Control & Optimization*, vol. 44, pp. 1–28, 2005.

[32] G. E. Dullerud and R. D'Andrea, "Distributed control of heterogeneous systems," *IEEE Trans. Autom. Control*, vol. 49, pp. 2113–2128, 2004.

[33] C. Langbort, R. Chandra, and R. D'Andrea, "Distributed control design for systems interconnected over an arbitrary graph," *IEEE Trans. Autom. Control*, vol. 49, pp. 1502–1519, 2004.

[34] F. Wu, "Distributed control for interconnected linear parameter-dependent systems," *IEEE Proceedings on Control Theory Applications*, vol. 150, pp. 518–527, 2003.

[35] C. Langbort, X. Lin, R. D'Andrea, and S. Boyd, "A decomposition approach to distributed analysis of networked systems," *IEEE Conference on Decision and Control*, 2004.

[36] A. Rantzer, "On prize mechanisms in linear quadratic team theory," *Proc. IEEE Conf. Decision and Control*, pp. 1112–1116, 2007.

[37] T. Keviczky, F. Borrelli, and G. J. Balas, "Decentralized receding horizon control for large scale dynamically decoupled systems," *Automatica*, vol. 42, pp. 2105–2115, 2006.

[38] A. Sarwar, "Spatiotemporal systems: Gradual variations, indentification, adaptation and robustness," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2009.

[39] A. P. Featherstone and R. D. Braatz, "Control relevant identification of sheet and film processes," *Proceedings of the American Control Conference*, pp. 2692–2696, 1996.

[40] P. Massioni and M. Verhaegen, "Subspace identification of circulant systems," *Automatica*, vol. 44, pp. 2825–2833, 2008.

[41] A. Sarwar and P. Voulgaris, "Spatially invariant systems: identification and adaptation," *Proc. of the 3rd Int. Conf. on App. Math., Sim., Modelling, Circuits, Systems and Signals*, pp. 208–215, 2009.

[42] P. Massioni and M. Verhaegen, "Subspace identification of distributed, decomposable systems," *IEEE Conference on Decision and Control*, pp. 3364–3369, 2009.

[43] M. Ali, S. S. Chughtai, and H. Werner, "Identification of spatially interconnected systems," *IEEE Conference on Decision and Control*, pp. 7163–7168, 2009.

[44] M. Verhaegen and V. Verdult, *Filtering and System Identification- A Least Squares Approach*. Cambridge University Press, 2007.

[45] D. E. Knuth, "Big omicron and big omega and big theta," *ACM SIGACT News*, vol. 8, pp. 18–24, 1976.

[46] G. H. Golub and C. F. V. Loan, *Matrix Computations*. JHU Press, 1996.

[47] P. Dewilde and A. J. V. D. Veen, *Time-Varying systems and Computations*. Kluwer Academic Publishers, 1998.

[48] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, A.-J. van der Veen, and D. White, "Fast stable solvers for sequentially semi- separable linear systems of equations," *Report, Lawrence Livermore National Laboratory*, 2003.

[49] W. Hackbusch, "A sparse matrix arithmetic based on $\mathcal{H}$-matrices, part i: Introduction to $\mathcal{H}$-matrices," *Computing*, vol. 62, pp. 1436–5057, 1999.

[50] U. Baur and P. Benner, "Gramian-based model reduction for data-sparse systems," *Preprint*, 2007.

[51] A. Bottcher and B. Silberman, *Introduction to Large Truncated Toeplitz Matrices*. Springer, 1991.

[52] I. Gohberg, S. Goldberg, and M. Kaashoek, *Classes of Linear Operators Vol. II*, I. Gohberg, Ed. Birkhauser Verlag, 1993.

[53] S. Boyd and V. Balakrishnan, "A regularity result for the singular values of a transfer matrix and a quadratically convergent algorithm for computing its L_∞ norm," *Systems & Control Letters*, vol. 15, pp. 1–7, 1990.

[54] P. S. Stanimirovi'c, "A finite algorithm for generalized inverses of polynomial and rational matrices," *Applied Mathematics and Computation*, vol. 144, pp. 199–214, 2003.

[55] P. S. Stanimirovi'c and M. B. Tasi'c, "Computing generalized inverses using lu factorization of matrix product," *International Journal of computer Mathematics*, vol. 85, pp. 1865–1878, 2008.

[56] L. M. Silverman, "Realization of linear dynamical systems," *IEEE Transactions on Automatic Control*, vol. AC-16, pp. 554–567, 1971.

[57] N. Motee and A. Jadbabaie, "Optimal control of spatially distributed systems," *IEEE Trans. Autom. Control*, vol. 53, pp. 1616–1629, 2008.

[58] S. Boyd and J. Doyle, "Comparison of peak and RMS gains for discrete-time systems," *Systems & Control Letters*, vol. 9, pp. 1–6, 1987.

[59] B. Moore, "Principal component analysis in linear systems: Controllability, observability, and model reduction," *IEEE Transactions on Automatic Control*, vol. AC-26, pp. 17–31, 1981.

[60] K. Zhou, K. Glover, and J. C. Doyle, *Robust and Optimal Control.* Prentice Hall, 1996.

[61] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*, T. Kailath, Ed. Prentice Hall, 2000.

[62] L.-S. Hahn and B. Epstein, *Classical Complex Analysis.* Jones & Bartlett, 1996.

[63] Y. Eidelman and I. Gohberg, "On a new class of structured matrices," *Integral Equations and Operator Theory*, vol. 34, pp. 292–324, 1999.

[64] T. Kailath, "Fredholm resolvents, weiner-hopf equations, and riccati differential equations," *IEEE Transactions on Information Theory*, vol. 15, pp. 665–672, 1969.

[65] A.-J. van der Veen and P. Dewilde, "Large matrix inversion using state space techniques," *Workshop on VLSI Signal Processing*, 1993.

[66] S. Chandrasekaran, M. Gu, X. Sun, J. Xia, and J. Zhu, "A superfast algorithm for toeplitz systems of linear equations," *SIAM Journal on Matrix Analysis and Applications*, vol. 29, pp. 1247–1266, 2007.

[67] T. Bella, V. Olshevsky, and P. Zhlobich, "Classifications of recurrence relations via subclasses of (h,m)-quasiseparable matrices," *SIAM Journal of Matrix Analysis*, p. in press, 2007.

[68] T. Bella, Y. Eidelman, I. Gohberg, and V. Olshevsky, "Computations with quasiseparable polynomials and matrices," *Theoretical Computer Science*, vol. 409, pp. 158–179, 2008.

[69] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, X. Sun, A. J. V. der Veen, and D. White, "Some fast algorithms for sequentially semi-separable representations," *SIAM Journal of Matrix Analysis and Applications*, vol. 27, pp. 341–364, 2005.

[70] Y. Eidelman and I. Gohberg, "On generators of quasiseparable finite block matrices," *Calcolo*, vol. 42, pp. 187–214, 2005.

[71] Y. Eidelman, I. Gohberg, and V. Olshevsky, "The QR iteration method for Hermitian quasiseparable matrices of an arbitrary order," *Linear Algebra and its Applications*, vol. 404, pp. 305–324, 2005.

[72] ——, "Eigenstructure of order-one-quasiseparable matrices. three-term and two-term recurrence relations," *Linear Algebra and its Applications*, vol. 405, pp. 1–40, 2005.

[73] L. Gemignani, “Quasiseparable structures of companion pencils under the qz algorithm,” *Cacolo*, vol. 42, 2005.

[74] E. Alijagic and P. Dewilde, “Minimal quasi-separable realizations for the inverse of a quasi-separable operator,” *Linear Algebra and its Applications*, vol. 414, pp. 445–463, 2006.

[75] S. Chandrasekaran, M. Gu, J. Xia, and J. Zhu, “A fast QR algorithm for companion matrices,” *Operator Theory: Advances and Applications*, vol. 179, pp. 111–143, 2007.

[76] Y. Eidelman and I. Gohberg, “Out-of-band quasiseparable matrices,” *Linear Algebra and its Applications*, vol. 429, pp. 266–289, 2008.

[77] S. Delvaux and M. van Barel, “A qr-based solver for rank structured matrices,” *SIAM. J. Matrix Anal. & Appl.*, vol. 30, pp. 464–490, 2008.

[78] R. Vandebril, G. Golub, and M. V. Barel, “A quasi-separable approach to solve the symmetric definite tridiagonal generalized eigenvalue problem,” *SIAM. J. Matrix Anal. & Appl.*, vol. 31, pp. 154–174, 2009.

[79] P. Dewilde and A.-J. van der Veen, “Innerouter factorization and the inversion of locally finite systems of equations,” *Linear Algebra and its Applications*, vol. 313, pp. 53–100, 2000.

[80] H. Sandberg and A. Rantzer, “Balanced truncation of linear time-varying systems,” *IEEE Trans. Autom. Control*, vol. 49, pp. 217–229, 2004.

[81] A. Householder, *The Theory of Matrices in Numerical Analysis.* Blaisdell Publishing Company, 1964.

[82] C. Kenney, A. J. Laub, and E. Jonckheere, “Positive and negative solutions of dual Riccati equations by matrix sign function iteration,” *Systems and Control Letters*, vol. 13, pp. 109–116, 1989.

[83] P. Gahinet, “Explicit controller formulas for LMI-based H_∞ synthesis,” *Automatica*, vol. 32, pp. 1007–1014, 1996.

[84] K. Glover and J. C. Doyle, “State-space formulae for all stablizing controllers that satisfy an H_∞-norm bound: and relations to risk sensitivity,” *Systems & Control Letters*, vol. 11, pp. 167–172, 1988.

[85] J. Roberts, “Linear model reduction and solution of the algebraic Riccati equation by use of the sign function,” *International Journal of Control*, vol. 32, pp. 677–687, 1980.

[86] C. S. Kenney and A. J. Laub, “The matrix sign function,” *IEEE Trans. Autom. Control*, vol. 40, pp. 1330–1348, 1995.

[87] N. J. Higham, *Functions of Matrices: Theory and Computation.* SIAM, 2008.

[88] B. D. O. Anderson, “Second-order convergent algorithms for the steady state Riccati equation,” *International Journal of Control*, vol. 28, pp. 295–306, 1978.

[89] N. J. Higham, “Stable iterations for the matrix square root,” *Numerical Algorithms*, vol. 15, pp. 227–242, 1997.

[90] X.-X. Guo, W.-W. Lin, and S.-F. Xu, “A structure-preserving doubling algorithm for nonsymmetric algebraic riccati equation,” *Numerische Mathematik*, vol. 103, pp. 393–412, 2006.

[91] R. Smith, “Matrix equation $XA+BX=C$,” *SIAM Journal of Applied Mathematics*, vol. 16, p. No. 1, 1968.

[92] A. Beavers and E. Denman, “A computational method for eigenvalues and eigevectors of a matrix with real eigenvalues,” *Numer. Math.*, pp. 389–396, 1973.

[93] A. Varga, “Balancing free square-root algorithm for computing singular perturbation approximations,” *Proc. IEEE Conf. Decision and Control*, pp. 1062–1065, 1991.

[94] C. Scherer, *Theory of Robust Control (Course notes).* Delft Center for Systems and Control, Unpublished, 2001.

[95] S. Boyd, V. Balakrishnan, and P. Kabamba, “A bisection method for computing the H_∞ norm of a transfer matrix and related problems,” *Mathematical control of signals and systems*, vol. 2, pp. 207–219, 1989.

[96] P. Gahinet, “On the game Riccati equations arising in H_∞ control problems,” *SIAM Journal on Control and Optimization*, vol. 32, pp. 635–647, 1994.

[97] P. Gahinet and P. Apkarian, “A linear matrix inequality approach to H_∞ control,” *International Journal of Nonlinear and Robust Control*, 1994.

[98] R. Byers, C. He, and V. Mehrmann, “The matrix sign function method and the computation of invariant subspaces,” *SIAM Journal of Matrix Analysis and Applications*, vol. 18, pp. 615–632, 1997.

[99] N. J. Higham and P. A. Knight, “Matrix powers in finite precision arithmetic,” *SIAM Journal of Matrix Analysis and Applications*, vol. 16, pp. 343–358, 1995.

[100] N. J. Higham, “Perturbation theory and backward error for $AX - XB = C$,” *BIT Numerical Mathematics*, pp. 124–136, 1993.

[101] C. Kenney, A. J. Laub, and M. Wette, “Error bounds for Newton refinement of solutions to algebraic Riccati equations,” *Mathematics of Control, Signals, and Systems*, vol. 3, pp. 211–224, 1990.

[102] L. Keel and S. Bhattacharyya, “Robust, fragile or optimal,” *Proc. Amer. Control Conf.*, pp. 1307–1313, 1997.

[103] J. K. Rice and M. Verhaegen, “Distributed control: A sequentially semi-separable approach,” *Proc. IEEE Conf. Decision and Control*, 2008.

[104] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis.* Cambridge University Press, 1991.

[105] M. Dettori, “LMI techniques for control with application to a compact disc player mechanism,” *Phd Thesis*, 2001.

[106] J. K. Rice and M. Verhaegen, “A structured matrix approach to efficient calculation of lqg repetitive learning controllers in the lifted setting,” *International Journal of Control*, vol. 83, pp. 1265–1276, 2010.

[107] ——, “Distributed control: A sequentially semi-separable approach for heterogeneous linear systems,” *IEEE Trans. Autom. Control*, vol. 54, pp. 1270–1283, 2009.

[108] E. Kreyszig, *Introductory Functional Analysis with Applications.* John Wiley and Sons, 1978.

[109] J. H. Justice and J. L. Stevens, “Stability criterion for N-dimensional digital filters,” *IEEE Transactions on Automatic Control*, vol. June, pp. 284–286, 1973.

[110] S. Samar and C. Beck, “Model reduction of heterogeneous distributed systems,” *Proc. IEEE Conf. Decision and Control*, 2003.

[111] H. Sandberg and R. M. Murray, “Model reduction of interconnected linear systems using structured gramians,” *IFAC World Congress*, 2008.

[112] D. Doan, T. Keviczky, I. Necoara, M. Diehl, and B. D. Schutter, "A distributed version of han's method for dmpc using local communications only," *Journal of control Engineering and Applied Informatics*, vol. 11, pp. 6–15, 2009.

[113] A. Robinson, "A survey of optimal control of distributed-parameter systems," *Automatica*, vol. 7, pp. 371–388, 1971.

[114] N. R. S. Jr., P. Varaiya, M. Athans, and M. G. Safonov, "Survey of decentralized control methods for large scale systems," *IEEE Trans. Autom. Control*, vol. 23, pp. 108–129, 1978.

[115] P. Sharma and C. Beck, "Modelling and distributed control of mobile offshore bases," *Proc. Amer. Control Conf.*, pp. 5238–5243, 2004.

[116] R. D'Andrea, "Linear matrix inequalities, multidimensional system optimization, and control of spatially distributed systems: An example," *Proc. Amer. Control Conf.*, pp. 2713–2717, 1999.

[117] A.-J. van der Veen and P. Dewilde, "Modeling computational networks by time-varying systems," *Integration, the VLSI Journal*, 1993.

[118] J. K. Rice and M. Verhaegen, "Distributed computations and control in multi-agent systems," *ICARA*, 2009.

[119] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods.* Athena Scientific, 1997.

[120] J.-M. Contet, F. Gechter, P. Gruer, and A. Koukam, "Multiagent system model for vehicle platooning with merge and split capabilities," *In Int. Conf. on Autonomous Robots and Agents*, 2006.

[121] F. Bullo, J. Cortes, and S. Martinez, *Distributed Control of Robotic Networks.* Manuscript preprint. Electronically available at http://coordinationbook.info, 2008.

[122] E. Shaw and J. K. Hedrick, "String stability analysis for heterogeneous vehicle strings," in *Proc. Amer. Control Conf.*, 2007.

[123] B. Shu and B. Bamieh, "Robust H_2 control of vehicular strings," *Submitted to ASME Journal of Dynamics, Meas. and Control.*

[124] S. Sheikholeslam and C. Desoer, "Longitudinal control of a platoon of vehicles," *In Proc. Amer. Control Conf.*, pp. 291–296, 1990.

[125] M. R. Jovanović and B. Bamieh, "On the ill-posedness of certain vehicular platoon control problems," *IEEE Trans. Autom. Control*, vol. 50, p. 9, 2005.

[126] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations.* Wadsworth Publ. Co, 1989.

[127] L. Ljung, "Asymptotic behavior of the extended kalman filter as a parameter estimator for linear systems," *IEEE Transactions on Automatic Control*, vol. 24, pp. 36–50, 1979.

[128] T. McKelvey and A. Helmersson, "System identification using an over-parametrized model class - improving the optimization algorithm," *IEEE Conference on Decision and Control*, pp. 2984–2989, 1997.

[129] A. Turing, "The chemical basis of morphogenesis," *Philosophical Trans. of the Royal Society of London, Series B*, vol. 237, pp. 37–72, 1952.

[130] J. Murray, *Mathematical Biology: An Introduction.* Springer-Verlag, 2002.

[131] J. D. Murray, *Mathematical Biology: Spatial Models and Biomedical Applications.* Springer-Verlag, 2003.

[132] M. S. Yeung, J. Tegner, and J. J. Collins, “Reverse engineering gene networks using singular value decomposition and robust regression,” *Proc. of the Nat. Academy of Sci. of the USA*, vol. 99, pp. 6163–6168, 2002.

[133] J. K. Rice and M. Verhaegen, “Robust and distributed control of a smart blade,” *Wind Energy*, vol. 13, pp. 103–116, 2010.

[134] W. Leithead, S. de la Salle, and D. Reardon, “Role and objectives of control for wind turbines,” *IEE Proceedings-C*, vol. 138, pp. 135–148, 1991.

[135] D. Benasciutti and R. Tovo, “Spectral methods for lifetime prediction under wide-band stationary random processes,” *International Journal of Fatigue*, vol. 27, pp. 867–877, 2005.

[136] K. Hammerum, P. Brath, and N. K. Poulsen, “A fatigue approach to wind turbine control,” *The Science of Making Torque from Wind, Journal of Physics: Conference Series*, vol. 75, 2007.

[137] J. Manwell, J. McGowan, and A. Rogers, *Wind Energy Explained: Theory, Design and Application*. Wiley, 2002.

[138] E. Welfonder, R. Neifer, and M. Spanner, “Development and experimental identification of dynamic models for wind turbines,” *Control Engineering Practice*, vol. 5, pp. 63–73, 1997.

[139] K. A. Stol, “Disturbance tracking control and blade load mitigation for variable-speed wind turbines,” *Journal of Solar Energy Engineering*, vol. 125, pp. 396–401, 2003.

[140] B. Kallesœ, “Equations of motion for a rotor blade, including gravity, pitch action and rotor speed variations,” *Wind Energy*, vol. 10, pp. 209–230, 2007.

[141] I. Houtzager, “Adaptive learning control for wind turbines equipped with smart rotor,” *MSc Thesis.*, 2007.

[142] S. Melzer and B. Kuo, “Optimal regulation of systems described by a countably infinite number of objects,” *Automatica*, vol. 7, pp. 359–366, 1971.

[143] N. Motee and A. Jadbabaie, “Approximation methods and spatial interpolation in distributed control systems,” *Proc. Amer. Control Conf.*, pp. 860–865, 2009.

[144] E. W. Kamen and P. P. Khargonekar, “On the control of linear systems whose coefficients are functions of parameters,” *IEEE Trans. Autom. Control*, vol. 29, pp. 25–33, 1984.

[145] M. Fardad and M. Jovanović, “On the state-space design of optimal controllers for distributed systems with finite communication speed,” *Proc. IEEE Conf. Decision and Control*, 2008.

[146] G. A. de Castro and F. Paganini, “Convex synthesis of localized controllers for spatially invariant systems,” *Automatica*, vol. 38, pp. 445–456, 2002.

[147] R. Fraanje and M. Verhaegen, “Subspace model identification of 2-d invariant systems - a decoupling approach,” *Technical Report, available at http://www.dcsc.tudelft.nl/∼rfraanje/*, 2006.

[148] J. Rice, P. Massioni, T. Keviczky, and M. Verhaegen, *Efficient Modeling and Control of Large-Scale Systems*. Springer, 2010, ch. Distributed Control methods for structured large-scale systems.

[149] F. Borrelli and T. Keviczky, “Distributed lqr design for identical dynamically decoupled systems,” *IEEE Transactions on Automatic Control*, vol. 53, pp. 1901–1912, 2008.

[150] D. L. Laughlin, M. Morari, and R. D. Braatz, “Robust performance of cross-directional basis-weight control in paper machines,” *Automatica*, vol. 29, pp. 1395–1410, 1993.

[151] T. Kailath, S. Y. Kung, and M. Morf, “Displacement ranks of matrices and linear equations,” *Journal of Mathematical Analysis and Applications*, vol. 68, pp. 395–407, 1979.

[152] R. D’Andrea, C. Langbort, and R. S. Chandra, “A state space approach to control of interconnected systems,” *Mathematical Systems Theory in Biology, Communication, Computation and Finance*, 2003.

[153] E. Osborne, “On pre-conditioning of matrices,” *Journal of the ASsoc. for Comp. Mach.*, vol. 7, pp. 338–349, 1960.

[154] J. C. Doyle, J. E. Wall, and G. Stein, “Performance and robustness analysis for structured uncertainty,” *Proc. IEEE Conf. Decision and Control*, p. 629:636, 1982.

[155] C. Beck and J. Doyle, “Mixed μ upper bound computation,” *Proceedings of the 31st CDC*, pp. 3187–3192, 1992.

[156] G. J. Balas, J. C. Doyle, K. Glover, A. Packard, and R. Smith, *μ-Analysis and Synthesis Toolbox: User’s Guide*. The MathWorks, 1993.

[157] R. Chandra, C. Langbort, and R. D’Andrea, “Distributed control design for spatially interconnected systems with robustness to small communication delays,” *Mathematical Theory of Networks and Systems*, 2004.

[158] L. Chen, R. Wang, T. Zhou, and K. Aihara, “Noise-induced cooperative behavior in a multi-cell system,” *Bioinformatics*, vol. 21, pp. 2722–2729, 2005.

[159] C. W. Reynolds, “Flocks, herds, and schools: A distributed behavioral model,” *Computer Graphics*, vol. 21, pp. 25–34, 1987.

[160] S. Arimoto, S. Kawamura, and F. Miyasaki, “Bettering operation of robots by learning,” *Journal of Robotic Systems*, vol. 1, pp. 123–140, 1984.

[161] D. A. Bristow, M. Tharayil, and A. G. Alleyne, “A survey of iterative learning control,” *IEEE Control Systems Magazine*, vol. Vol. 26, no.3, pp. 96–114, 2006.

[162] M. Q. Phan and R. Longman, “A mathematical theory of learning control for linear discrete multivariable systems,” in *Proceedings of the AIAA/AAS Astrodynamics Specialist Conference, Mineeapolis, Minnesota,*, 1988.

[163] P. Goldsmith, “On the equivalence of causal lti iterative learning control and feedback control,” *Automatica*, vol. 38, pp. 703–708, 2002.

[164] M. Verwoerd, G. Meinsma, and T. de Vries, “On the use of noncausal lti operators in iterative learning control,” in *Proceedings of the 41st IEEE CDC*, 2002.

[165] K. J. Astrom, P.Hagander, and J. Sternby, “Zeros of sampled systems,” *Automatica*, vol. 20, pp. 31–38, 1984.

[166] M. Q. Phan, R. Longman, and K. L. Moore, “Unified formulation of linear iterative learning control,” in *Spaceflight mechanics 2000; Proceedings of the AAS/AIAA Space Flight Mechanics Meeting*, 2000.

[167] R. Tousain, E. van der Meché, and O. Bosgra, “Design strategy for itertive learning control based on optimal control,” *Proceedings of the 40th IEEE CDC*, pp. FrM10–3, 2001.

[168] M. Q. Phan and R. W. Longman, “Higher-order iterative learning control by pole placement and noise filtering,” *IFAC World Congress*, 2002.

[169] K. S. Lee, J. Lee, I. Chin, and J. Choi, “Control of wafer temperature uniformity in rapid thermal processing using an optimal iterative learning control technique,” *Industrial & Engineering Chemistry Research*, vol. 40, pp. 1661–1672, 2001.

[170] M. Cho, Y. Lee, S. Joo, and K. S. Lee, “Semi-empirical model-based multivariable learning control of an rtp system,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 18, p. No.3, 2005.

[171] J. H. Lee, S. Natarajan, and K. S. Lee, “A model-based predictive control approach to repetitive control of continuous processes with periodic operations,” *Journal of Process Control*, vol. 11, pp. 195–207, 2001.

[172] M. G. Wassink, O. Bosgra, D. Rixen, and S. Koekebakker, “Modeling of an inkjet printhead for iterative learning control using bilaterally coupled multiports,” *Proceedings of the ECC*, 2005.

[173] I. Rotariu, B. Dijkstra, and M. Steinbuch, “Standard and lifted approaches of iterative and learning control applied on a motion system,” *MTNS*, 2004.

[174] D. A. Bristow and A. G. Alleyne, “A high precision motion control system with application to microscale robotic deposition,” *IEEE Transactions on Control Systems Technology*, vol. 14, pp. 1008–1020, 2006.

[175] K. Mainali, S. Panda, J. Xu, and T. Senjyu, “Position tracking performance enhancement of linear ultrasonic motor using iterative learning control,” *IEEE Power Electronics Specialists Conference*, pp. 4844–4849, 2004.

[176] K. Galkowski, E. Rogers, S. Xu, J. Lam, and D. Owens, “LMIs-a fundamental tool in analysis and controller design for discrete linear repetitive processes,” *IEEE Transactions on Circuits and Systems–I: Fundamental Theory and Applications*, vol. 49, pp. 768–778, 2002.

[177] K. Galkowski, J. Lam, E. Rogers, S. Xu, B. Sulikowski, W. Paszke, and D. Owens, “Lmi based stability analysis and robust controller design for discrete linear repetitive processes,” *International Journal of Robust and Nonlinear Control*, vol. 13, pp. 1195–1211, 2003.

[178] W. Paszke, K. Galkowski, E. Rogers, and D. Owens, “H_∞ and guaranteed cost control of discrete linear repetitive processes,” *Linear Algebra and its Applications*, vol. 412, pp. 93–131, 2006.

[179] D. Leith and W. E. Leithead, “Gain-scheduled and nonlinear systems: dynamic analysis by velocity-based linearization families,” *International Journal of Control*, vol. 70, pp. 289–317, 1998.

[180] M. Verhaegen and X. Yu, “A class of subspace model identification algorithms to identify periodically and arbitrarily time-varying systems,” *Automatica*, vol. 31, pp. 201–216, 1995.

[181] F. Felici, J.-W. van Wingerden, and M. Verhaegen, “Subspace identification of MIMO LPV systems using a periodic scheduling sequence,” *Automatica*, vol. 43, pp. 1684–1697, 2007.

[182] I. Chin, S. J. Qin, K. S. Lee, and M. Cho, “A two-stage iterative learning control technique combined with real-time feedback for independent disturbance rejection,” *Automatica*, vol. 40, pp. 1913–1922, 2004.

[183] R. W. Longman, “Iterative learning control and repetitive control for engineering practice,” *International Journal of Control*, vol. 73, pp. 930–954, 2000.

[184] Y. Chen and K. Moore, “Harnessing the non-repetitiveness in iterative learning control,” *Proceedings of the 41st IEEE CDC*, pp. 3350–3355, 2002.

[185] K. J. Astrom and B. Wittenmark, *Computer-Controlled Systems.* Prentice-Hall, 1984.

[186] R. Tousain and D. van Casteren, "Iterative learning control in a mass product: Light on demand in dlp projection systems," in *Proceedings of the 2007 ACC*, 2007.

[187] D. Leith and W. Leithead, "Appropriate realization of gain-scheduled controllers with application to wind turbine regulation," *International Journal of Control*, vol. 65, pp. 223–248, 1996.

[188] P. Hingwe, H.-S. Tan, A. Packard, and M. Tomizuka, "Linear parameter varying controller for automated lane guidance: experimental study on tractor-trailers," *IEEE Transactions on Control Systems Technology*, vol. 10, pp. 793–806, 2002.

[189] M. Kothare, B. Mettler, M. Morari, P. Bendotti, and C. Falinower, "Linear parameter varying model predictive control for steam generator level control," *Computers and Chemical Engineering*, vol. 21(Supplement), pp. S861–866, 1997.

[190] K. Takahashi and S. Massaquoi, "Neuroengineering model of human limb control-gain scheduled feedback control approach," *In Proceedings of the CDC*, 2007.

[191] W. Qin and Q. Wang, "An LPV approximation for admission control of an internet web server: identification and control," *Control Engineering Practice*, vol. 15, pp. 1457–1467, 2007.

[192] S. C. Baslamisli, I. Polat, and I. E. Kose, "Gain scheduled active steering control based on a parametric bicycle model," *in Proceedings of the 2007 IEEE Intelligent Vehicles Symposium.*

[193] A. Packard, "Gain scheduling via linear fractional transformations," *Systems and Control Letters*, vol. 22, pp. 79–92, 1994.

[194] P. Apkarian and P. Gahinet, "A convex characterization of gain-scheduled H_∞ controllers," *IEEE Trans. Autom. Control*, vol. 40, pp. 853–864, 1995.

[195] P. Apkarian and R. J. Adams, "Advanced gain-scheduling techniques for uncertain systems," *IEEE Transactions on control systems technology*, vol. 6, pp. 21–32, 1998.

[196] C. Scherer, "LPV control and full block multipliers," *Automatica*, vol. 37, pp. 361–375, 2001.

[197] Y. Fujisaki, F. Dabbene, and R. Tempo, "Probabilistic design of LPV control systems," *Automatica*, vol. 39, pp. 1323–1337, 2003.

[198] F. Wu and K. Dong, "Gain-scheduling control of LFT systems using parameter-depedent Lyapunov functions," *Automatica*, vol. 42, pp. 39–50, 2006.

[199] C. Scherer and S. Weiland, *Linear Matrix Inequalities in Control.* unpublished, 2005.

[200] W. J. Rugh and J. S. Shamma, "Research on gain scheduling," *Automatica*, vol. 36, pp. 1401–1425, 2000.

[201] E. Sontag, "An introduction to the stabilization problem for parametrized families of linear systems," *Contemporary Mathematics*, vol. 47, pp. 369–400, 1985.

[202] P. Gahinet, P. Apkarian, and M. Chilali, "Affine parameter-dependent Lyapunov functions and real parametric uncertainty," *IEEE Transactions on Automatic Control*, vol. 41, pp. 436–442, 1996.

[203] P. Apkarian and H. Tuan, "Parametrized LMIs in control theory," *in Proceedings of the 1998 CDC.*

[204] H. D. Tuan and P. Apkarian, "Monotonic relaxations for robust control: New characterizations," *IEEE Transactions on Automatic Control*, vol. 47, pp. 378–384, 2002.

[205] C. Scherer, “LMI relaxations in robust control,” *European Journal of Control*, vol. 12, pp. 3–29, 2006.

[206] F. Wu, X. H. Yang, A. Packard, and G. Becker, “Induced l_2 norm control for LPV system with bounded parameter variation rates,” *Proceedings of the 1996 ACC*.

[207] R. C. Oliveira, M. C. de Oliveira, and P. L. Peres, “Convergent LMI relaxations for robust analysis of uncertain linear systems using lifted polynomial parameter-dependent Lyapunov functions,” *Systems & Control Letters*, vol. 57, pp. 680–689, 2008.

[208] P. A. Bliman, “An existence result for polynomial solutions of parameter-dependent LMIs,” *Systems & Control Letters*, vol. 51, pp. 165–169, 2004.

[209] C. Scherer and C. Hol, “Matrix sum-of-squares relaxations for robust semi-definite programs,” *Mathematical Programming Series B*, vol. 107, pp. 189–211, 2006.

[210] N. Higham, “Computing real square roots of a real matrix,” *Linear Algebra Applications*, vol. 88, pp. 405–430, 1987.

[211] D. F. Delchamps, “Analytic stabilization and the algebraic Riccati equation,” *Proc. IEEE Conf. Decision and Control*, pp. 1396–1401, 1983.

[212] J.-W. van Wingerden, “Control of wind turbines with ‘smart’ rotors: Proof of concept & lpv identification,” *PhD Thesis, Delft University of Technology*, 2008.

[213] B. W. Gaude, “Solving nonlinear aeuronautical problems using the carleman linearization method,” *SAND report*, 2001.

[214] P. Dewilde, K. Diepold, and W. Bamberger, “Optic flow computation and time-varying system theory,” *Symposium on Mathematical Theory of Networks and Systems*, 2004.

[215] P. Dewilde and S. Chandrasekaran, “A hierarchical semi-separable moore-penrose equation solver,” *Wavelets, Multiscale Systems and Hypercomplex Analysis*, pp. 69–85, 2006.

[216] R. Wallin, C.-Y. Kaob, and A. Hansson, “A cutting plane method for solving KYP-SDPs,” *Automatica*, vol. 44, pp. 418–429, 2008.

Summary

Efficient Algorithms for Distributed Control: A Structured Matrix Approach

Justin K. Rice

Distributed systems are all around us, and they are fascinating, and have an enormous potential to improve our lives, if their complexity can be properly harnessed. All scientists and engineers are aware of the great potential of this subject, since we witness fantastic distributed control systems every day, in the flocks of birds in the sky and fish in the sea. However, the collective behavior of millions of dynamically-coupled heterogeneous subsystems is hard to analyze and control for computational reasons.

Our approach to the problems of analysis and control of distributed systems is to exploit the matrix structure of array-interconnected systems in fast iterative algorithms. Since these algorithms preserve the original matrix structure of the system, the resulting centrally optimal controller realizations have the same structure, which can conveniently be 'redistributed' into a set of subcontrollers linked in the same interconnection topology as the original system.

For P interconnected subsystems, traditional analysis and control synthesis methods are $\mathcal{O}(P^3)$ computational complexity, but for N heterogeneous subsystems on a line, the above method is only $\mathcal{O}(N)$ complexity. If the system is homogeneous with only heterogeneous boundary conditions, the complexity can be reduced to $\mathcal{O}(1)$, independent of the size of the homogeneous section. These results also extend to multiple spatial dimensions: for heterogeneous or homogeneous subsystems on an $N \times M$ 2-D cartesian grid, the complexity is reduced to $\mathcal{O}(MN)$ or $\mathcal{O}(1)$ complexity respectively, as compared to $\mathcal{O}(M^3N^3)$ complexity of traditional techniques, an impressive improvement for very large systems $N, M > 1000$. Furthermore, due to the special form of the structured matrix arithmetic, the computations can actually be performed in a distributed way, on a distributed processor and memory system, with only linear complexity communication and memory requirements.

Using these efficient structured techniques, one can perform stability and H_2 and H_∞ analysis to an arbitrary degree of accuracy, and sub-optimally upper-bound the structured singular value for robustness analysis. For synthesis, controllers with H_2 and H_∞ performance arbitrarily close to optimal are possible, and D-K iterations can be performed for robust design. Structure preserving model order reduction, and even system identification are also possible.

It is also possible to apply this approach to analysis and synthesis of controllers for linear parameter varying(LPV) systems, and of repetitive controllers for trials with many time steps, T,in only $\mathcal{O}(T)$ complexity which would otherwise be $\mathcal{O}(T^3)$.

Samenvatting

Efficinte algoritmen voor gedistribueerde besturing: Een gestructureerde aanpak van matrix

Justin K. Rice

Gedistribueerde systemen zijn overal om ons heen, en ze zijn fascinerend, en hebben een enorme potentieel om onze levens te verbeteren, als hun complexiteit naar behoren kan worden benut. Alle wetenschappers en ingenieurs zijn zich bewust van de grote potentieel van dit onderwerp, omdat we getuige van een fantastisch verdeeld controlesystemen elke dag, in de kudden van vogels in de lucht en vissen in de zee. Echter, het collectieve gedrag van miljoenen dynamisch gekoppelde subsystemen is moeilijk te analyseren en controle voor de computationele redenen.

Onze aanpak van de problemen van de analyse en controle van gedistribueerde systemen te benutten is de matrix-structuur van de array met elkaar verbonden systemen in de snelle iteratieve algoritmes. Aangezien deze algoritmen het behoud van de originele matrixstructuur van het systeem, de resulterende centraal optimale regelaar realisaties hebben dezelfde structuur, , die gemakkelijk kan worden ‘herverdeeld’ in een reeks van subcontrollers elkaar verbonden zijn in een en dezelfde interconnectie als de topologie oorspronkelijke systeem.

Voor P onderling verbonden subsystemen, traditionele analyse en controle synthese methoden zijn $\mathcal{O}(P^3)$ computationele complexiteit, maar voor N heterogene subsystemen op een lijn, de hierboven beschreven methode wordt slechts $\mathcal{O}(N)$ complexiteit. Als het systeem is homogeen Alleen heterogene randvoorwaarden, kan de complexiteit worden verlaagd tot $\mathcal{O}(1)$, onafhankelijk van de grootte van de homogene sectie. Deze resultaten worden uitgebreid tot meerdere ruimtelijke Afmetingen: voor heterogene of homogene subsystemen op een $N \times M$ 2-D cartesische rooster, wordt de complexiteit verminderd tot $\mathcal{O}(MN)$ of $\mathcal{O}(1)$ complexiteit respectievelijk als vergeleken met $\mathcal{O}(M^3N^3)$ complexiteit van de traditionele technieken, een indrukwekkende verbetering voor zeer grote systemen $N, M > 1000$. Bovendien, als gevolg van de bijzondere vorm van de gestructureerde matrix rekenen, kan de berekeningen daadwerkelijk worden uitgevoerd in een gedistribueerde manier, op een gedistribueerd systeem processor en het geheugen, met Enkel lineaire complexiteit communicatie en geheugenfuncties.

Met behulp van deze technieken efficint gestructureerd, kan men uitvoeren stabiliteit en H_2 en H_∞ analyse naar een willekeurige mate van nauwkeurigheid, en sub-optimaal bovengrensconcentraties de gestructureerde enkelvoud waarde voor robuustheid analyse. Voor de synthese, controllers met H_2 end H_∞ prestaties willekeurig dicht bij optimaal zijn mogelijk is, en DK iteraties kan worden uitgevoerd voor robuuste ontwerp. Structuur-model om het behoud van de korting, en zelfs systeem identificatie zijn ook mogelijk.

Het is ook mogelijk om deze aanpak toe te passen voor analyse en synthese van controllers voor lineaire parameter varirende (LPV) systemen, en van repetitieve controllers voor proeven met veel tijd stappen, T, slechts in $\mathcal{O}(T)$ complexiteit die anders zou zijn $\mathcal{O}(T^3)$.

Curriculum Vitae

Justin Rice was born on February 1, 1983, in Independence, Missouri, and raised in the forests to the east of Lake Lotawana. After graduating from the world-renowned Missouri public school system, he obtained his B.S. in 2005 followed by his Master of Engineering degree in 2006 under the supervision of Raffaello D'Andrea, both in Mechanical & Aerospace Engineering at Cornell University.

In September 2006 he moved to the Netherlands and started his PhD project at the Delft Center for Systems and Control, funded by the Dutch technology foundation STW, under the supervision of Michel Verhaegen. The main focus for the project was the investigation of iterative algorithms for analysis and control of large-scale systems.

In September 2010 he plans to relocate to New York, where he will work as a post-doctorate researcher in neuroscience at CCNY, under the supervision of Lucas Parra.