LUND UNIVERSITY

Gradient Methods for Large-Scale and Distributed Linear Quadratic Control

Mårtensson, Karl

2012

Gradient Methods for Large-Scale and Distributed Linear Quadratic Control

Gradient Methods for Large-Scale and Distributed Linear Quadratic Control

Karl Mårtensson

Department of Automatic Control
Lund University
Lund, April 2012

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

ISSN 0280–5316
ISRN LUTFD2/TFRT--1091--SE

Printed in Sweden by Media-Tryck.
Lund 2012

Abstract

This thesis considers methods for synthesis of linear quadratic controllers for large-scale, interconnected systems. Conventional methods that solve the linear quadratic control problem are only applicable to systems with moderate size, due to the rapid increase in both computational time and memory requirements as the system size increases. The methods presented in this thesis show a much slower increase in these requirements when faced with system matrices with a sparse structure. Hence, they are useful for control design for systems of large order, since they usually have sparse systems matrices. An equally important feature of the methods is that the controllers are restricted to have a distributed nature, meaning that they respect a potential interconnection structure of the system.

The controllers considered in the thesis have the same structure as the centralized LQG solution, that is, they are consisting of a state predictor and feedback from the estimated states. Strategies for determining the feedback matrix and predictor matrix separately, are suggested. The strategies use gradient directions of the cost function to iteratively approach a locally optimal solution in either problem. A scheme to determine bounds on the degree of suboptimality of the partial solution in every iteration, is presented. It is also shown that these bounds can be combined to give a bound on the degree of suboptimality of the full output feedback controller. Another method that treats the synthesis of the feedback matrix and predictor matrix simultaneously is also presented.

The functionality of the developed methods is illustrated by an application, where the methods are used to compute controllers for a large deformable mirror, found in a telescope to compensate for atmospheric disturbances. The model of the mirror is obtained by discretizing a partial differential equation. This gives a linear, sparse representation of the mirror with a very large state space, which is suitable for the methods presented in the thesis. The performance of the controllers is evaluated using performance measures from the adaptive optics community.

Acknowledgments

In my years at the Department of Automatic Control I have met numerous people who have helped me on my path towards this thesis. I would like to take this opportunity to thank as many as possible.

First of all, as my supervisor, Anders Rantzer deserves a lot of my gratitude. It has been an amazing experience to work with such a respected researcher. I have learnt a lot by just sitting back and observe Anders' thought process as he presents a solution on the blackboard during the many discussions that we have had. At first, Anders helped me to find my direction in the research. As time progressed, his encouragement and his valuable comments helped me to believe in myself and to be able to do more independent work. But not only did he recognize my academic accomplishments, also other achievements did not pass him by, for example me completing the Swedish classic circuit as the first person at the department.

I would like to thank Per Hagander, my co-supervisor, who showed and taught me about many of the intricate problems in the field of control when I was a teacher assistant in Control Theory. In later years we have had valuable talks about my research.

Bo Bernhardsson deserves my thanks for taking his time to read and correct a number of proofs essential to my work. With his expertise in mind, I would always feel confident that there was no mistakes left after he had examined them.

The person who opened my eyes to the department is Andreas Wernrud and for that I thank him. He told what a great place the department was to work at and convinced me, even though it did not take much, to apply for a position. After I started, it was with him that I wrote my first paper. Without Andreas I would never be where I am today.

Since I started working I have always shared office with Pontus Giselsson. This has been a truly fortunate choice; within a few meters I have someone with a sharp mind who always shows interest in the problems

that I might encounter. Although, I must admit that our discussions sometime leads us to try to solve completely off-topic puzzles.

I would like to thank Vladimeros Vladimerou for the trips we made together to Italy for the EU project meetings. It was good to have someone with his experience to understand the tasks of the project.

I want to thank Rikard Heimsten, who has provided invaluable insights in the field of adaptive optics. His efforts made it possible to produce results of a real application to the theories in the thesis.

In the process of writing this thesis, a number of colleagues have taken their time to read it, giving me comments and suggestions to help me improve it. Per-Ola Larsson, Pontus Giselsson, Erik Johannesson and Leif Andersson, for this I am very grateful.

A special thanks goes to the administrative and technical staff for making the department running as smoothly as it does and taking care of all the problems that I am clueless to solve.

I have a number of fellow Ph.D. student colleagues that I would like to thank. My most sincere thanks go to: Per-Ola Larsson for being convinced to joining me all over Sweden to complete different races in our pursuit towards the Swedish classic circuit. Erik Johannesson who showed me that drinking beer is more than just to drink a beer. Anders Widd for keeping me company when sweating in the Napa Valley heat outside of the vineyard. Daria Madjidian and Anna Lindholm for the amazingly hilarious night out i Milan.

Finally, I would like to thank my family and my friends for always believing in me and never giving up on me. I am truly grateful for all the support you have given me and all the joys you have showed me exist outside the office.

Karl

Financial Support

Grateful acknowledgements go to European Community's Seventh Framework Programme under the grant agreement number 224428, acronym CHAT; and the Swedish Research Council through the Linnaeus Center, LCCC.

Contents

Preface

Contributions of the Thesis

The thesis consists of one introductory chapter and four papers. In this section the contents of the introductory chapter and the contributions of the papers will be described.

Chapter 1 – Background

In this chapter different topics related to the work of this thesis will be given. Since not all topics are directly related, the presentations will be free-standing. In a section in the end of the chapter a discussion of how they connect to the work of the thesis will be given.

Paper I

Mårtensson, K. and A. Rantzer (2011) "Synthesis of structured controllers for large-scale systems." Submitted to *IEEE Transactions on Automatic Control*.

This paper presents methods to calculate structured state feedback controllers for large-scale systems. The methods use trajectories of the closed loop system to determine descent directions in which the feedback matrix is changed to improve the performance. These trajectories can be obtained either by simulation of a model or measurements from of the actual system, thus producing either offline or online design schemes. The schemes are shown to be have linear scalability properties when the system satisfies certain sparsity assumptions.

The idea behind the methods presented in the paper was developed by K. Mårtensson and A. Rantzer. The analysis of the scalability of the methods was performed by K. Mårtensson.

Paper II

Mårtensson, K. and A. Rantzer (2012) "Synthesis of Structured Output Feedback Controllers for Large-Scale Systems." Submitted to *Automatica*.

This paper extends the ideas of Paper I to synthesis methods for structured output feedback controllers for large-scale systems. The objective of the methods is to obtain a suboptimal feedback matrix and a suboptimal state predictor matrix, respectively. An analysis of how to combine the degree of suboptimality of each of these matrices into a degree of suboptimality for the complete controller is performed.

The extensions of the methods in Paper I were made by K. Mårtensson. The analysis of the combination of the degrees of suboptimality was also performed by K. Mårtensson. A. Rantzer contributed with support and insights during the work.

Paper III

Mårtensson, K. and A. Rantzer (2012) "A Scalable Method for Continuous-Time Distributed Control Synthesis." To appear in *Proceedings of the 2012 American Control Conference*, Montréal, Canada.

This paper presents a method to determine structured state feedback controllers for continuous-time systems. Similarly to Paper I, the method works by simulating the closed loop system and using the trajectories to determine descent directions. Also, it has been shown the degree of suboptimality can be determined through the obtained trajectories.

The work of the paper is performed by K. Mårtensson. Useful support and comments were provided by A. Rantzer.

Paper IV

Mårtensson, K. and R. Heimsten (2012) "Synthesis of Structured State Feedback Controllers for a Large Deformable Mirror." Submitted to *IEEE Transactions on Control Systems Technology*.

In this paper, the methods developed in Paper I are used to find state feedback controllers for a large deformable mirror. The model of the mirror is described by large, sparse system matrices and is thus suitable for the theory in Paper I. The performance of the obtained controllers are evaluated by measures found in the adaptive optics literature.

The model used for the simulations was provided by R. Heimsten. The implementation of the methods for finding the state feedback matrices for large-scale systems was done by K. Mårtensson. The simulations and evaluations of the controllers were performed by K. Mårtensson.

Additional Publications

Other publications by the author of the thesis are listed below.

Mårtensson, K. and A. Wernrud (2008): "Dynamic Model Predictive Control." In *Proceedings of the 17th IFAC World Congress*, Seoul, Korea.

Mårtensson, K. and A. Rantzer (2009): "Gradient methods for iterative distributed control." In *Proceedings of the 48th IEEE Conference on Decision and Control*, Shanghai, China.

Mårtensson, K. and A. Rantzer (2010): "Sub-optimality bound on a gradient method for iterative distributed control synthesis." In *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems*. Budapest, Hungary.

Mårtensson, K. and V. Vladimerou (2011): "Distributed resource management using iterative gradient update synthesis." In *Proceedings of the 2011 American Control Conference*, San Francisco, CA, USA.

Madjidian D. and K. Mårtensson and A. Rantzer (2011): "A Distributed Power Coordination Scheme for Fatigue Load Reduction in Wind Farms." In *Proceedings of the 2011 American Control Conference*, San Francisco, CA, USA.

1

Background

In this chapter concepts important to the thesis are described. Since the objective of the thesis is to find methods for large-scale, structured linear quadratic Gaussian (LQG) control, the chapter begins with a brief overview of the theory of ordinary LQG. Common numerical methods for solving this problem are discussed. In this discussion it is indicated why there is a problem when using these methods for large-scale systems. Some approaches to handle this issue for large-scale systems are also given in this chapter. As the thesis considers structured controllers, a review of research in distributed control is given. Since the methods of the thesis build upon the gradient descent direction method, a section is devoted to treat this topic. Finally, similar ideas used in optimal control for general non-linear systems are described.

1.1 Linear Quadratic Gaussian Control

When designing a controller we would usually like to evaluate its performance. To quantify the performance, a function, called the cost function or the performance measure, needs to be selected. This function maps the closed loop system to a number that describes the performance of the controller. Usually, the lower the number is, the better the performance. In optimal control the objective is to find a controller with the lowest cost for a given cost function, possibly with additional constraints that have to be satisfied. In linear quadratic Gaussian (LQG) control we consider linear systems and quadratic cost functions. The theory is well-studied and it is known that under certain assumptions, the optimal controller is linear and separates into two parts, one state feedback and one state predictor. The theory can be found in numerous books, among others [Zhou *et al.*, 1996, Åström, 2006, Boyd and Barratt, 1991]. In this section a brief overview of the theory of LQG will be given, starting by describing the

linear quadratic regulator problem and then the problem of state reconstruction.

Linear Quadratic Regulator Problem

In the majority of the papers of this thesis we consider linear, time-invariant systems in discrete time. For this reason we will consider such systems in the following presentation of the theory of LQG. In this part we will also assume that the system has some given initial state,

$$x(k+1) = Ax(k) + Bu(k), \quad x(0) = x_0. \tag{1.1}$$

Here, $x(k) \in \mathbb{R}^n$ is the state vector of the system with the given, but arbitrary, initial state x_0, and $u(k) \in \mathbb{R}^m$ is the control signal used to regulate the system. We will assume that the pair (A, B) is stabilizable. The objective is to find u that drives the state from x_0 towards zero in an optimal way. In order to quantify what we mean by optimal, a cost or performance function needs to be specified. In the discrete time case the quadratic cost is given by

$$J(x,u) = \sum_{k=0}^{\infty} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \underbrace{\begin{bmatrix} Q_x & Q_{xu} \\ Q_{xu}^T & Q_u \end{bmatrix}}_{Q} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}. \tag{1.2}$$

In this discussion the control weight Q_u will be required to be positive definite and the total weight Q to be positive semidefinite. It is possible to have more general assumptions on the weights, but these assumptions are not essential for the presentation. We define a control strategy by the causal function μ that takes a trajectory x and determines the control u. The optimal cost is the infimum

$$\inf_{\mu} J(x, \mu(x)),$$

where x satisfies (1.1) with $u = \mu(x)$. The optimal control strategy, denoted μ_{opt}, is the μ for which the infimum is attained. This setup is usually referred to as the discrete time linear quadratic regulator (LQR) problem.

The solution to the discrete time LQR problem is well-known and relies on the solution of the discrete algebraic Riccati equation

$$X = A^T X A + Q_x - (Q_{xu} + A^T X B)(Q_u + B^T X B)^{-1}(Q_{xu}^T + B^T X A). \tag{1.3}$$

A few methods to solve the discrete algebraic Riccati equation will be discussed in Section 1.2. Let X be the positive definite solution to (1.3).

Then the optimal control law in the discrete time case is linear and defined by

$$u_{\text{opt}}(k) = -\underbrace{(Q_u + B^T X B)^{-1}(Q_{xu}^T + B^T X A)}_{K_{\text{opt}}} x(k). \tag{1.4}$$

For any stabilizing linear control law $u(k) = -Kx(k)$, the value of the cost can be calculated by determining the solution to the following discrete time Lyapunov equation

$$S = (A - BK)^T S(A - BK) + \underbrace{Q_x - Q_{xu}K - K^T Q_{xu}^T + K^T Q_u K}_{Q_K}. \tag{1.5}$$

It is easily realized that the solution to (1.5) can be expressed as an infinite sum, that is,

$$S = \sum_{k=0}^{\infty} ((A - BK)^T)^k Q_K (A - BK)^k. \tag{1.6}$$

Using (1.6), an expression for the value of the cost function with the control law μ, defined by $u(k) = -Kx(k)$, can be determined as

$$\begin{aligned} J(x, \mu(x)) &= \sum_{k=0}^{\infty} \begin{bmatrix} x(k) \\ -Kx(k) \end{bmatrix}^T Q \begin{bmatrix} x(k) \\ -Kx(k) \end{bmatrix} = \sum_{k=0}^{\infty} x(k)^T Q_K x(k) \\ &= \sum_{k=0}^{\infty} x_0^T ((A - BK)^T)^k Q_K (A - BK)^k x_0 = x_0^T S x_0. \end{aligned}$$

When the optimal control law is used, it turns out that the solution of the discrete algebraic Riccati equation, X, and S coincide. Hence, the value of the cost function is easily determined by

$$J(x, \mu_{\text{opt}}(x)) = x_0^T X x_0. \tag{1.7}$$

The continuous time LQR problem is solved in a similar manner and can for example be found in [Zhou *et al.*, 1996, Åström, 2006, Boyd and Barratt, 1991].

State Reconstruction

The purpose of state reconstruction is to determine estimates of all the states of a system using measurements of the output. The system considered here is

$$\begin{aligned} x(k+1) &= Ax(k) + w(k), \quad x(0) = x_0, \\ y(k) &= Cx(k) + e(k), \end{aligned} \tag{1.8}$$

where $x(k) \in \mathbb{R}^n$ is the state that is to be estimated and $y(k) \in \mathbb{R}^p$ the observation that is used for the estimation. The pair (A, C) is assumed to be detectable. The noise corrupting the states and measurements, $w(k)$ and $e(k)$, respectively, are assumed to come from a Gaussian process with zero mean and variance

$$\mathbf{E} \begin{bmatrix} w(k) \\ e(k) \end{bmatrix} \begin{bmatrix} w(k) \\ e(k) \end{bmatrix}^T = \begin{bmatrix} R_w & R_{we} \\ R_{we}^T & R_e \end{bmatrix} = R,$$

where the variance of the measurement noise R_e is assumed to be positive definite, and the total variance R to be positive semidefinite. As in the discussion for the LQR problem, there are more general assumptions for the variances. The objective of the state reconstruction is to use the measurement trajectory $\mathscr{Y}_k = \{y_k, y_{k-1}, \ldots\}$ to give an estimate $\hat{x}(k+1)$ of the state $x(k+1)$. Define the state estimation policy $\varphi(\mathscr{Y}_k; k) = \hat{x}(k+1)$ and the estimation error of this policy as $\tilde{x}(k) = x(k) - \hat{x}(k)$. By specifying any constant, positive semidefinite weight W, we define the cost of a state estimation policy as

$$J(\varphi) = \lim_{k \to \infty} \mathbf{E}\, \tilde{x}(k)^T W \tilde{x}(k). \tag{1.9}$$

Now, we would like to find the optimal state estimation policy, that is, the policy that minimizes (1.9). The solution again relies on the solution of a discrete algebraic Riccati equation

$$Y = AYA^T + R_w - (R_{we} + AYC^T)(R_e + CYC^T)^{-1}(R_{we}^T + CYA^T). \tag{1.10}$$

With the solution Y to (1.10) the optimal policy is given by the linear dynamical system

$$\hat{x}(k+1) = (A - L_{\text{opt}}C)\hat{x}(k) + L_{\text{opt}}y(k), \quad \hat{x}(0) = 0, \tag{1.11}$$

where

$$L_{\text{opt}} = (R_{we} + AYC^T)(R_e + CYC^T)^{-1}. \tag{1.12}$$

Using this estimation policy, an expression of the resulting estimation error is given by

$$\tilde{x}(k) = \sum_{i=0}^{k} (A - L_{\text{opt}}C)^{i-k}(w(i) - L_{\text{opt}}e(i)) + (A - L_{\text{opt}}C)^k x_0. \tag{1.13}$$

Similarly to the LQR problem, for any prediction matrix L, for which $A - LC$ is stable, used instead of L_{opt} in (1.11), the cost (1.9) can be determined with the use of the solution to the Lyapunov equation

$$P = (A - LC)P(A - LC)^T + \underbrace{R_w - R_{we}L^T - LR_{we}^T + LR_eL^T}_{R_L}. \tag{1.14}$$

In similarity to the solution of (1.5), the solution of (1.14) can be expressed as an infinite sum

$$P = \sum_{k=0}^{\infty} (A - LC)^k R_L ((A - LC)^T)^k. \tag{1.15}$$

Denoting $A_L = A - LC$, an expression for the value of the cost function can now be determined as

$$\begin{aligned} J(\varphi_L) &= \lim_{k\to\infty} \mathbf{E}\, \mathrm{tr}\left(\tilde{x}(k)\tilde{x}(k)^T W\right) \\ &= \lim_{k\to\infty} \mathrm{tr}\left(\left(\sum_{i=0}^{k} A_L^i R_L (A_L^T)^i + A_L^k x_0 x_0^T (A_L^T)^k\right) W\right) \\ &= \mathrm{tr}\,(PW)\,. \end{aligned}$$

When the optimal control policy is used, Y and P coincide giving the optimal cost

$$J(\varphi_{\mathrm{opt}}) = \mathrm{tr}\,(YW)\,. \tag{1.16}$$

Linear Quadratic Gaussian Control

In linear quadratic Gaussian control we are concerned with finding the optimal output feedback controller for the system

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + w(k), \\ y(k) &= Cx(k) + e(k). \end{aligned} \tag{1.17}$$

Now, the objective is to find the optimal control policy $u(k) = \mu(\mathscr{Y}_{k-1}; k)$ that minimizes the cost

$$J(\mu) = \lim_{k\to\infty} \mathbf{E} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q_x & Q_{xu} \\ Q_{xu}^T & Q_u \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}, \tag{1.18}$$

where x satisfies (1.17) with the control policy μ. The well-known *separation principle* tells us that the optimal LQG controller is separated in two parts; the optimal state estimation policy and the optimal feedback law applied to the state estimates. Hence, the optimal output feedback controller is given by the linear system

$$\begin{aligned} \hat{x}(k+1) &= (A - LC)\hat{x}(k) + L_{\mathrm{opt}} y(k), \\ u(k) &= -K_{\mathrm{opt}} \hat{x}(k), \end{aligned} \tag{1.19}$$

where K_{opt} and L_{opt} are the previously defined matrices. The optimal value of the cost function also separates into two terms, which relate

to the optimal LQR problem and the optimal state estimation problem, respectively. The optimal cost is given by

$$\begin{aligned} J(\mu) &= \operatorname{tr}\left(X R_w\right) + \operatorname{tr}\left(Y K_{\text{opt}}^T (Q_u + B^T X B) K_{\text{opt}}\right) \\ &= \operatorname{tr}\left(X L_{\text{opt}} (R_e + C Y C^T) L_{\text{opt}}^T\right) + \operatorname{tr}\left(Y Q_x\right). \end{aligned} \tag{1.20}$$

Any suboptimal controller can be parametrized using the optimal controller, see [Zhou *et al.*, 1996]. With this parametrization the cost separates into two terms, the optimal cost and an additional cost term. The added cost of using any stabilizing feedback matrix K and any stabilizing predictor matrix L instead of the optimal ones in (1.19), is easily determined by introducing

$$\begin{aligned} A_{\text{ext}} &= \begin{bmatrix} A - BK & BK \\ 0 & A - LC \end{bmatrix}, \\ B_{\text{ext}} &= \begin{bmatrix} L_{\text{opt}} \\ (L_{\text{opt}} - L) \end{bmatrix} (R_e + C Y C^T)^{1/2}, \\ C_{\text{ext}} &= (Q_u + B^T X B)^{1/2} \begin{bmatrix} (K_{\text{opt}} - K) & K \end{bmatrix}. \end{aligned} \tag{1.21}$$

The total cost of the control policy that arises when using K and L, is

$$J(\mu) = J(\mu_{\text{opt}}) + \operatorname{tr}\left(S_{\text{ext}} B_{\text{ext}} B_{\text{ext}}^T\right), \tag{1.22}$$

where S_{ext} is the solution to the Lyapunov equation

$$S_{\text{ext}} = A_{\text{ext}}^T S_{\text{ext}} A_{\text{ext}} + C_{\text{ext}}^T C_{\text{ext}}. \tag{1.23}$$

1.2 Numerical Methods for the Algebraic Riccati Equation

In the previous section we saw the importance of computing the solution to the algebraic Riccati equation. There has been considerable research efforts devoted to numerical methods for calculating the solution of the algebraic Riccati equation. The methods are usually alterations to already existing ones, improved to increase the computational performance. However, a recurring property of the computational time requirements of the methods is that it scales as $O(n^3)$ for general systems of size n. Also, since the solution X is a full matrix, the memory or workspace requirement scales at least as $O(n^2)$. These two requirements make such methods inapplicable to large-scale systems. In order to find methods for these systems, special properties, such as rank or sparsity structure, must be exploited.

In this section we will give outlines of three of the main methods used to solve the algebraic Riccati equation: the Newton–Kleinman method, the Schur method and the matrix sign function method. It should be noted that, most commonly, a method for solving the continuous algebraic Riccati equation is developed. This methodology is then extended to obtain a method that solves the discrete algebraic Riccati equation. We will present the methods in their discrete time format. In the section a discussion of different approaches to deal with large-scale algebraic Riccati equations will also be given. A thorough review of the topics can be found in [Bini *et al.*, 2011].

Newton–Kleinman Method

The Newton–Kleinman method is an iterative method where a guess of the solution to the algebraic Riccati equation is brought closer to the true solution in each iterate. The method works by applying Newton's method for solving (1.3). It was first suggested for the continuous case in [Kleinman, 1968]. A few years later, the discrete version was given in [Hewer, 1971].

We approach the problem by rewriting (1.3) as

$$\mathcal{R}(X) = A^T X A - X + Q_x - (Q_{xu} + A^T X B) \cdot (Q_u + B^T X B)^{-1}(Q_{xu}^T + B^T X A) = 0. \quad (1.24)$$

The method is initialized with a stabilizing guess X_0, that is, $A - BK_0$ is stable for $K_0 = (Q_u + B^T X_0 B)^{-1}(Q_{xu}^T + B^T X_0 A)$. Finding an initial stabilizing guess for unstable systems is a problem in itself. In [Fassbender and Benner, 1999, Gallivan *et al.*, 2006] ways to find a stabilizing X_0 are described. Let the sequence of X_i be recursively determined by

1. Let $K_i = (Q_u + B^T X_i B)^{-1}(Q_{xu}^T + B^T X_i A)$ and $A_i = A - BK_i$.
2. Solve the Lyapunov equation $S_i = A_i^T S_i A_i + \mathcal{R}(X_i)$.
3. Set $X_{i+1} = X_i + S_i$.

The sequence X_i converges quadratically to the solution X of (1.24).

The step in this method that mainly contributes to the computational cost is step 2, solving the Lyapunov equation. A common method to solve such equations is to use the Bartels–Stewarts algorithm, [Bartels and Stewart, 1972] for the continuous case and [Barraud, 1977] for the discrete case. This algorithm relies on finding the Schur decomposition of A_i and a Cholesky factorization of $\mathcal{R}(X_i)$. The computational cost of finding the Schur decomposition of a matrix with size n scales as $O(n^3)$, [Golub and Van Loan, 1996]. The same holds for finding a Cholesky factorization of a matrix. The conclusion is that the ordinary Newton–Kleinman method has a computational complexity of $O(n^3)$.

Schur Method

A way of determining the solution to (1.3) is to consider the associated Hamiltonian matrix. The method was originally proposed in [Laub, 1979].

Define $\hat{A} = A - BQ_u^{-1}Q_{xu}$ and $\hat{Q}_x = Q_x - Q_{xu}Q_u^{-1}Q_{xu}^T$. If the matrix $\hat{A}$ is assumed to be non-singular, the Hamiltonian, in discrete time, is given by

$$H = \begin{bmatrix} \hat{A} + BQ_u^{-1}B^T\hat{A}^{-T}\hat{Q}_x & -BQ_u^{-1}B^T\hat{A}^{-T} \\ -\hat{A}^{-T}\hat{Q}_x & \hat{A}^{-T} \end{bmatrix}. \tag{1.25}$$

The problem when $\hat{A}$ is singular can be circumvented by instead examining a certain matrix pencil. For more details see [Lancaster and Rodman, 1995].

The Hamiltonian has exactly n stable eigenvalues. Let $[\, X_1^T \quad X_2^T \,]^T$ span the n-dimensional, stable, invariant subspace of H, that is, the subspace corresponding to all the stable eigenvalues of H. Then the solution to (1.3) is $X = X_2X_1^{-1}$. The objective in the Schur method is to find this invariant subspace. The method simply works by determining the Schur decomposition of H

$$U^*HU = T \tag{1.26}$$

where U is unitary and T is "quasi-upper" (meaning block-upper where the blocks are of size 1 or 2). By applying a sorting technique, the diagonal blocks of T, corresponding to the stable eigenvalues, can be moved to the top $n \times n$ block of T. The first n columns $[\, U_{11}^T \quad U_{21}^T \,]^T$ of the transformed U now spans the stable, invariant subspace of H, thus $X = U_{21}U_{11}^{-1}$.

The main step of the method that influences the computational cost is the Schur decomposition. Hence, this implies that the computational cost is at least $O(n^3)$.

The Schur method has some problems associated with it. Due to rounding errors when finding U and T through the Schur decomposition and the iterative permutation procedure, the matrix U_{11} may be ill-conditioned, meaning that there will be numerical difficulties in the inversion of $X = U_{21}U_{11}^{-1}$. Also, even when U_{11} is well-conditioned, there are no guarantees that $U_{21}U_{11}^{-1}$ is symmetric. For these reasons it is useful to complement the Schur method with a correction method in the end, see for example [Mehrmann and Tan, 1988]. Today, most Riccati solvers, for example the one in the MATLAB control toolbox [The MathWorks, Inc, 2010], are implemented using the Schur method.

Matrix Sign Function Method

The matrix sign function method is another method to solve (1.3). As the name suggest, the method relies on determining the matrix sign. The continuous version of the method can be found in [Denman and Beavers,

1976] while the discrete version, which requires a bit more work, is given in [Gardiner and Laub, 1986].

For any square matrix Y with Jordan canonical form VJV^{-1}, $\text{sign}(Y) = VSV^{-1}$, where S is a diagonal matrix with $S_{ii} = \text{sign}(\text{Re}\, J_{ii})$ (here sign refers to the usual, scalar function). Now, define the matrices

$$F = \begin{bmatrix} I & BQ_u^{-1}B^T \\ 0 & \hat{A}^T \end{bmatrix} \quad \text{and} \quad G = \begin{bmatrix} \hat{A} & 0 \\ -\hat{Q}_x & I \end{bmatrix}. \tag{1.27}$$

The Hamiltonian can be expressed in these matrices, $H = F^{-1}G$. It should be noted that the matrix F is invertible if and only if $\hat{A}$ is invertible. Form the transformed matrix $\widehat{H} = (G - F)^{-1}(G + F)$, where the stable eigenvalues of H are transformed to eigenvalues of $\widehat{H}$ with negative real part. By letting

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} = \text{sign}(\widehat{H}), \tag{1.28}$$

the solution X to (1.3) can be determined by solving the overdetermined equation system

$$\begin{bmatrix} W_{12} \\ W_{22} + I \end{bmatrix} X = - \begin{bmatrix} W_{11} + I \\ W_{21} \end{bmatrix}. \tag{1.29}$$

For similar reasons as with the Schur method, it is advisable to use a correction step on the obtained solution X.

Now, what needs to be determined is $\text{sign}(\widehat{H})$. An iterative procedure, which builds on Newton's method for solving $Z^2 = I$, can be used for this purpose. We initialize the iterative procedure by setting $Z_0 = \widehat{H}$. Now, Z_k is determined recursively by

$$Z_{k+1} = \frac{1}{2(\det Z_k)^{1/c}} \left(Z_k + Z_k^{-1} \right), \tag{1.30}$$

where c is the size of $\widehat{H}$.

Analysing the method with respect to the computational cost, the main steps that require computations are the inversion of Z_k in (1.30) and solving (1.29). Both of these operations scale as $O(n^3)$. A strength of the method is that the operations within it are well-suited for parallel implementation, see for example [Gardiner and Laub, 1988].

Methods for Large-Scale Riccati Equations

When the size of the system is of an order of 10^4 and above the methods described previously are no longer applicable due to computational and memory limitations. Methods for large-scale systems need to handle both

of these aspects. To deal with the memory requirements, the solution X is usually approximated by another matrix $\hat{X}$ that can be described in a more memory efficient way. Even though a system has a large state space, the number of inputs and outputs are typically small. [Penzl, 2000, Antoulas *et al.*, 2002] discuss the rapid decay of the eigenvalues of X for such systems. By disregarding the eigenvectors with very low eigenvalues, X can be approximated by a low-rank matrix, that is, $\hat{X} = ZZ^T$ for $Z \in \mathbb{R}^{n\times r}$ where $r \ll n$. The objective of the Riccati solver is now to find Z. Most methods using this approach apply the methodology of the Newton–Kleinman method. To solve the Lyapunov equation, which constitutes the main part of the calculations of the method, different techniques can be resorted to. The use of the Krylov subspace method was first proposed in [Saad, 1990]. The matrix Z is iteratively determined as an element of the Krylov subspace $\mathcal{K}_m = \{B, AB, \ldots, A^{m-1}B\}$ using either the Arnoldi or the Lanczos process. Other works that elaborate this approach are for example [Jaimoukha and Kasenally, 1994, Li and White, 2004, Jbilou and Riquet, 2006]. Another technique to solve the Lyapunov equation is to use a low rank version of the alternating direction implicit (ADI) iteration, presented in [Penzl, 1998, Gugercin *et al.*, 2003, Simoncini, 2007, Benner *et al.*, 2008, Benner and Faßbender, 2011]. The process relies on the ADI iteration

$$\begin{aligned} X_{i-1/2}(I - \mu_i A^T) &= (A - \mu_i I)X_{i-1}A^T + Q, \\ (I - \eta_i A)X_i &= AX_{i-1/2}(A^T - \eta_i I) + Q \end{aligned}$$

to solve the Lyapunov equation $X = AXA^T + Q$. If the ADI parameters μ and η are optimally chosen, the convergence of X_i towards the solution X can be shown to be superlinear. For a low-rank algorithm, the steps are altered to determine Z_i instead of X_i. For sparse matrices with $O(n)$ non-zero elements, the computational cost can be as low as $O(n)$, [Benner *et al.*, 2008].

The Newton–Kleinman method with a slightly different parametrization of $\hat{X}$ by $\mathscr{H}$-matrices is presented in [Grasedyck, 2008]. When the system matrices are of the same structure, they show that the computational complexity is $O(n(\log n)^p)$ for a small constant p. The concept of $\mathscr{H}$-matrices is also used in combination with the matrix sign function methodology, giving similar results in regard of the computational cost, [Grasedyck *et al.*, 2003].

Inspired by the use of Krylov subspaces to solve Lyapunov equations, [Jbilou, 2003, Heyouni and Jbilou, 2009] propose to directly solve the Riccati equation in a similar manner. Similarly, the iterative procedure works by projecting the solution X onto a Krylov subspace and solve a low-order Riccati equation with, for example, the Schur method.

1.3 Interconnected Systems and Distributed Control

A real system usually consists of a number of components that are linked together to form a functioning unit. It is common that the dynamics for each separate component is known. Connecting two components means that the output of one component will be the input of the other. We call such a system an interconnected system and the components are called subsystems or agents. The structure of the interconnections can then be represented in an interconnection graph, where the nodes correspond to the subsystems and the edges refer to the connections. If the system is linear, the resulting dynamics matrix A of the interconnected system has a structure that resembles the interconnection graph. Hence, if the interconnection matrix is sparse, that is, each component is only connected to a few other components, the dynamics matrix is also sparse. An actuator in an interconnected system usually only directly effects one component, or a few neighboring components. In case of linear systems, this means that the control matrix B also is sparse.

In a centralized controller the control action for each actuator is determined by a central decision maker. This means that the controller treats the systems as one single, possibly large-scale, system. In a distributed control system the decision making is divided out among the actuator. Each actuator has only limited access to measurements of the system, commonly only from subsystems in its neighborhood. In an ideal setup, for most cost functions, a centralized controller exhibits better performance than a distributed controller, since all decisions in a centralized controller is made with at least the same information available to a distributed controller. There are still a number of reasons to choose the latter:

- **Connection limitations.** The communication network may not allow every component to communicate. The cost of introducing a communication channel between distant agents may be much larger than the benefit given by the channel. Also, in some systems, not every component wants to share its internal information.

- **Robustness against failures.** If the centralized controller suffers from a failure, the entire system breaks down. If one controller in a distributed control system fails, the remaining systems may still be able to function. They may even be able to handle the tasks of the failed controller. Similarly, a centralized system is vulnerable to partial network failure, while only the failed part is effected in a distributed control system.

- **Difficulties to design large-scale controllers.** When only considering the system as a whole, the state space may be extremely

large. As seen in the previous section, this causes a problem when determining the optimal controller. For this reason it may be beneficial to use the interconnection topology in the design method for the controller to reduce the computational cost.

- **Introducing new agents.** If new agents are introduced into the system, a centralized controller needs to be completely recomputed. In a distributed solution we may only need to recompute the parts where the new agents are located.

Examples of interconnected systems and distributed control can be found in control of unmanned vehicles concerning their formation and task allocation, [Raffard *et al.*, 2004, Earl and D'Andrea, 2007, Dunbar and Murray, 2006], leader-follower problems in vehicular platoons in [Jovanovic *et al.*, 2008] and distributed control of power systems in [Jokic *et al.*, 2007, Hermans *et al.*, 2011].

The ideas of decision problems and distributed control have been studied since the 50s and have their origin in the economics literature. The theory of teams, defined as a group of people making decisions for a common reward, and how their common payoff is determined with respect to the uncertain environment they live in, were presented in [Marschak, 1955]. Each member of the team makes decisions with different information about the environment, resulting in a different problem than classical optimization where everyone has access to all information. It is also discussed how different communication networks can increase the payoff for the team. The ideas are more formally and mathematically treated and extended in [Radner, 1962].

In the late 60s the difficulties of certain information structures in the problem formulations were pointed out in [Witsenhausen, 1968]. With a simple example of a linear system with a quadratic cost function it was shown that with information constraints, a linear solution does not have to be optimal. The conclusion is that for systems with general information structures it is hard to find optimal solutions. However, it is possible to solve the problem for special classes of information constraints. The concept of a partially nested information structure was introduced in [Ho and Chu, 1972]. This concept means that any follower to an agent receives at least the same information as that agent does. Under this assumption of the information pattern, the optimal LQG controller is shown to be linear.

The most restrictive information constraint is that each agent only has access to local measurements. We denote this scenario by decentralized control. In [Wang and Davison, 1973] a necessary and sufficient condition for the existence of a linear, stabilizing controller is given. For a survey of the classical results in decentralized control, see [Sandell *et al.*, 1978].

In recent years distributed control has attracted a lot of attention. Systems that consist of identical subsystems, so-called spatially invariant systems, are considered in [Bamieh *et al.*, 2002]. This means that the systems will be of a spatially infinite or periodic nature. The systems are parametrized in the frequency domain and the controller is then designed. This will result in a spatially localized structure, which allows for a distributed implementation. Design of distributed controllers for spatially invariant systems with an additional assumption of propagation of information through the system were treated in [Bamieh and Voulgaris, 2005]. If the controller is assumed to have at least the same propagation speed as the system, it was shown how to convexify the controller design. This communication constraint means that any correction made by a subcontroller is communicated to its neighboring subcontrollers before the correction effects the neighboring subsystems.

[D'Andrea and Dullerud, 2003] was also concerned with control of infinitely many spatially interconnected, identical subsystems. The objective was to determine a controller minimizing the induced ℓ_2-norm and satisfying the interconnection structure of the system, that is, distributed $\mathcal{H}_\infty$ control synthesis. This was accomplished by solving linear matrix inequalities (LMIs) of a size comparable to each of the identical subsystem. The theories were extended to allow for heterogeneous systems in [Dullerud and D'Andrea, 2004, Langbort *et al.*, 2004]. In the latter publication, nonideal communications between subsystems, for example lossy or delayed channels, was also treated.

Quadratic invariance is another way of describing the information structure of the controller, introduced in [Rotkowitz and Lall, 2006]. The posed controller design problem was to minimize a given norm of the closed-loop system, while respecting the controller structure. If the set of admissible controllers, S, is quadratically invariant, the closed-loop system can be parametrized using the Youla-parametrization, where the Youla-parameter Q also is in S. Hence, the minimization problem was cast as a convex optimization program. The concept of quadratic invariance actually coincides with partially nestedness, see [Rotkowitz, 2008].

In [Borrelli and Keviczky, 2008] identical, dynamically decoupled systems were considered. They address the problem of finding optimal LQR controllers for a coupled cost function, where the controller respects an arbitrary, but given, communication graph. The problem is reduced to solving a Riccati equation of a size comparable to the maximum vertex degree of the interconnection graph.

A popular area of recent research is distributed model predictive control, in which distributed optimal control is to be accomplished in the presence of state and control constraints. The idea in ordinary model predictive control (MPC) is to solve an optimization problem, for example a

quadratic program, in every time sample of a discrete time system and to apply the first control action to the system. There are approaches to guarantee stability of MPC schemes, see [Mayne *et al.*, 2000]. When considering distributed MPC, instead of solving one large optimization problem in a centralized fashion, local optimization problems are posed for each agent or small collection of agents to achieve global performance. In [Keviczky *et al.*, 2006] a decentralized MPC scheme for dynamically decoupled systems was treated. The coupling comes from the cost function and constraints of the states between agents.

A methodology to solve optimal control problems for large-scale systems is to introduce Lagrange multipliers to remove the coupling in the dynamics equation. This has been known for many years, see for example [Mesarovic *et al.*, 1970]. The approach is well-suited for distributed MPC and the methods presented in for example [Negenborn *et al.*, 2008, Giselsson and Rantzer, 2011] build on it. The Lagrange multipliers are usually updated by gradient methods and require substantial communication in between neighbors. A survey of distributed and hierarchical MPC is given in [Scattolini, 2009].

1.4 Gradient Descent Method

The gradient descent method is a simple, iterative procedure to find a local minimizer to the unconstrained minimization problem

$$\min_x f(x) \tag{1.31}$$

for the continuously differentiable function $f : \boldsymbol{D} \to \mathbb{R}$, with domain $\boldsymbol{D} \subset \mathbb{R}^n$. The method works by starting with an initial guess $x^{(0)} \in \boldsymbol{D}$, and define the sequence $x^{(k+1)} = x^{(k)} - \gamma_k \nabla f(x^{(k)})$ where the step length γ_k is chosen such that $x^{(k+1)} \in \boldsymbol{D}$ and $f(x^{(k+1)}) < f(x^{(k)})$. The sequence $\{x^{(k)}\}$ now approaches a local minimizer to f, not necessarily ever reaching it. The choice of step length of course influences the rate with which the iterates approach the local minimum. It can for example be chosen through exact line search, meaning that we solve $\min_{\gamma_k > 0} f(x^{(k)} - \gamma_k \nabla f(x^{(k)}))$, or by the use of inexact techniques, for example Armijo's rule, see for example [Boyd and Vandenberghe, 2004]. The reason for resorting to such techniques is that exact line search is associated with many function evaluations. In problems where the gradient descent method is used it is not uncommon that function evaluations imply a high computational cost. The idea behind the inexact techniques is to instead determine a "good enough" step length by less function evaluations and determine a new search direction and start anew.

A problem associated with the gradient descent method is the slow convergence for poorly scaled problems. By investigating the gradient descent method for the objective function $f(x_1, x_2) = x_1^2 + \alpha x_2^2$, where $\alpha \gg 1$, we find that the solution can jump back and forth and hardly reducing f, implying a very slow convergence. The reason is the long and narrow level surfaces of f. The gradient descent method will show this behavior for any objective function with ill-conditioned level surfaces. There are ways to modify the gradient to account for the described problem. For any positive definite matrix B, $-B\nabla f$ is still a descent direction. For example, in Newton's method the scaling is chosen as the inverse of the Hessian of f. Hence, by an appropriate choice of the scaling B, the gradient method converges faster. Using *faster gradient descent methods* first presented in [Barzilai and Borwein, 1988] is also a possibility that can solve the problem. The idea is to keep the search direction but to choose the step length in a way that does not guarantee a strict monotone sequence of the function values. Many methods extends these ideas, see for example [Dai and Fletcher, 2005, Dai *et al.*, 2006] and references therein. It should be noted that there are some problems with these methods, see [van den Doel and Ascher, 2011].

The gradient method can be applied to constrained minimization problem as well. Assume that the minimization problem (1.31) is subjected to linear equality constraints

$$Ax = b.$$

If we assume that the initial point $x^{(0)}$ is feasible, the direction used in each step of the constrained gradient descent method, or more appropriately steepest descent method, is the gradient projected onto the constraint set. That is, we orthogonally project the gradient $\nabla f(x^{(k)})$ onto the subspace $\{x \mid Ax = 0\}$ to find the search direction. It is also possible to handle non-linear equality constraints and inequality constraints in a similar manner, see for example [Snyman, 2006].

Stochastic Gradient Descent Method

A concept in machine learning is the *stochastic* gradient descent method. The idea is to update the decision variable not by the true gradient but by a stochastic approximation of it. The reason could be that the true gradient is very difficult or impossible to determine. The presentation here is inspired by the one found in [Bottou, 2004].

The objective function in (1.31) now has a stochastic formulation

$$f(x) = \mathbf{E}_z Q(x, z) = \int Q(x, z)\, dP(z). \tag{1.32}$$

In the machine learning community f is referred to the *expected risk* function. The interpretation of the components is that x is a parameter

that should be tuned to minimize the influence of the surrounding environment described by the event z with probability distribution dP. The influence is given by the cost Q. The expected risk can be approximated by a finite set of independent observations $z^{(1)}, \ldots, z^{(N)}$, called a *training set*.

$$f(x) \approx f_{\text{approx}}(x) = \frac{1}{N} \sum_{k=1}^{N} Q(x, z^{(k)}). \tag{1.33}$$

One approach is now to approximate ∇f with ∇f_{approx} and use this approximation in the gradient descent method. This is referred to as the *batch* gradient descent method. A drawback with this approach is that to determine ∇f_{approx}, we need to determine $\nabla_x Q$ for the whole training set. Since the training set needs to be large in order to provide a good approximation, this can potentially result in a large computational cost. In the stochastic gradient descent method we instead choose an observation $z^{(k)}$ at random and update the value of the decision variable by

$$x^{(k+1)} = x^{(k)} - \gamma_k \nabla_x Q(x^{(k)}, z^{(k)}). \tag{1.34}$$

This approach substantially cuts the computational cost. In [Bottou, 2004] some convergence analysis of the method can be found.

The fairly cheap computational cost of the method makes it an attractive choice for solving large-scale problems. Examples of publications looking into this are [Langford *et al.*, 2009, Bottou, 2010, Xu, 2011].

1.5 Optimal Control Using Adjoint Variables

In this section we will review some aspects of finite horizon, optimal control. In the presentation, continuous time, non-linear systems will be considered and the cost function will not be required to be quadratic. More specifically, we consider the system

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \tag{1.35}$$

and the cost function

$$J(x, u) = \varphi(x(t_{\text{final}})) + \int_0^{t_{\text{final}}} L(x(t), u(t)) dt. \tag{1.36}$$

The objective of the optimal control problem is to find the control signal $u_{\text{opt}}(t)$ that minimizes (1.36) when $x(t)$ is subject to (1.35) for the given control signal. The numerical treatment of optimal control problems can be mainly divided into two branches: the *indirect* and the *direct* methods. These methods will briefly be described here, a more detailed discussion can be found in for example [Binder *et al.*, 2001].

Indirect Methods

For the indirect methods we introduce the Hamiltonian to (1.35) and (1.36)

$$\mathcal{H}(x(t), u(t), \lambda(t)) = L(x(t), u(t)) + \lambda(t)^T f(x(t), u(t)), \tag{1.37}$$

where $\lambda(t)$ are the so-called adjoint states. By Pontryagin's maximum principle, the Hamiltonian gives necessary conditions for the optimal solution $x_{\text{opt}}(t)$, $u_{\text{opt}}(t)$ and $\lambda_{\text{opt}}(t)$ to the optimal control problem previously described. They are

$$\dot{x}_{\text{opt}}(t) = f(x_{\text{opt}}(t), u_{\text{opt}}(t)), \qquad x_{\text{opt}}(0) = x_0, \tag{1.38}$$

$$\dot{\lambda}_{\text{opt}}(t) = -\frac{\partial \mathcal{H}}{\partial x}(x_{\text{opt}}(t), u_{\text{opt}}(t), \lambda_{\text{opt}}(t)), \quad \lambda_{\text{opt}}(t_{\text{final}}) = \left.\frac{\partial \varphi}{\partial x}\right|_{t=t_{\text{final}}}, \tag{1.39}$$

$$u_{\text{opt}}(t) = \arg\min_u \mathcal{H}(x_{\text{opt}}(t), u, \lambda_{\text{opt}}(t)), \tag{1.40}$$

where (1.40) is pointwise minimization. In the indirect methods, these sets of relations are to be solved. One possibility is to use gradient methods to solve the equations (1.38)–(1.40). The procedure will work by starting with an initial guess of the optimal u. With this guess the trajectories of x and λ are determined through (1.38) and (1.39). Using the trajectories, a gradient to the objective in (1.40) can be determined and the control is updated in this direction. This implies that in every iteration of updating the solution guess u, (1.38) needs to be numerically integrated in the forward time direction and (1.39) in the backwards time direction.

Another possibility comes from noticing the initial condition on the states and the final condition on the adjoint states. This means that we are faced with a two-point boundary value problem. The necessary conditions of optimality transform the optimal control problem into a multi–point boundary value problem, which can then be solved by appropriate numerical methods.

Direct Methods

In direct methods the control signal $u(t)$ is parametrized by a finite number of parameters $p_1, \ldots, p_n$. For example, by partitioning the time interval $[0, t_{\text{final}}]$ into a number of time intervals, $u(t)$ is described by polynomials in these partitions, where the parameters $\boldsymbol{p}$ are the coefficients of the polynomials. With the given parametrization, the optimal control problem in (1.35) and (1.36) is transformed to

$$\dot{x}(t) = f(x(t), u(t, \boldsymbol{p})), \quad x(0) = x_0, \tag{1.41}$$

$$\bar{J}(x, \boldsymbol{p}) = \varphi(x(t_{\text{final}})) + \int_0^{t_{\text{final}}} L(x(t), u(t, \boldsymbol{p}))dt. \tag{1.42}$$

Hence, in direct methods the original infinite optimal control problem is transformed into a finite dimensional non-linear programming problem. The objective of the direct methods is to minimize (1.42) with respect to the finite number of parameters $\boldsymbol{p}$ subject to the dynamics (1.41). Two related approaches for this are described below.

Direct Single Shooting Methods: In direct single shooting methods the minimization problem is solved using the gradient of $\bar{J}$ with respect to $\boldsymbol{p}$. The method can be found as early as in [Hicks and Ray, 1971]. For any set of parameters $\boldsymbol{p}$ the trajectory of x can be determined through forward integration of the dynamic system. Hence, the cost is actually defined solely by the parameters. We will assume that L is zero, that is, there is only a final cost. Since a non-zero L can be included in the final cost by introducing an auxiliary state for the integral, this is no limitation. With the trajectory of x, the gradient can be calculated by determining either a forward or an adjoint sensitivity analysis. With forward sensitivity analysis the expression for the gradient becomes

$$\nabla_{\boldsymbol{p}}\bar{J} = x_{\boldsymbol{p}}(t_{\text{final}}) \cdot \nabla_x \varphi(x(t_{\text{final}})), \tag{1.43}$$

$$\dot{x}_{\boldsymbol{p}} = x_{\boldsymbol{p}} \cdot \nabla_x f + u_{\boldsymbol{p}} \cdot \nabla_u f, \quad x_{\boldsymbol{p}}(0) = 0, \tag{1.44}$$

where $x_{\boldsymbol{p}}$ and $u_{\boldsymbol{p}}$ are short for $\nabla_{\boldsymbol{p}} x$ and $\nabla_{\boldsymbol{p}} u$, respectively. In the dynamics of $x_{\boldsymbol{p}}$, the gradients $\nabla_x f$ and $\nabla_u f$ are evaluated along the trajectories of x and u, respectively. Similarly, the adjoint sensitivity analysis gives that

$$\nabla_{\boldsymbol{p}}\bar{J} = \int_0^{t_{\text{final}}} u_{\boldsymbol{p}} \cdot \nabla_u f \cdot \lambda(t) dt, \tag{1.45}$$

$$\dot{\lambda} = -\nabla_x f \cdot \lambda, \quad \lambda(t_{\text{final}}) = \nabla_x \varphi(x(t_{\text{final}})). \tag{1.46}$$

That the method is referred to as single shooting comes from the fact that the trajectory of x is determined in a single shot. This may lead to numerical difficulties when unstable systems are considered, since the trajectories may diverge. In such cases it can be useful to apply the direct multiple shooting method.

Direct Multiple Shooting Methods: The problem of diverging trajectories in case of unstable systems is circumvented in the direct multiple shooting method by resetting the trajectory in a number of time instants. This method was first introduced in [Bock and Plitt, 1984]. Let the reset instants be the times $t_0 = 0 < t_1 < \ldots < t_k = t_{\text{final}}$. The resetting is accomplished by introducing artificial initial values s_i and to define the reset state trajectory x_{res}

$$\dot{x}_{\text{res}}(t, s_i, \boldsymbol{p}) = f(x_{\text{res}}(t, s_i, \boldsymbol{p}), u(t, \boldsymbol{p})), \quad t \in [t_i, t_{i+1}],$$
$$x_{\text{res}}(t_i, s_i, \boldsymbol{p}) = s_i.$$

With this reset state trajectory defined, the non-linear programming problem in (1.42) is extended by the equality constraints

$$\begin{aligned} x_0 &= s_0, \\ x_{\text{res}}(t_{i+1}, s_i, \boldsymbol{p}) &= s_{i+1}, \quad i = 0, \ldots, k-1. \end{aligned}$$

Hence, the non-linear program in the direct single shooting method has been transformed to one with more decision variables and more constraints. In a similar way as the direct single shooting method, either a forward or an adjoint sensitivity analysis should be applied to find the gradient of $\bar{J}$ with respect to the parameters $\boldsymbol{p}$.

1.6 Connections with the Thesis Contributions

In this section we connect the topics discussed in previous sections to the work in remaining part of the thesis. The thesis is devoted to methods for finding distributed LQG controllers for large-scale systems are treated in this thesis. This is accomplished by using the gradient descent method to iteratively improve the controller. The gradients are calculated using a trajectory of the adjoint system, similarly to how gradients are determined in optimal control for non-linear systems.

From Section 1.2 it is understood that solving the LQG problem becomes extremely difficult for large-scale problems, since finding the solution of the Riccati equation and the related Lyapunov equation requires too much computational time and memory. There is work towards solving the Riccati equation for some special types of large-scale systems, references found in Section 1.2. The aim of this thesis is to develop methods to find controllers that minimize a quadratic cost function, without using solutions of Riccati or Lyapunov equations. In this way we circumvent the time and memory problem when designing LQG controllers for large-scale systems.

The underlying idea of the methods in the thesis can be interpreted as the stochastic gradient descent method introduced in Section 1.4. For example, in the LQR problem, we consider a slightly different cost function than the one introduced in Section 1.1. The cost function is instead the average of the cost in Section 1.1 when the initial state is Gaussian with zero mean and known variance. A possibility to minimize this cost function is to use the gradient descent method. When dealing with large-scale systems it is not possible to evaluate the reformulated cost function, nor its gradient. However, using a similar approach as the ones described in Section 1.5, the gradient for a given initial state can be determined. Specifically, the gradients are calculated by combining trajectories of the

states and the adjoint states. Since the control is determined through state feedback, this can be thought of as a parametrization of the control, relating the calculations to the single and multiple shooting methods. A difference is that the problem in Section 1.5 is to determine an open loop control signal while the thesis focuses on finding a feedback control law. Now, following the ideas of the stochastic gradient descent method, in each update iteration of the feedback matrix, an initial state is chosen for which the gradient is computed. Using different initial states in every update iteration will result in that the final feedback matrix will be a local minimum of the reformulated cost function.

An assumption is that the systems have a structure that can be exploited, for example rank or sparsity structure. The calculated controllers will also be required to satisfy some structure. This means that the methods can be used to find distributed controllers for interconnected systems, described in Section 1.3. In Section 1.3 a number of different techniques to determine distributed controllers are discussed. They use a specified structure of the problem to cast it into a formulation that is recognized to have tractable techniques to find the optimal solution, for example to a set of LMIs or a convex optimization problem. In this thesis we do not focus on finding theoretical proofs that the actual optimal distributed solution can be obtained. Instead we focus on finding efficient methods that determine suboptimal solutions in a reasonable time, even for large-scale systems. For example, even though there are methods to solve LMIs, the computational time for such methods grows rapidly when the LMIs grows in size, unless some special type of structure of the LMI can be exploited. The same is true for methods that solve convex optimization problems. Another problem with methods that transform the controller design into a convex optimization problem is that even though the optimization problem is convex, it may be infinite dimensional. Hence, it requires a parametrization of the decision variables, which introduces approximations. In order to obtain good approximations, a large parameter space may be needed, thus increasing computational costs. Another aspect of the techniques in Section 1.3 is that for each of them only certain systems and controller structures are allowed, limiting their use to systems with general interconnection graphs. The methods given in this thesis do not require anything of the interconnection structure, except that it allows for the systems to be simulated efficiently.

Another way to deal with the interconnected property in a control system is to use distributed MPC, described in Section 1.3. This technique requires heavy online computations and is only applicable where this is not an issue. Also, methods for distributed MPC may require lots of communication to determine the control action. Most of the methods in this thesis rely on offline computations and the obtained controller does not

require much communication when controlling the system.

1.7 Future Work

This section presents a number of extensions of the ideas found in the thesis, which could be pursued. It is advised that this section is read after reading the papers of the thesis.

In the major part of the thesis, the methods rely on simulation of the system from initial states. This will generate descent directions with respect to the initial state and after a number of iterations the controller converges to a suboptimal controller for the average cost. But the gradient in each update iteration may be a poor estimate of the gradient to the average cost. If the systems are driven with noise instead of an initial state, as done in a section of Paper I, the approximation of the average gradient would be better. The reason for not using this strategy is that the suboptimality calculations proved difficult, due to a causality issue in the related optimization problem. It could be possible to avoid this issue by restating the optimization problem to a recursive setting.

Another issue that could be looked into is related to the one previously discussed about the estimation of the gradients. When the gradient of the average cost is approximated by a gradient of the cost from an initial state, it would be interesting to determine a measure of the error of the approximation. This measure could for example be the variance of the approximation error. Such a measure could prove valuable when investigating the needed size of the training set of the stochastic gradient descent method.

When a gradient is determined in the methods in this thesis, it is approximated by truncating infinite sums to finite sums. For general systems it is not trivial to choose a final time to guarantee that the approximation is still a descent direction. This problem could be resolved by allowing a variable final time. The idea is to find a criterion for the state trajectory that verifies that the truncated part of the infinite sum is negligible. Now, for each iteration, the state is simulated for enough time to satisfy this criterion.

A problem of the methods in the thesis is that a stabilizing solution is assumed to be known. For stable systems this is not a problem, but for general unstable systems it is far from trivial to find an initial stabilizing structured controller. As the methods resemble the direct single shooting method in optimal control, see Section 1.5, it could be possible to use an approach similar to the one taken in the direct multiple shooting method to circumvent this problem. It should be noted that it is not expected that this approach will work on general unstable systems, since this would

solve the problem of finding stabilizing structured controllers to unstable systems. But it would at least broaden the class of systems for which the methods presented in this thesis are applicable.

In Paper IV structured state feedback matrices are found by applying the methods. This should be considered initial work in designing a controller to a deformable mirror, since a few important issues are neglected. For example, since state feedback is assumed, the influence of limited sensing is not addressed. It would be interesting to examine the performance degradation when introducing state estimates instead of the true states. Also, both actuators and sensors may have noticeable dynamics that have to be included into the model.

References

Antoulas, A. C., D. C. Sorensen, and Y. Zhou (2002): "On the decay rate of Hankel singular values and related issues." *Systems & Control Letters*, **46**, pp. 323–342.

Åström, K. J. (2006): *Introduction to Stochastic Control Theory*. Dover, New York.

Bamieh, B., F. Paganini, and M. A. Dahleh (2002): "Distributed control of spatially invariant systems." *IEEE Transactions on Automatic Control*, **47:7**, pp. 1091–1107.

Bamieh, B. and P. G. Voulgaris (2005): "A convex characterization of distributed control problems in spatially invariant systems with communication constraints." *Systems & Control Letters*, **54:6**, pp. 575–583.

Barraud, A. (1977): "A numerical algorithm to solve $A^T X A - X = Q$." *IEEE Transactions on Automatic Control*, **22:5**, pp. 883–885.

Bartels, R. H. and G. W. Stewart (1972): "Solution of the matrix equation $AX + XB = C$ [F4]." *Commun. ACM*, **15**, September, pp. 820–826.

Barzilai, J. and J. Borwein (1988): "Two-point step size gradient methods." *IMA Journal of Numerical Analysis*, **8:1**, pp. 141–148.

Benner, P. and H. Faßbender (2011): "On the numerical solution of large-scale sparse discrete-time Riccati equations." *Advances in Computational Mathematics*, **35:2**, pp. 119–147.

Benner, P., J.-R. Li, and T. Penzl (2008): "Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems." *Numerical Linear Algebra with Applications*, **15**, November, pp. 755–777.

Binder, T., L. Blank, H. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J. Schlöder, and O. Stryk (2001): "Introduction to model based optimization of chemical processes on moving horizons." In Grötschel *et al.*, Eds., *Online Optimization of Large Scale Systems: State of the Art*, pp. 295–340. Springer.

Bini, D., B. Iannazzo, and B. Meini (2011): *Numerical Solution of Algebraic Riccati Equations*. Society for Industrial and Applied Mathematics.

Bock, H. G. and K. J. Plitt (1984): "A multiple shooting algorithm for direct solution of optimal control problems." In *Proceedings 9th IFAC World Congress Budapest*, pp. 243–247. Pergamon Press.

Borrelli, F. and T. Keviczky (2008): "Distributed LQR design for identical dynamically decoupled systems." *IEEE Transactions on Automatic Control*, **53:8**, pp. 1901–1912.

Bottou, L. (2004): "Stochastic learning." In *Advanced lectures on machine learning*, vol. 3176 of *Lecture notes in artificial intelligence*, pp. 146–168. Springer–Verlag Berlin. Machine Learning Summer School Conference 2003, Max Planck Inst Biol Cybernet, Tubingen, Australia, aug, 2003.

Bottou, L. (2010): "Large-scale machine learning with stochastic gradient descent." In Lechevallier and Saporta, Eds., *Proceedings of COMPSTAT'2010*, pp. 177–186. Physica-Verlag HD.

Boyd, S. and C. Barratt (1991): *Linear Controller Design—Limits of Performance*. Prentice Hall, Englewood Cliffs, New Jersey 07632.

Boyd, S. and L. Vandenberghe (2004): *Convex Optimization*. Cambridge University Press.

Dai, Y.-H. and R. Fletcher (2005): "On the asymptotic behaviour of some new gradient methods." *Mathematical Programming*, **103**, pp. 541–559. 10.1007/s10107-004-0516-9.

Dai, Y.-H., W. W. Hager, K. Schittkowski, and H. Zhang (2006): "The cyclic Barzilai–Borwein method for unconstrained optimization." *IMA Journal of Numerical Analysis*, **26:3**, pp. 604–627.

D'Andrea, R. and G. Dullerud (2003): "Distributed control design for spatially interconnected systems." *IEEE Transactions on Automatic Control*, **48:9**, pp. 1478–1495.

Denman, E. D. and A. N. Beavers, Jr. (1976): "The matrix sign function and computations in systems." *Applied Mathematics and Computation*, **2:1**, pp. 63–94.

Dullerud, G. and R. D'Andrea (2004): "Distributed control of heterogeneous systems." *IEEE Transactions on Automatic Control*, **49:12**, pp. 2113–2128.

Dunbar, W. B. and R. M. Murray (2006): "Distributed receding horizon control for multi-vehicle formation stabilization." *Automatica*, **42:4**, pp. 549–558.

Earl, M. G. and R. D'Andrea (2007): "A decomposition approach to multi-vehicle cooperative control." *Robotics and Autonomous Systems*, **55**, April, pp. 276–291.

Fassbender, H. and P. Benner (1999): "Initializing Newton's method for discrete-time algebraic Riccati equations using the butterfly SZ algorithm." In *Proceedings of the 1999 IEEE International Symposium on Computer Aided Control System Design, 1999.*, pp. 70–74.

Gallivan, K., X. Rao, and P. V. Dooren (2006): "Singular Riccati equations stabilizing large-scale systems." *Linear Algebra and its Applications*, **415:2–3**, pp. 359–372.

Gardiner, J. and A. Laub (1988): "Matrix sign function implementations on a hypercube multiprocessor." In *Proceedings of the 27th IEEE Conference on Decision and Control*, vol. 2, pp. 1466–1471.

Gardiner, J. D. and A. J. Laub (1986): "A generalization of the matrix-sign-function solution for algebraic Riccati equations." *International Journal of Control*, **44:3**, pp. 823–832.

Giselsson, P. and A. Rantzer (2011): "On stopping conditions in duality-based distributed model predictive control." *IEEE Transactions on Automatic Control*. Submitted.

Golub, G. H. and C. F. Van Loan (1996): *Matrix computations*. Johns Hopkins Univ. Press, Baltimore.

Grasedyck, L. (2008): "Nonlinear multigrid for the solution of large-scale Riccati equations in low-rank and $\mathscr{H}$-matrix format." *Numerical Linear Algebra with Applications*, **15:9**, pp. 779–807.

Grasedyck, L., W. Hackbusch, and B. N. Khoromskij (2003): "Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices." *Computing*, **70**, pp. 121–165.

Gugercin, S., D. C. Sorensen, and A. C. Antoulas (2003): "A modified low-rank Smith method for large-scale Lyapunov equations." *Numerical Algorithms*, **32**, pp. 27–55.

Hermans, R. M., M. Lazar, and A. Jokic (2011): "Distributed predictive control of the 7-machine CIGRÉ power system." In *Proceedings of the 2011 American Control Conference*, pp. 5225–5230.

Hewer, G. (1971): "An iterative technique for the computation of the steady state gains for the discrete optimal regulator." *IEEE Transactions on Automatic Control*, **16:4**, pp. 382–384.

Heyouni, M. and K. Jbilou (2009): "An extended block Arnoldi algorithm for large-scale solutions of the continuous-time algebraic Riccati equation." *Electronic Transactions on Numerical Analysis*, **33**, pp. 53–62.

Hicks, G. and W. Ray (1971): "Approximation methods for optimal control systems." *Can. J. Chem. Engng.*, **49**, pp. 522–528.

Ho, Y.-C. and K.-C. Chu (1972): "Team decision theory and information structures in optimal control problems–Part I." *IEEE Transactions on Automatic Control*, **17:1**, pp. 15–22.

Jaimoukha, I. M. and E. M. Kasenally (1994): "Krylov subspace methods for solving large Lyapunov equations." *SIAM Journal on Numerical Analysis*, **31:1**, pp. 227–251.

Jbilou, K. (2003): "Block Krylov subspace methods for large algebraic Riccati equations." *Numerical Algorithms*, **34**, pp. 339–353.

Jbilou, K. and A. J. Riquet (2006): "Projection methods for large Lyapunov matrix equations." *Linear Algebra and its Applications*, **415:2 3**, pp. 344–358.

Jokic, A., P. P. J. van den Bosch, and M. Lazar (2007): "Distributed price-based optimal control of power systems." In *IEEE International Conference on Control Applications*, pp. 910–915.

Jovanovic, M. R., J. M. Fowler, B. Bamieh, and R. D'Andrea (2008): "On the peaking phenomenon in the control of vehicular platoons." *Systems & Control Letters*, **57:7**, pp. 528–537.

Keviczky, T., F. Borrelli, and G. J. Balas (2006): "Decentralized receding horizon control for large scale dynamically decoupled systems." *Automatica*, **42:12**, pp. 2105–2115.

Kleinman, D. (1968): "On an iterative technique for Riccati equation computations." *IEEE Transactions on Automatic Control*, **13:1**, pp. 114–115.

Lancaster, P. and L. Rodman (1995): *Algebraic Riccati Equation*. Oxford University Press.

Langbort, C., R. S. Chandra, and R. D´Andrea (2004): "Distributed control design for systems interconnected over an arbitrary graph." *IEEE Transactions on Automatic Control*, **49:9**, pp. 1502–1519.

Langford, J., L. Li, and T. Zhang (2009): "Sparse online learning via truncated gradient." *Journal of Machine Learning Research*, **10**, June, pp. 777–801.

Laub, A. (1979): "A Schur method for solving algebraic Riccati equations." *IEEE Transactions on Automatic Control*, **24:6**, pp. 913–921.

Li, J.-R. and J. White (2004): "Low-rank solution of Lyapunov equations." *SIAM Review*, **46:4**, pp. 693–713.

Marschak, J. (1955): "Elements for a theory of teams." *Management Sci.*, **1**, pp. 127–137.

Mayne, D., J. Rawlings, C. Rao, and P. Scokaert (2000): "Constrained model predictive control: Stability and optimality." *Automatica*, **36:6**, pp. 789–814.

Mehrmann, V. and E. Tan (1988): "Defect correction method for the solution of algebraic Riccati equations." *IEEE Transactions on Automatic Control*, **33:7**, pp. 695–698.

Mesarovic, M. D., D. Macko, and Y. Takahara (1970): *Theory of hierarchical, multilevel, systems*. Academic Press, New York.

Negenborn, R., B. De Schutter, and J. Hellendoorn (2008): "Multi-agent model predictive control for transportation networks: Serial versus parallel schemes." *Engineering Applications of Artificial Intelligence*, **21:3**, pp. 353–366.

Penzl, T. (1998): "A cyclic low rank Smith method for large sparse Lyapunov equations with applications in model reduction and optimal control." *SIAM J. Sci. Comput*, **21**, pp. 1401–1418.

Penzl, T. (2000): "Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case." *Systems & Control Letters*, **40:2**, pp. 139–144.

Radner, R. (1962): "Team decision problems." *Ann. Math. Statist.*, **33:3**, pp. 857–881.

Raffard, R. L., C. J. Tomlin, and S. P. Boyd (2004): "Distributed optimization for cooperative agents: Application to formation flight." In *Proceedings of the 43rd IEEE Conference on Decision and Control*.

Rotkowitz, M. (2008): "On information structures, convexity, and linear optimality." In *Proceedings of the 47th IEEE Conference on Decision and Control*, pp. 1642–1647.

Rotkowitz, M. and S. Lall (2006): "A characterization of convex problems in decentralized control." *IEEE Transactions on Automatic Control*, **51:2**, pp. 274–286.

Saad, Y. (1990): "Numerical solution of large Lyapunov equations." In *Signal Processing, Scattering and Operator Theory and Numerical Methods*, pp. 503–511. Birkhäuser.

Sandell, Jr., N. R., P. Varaiya, M. Athans, and M. G. Safonov (1978): "Survey of decentralized control methods for large scale systems." *IEEE Transactions on Automatic Control*, **23:2**, pp. 108–128.

Scattolini, R. (2009): "Architectures for distributed and hierarchical model predictive control – a review." *Journal of Process Control*, **19:5**, pp. 723–731.

Simoncini, V. (2007): "A new iterative method for solving large-scale Lyapunov matrix equations." *SIAM J. Sci. Comput*, **29**, pp. 1268–1288.

Snyman, J. A. (2006): *Practical mathematical optimization: An introduction to basic optimization theory and classical and new gradient-based algorithms*. Springer Verlag.

The MathWorks, Inc (2010): *Control System Toolbox™ User's Guide, Version 9*. The MathWorks Inc, 3 Apple Hill Drive, Natick, MA 01760-2098.

van den Doel, K. and U. Ascher (2011): "The chaotic nature of faster gradient descent methods." *Journal of Scientific Computing*, pp. 1–22.

Wang, S.-H. and E. Davison (1973): "On the stabilization of decentralized control systems." *IEEE Transactions on Automatic Control*, **18:5**, pp. 473–478.

Witsenhausen, H. S. (1968): "A counterexample in stochastic optimum control." *SIAM Journal on Control*, **6:1**, pp. 138–147.

Xu, W. (2011): "Towards optimal one pass large scale learning with averaged stochastic gradient descent." *CoRR*.

Zhou, K., J. Doyle, and K. Glover (1996): *Robust and Optimal Control*. Prentice-Hall.

Paper I

Synthesis of Structured Controllers for Large-Scale Systems

Karl Mårtensson and Anders Rantzer

Abstract

We present a synthesis procedure to design structured controllers for linear systems to optimize a quadratic performance criterion. Controllers are updated in an iterative fashion to reduce the cost. Descent directions are determined by simulating the system itself and the corresponding adjoint system. In each iterate suboptimality bounds are calculated in order to validate the current controller. An important property of the proposed method is that the computational complexity scales linearly when the system matrices are sparse. Hence it is useful when designing controllers for large-scale sparse systems, for example distributed systems or systems resulting from discretized PDEs.

1. Introduction

The theory of optimal linear quadratic control has been around for several decades and is well documented in the literature, for example [Åström, 2006, Bryson and Ho, 1975, Lancaster and Rodman, 1995]. The solutions are closely related to solving the algebraic Riccati equation to get an expression for an optimal feedback matrix.

Solving optimal linear quadratic control problems with the use of Riccati equations works well when considering small and medium sized systems. Due to the fact that workspace complexity is $O(n^2)$ and the computational complexity is $O(n^3)$, see [Benner *et al.*, 2008], this method of finding the optimal solution is less tractable for large-scale systems. When solving general large-scale optimization problems, one can not use conventional approaches but one needs to find and exploit some structure in the problem [Lasdon, 2002]. One common structure of linear dynamic systems is that the system matrices are sparse. Such sparse systems are for example obtained from discretization of PDEs, for example [Heimsten, 2011, Lasiecka and Tuffaha, 2009, Staffans, 1996], or when considering large interconnected systems. The papers [Benner *et al.*, 2008, Benner and Faßbender, 2011] treat systems with a large state space but only a few inputs, for example systems resulting from discretization of PDEs. The solution to the Riccati equation is approximated with low rank Cholesky factors, and the feedback matrix is determined by a tractable product of these. Another low rank approximation method based on stable invariant subspaces of the Hamiltonian is given in [Amodei and Buchot, 2010]. A Riccati based method for stabilization of large-scale systems can be found in [Rao *et al.*, 2000].

Except for the possible large state space, distributed interconnected systems introduces another constraint on the controllers. When controlling these systems, it is often desirable to respect a communication structure in which a subsystem is only allowed to communicate with a few other subsystems. Already in the late 1960s it was pointed out that such problems are fundamentally difficult to solve. In particular [Witsenhausen, 1968] showed that even a small quadratic control problem does not have a linear optimal solution. The stabilizability and optimality of linear decentralized controllers for large-scale interconnected systems was investigated in [Ikeda *et al.*, 1983]. In recent years a lot of work has been made to provide synthesis methods for classes of distributed systems. A concept of quadratic invariance was introduced in [Rotkowitz and Lall, 2006, Rotkowitz and Lall, 2002], which transforms the problem of finding an optimal controller into a convex optimization program. [Bamieh *et al.*, 2002, Bamieh and Voulgaris, 2005] investigated systems that are spatially invariant. A structured state and output feedback $\mathcal{H}_2$ control synthesis

method by relaxing the Riccati equations via linear matrix inequalities was discussed in [Li and Paganini, 2006]. Other methods which involve the use of linear matrix inequalities were presented in for example [Gattami, 2006, Langbort *et al.*, 2004, Rantzer, 2006].

It is possible to investigate the solution of a linear quadratic control problem using Pontryagin's Maximum principle, by introducing adjoint states, [Bryson and Ho, 1975, Heinkenschloss, 2005]. In [Bryson and Ho, 1975] it was shown how the express the optimal control signal in terms of adjoint states. In [Heinkenschloss, 2005] large-scale, sparse continuous-time system were discretized to solve a open-loop linear quadratic control problem. By introducing adjoint variables, the problem was transformed to a set of sparse linear equations, which was solved by Gauss-Seidel (GS) iterations. It was also suggested how solve the equations with GS as a preconditioner in a Krylov subspace method.

In this paper we present a method to iteratively improve a structured linear controller with respect to a given linear quadratic performance. The underlying idea works by determining a descent direction to the performance and to take a step in that direction to reduce the cost. A similar technique has been studied in [Geromel and Bernussou, 1982] where the descent direction was determined by calculating solutions to Riccati equations, which means that it is not applicable for large-scale systems. In this work we determine the descent direction by simulating the dynamical system and the corresponding adjoint system. The trajectories are used to find the descent direction. When the dynamical system is sparse it turns out that this will produce a scalable method. The trajectory determined for calculating a descent direction can also be used to find a suboptimality bound of the current controller. This gives a way to verify that the controller is close to the optimal solution. Hence the bound can be used as a criterion when to stop the process of updating the controller to improve the performance. The paper builds on the recent papers [Mårtensson and Rantzer, 2009, Mårtensson and Rantzer, 2010].

The paper is organized as follows. In Section 2 the general problem setup is given. The process of finding the descent direction to the quadratic cost function using the trajectories of the dynamical and adjoint system is found here. This section also shows how to determine the suboptimality bounds. In Section 3 the dynamical system is restricted to sparse distributed systems and it is shown how the previously presented method is used to give a scalable scheme. If the dynamical system is modified to include noise, a real-time scheme can be handled in a similar fashion. The details of this process is found in Section 4. Two examples illustrating the methods are presented in Section 5.

1.1 Mathematical Notation

The set of real numbers is denoted by $\mathbb{R}$, real vectors of dimension n by $\mathbb{R}^n$ and real $n \times m$ matrices by $\mathbb{R}^{n\times m}$. When a partition of a vector or a matrix exists, subscripts will refer to that partition. For example, for $x \in \mathbb{R}^n$ then $x_i \in \mathbb{R}^{n_i}$ refers to the ith partition of x and $A \in \mathbb{R}^{n\times m}$ then $A_{i,j} = A_{ij} \in \mathbb{R}^{n_i \times m_j}$ refers to the i,jth partition of A. When the components of the vectors or matrices are ordered with respect to the partition, x_i or A_{ij} equivalently means the ith or i,jth *block* of x or A, respectively. For a matrix valued function $f : \mathbb{R}^{n\times m} \to \mathbb{R}$ we define the differential df as the part of $f(X + dX) - f(X)$ that is linear in dX, that is the linearized part of f. The gradient of f with respect to X is denoted $\nabla_X f$ and means

$$\nabla_X f = \begin{bmatrix} \frac{\partial f}{\partial X_{1,1}} & \frac{\partial f}{\partial X_{1,2}} & \cdots & \frac{\partial f}{\partial X_{1,m}} \\ \frac{\partial f}{\partial X_{2,1}} & \frac{\partial f}{\partial X_{2,2}} & \cdots & \frac{\partial f}{\partial X_{2,m}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial X_{n,1}} & \frac{\partial f}{\partial X_{n,2}} & \cdots & \frac{\partial f}{\partial X_{n,m}} \end{bmatrix}$$

For a pair of $(A, B) \in \mathbb{R}^{m\times m} \times \mathbb{R}^{m\times p}$ we say that the matrix $K \in \mathbb{R}^{p\times m}$ stabilizes (A, B) if $A - BK$ has all its eigenvalues in the unit circle. A pair (A, B) is said to be stabilizable if such K exists.

2. General Theory

In this paper we consider linear time invariant systems in discrete time. Hence, the dynamic equation for these systems is

$$x(t + 1) = Ax(t) + Bu(t), \quad x(0) = x_0 \tag{1}$$

where $x(t) \in \mathbb{R}^m$ and $u(t) \in \mathbb{R}^p$ and $t \geq 0$. The matrices $A \in \mathbb{R}^{m\times m}$ and $B \in \mathbb{R}^{m\times p}$. We assume that (A, B) is stabilizable in the paper.

2.1 Problem Formulation

We wish that the controllers used to control the system (1) should minimize a quadratic expression in the system states and control variables. Hence we are looking at linear quadratic control (LQR) synthesis. With the knowledge that linear state feedback controllers are optimal for this problem, the controllers we consider are such. That is, we consider controllers on the form

$$u(t) = -Kx(t) \tag{2}$$

A difference from usual LQR design is that we are interested in structured controllers and restrict the allowed feedback matrices to some subspace $\mathscr{K} \subset \mathbb{R}^{p \times m}$. Also, let the set of admissible *stabilizing* feedback matrices be

$$\mathscr{K}_{\text{stab}} = \{K \mid K \in \mathscr{K} \text{ and } K \text{ stabilizes (1)}\}$$

With the dynamics in (1) and the control in (2) in mind, for every $K \in \mathscr{K}_{\text{stab}}$ we define the linear quadratic cost function

$$J(K, x_0) = \sum_{t=0}^{\infty} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T \underbrace{\begin{bmatrix} Q_x & Q_{xu} \\ Q_{xu}^T & Q_u \end{bmatrix}}_{Q} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \tag{3}$$

where Q_x is positive semi-definite and Q_u is positive definite. The objective for the control synthesis is then to find the admissible stabilizing feedback matrix that minimizes $J(K, x_0)$.

2.2 Analysis

By solving certain Lyapunov equations we can get tractable expressions for (3). Given the solution unique X_0 for the Lyapunov equation

$$X_0 = (A - BK)X_0(A - BK)^T + x_0 x_0^T \tag{4}$$

we have that

$$J(K, x_0) = \operatorname{tr}\left((Q_x - 2Q_{xu}K + K^T Q_u K)X_0\right)$$

By the unique solution P to another Lyapunov equation

$$P = (A - BK)^T P(A - BK) + Q_x - Q_{xu}K - K^T Q_{xu}^T + K^T Q_u K \tag{5}$$

we have another expression for the cost function

$$J(K, x_0) = \operatorname{tr}\left(P x_0 x_0^T\right)$$

With these expressions we are able to determine the gradient of J with respect to K.

Proposition 1
Given the system (1) and a stabilizing K, the gradient of the cost function J defined in (3) with respect to K is

$$\nabla_K J = 2\left(Q_u K - Q_{xu}^T - B^T P(A - BK)\right) X_0, \tag{6}$$

where X_0 and P satisfy the Lyapunov equations (4) and (5). □

Proof. We will determine an expression for the differential of P. Let's start with defining the following notation to simplify the future expressions.

$$\begin{aligned} A_K &= A - BK \\ M &= dK^T \left(Q_u K - Q_{xu}^T - B^T P A_K \right) \end{aligned}$$

Differentiating (5) shows that dP satisfies the Lyapunov equation

$$dP = A_K^T dP A_K + M + M^T \iff dP = \sum_{k=0}^{\infty} (A_K^T)^k (M + M^T) A_K^k$$

Hence, since $dJ = \mathrm{tr}\left(dP x_0 x_0^T\right)$, we get that

$$\begin{aligned} dJ &= 2 \cdot \mathrm{tr} \left(\sum_{k=0}^{\infty} (A_K^T)^k M A_K^k x_0 x_0^T \right) \\ &= 2 \cdot \mathrm{tr} \left(M \sum_{k=0}^{\infty} A_K^k x_0 x_0^T (A_K^T)^k \right) \\ &= 2 \cdot \mathrm{tr} \left(dK^T (Q_u K - Q_{xu}^T - B^T P A_K) X_0 \right) \end{aligned}$$

By using the relation about differentials

$$dZ \;=\; \mathrm{tr}\left(dX^T \cdot Y\right) \implies \nabla_X Z = Y$$

the relation (6) is verified. □

The expression in (6) involves the solution of the Lyapunov equation (5). Finding the solution to a Lyapunov equation is for large system a time consuming operation. In [Bryson and Ho, 1975] it is shown how to express the optimal control signal with adjoint states and in that way not have to solve a Riccati equation. In the following proposition we show how to get rid of the matrix P in the expression for $\nabla_K J$ by introducing adjoint states.

Proposition 2
Given the system (1) and a stabilizing K, let the adjoint states λ be defined by the backwards iteration

$$\lambda(t-1) = (A - BK)^T \lambda(t) - (Q_x - Q_{xu} K - K^T Q_{xu}^T + K^T Q_u K) x(t) \quad (7)$$

where $x(t)$ are the states of (1), with $\lim_{t\to\infty} \lambda(t) = 0$. Then

$$\nabla_K J = 2\left(\sum_{t=0}^{\infty} \left(-Q_u u(t) - Q_{xu}^T x(t) + B^T \lambda(t)\right) x(t)^T\right) \tag{8}$$

□

Proof. For simplicity, let $Q_K = Q_x - Q_{xu}K - K^T Q_{xu}^T + K^T Q_u K$ and keep the notation from Proposition 1. For any j,

$$\lambda(j) = -\sum_{k=j+1}^{\infty} (A_K^T)^{k-j-1} Q_K x(k) = -\sum_{k=0}^{\infty} (A_K^T)^k Q_K A_K^{k+1} x(j)$$

Hence

$$\sum_{t=0}^{\infty} \lambda(t)x(t)^T = -\sum_{t=0}^{\infty}\sum_{k=0}^{\infty} (A_K^T)^k Q_K (A_K)^{k+1} x(t)x(t)^T = -PA_K X_0$$

Fitting this into (6) and using that $X_0 = \sum_{t=0}^{\infty} x(t)x(t)^T$ gives the desired result. □

Given the gradient for a stabilizing admissible feedback matrix K, the projected gradient, $\nabla_K J|_{\mathscr{K}}$, is clearly a descent direction to $J(K, x_0)$. We realize this by noting that $\nabla_K J|_{\mathscr{K}}$ is the gradient to the restricted function $J|_{\mathscr{K}}$ and is hence descent direction to $J|_{\mathscr{K}}$. But by definition, $J|_{\mathscr{K}} \equiv J$ on $\mathscr{K}$.

2.3 General Algorithm

Instead of infinite time, we truncate the sum in (8), implying that we approximate the gradient of J. Let the final time of the sum be t_{final}. We will have to simulate the states from $t = 0$ to $t = t_{\text{final}}$ in increasing time and after which the adjoint states are simulated from $t = t_{\text{final}}$ to $t = 0$ in decreasing time. With these trajectories we are able to determine an approximation of the gradient. We summarize the procedure into the following algorithm.

ALGORITHM 1
Consider a system (1) with control $u(t) = -Kx(t)$ where $K \in \mathscr{K}_{\text{stab}}$. To find a local minimizer to (3), start with $K^{(0)} \in \mathscr{K}_{\text{stab}}$ and for each $\tau \geq 0$,

1. Simulate the states of (1) with control $u(t) = -K^{(\tau)}x(t)$ for times $t = [0, t_{\text{final}}]$.

2. Simulate the adjoint states of (7) for times $t = [0, t_{\text{final}}]$ in the backwards time direction with $\lambda(t_{\text{final}}) = 0$.

3. Calculate an approximation of $\nabla_K J$

$$\nabla_K J_{\text{approx}} = 2 \sum_{t=0}^{t_{\text{final}}} \Big(-[Q_u]u(t) - [Q_{xu}]^T x(t) + B^T \lambda(t) \Big) x(t)^T$$

and project the approximation on the admissible set of feedback matrices

$$G = \text{proj}_{\mathscr{K}} (\nabla_K J_{\text{approx}})$$

4. Update the feedback matrix in the direction of the projected gradient

$$K^{(\tau+1)} = K^{(\tau)} - \gamma_\tau G$$

for some step length γ_τ.

5. Increase τ with 1 and goto 1).

□

Remark 1
In step 4 of the algorithm the feedback matrix is updated in the direction of the projected gradient with some step length. How to pick an appropriate step length is of course some concern. In a way, the step length is a design parameter of the algorithm, but there are approaches to estimate a suitable step length for a direction. One way is to simulate (1) with the $K = K^{(\tau)} - \gamma_\tau G$. If this K does not lead to a decreased cost, the step length is scaled by some factor < 1. □

As posed, there is no stopping criteria for Algorithm 1. It is possible to specify the number of iterations to update the feedback matrix. This strategy will not guarantee that any kind of performance of the acquired feedback matrix is met. In the following section we describe how to calculate a bound of the suboptimality. By specifying the amount of suboptimality required of the sought feedback matrix, the suboptimality bound can function as a stopping criteria.

2.4 Suboptimality Bounds

Solving the ordinary LQR control problem is a well-studied problem and has a tractable solution. But finding the minimizing feedback matrix, when imposing a structure, is not even guaranteed to be convex. The

underlying method in Algorithm 1 is a descent method, and hence we can not guarantee that the globally optimal structured feedback matrix is ever attained. As mentioned in Algorithm 1, this method produces a locally optimal solution. A measure of the suboptimality for the feedback matrix in each iteration step of the update algorithm, is $\alpha \geq 1$ such that

$$J(K, x_0) \leq \alpha J(K_{\text{opt}}, x_0), \tag{9}$$

where $K_{\text{opt}} = \arg\min_K J(K, x_0)$. That is, $J(K, x_0)$ is within a factor of α of the actual optimal value. This means that if we can verify that an α close to 1 must satisfy (9), then even though K might not be the optimal feedback matrix, we will not find one that reduces the cost greatly compared to this one.

We introduce the truncated version of the cost by

$$J(K, x_0, t_{\text{final}}) = \sum_{t=0}^{t_{\text{final}}} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \tag{10}$$

where the states $x(t)$ satisfy (1) and $u(t) = -Kx(t)$. The following theorem gives us a suboptimality bound telling us that in the time interval $[0, t_{\text{final}}]$ we are within a factor of α of the optimal solution on this interval.

THEOREM 1
If $\alpha \geq 1$ is such that for a given sequence of dual (or adjoint) variables $\lambda(t)$, with $\lambda(t_{\text{final}}) = 0$

$$J(K, x_0, t_{\text{final}}) \leq \alpha \min_{\substack{x,u \\ x(0)=x_0}} \sum_{t=0}^{t_{\text{final}}} \Bigg(\begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} + 2\lambda(t)^T \left(x(t+1) - Ax - Bu(t) \right) \Bigg) \tag{11}$$

then

$$J(K, x_0, t_{\text{final}}) \leq \alpha J(K_{\text{opt}}, x_0, t_{\text{final}}), \tag{12}$$

where

$$K_{\text{opt}} = \arg\min_K J(K, x_0, t_{\text{final}})$$

□

Proof. Assume that α is such that for a given sequence of $\lambda(t)$, (11) holds. We have that

$$
\begin{aligned}
J(K_{\text{opt}}, x_0, t_{\text{final}}) &= \begin{cases} \min\limits_{K,x} \sum\limits_{t=0}^{t_{\text{final}}} \begin{bmatrix} x(t) \\ -Kx(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ -Kx(t) \end{bmatrix} \\ \text{subject to: } x(t+1) = (A - BK)x(t) \\ \qquad\qquad x(0) = x_0 \end{cases} \\
&\geq \begin{cases} \min\limits_{x,u} \sum\limits_{t=0}^{t_{\text{final}}} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \\ \text{subject to: } x(t+1) = Ax(t) - Bu(t) \\ \qquad\qquad x(0) = x_0 \end{cases} \\
&\geq \min_{\substack{x,u \\ x(0)=x_0}} \sum_{t=0}^{t_{\text{final}}} \Bigg(\begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \\
&\qquad\qquad + 2\lambda(t)^T (x(t+1) - Ax(t) - Bu(t)) \Bigg)
\end{aligned}
$$

where the second inequality comes from introducing dual variables. Hence, if (11) holds, so must (12). □

Remark 2
If the matrix Q is not positive definite (that is, only positive semi-definite) it is easily realized the minimization program in (11) is not necessarily bounded. For any direction in $[\,x(t)^T \quad u(t)^T\,]^T$ that is not penalised, $\lambda(t)$ needs to be such that the term $2\lambda(t)^T(x(t+1) - Ax(t) - Bu(t))$ equals 0 in this direction. Given a sequence of $\lambda(t)$ and all the directions in which Q is singular, the way to construct an admissible sequence of adjoint variables is then to project $[\,(\lambda(t-1)^T - \lambda(t)^T A) \quad \lambda(t)^T B\,]^T$ onto the orthogonal subspace of these directions (meaning the range of Q). □

With a large-scale system and a long simulation horizon t_{final}, the minimization program is potentially huge. It turns out that there is an easy explicit solution.

Proposition 3
Assume that Q is positive definite. The minimal value (denoted $V_{\min}$) of the minimization program in (11) can be determined explicitly and is

$$
V_{\min} = x_0^T Q_x x_0 - 2\lambda(0)^T A x_0 - f^T Q_u^{-1} f - \sum_{t=1}^{t_{\text{final}}} g(t)^T Q^{-1} g(t)
$$

where

$$f = Q_{xu}^T x_0 - B^T \lambda(0)$$
$$g(t) = \begin{bmatrix} \lambda(t-1) - A^T \lambda(t) \\ -B^T \lambda(t) \end{bmatrix}$$

□

Proof. Introduce

$$F = Q_u u(0) + f$$
$$G(t) = Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} + g(t)$$

We get that

$$\begin{aligned}
&\sum_{t=0}^{t_{\text{final}}} \left(\begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} + 2\lambda(t)^T \left(x(t+1) - Ax(t) - Bu(t) \right) \right) \\
&\quad = \begin{bmatrix} x_0 \\ u(0) \end{bmatrix}^T Q \begin{bmatrix} x_0 \\ u(0) \end{bmatrix} - 2\lambda(0)(Ax_0 + Bu(0)) \\
&\qquad + \sum_{t=1}^{t_{\text{final}}} \left(\begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} + 2\lambda(t-1)x(t) - \lambda(t)(Ax(t) + Bu(t)) \right) \\
&\quad = x_0^T Q_x x_0 - 2\lambda(0)Ax_0 + F^T Q_u^{-1} F - f^T Q_u^{-1} f \\
&\qquad + \sum_{t=1}^{t_{\text{final}}} \left(G(t)^T Q^{-1} G(t) - g(t)^T Q^{-1} g(t) \right)
\end{aligned}$$

The equalities arise by inserting $x(0) = x_0$, using that $\lambda(t_{\text{final}}) = 0$ and completing the squares. To minimize the final expression, the squares $F^T Q_u^{-1} F$ and $G(t)^T Q^{-1} G(t)$ are set to 0 and the relation for $V_{\min}$ is reached. □

Remark 3
With similar reasoning as in Remark 2 it is also possible to deal with positive semi-definite Q. Then $Q^{-1}g(t)$ will mean any vector v satisfying $Qv = g(t)$ (this v will actually equal some minimizing $-[\,x(t)^T \quad u(t)^T\,]^T$). □

The Theorem 1 gives a method to evaluate the expected performance an updated feedback matrix will give to the system. We only have to choose

the dual or adjoint variables. The name suggest that we choose the adjoint variables defined by (7). To motivate this choice, we could refer to Pontryagin's maximum principle. The motivation comes from examining

$$\max_{\lambda}\min_{x,u}\underbrace{\sum_{t=0}^{t_{\text{final}}}\left(\begin{bmatrix}x(t)\\u(t)\end{bmatrix}^T Q\begin{bmatrix}x(t)\\u(t)\end{bmatrix}+2\lambda(t)^T(x(t+1)-Ax(t)-Bu(t))\right)}_{\mathcal{L}(x,u,\lambda)}$$

from Theorem 1. We let the objective function be denoted by $\mathcal{L}(x,u,\lambda)$. To find a saddle point for $\mathcal{L}$ then

$$\begin{aligned}0&=\nabla_{x(t)}\mathcal{L}=2(Q_x x(t)+Q_{xu}u(t)+\lambda(t-1)-A^T\lambda(t))\\0&=\nabla_{u(t)}\mathcal{L}=2(Q_{xu}^T x(t)+Q_u u(t)-B^T\lambda(t))\end{aligned}$$

We get (7) by $\nabla_{x(t)}\mathcal{L}+K^T\nabla_{u(t)}\mathcal{L}=0$ and replacing $u(t)=-Kx(t)$.

3. Synthesis of Distributed Controllers

To get a scalable scheme from the theory developed in the previous section, we need to impose structure on both the systems and the admissible controllers. Such structure is for example distributed systems, that is systems, which are partitioned in distinct subsystems, denoted *agents*. These agents are only connected to a few other agents. This connection has two meanings. First, the dynamics for each agent only directly depends on the states of the agents it is connected to. Secondly, each agent is only able to communicate with its connected agents, meaning that for example only measurements of the states from these agents can be used to determine the control signal. The connection between agents can be formulated by a directed graph structure. We will explain the details below.

3.1 Underlying Graph Structure

With a graph $\mathcal{G}$ we mean the pair $(\mathcal{V},\mathcal{E})$ of vertices and edges, respectively. The set of vertices $\mathcal{V}=\{v_1,v_2,\dots,v_n\}$ will later represent the partition of the system. Each vertex v_k will denote an agent of the system. The set of edges $\mathcal{E}$ is a collection of ordered pairs (v_i,v_k) (or in short (i,k)) which means that there is a connection starting from vertex v_i and ends in vertex v_k. We will require all considered edge sets to include the pairs (v_i,v_i) for all vertices $1\le i\le n$. For each vertex v_i we define its *up-neighbors* as $\mathcal{N}_i^{\text{up}}=\{v_k\mid(v_k,v_i)\in\mathcal{E},k\neq i\}$ and its *down-neighbors* as $\mathcal{N}_i^{\text{down}}=\{v_k\mid(v_i,v_k)\in\mathcal{E},k\neq i\}$. We will denote an agents *neighbors* by $\mathcal{N}_i=\mathcal{N}_i^{\text{up}}\cup\mathcal{N}_i^{\text{down}}$.

3.2 Modelling the Subsystems

As mentioned in the previous section, we consider LTI systems in discrete time (1) with the same assumptions as in this section. The systems will also be assumed to be distributed, a property described by an associated graph $\mathcal{G}$. The vertices of the graph constitute a partition of the states of (1). We assume that the states are ordered according to the vertices, i.e. we collect the states corresponding to v_1 in the beginning of the state vector x and so on. This gives us the state vector $x = [\,x_1^T \quad x_2^T \quad \dots \quad x_n^T\,]^T$, where each x_i is the states of vertex or agent v_i. Now, the distributed structure of the system can be described by the edges of $\mathcal{G}$. If there is an edge $(v_i, v_k) \in \mathcal{E}$, then agent v_i may directly influence agent v_k through the dynamics. That is, the dynamics matrix has a sparsity structure which resembles the graph associated with the distributed system

$$A_{ik} = 0 \quad \text{if} \quad (v_k, v_i) \notin \mathcal{E}$$

We will assume that the control inputs are partitioned over the agents, meaning that each agent has a distinct set of control signals which it uses to control its states. This set of control signals may not affect any other agent directly. The way this shows up in the dynamics equation is that the matrix B will be block diagonal,

$$B = \operatorname{diag}(B_1, B_2, \dots, B_n)$$

where B_i is the local B matrix for agent v_i.

Remark 4
The reason for assuming that the matrix B to be block-diagonal will become evident in a later section. This assumption can be relaxed to allow certain sparse matrices. □

The dynamics for each subsystem can then be written as

$$x_i(t+1) = A_{ii}x_i(t) + B_i u_i(t) + \sum_{k \in \mathcal{N}_i^{\text{up}}} A_{ik}x_k(t) \tag{13}$$

where $x_i(t) \in \mathbb{R}^{m_i}$ and $u_i(t) \in \mathbb{R}^{p_i}$.

A system with 4 agents can be found in Figure 1. For example, we have that

$$x_3(t+1) = A_{33}x_3(t) + B_3 u_3(t) + [A_{31}x_1(t) + A_{34}x_4(t)]$$

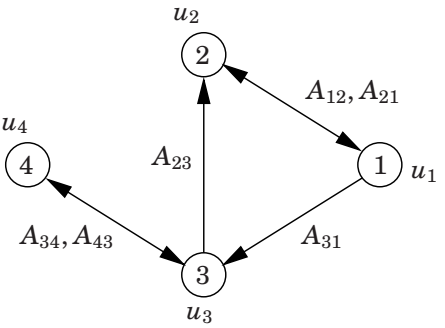


Figure 1. Graphical representation of a distributed system. The arrows shows how each agent affects the others. The set $\mathcal{E} = \{(1,1),(2,2),(3,3),(4,4),(1,2),(2,1),(1,3),(3,2),(3,4),(4,3)\}$

3.3 Problem Formulation

In section 2.1 we saw how to impose restrictions on the admissible controllers. This suits well with the idea of distributed controllers. We assume that an agent v_i may only use measurements from the states of its neighbors to determine its control input. In terms of the structure of the feedback matrix

$$K_{ik} = 0 \quad \text{if} \quad (v_i, v_k) \notin \mathcal{E}$$

This defines the set of admissible feedback matrices as

$$\mathcal{K} = \{K \mid K_{ik} = 0 \text{ if } (v_i, v_k) \notin \mathcal{E},\ \forall i,k\} \tag{14}$$

We will also introduce a restriction on the cost function in (3). The matrix Q will be assumed to have a structure which allows us to separate the cost function in a term for each agent, i.e.

$$J(K,x_0) = \sum_{i=1}^{n}\sum_{t=0}^{\infty} \begin{bmatrix} x_i(t) \\ u_i(t) \end{bmatrix}^T \underbrace{\begin{bmatrix} [Q_x]_i & [Q_{xu}]_i \\ [Q_{xu}]_i^T & [Q_u]_i \end{bmatrix}}_{Q_i} \begin{bmatrix} x_i(t) \\ u_i(t) \end{bmatrix}$$

This means that we assume that the matrices Q_x, Q_u and Q_{xu} are block diagonal. Hence, Q_x is positive semi-definite and Q_u is positive definite if and only if all $[Q_x]_i$ and $[Q_u]_i$ are positive semi-definite and positive definite, respectively.

REMARK 5
The reason for restricting Q to this structure is the same as for the restriction of the B matrix, and will become evident later on. As with the B matrix, the assumption can be relaxed to allow certain sparse Q. □

3.4 Scalable Calculations

With the distributed structure of the system and problem, i.e. the assumptions of the structure A and B in the dynamics equation and Q in the cost function, we will show that the algorithm in section 2.3 will be both scalable when considering computational time and modularized in the sense that computations can be partitioned. Specifically, we will show that the calculations of the part of the gradient in agent i only requires the dynamics matrix, states and adjoint states for neighboring agents, i.e. agents in $\mathcal{N}_i$.

In section 2.2 a descent direction is obtained by projecting the gradient onto the subspace $\mathscr{K}$. With the structure of $\mathscr{K}$ in (14), the only part of the gradient that needs to be determined is

$$\begin{aligned}[\nabla_K J]_{ik} &= 2\sum_{t=0}^{\infty}\left[\left(-Q_u u(t) - Q_{xu}^T x(t) + B^T\lambda(t)\right)x(t)^T\right]_{ik} \\ &= 2\sum_{t=0}^{\infty}\left(-[Q_u]_i u_i(t) - [Q_{xu}]_i^T x_i(t) + B_i^T\lambda_i(t)\right)x_k(t)^T\end{aligned}$$

for all $k \in \{i\}\cup\mathcal{N}_i^{\text{down}}$. Here we understand that for each agent i, we need to determine both its state evolution and its adjoint states evolution.

Obviously, by the restriction of A, B and K, an agent i uses only the states from neighboring agents to simulate its states x_i. By (7), the adjoint state evolution of agent i is

$$\begin{aligned}\lambda_i(t+1) = &\left[(A-BK)^T\lambda(t)\right]_i \\ &- \left[(Q_x - Q_{xu}K - K^TQ_{xu}^T + K^TQ_uK)x(t)\right]_i \quad (15)\end{aligned}$$

The first term of (15) simplifies to

$$\left[(A-BK)^T\lambda(t)\right]_i = [A-BK]_{ii}^T\lambda_i(t) + \sum_{k\in\mathcal{N}_i^{\text{down}}}[A-BK]_{ki}^T\lambda_k(t)$$

The second term of (15) becomes

$$\begin{aligned}&\left[(Q_x - Q_{xu}K - K^TQ_{xu}^T + K^TQ_uK)x(t)\right]_i \\ &\quad= [Q_x]_i x_i(t) + [Q_{xu}]_i u_i(t) - \sum_{k\in\{i\}\cup\mathcal{N}_i^{\text{down}}} K_{ki}^T\left([Q_{xu}]_k x_k(t) + [Q_u]u_k(t)\right)\end{aligned}$$

Now, with these expressions, algorithm 1 can be updated to

ALGORITHM 2
Consider a system (1) with control $u(t) = -Kx(t)$ where $K \in \mathcal{K}_{\text{stab}}$. To find a local minimizer to (3), start with $K^{(0)} \in \mathcal{K}_{\text{stab}}$ and for each $\tau \geq 0$,

1. Simulate the states of (1) with control $u(t) = -K^{(\tau)}x(t)$ for times $t \in [0, t_{\text{final}}]$.
2. Simulate the adjoint states of (7) for times $t \in [0, t_{\text{final}}]$ in the backwards time direction with $\lambda(t_{\text{final}}) = 0$.
3. For all agents i and all $k \in \mathcal{N}_i^{\text{down}}$,

   I) Calculate

$$G_{ik} = 2\sum_{t=0}^{t_{\text{final}}} \Big(-[Q_u]_i u_i(t) - [Q_{xu}]_i^T x_i(t) + B_i^T \lambda_i(t) \Big) x_k(t)^T$$

   II) Update the feedback matrix

$$K_{ik}^{(\tau+1)} = K_{ik}^{(\tau)} - \gamma_\tau G_{ik}$$

   for some step length γ_τ.
4. Increase τ with 1 and goto 1).

□

The operations that are performed for each agent in the algorithm only needs information of its neighbors meaning that the scheme is modularized. If, for all i, $|\mathcal{N}_i| \leq N_{\max} \ll n$, i.e. each agent only has a few number of neighbors, this also shows us that it is scalable, since the number of calculations required to obtain the descent direction is related to $n \cdot N_{\max}$.

REMARK 6
The reason for limiting the structure of B and Q becomes evident when investigating the scalability of the method. If B is not block-diagonal the matrix $A - BK$ does not have the same structure as A. The structure of Q will affect the structure of $(Q_x - Q_{xu}K - K^T Q_{xu}^T + K^T Q_u K)x(t)$. But if B and Q are chosen in a way such that the structures of these matrices are still sparse, the method will still be scalable. For example, if B and Q also have non-zero blocks for agents that are neighbors the sparsity structure still holds (when the number of neighbors for each agent is still a lot less than the total number of agents). □

We also notice that the procedure is robust in the aspect of adding new agents to the system. If the system is enlarged with new agents, only the calculations to agents in the neighborhood of where these new agents come in, are changed.

3.5 Scalable Suboptimality Bound Calculations

When determining the suboptimality bound using Proposition 3, the structure of the included matrices ensures a scalable method. The matrix inverses of Q and Q_u are solved efficiently using the block-diagonal structure of these matrices.

By exploiting the structure of the matrices of the minimization program of (11) of Theorem 1 for determining the suboptimality bound, it is easily seen that it can be separated into minimization programs for each subsystem. Each minimization program will have a number of decision variable in the order of the size of each subsystem.

4. Real-Time Synthesis of Distributed Controllers

In section 2 and 3 the focus was on finding a scheme for obtaining controllers solving the distributed deterministic linear quadratic control problem. The systems considered are deterministic and the scheme worked by simulating them offline. It turns out that by changing the model description to include noise, we can follow similar steps to find a scheme that solves the stochastic linear quadratic control problem (LQG) for a real plant in real-time. By using measured values of the states of the actual plant we can determine a descent direction of the feedback matrix. This means that we get a scheme that works in real-time to improve performance of a distributed system.

4.1 Restated System Model

We now consider the discrete time stochastic LTI system

$$x(t+1) = Ax(t) + Bu(t) + w(t) \tag{16}$$

where w is white noise with variance W, and $w(t)$ is independent of $x(s)$ for $s \leq t$. In all other aspects we use the same assumptions and notation as in Section 3.2, i.e. the system consists of n agents, and the connection of the agents is described by a graph $\mathcal{G}$, the structure of the matrices in (16) are determined by the edges of $\mathcal{G}$ and so on.

In this stochastic setting the cost function will now be

$$J(K) = \mathbf{E} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \tag{17}$$

where $x(t)$ satisfies (16) and $u(t) = -Kx(t)$.

4.2 Analysis

Similar to the analysis in section 2 we state the following two propositions, without proofs.

PROPOSITION 4
Given the stationary stochastic process (16) where $u(t) = -Kx(t)$ for a stabilizing K and where w is white noise with covariance W. Then $J(K)$, defined by (17), has the gradient

$$\nabla_K J = 2\left[Q_u K - Q_{xu}^T - B^T P(A - BK)\right] X \tag{18}$$

where X and P satisfy the Lyapunov equations

$$X = (A - BK)X(A - BK)^T + W \tag{19}$$

$$P = (A - BK)^T P(A - BK) + Q_x - Q_{xu}K - K^T Q_{xu}^T + K^T Q_u K \tag{20}$$

□

PROPOSITION 5
Under the conditions of Proposition 4, consider the stationary stochastic process λ defined by the backwards iteration

$$\lambda(t-1) = (A - BK)^T \lambda(t) - (Q_x - Q_{xu}K - K^T Q_{xu}^T + K^T Q_u K)x(t) \tag{21}$$

where $x(t)$ are the states of the original system. Then

$$\nabla_K J = 2\left((Q_u K - Q_{xu}^T)\mathbf{E}\, xx^T + B^T \mathbf{E}\, \lambda x^T\right)$$

□

4.3 Real-Time Scheme

Similar to section 3.4 we use proposition 5 to form an online scheme to update the feedback matrix K to improve the performance given in (17) while $K \in \mathcal{K}_{\text{stab}}$. The difference is that instead of simulating the states of the system, we collect measurement of the states and use them to simulate the adjoint state equations.

ALGORITHM 3

Consider a system (16) with control $u(t) = -Kx(t)$ where $K \in \mathscr{K}_{\text{stab}}$. To iteratively improve the performance (17) and approach a local minimizer, at time t_τ, let the feedback matrix be $K^{(\tau)}$, and in each agent i:

1. Measure the state $x_i(t)$ of the agent and collect measurements of the states and control signals of the agent's neighbors, for times $t = t_\tau, \ldots, t_\tau + N$.
2. Simulate the adjoint states $\lambda_i(t)$ of the system (21) for times $t = t_\tau, \ldots, t_\tau + N$ in the backwards direction, by communicating adjoint states from and to neighboring agents.
3. For every $j \in \{i\} \cup \mathcal{N}_i^{\text{down}}$, calculate the estimates of $\mathbf{E}\, u_i x_j^T$ and $\mathbf{E}\, \lambda_i x_j^T$ by

$$\left(\mathbf{E}\, x_i x_j^T\right)_{\text{est}} = \frac{1}{N} \sum_{t=t_\tau}^{t_\tau+N} x_i(t) x_j(t)^T$$

$$\left(\mathbf{E}\, u_i x_j^T\right)_{\text{est}} = \frac{1}{N} \sum_{t=t_\tau}^{t_\tau+N} u_i(t) x_j(t)^T$$

$$\left(\mathbf{E}\, \lambda_i x_j^T\right)_{\text{est}} = \frac{1}{N} \sum_{t=t_\tau}^{t_\tau+N} \lambda_i(t) x_j(t)^T$$

4. For every $j \in \{i\} \cup \mathcal{N}_i^{\text{down}}$ the estimate of the i,j-block of the gradient becomes

$$G_{ij} = -2\Big([Q_u]_i \left(\mathbf{E}\, u_i x_j^T\right)_{\text{est}} - [Q_{xu}]_i^T \left(\mathbf{E}\, x_i x_j^T\right)_{\text{est}} + B_i^T \left(\mathbf{E}\, \lambda_i x_j^T\right)_{\text{est}} \Big)$$

5. For every $j \in \{i\} \cup \mathcal{N}_i^{\text{down}}$, update $K_{ij}^{(\tau+1)} = K_{ij}^{(\tau)} - \gamma_\tau G_{ij}$ for some step length γ_τ.
6. Let $t_{\tau+1} = t_\tau + N$, increase k by one and go to 1).

□

5. Numerical Examples

Two numerical examples will be investigated. The first is a small-scale example which will illustrate the method and compare it to the optimal LQR solution in order to show the convergence and suboptimality of the method. In the second example we examine the scalability of Algorithm 2 by using it on a sequence of large-scale systems with increasing size.

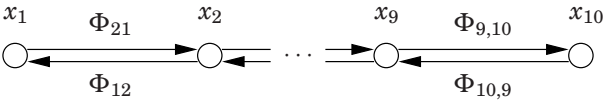


Figure 2. Graphical representation of the system in Example 5.1. The arrows shows how each agent affects the others.

5.1 Small-scale Example

The system

$$x(t+1) = Ax(t) + Bu(t)$$

that is considered, consists of 10 agents, each with one state, where the agents are connected in a linear fashion, see Figure 2. This leads to a tri-diagonal dynamics matrix, which, in this example, is

$$A = \begin{bmatrix} 0.5 & 0.5 & & & & & & & & \\ -0.5 & 0.1 & -0.3 & & & & & & & \\ & 0.4 & -0.2 & -0.5 & & & & & & \\ & & -0.4 & -0.5 & 0.2 & & & & & \\ & & & 0.2 & 0.3 & -0.1 & & & & \\ & & & & -0.3 & 0.1 & 0.3 & & & \\ & & & & & 0.2 & -0.4 & -0.4 & & \\ & & & & & & 0.2 & -0.2 & 0.3 & \\ & & & & & & & 0.5 & -0.5 & 0.3 \\ & & & & & & & & -0.1 & -0.1 \end{bmatrix}$$

and with the remaining entries equal to zero. We allow each agent to have an input and set $B = I$. We wish to minimize the cost

$$J(K, x_0) = \sum_{t=0}^{t_{\text{final}}} \left(x(t)^T Q_x x(t) + u(t)^T Q_u u(t) \right)$$

where $u = -Kx$, $Q_x = Q_u = I$ and $x_0 \in \mathcal{N}(0, I)$.

The magnitude of the maximal eigenvalue of A, $\rho(A) \approx 0.81$, hence we can initially let the system be uncontrolled, i.e. let $K = 0$. The algorithm is used for 50 iterations where the systems are simulated for times $t = 0, \ldots, 10$ in each iteration. The step length is constant in all iterations, and $\gamma_\tau = 10^{-2}$. The method for estimating suboptimality bounds is done in each update iteration. The result is given in Figures 3-4.

In Figure 3 the estimated suboptimal bound is denoted by α and shown in blue. A first remark is that the in the first iteration we get a negative value of the suboptimality bound. This is due to the fact that minimization program in (11) is not guaranteed to give a positive value. In case a negative value is obtained nothing can really be said about the suboptimality. Though, as we get closer to the optimal feedback matrix, the adjoint trajectory will approach the optimal (with respect to the Lagrangian in

Pontryagin's maximum principle) and the inequalities in the proof of Theorem 1 will almost be equal implying that we can expect a positive value from (11).

When positive, the suboptimality bound is always larger than 1 which is natural. In the same figure denoted by α_{exact} and shown in green, is the true suboptimality determined by

$$\alpha_{\text{exact}} = \frac{J(K^{(k)}, x_0)}{J(K_{\text{opt}}, x_0)}$$

As expected the suboptimality bound (when positive) is also always larger than the true suboptimality. As the true suboptimality approaches 1, that is the cost with the feedback matrix approaches the optimal cost, the suboptimality bound also approaches 1.

In Figure 4 the relative difference between α and α_{exact} is shown, that is

$$\Delta\alpha_{\text{rel}} = \frac{\alpha - \alpha_{\text{exact}}}{\alpha_{\text{exact}} - 1}$$

Hence, the relative difference measures how far the suboptimality bound is from the true suboptimality compared to the distance from the true suboptimality to the optimal value 1. When the suboptimality bound is positive the plot shows us that the relative difference is below 3.5. The relative distance decreases and when the true suboptimality approaches 1 the relative difference is below 1.5, meaning that the suboptimality bound does not underestimates the true suboptimality by more than a factor of 1.5. This shows that the suboptimality bound is a descent measure of the suboptimality and can hence be used for a stopping criterion for the method.

5.2 Large-scale Example

In this example a sequence of systems with increasing size will be examined. The systems will be randomly generated by some rules described below. For one system in the sequence with n agent

- Each agent will consist of 10 states. The part of dynamics matrix A_{ii} will be uniformly drawn and scaled to have the largest eigenvalue equal 0.7.
- Each agent will have one control signal. The part of control matrix, B_i will be uniformly drawn from $[-1, 1]^{10}$.
- The adjacency matrix will be randomly generated, to give all agents 5 neighbors and a connected graph. The adjacency matrix will be symmetric, hence a undirected graph.

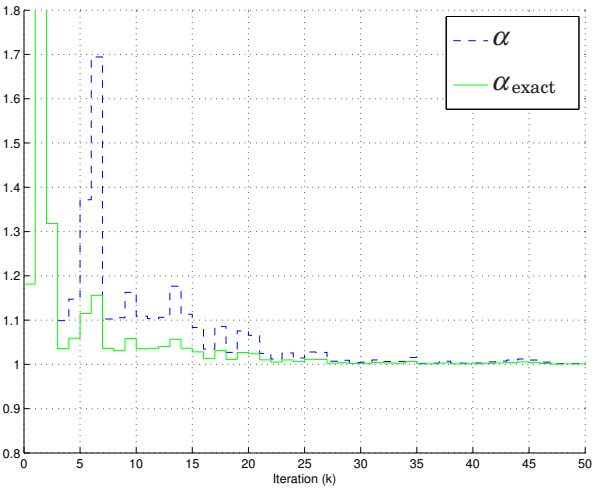


Figure 3. Example 5.1: Plots of the estimated suboptimality using the described method (in blue) and the exact suboptimality (in green).

- The neighbors affect the agent by $A_{ik} = b \cdot c^T$ where b and c are uniformly drawn from $[-0.22, 0.22]^{10}$.

The resulting systems not guaranteed to be stable, but the parameters are chosen to give stable systems in most cases. If a system is not stable it is regenerated until it is.

The number of agents for the systems will be $n = 50k$ for $k = 1, \ldots, 20$. Hence, the size of the smallest system will be 500 states and for the largest 1000 states. For all systems, the weights $Q_x = I$ and $Q_u = I$ (where I has the appropriate size). In each update iteration of Algorithm 2, the systems will be simulated for times $t = 1, \ldots, 20$. The step length $\gamma_\tau = 5 \cdot 10^{-4}$ in every iteration for every system.

When the method is applied to a system, it is keeps iterating to update the feedback matrices until the suboptimality bound is below 1.01 in 20 consecutive update iterations. The resulting computation times for completing the method is shown in Figure 5. We find the computation times for the method in blue. It shows that the complexity of the method is linear when we increase the number of agents. This should be compared the curve in green corresponding to the time required to determine the centralized optimal solution by solving the Riccati equation. This solution is only determined for the systems with $n \leq 200$ due to time requirements as well as workspace requirements. If we exclude the workspace requirement, and fit a third order polynomial to the curve (since the com-

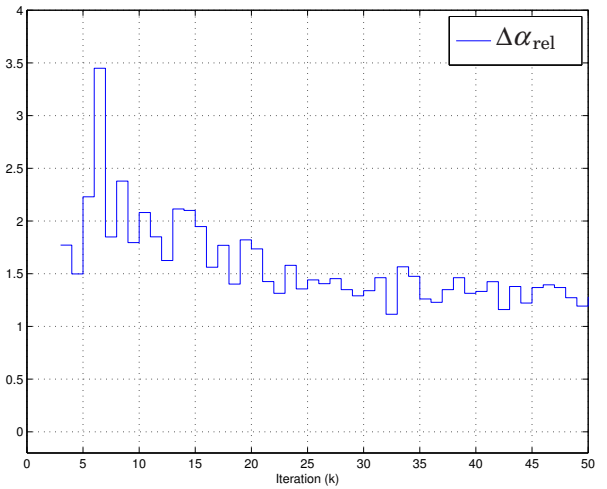


Figure 4. Example 5.1: Plot of the relative difference between the estimated and the exact suboptimality.

putational complexity should be $O(n^3)$), the expected time to complete the calculations for the system with $n = 1000$ would take more than 55 hours (but the workspace requirement would not even permit these calculations). It should be noted that it is only a coincidence that both computation times for $n = 50$ are approximately equal.

The number of update iterations in Algorithm 2 needed vary between 283 and 343. The fact that the number of iterations are almost constant is the reason why the method has linear complexity with respect to the number of agents. The number of iterations is actually decreasing for the sequence, the more agents generally means that fewer iterations were needed.

6. Conclusions and Future Works

6.1 Conclusions

We have shown a method for finding structured linear controllers to improve the LQR or LQG performance criterion. Given an initial stabilizing controller for a linear system and a cost function, the method works by iteratively updating the controller in a descent direction to reduce the cost. By simulating the system controlled by the current controller and

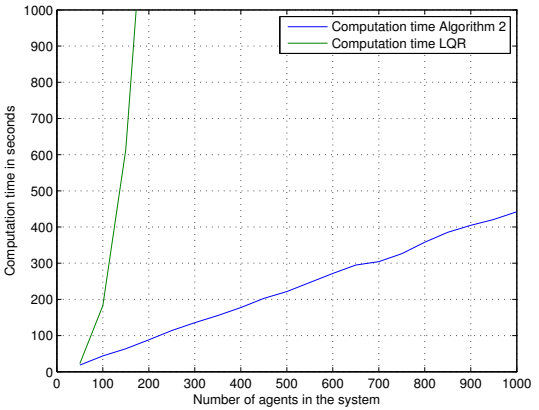


Figure 5. Example 5.2: Plot of the computation time with respect to the size of the system.

the corresponding adjoint state equation, the descent direction are given by a relation which involve multiplying the trajectories together. The only operations performed are matrix or vector multiplications, implying that when sparse matrices constitutes the system, the method will be scalable. The same trajectories determined for the evaluating the descent direction, are also used to calculate a suboptimality bound for the current controller. This bound validates the performance of the controllers and can be used as a stopping criteria for the iteration process.

6.2 Future Works

In [Mårtensson and Rantzer, 2011] a similar method for designing Kalman filters was presented. When these works are connected we get a scheme for output feedback synthesis. An interesting question is if the suboptimality bounds still are valid in this setting.

An example of a large-scale system is the large deformable mirrors presented in for example [Heimsten, 2011]. The system has more than 10000 states and the system matrices are of a sparse nature. Hence this system is suitable for the methods presented here, and it will be investigated the improvements that can be made to the controller given in [Heimsten, 2011].

References

Amodei, L. and J. Buchot (2010): "An invariant subspace method for large-scale algebraic Riccati equation." *Applied Numerical Mathematics*, **60:11**, pp. 1067–1082.

Åström, K. J. (2006): *Introduction to Stochastic Control Theory*. Dover, New York.

Bamieh, B., F. Paganini, and M. A. Dahleh (2002): "Distributed control of spatially invariant systems." *IEEE Transactions on Automatic Control*, **47:7**, pp. 1091–1107.

Bamieh, B. and P. G. Voulgaris (2005): "A convex characterization of distributed control problems in spatially invariant systems with communication constraints." *Systems & Control Letters*, **54:6**, pp. 575–583.

Benner, P. and H. Faßbender (2011): "On the numerical solution of large-scale sparse discrete-time Riccati equations." *Advances in Computational Mathematics*, **35:2**, pp. 119–147.

Benner, P., J.-R. Li, and T. Penzl (2008): "Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems." *Numerical Linear Algebra with Applications*, **15**, November, pp. 755–777.

Bryson, A. E. and Y.-C. Ho (1975): *Applied Optimal Control*. Hemisphere Pub. Corp.

Gattami, A. (2006): "Generalized linear quadratic control theory." In *Proceedings of the 45th IEEE Conference on Decision and Control*. San Diego, CA.

Geromel, J. and J. Bernussou (1982): "Optimal decentralized control of dynamic systems." *Automatica*, **18:5**, pp. 545–557.

Heimsten, R. (2011): *Study of a Large Deformable Mirror Concept*. PhD thesis, Department of Astronomy and Theoretical Physics, Lund University, Sweden.

Heinkenschloss, M. (2005): "A time-domain decomposition iterative method for the solution of distributed linear quadratic optimal control problems." *Journal of Computational and Applied Mathematics*, **173:1**, pp. 169–198.

Ikeda, M., D. Siljak, and K. Yasuda (1983): "Optimality of decentralized control for large-scale systems." *Automatica*, **19:3**, pp. 309–316.

Lancaster, P. and L. Rodman (1995): *Algebraic Riccati Equation*. Oxford University Press.

Langbort, C., R. S. Chandra, and R. D´Andrea (2004): "Distributed control design for systems interconnected over an arbitrary graph." *IEEE Transactions on Automatic Control*, **49:9**, pp. 1502–1519.

Lasdon, L. S. (2002): *Optimization Theory for Large Systems*. Dover Publications Inc.

Lasiecka, I. and A. Tuffaha (2009): "Riccati theory and singular estimates for a Bolza control problem arising in linearized fluid structure interaction." *Systems and Control Letters*, **58:7**, pp. 499–509.

Li, L. and F. Paganini (2006): "LMI relaxation to Riccati equations in structured $\mathcal{H}_2$ control." In *Proceedings of the 2006 American Control Conference*, pp. 644–649.

Mårtensson, K. and A. Rantzer (2009): "Gradient methods for iterative distributed control." In *Proceedings of the 48th IEEE Conference on Decision and Control*. Shanghai, China.

Mårtensson, K. and A. Rantzer (2010): "Sub-optimality bound on a gradient method for iterative distributed control synthesis." In *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems*. Budapest, Hungary.

Mårtensson, K. and A. Rantzer (2011): "A scalable modularized synthesis method for distributed Kalman filters." In *Proceedings of the 18th IFAC World Congress*. Milano, Italy.

Rantzer, A. (2006): "A separation principle for distributed control." In *Proceedings of the 45th IEEE Conference on Decision and Control*. San Diego, CA.

Rao, X., K. Gallivan, and P. Van Dooren (2000): "Riccati equation-based stabilization of large scale dynamical systems." In *Proceedings of the 39th IEEE Conference on Decision and Control*, vol. 5, pp. 4672–4677.

Rotkowitz, M. and S. Lall (2002): "Decentralized control information structures preserved under feedback." In *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 1, pp. 569–575.

Rotkowitz, M. and S. Lall (2006): "A characterization of convex problems in decentralized control." *IEEE Transactions on Automatic Control*, **51:2**, pp. 274–286.

Staffans, O. (1996): "On the discrete and continuous time infinite-dimensional algebraic Riccati equations." *Systems and Control Letters*, **29:3**, pp. 131–138.

Witsenhausen, H. S. (1968): "A counterexample in stochastic optimum control." *SIAM Journal on Control*, **6:1**, pp. 138–147.

Paper II

Synthesis of Structured Output Feedback Controllers for Large-Scale Systems

Karl Mårtensson and Anders Rantzer

Abstract

In this paper we present a scheme to find structured output feedback controllers for large-scale systems. When the size of the state space grows beyond a moderate size, classical methods for LQG and $\mathcal{H}_\infty$ soon becomes intractable due to both computational time and memory requirements. Moreover, it is not possible to treat controllers with interconnection constraints. We address these problems by suggesting gradient methods that update the controllers iteratively. The gradient directions are determined using trajectories from simulations. With the same trajectories, it is also shown how to determine bounds of the distance from optimal performance. By exploiting sparsity structure in the systems, it can be shown that both the computational time and memory requirement for the gradient iterations grow linearly with the number of state interconnections, making them suitable for large-scale systems.

1. Introduction

The theory of optimal linear quadratic Gaussian (LQG) control has been around for several decades and is well documented in the literature, for example [Åström, 2006, Bryson and Ho, 1975, Lancaster and Rodman, 1995, Zhou *et al.*, 1996]. The solution is closely related to solving two algebraic Riccati equations to get the expressions for the optimal feedback matrix and the optimal estimation matrix. Using the separation principle, these matrix gains are combined to an optimal output feedback controller.

Solving the algebraic Riccati equation or the related Lyapunov equation have in both cases memory requirement complexity $O(n^2)$ and computational complexity $O(n^3)$, see [Bini *et al.*, 2011] or [Benner *et al.*, 2008]. Due to this fact, it is reasonable to solve small to medium size LQG control problem with the use of the solution of the algebraic Riccati equation. However, for large-scale systems, it is not computationally possible to compute this solution. When solving general large-scale optimization problems, conventional approaches can not be used. Instead one needs to find and exploit some structure in the problem, [Lasdon, 2002]. One such structure that often turns up when dealing with large-scale systems is that the system matrices are very sparse. This is the case for example with large distributed systems and systems arising from discretization of partial differential equations (PDEs), for example [Heimsten, 2011, Lasiecka and Tuffaha, 2009, Staffans, 1996]. Even though discretized PDEs have a large state space, the number of inputs and outputs are not changed and are typically not many. Such systems are treated in the papers [Benner *et al.*, 2008, Benner and Faßbender, 2011] by approximating the solution to the Riccati equation with low rank Cholesky factors. The feedback and predictor matrix are determined by a tractable product of these factors. Another low rank approximation method based on stable invariant subspaces of the Hamiltonian is given in [Amodei and Buchot, 2010]. A Riccati based method for stabilization of large-scale systems can be found in [Rao *et al.*, 2000].

In distributed systems a number of components, subsystems or agents, are joined to form an interconnected system. A distributed system, even if the size is moderate, produces another problem for the usual solution the LQG problem, namely that the interconnection may impose restrictions of the information available for control in each subsystem. Already in the late 1960s it was pointed out that such problems are fundamentally difficult to solve. In particular, [Witsenhausen, 1968] showed that even a small quadratic control problem does not have a linear optimal solution. In [Ikeda *et al.*, 1983] decentralized control, that is, when there is no communication in between agents, for large-scale interconnected systems is investigated with respect to stability and optimality. Decentralized Kalman

filtering for data fusion in sensor networks has been examined in [Rao *et al.*, 1993]. In [Hodzic and Siljak, 1985] the estimation and control problem are examined for large-scale systems with a hierarchical structure. The procedure works by designing LQG controllers piece-by-piece, giving a structured controller. Both stability and degree of suboptimality are investigated. In recent years a lot of work has been put into finding structures of the systems and the controllers that for which there are methods to find the optimal solution. A concept of quadratic invariance for the set of admissible controllers was introduced in [Rotkowitz and Lall, 2006, Rotkowitz and Lall, 2002], by which the problem of finding an optimal controller is transformed into a convex optimization program. In [Bamieh *et al.*, 2002, Bamieh and Voulgaris, 2005] spatially invariant systems, meaning that a translation in spatial coordinates does not affect the dynamics, are investigated. They show for example that such controllers are distributed and spatially localized. A structured state and output feedback $\mathcal{H}_2$ control synthesis method by relaxing the Riccati equations via linear matrix inequalities was discussed in [Li and Paganini, 2006]. Other methods which involve the use of linear matrix inequalities were presented in for example [Gattami, 2006, Langbort *et al.*, 2004, Rantzer, 2006]. A distributed estimation problem is solved in [Spanos *et al.*, 2005, Olfati-Saber, 2007] by using consensus techniques to accomplish the data fusion. Algorithms where each agent only determines a local state estimate of a large-scale, sparse system were presented in [Khan and Moura, 2008]. Here, neither the full state nor the full system model is known at any node of the system. Examples of applications with systems of large state space and decentralized control can be found in [Stankovic *et al.*, 1999, Stipanovic *et al.*, 2004] in where the control problem is solved using the stochastic inclusion principle [Hodzic *et al.*, 1983].

The focus of this paper is on finding linear structured output feedback controllers to minimize an LQG cost. The underlying method in the schemes presented is to update the controller matrix gains in descent directions in an iterative fashion to improve the cost. A similar technique has been studied in [Geromel and Bernussou, 1982] where the descent direction was determined by calculating solutions to Riccati equations. However, in this setting the method is not applicable for large-scale systems. In this work we will see how to circumvent this problem. Other descent schemes have been used in LQG design. A method to find low-order LQG controllers is presented in [Harn and Kosut, 1993]. In [Beseler *et al.*, 1992] it is shown how to use Newton's method to solve constrained LQG control problems. An iterative scheme to find locally-optimal solutions to constrained nonlinear stochastic systems by using iterative linearizations around the current trajectory, can be found in [Todorov and Li, 2005].

In this work we determine the descent direction by simulating the dy-

namical system and the corresponding adjoint system. This idea is used both for the separate state feedback and state estimation problem, or for the complete system controlled with the output feedback controller. The simulated trajectories are combined to find the descent direction of the given LQG cost. It turns out that when the dynamical system is sparse this procedure will provide a scalable method. When treating the state feedback and state estimation problem separately, the trajectory determined for calculating a descent direction can also be used to find a bound of the degree of suboptimality of the current controller. This gives a way to verify that the controller is close to the optimal solution. Hence, the bound can be used as a criterion when to stop the process of updating the controller to improve the performance. The paper extends the results from the recent papers [Mårtensson and Rantzer, 2011b, Mårtensson and Rantzer, 2010, Mårtensson and Rantzer, 2011a].

In the remainder of this section some mathematical notation will be introduced and the problem formulation will be described. Section 2 shows how to determine suboptimal output feedback controllers in by regarding the state prediction and state feedback as separate problems. This section also contains how the degree of suboptimality of each separate problem can be combined to a degree of suboptimality of the complete system. In Section 3 a method to treat the problem of finding the optimal output feedback controller in a simultaneous fashion will be given. In Section 4 we discuss the computational time and memory requirement for the methods. We present two numerical examples in Section 5. Finally, Section 6 gives come concluding remarks and future works.

1.1 Mathematical Notation

The set of real numbers is denoted by $\mathbb{R}$, real vectors of dimension n by $\mathbb{R}^n$ and real $n \times m$ matrices by $\mathbb{R}^{n\times m}$. The matrix I refers to the identity matrix with the appropriate size. The matrix $0^{n\times m}$ will refer to the zero matrix in $\mathbb{R}^{n\times m}$ and will be used when it is important to stress the size of such matrix. When two symmetric matrices X and Y are involved, the relation $X \geq Y$ refers to inequality in a positive definite meaning. When a partition of a vector or a matrix exists, subscripts will refer to that partition. For example, for $x \in \mathbb{R}^n$ then $x_i \in \mathbb{R}^{n_i}$ refers to the ith partition of x and $A \in \mathbb{R}^{n\times m}$ then $A_{i,j} = A_{ij} \in \mathbb{R}^{n_i \times m_j}$ refers to the i,jth partition of A. When the components of the vectors or matrices are ordered with respect to the partition, x_i or A_{ij} equivalently means the ith or i,jth *block* of x or A, respectively. For a matrix valued function $f : \mathbb{R}^{n\times m} \to \mathbb{R}$ we define the differential df as the part of $f(X + dX) - f(X)$ that is linear in dX, that is the linearized part of f. The gradient of f with respect to

X is denoted $\nabla_X f$ and means

$$\nabla_X f = \begin{bmatrix} \frac{\partial f}{\partial X_{1,1}} & \frac{\partial f}{\partial X_{1,2}} & \cdots & \frac{\partial f}{\partial X_{1,m}} \\ \frac{\partial f}{\partial X_{2,1}} & \frac{\partial f}{\partial X_{2,2}} & \cdots & \frac{\partial f}{\partial X_{2,m}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial X_{n,1}} & \frac{\partial f}{\partial X_{n,2}} & \cdots & \frac{\partial f}{\partial X_{n,m}} \end{bmatrix}.$$

For a pair of $(A, B) \in \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times m}$ we say that the matrix $K \in \mathbb{R}^{m\times n}$ stabilizes (A, B) if $A - BK$ has all its eigenvalues in the unit circle. A pair (A, B) is said to be stabilizable if such K exists. Similarly, the pair $(A, C) \in \mathbb{R}^{n\times n} \times \mathbb{R}^{p\times n}$ is said to be detectable if there exist $L \in \mathbb{R}^{n\times p}$ such that $A - LC$ has all its eigenvalues in the unit circle.

We use the notation $\mathcal{N}(\mu, \sigma^2)$ for Gaussian noise with mean μ and variance σ^2.

1.2 Problem Formulation

In this paper we consider linear time invariant systems in discrete time. Hence, the dynamic equation for these systems is

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + w(t), \\ y(t) &= Cx(t) + e(t), \end{aligned} \tag{1}$$

where the time $t \geq 0$ and $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$ and $y(t) \in \mathbb{R}^p$. The signals $w(t) \in \mathbb{R}^n$ and $e(t) \in \mathbb{R}^p$ are Gaussian noise with the variance

$$\mathbf{E} \begin{bmatrix} w(t) \\ e(t) \end{bmatrix} \begin{bmatrix} w(t) \\ e(t) \end{bmatrix}^T = \begin{bmatrix} R_w & R_{we} \\ R_{we}^T & R_e \end{bmatrix} = R.$$

Now, R is assumed to be positive semi-definite and R_e positive definite. The matrices are $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ and $C \in \mathbb{R}^{p\times n}$. We assume that (A, B) is stabilizable and that (A, C) is detectable in the paper.

The system is to be controlled with an output feedback controller, that is, a controller that in time t uses the available measurement trajectory, y, to determine the control $u(t)$. If the controller is not allowed to have a direct term, that is, it has to be strictly proper, the measurements that are available in time t will be $\{y(t-1), y(t-2), \ldots\}$. Otherwise, if a direct term is allowed, the measurements $\{y(t), y(t-1), \ldots\}$ are available. The objective is to construct a controller that minimizes a quadratic expression in the system states and control variables, that is, an expression

involving the variance of the system states and control variables. This means that we are looking at linear quadratic Gaussian (LQG) control design. The optimal controller solving this problem is linear and consists of a linear state estimator and a linear feedback law using the estimated states to determine the control. This will be the parametrization of the controllers used in this paper. Specifically, if no direct term is allowed in the controller, we consider

$$\begin{aligned} \hat{x}(t+1) &= A_{KL}\hat{x}(t) + Ly(t), \\ u(t) &= -K\hat{x}(t). \end{aligned} \tag{2}$$

Otherwise, if a direct term is allowed, the controllers are on the form

$$\begin{aligned} \hat{x}(t+1) &= (A_{KL} + BL_0C)\hat{x}(t) + (L - BL_0)y(t), \\ u(t) &= -(K - L_0C)\hat{x}(t) - L_0y(t), \end{aligned} \tag{3}$$

where $A_{KL} = A - BK - LC$. The matrix gains that are to be determined are the feedback matrix $K \in \mathbb{R}^{m\times n}$, the predictor matrix $L \in \mathbb{R}^{n\times p}$ and in case of direct term, the matrix $L_0 \in \mathbb{R}^{m\times p}$.

The difference from the usual LQG design is that a structure is imposed on the admissible controllers. The admissible controller gains are restricted to given the subspaces $\mathscr{K} \subset \mathbb{R}^{m\times n}$, $\mathscr{L} \subset \mathbb{R}^{n\times p}$ and $\mathscr{L}_0 \subset \mathbb{R}^{m\times p}$, such that $K \in \mathscr{K}$, $L \in \mathscr{L}$ and $L_0 \in \mathscr{L}_0$. Also, let the set of admissible stabilizing feedback matrices be denoted by

$$\mathscr{K}_{\text{stab}} = \{K \mid K \in \mathscr{K} \text{ and } A - BK \text{ is stable}\}$$

and all the stabilizing predictor matrices be

$$\mathscr{L}_{\text{stab}} = \{L \mid L \in \mathscr{L} \text{ and } A - LC \text{ is stable}\}.$$

For $K \in \mathscr{K}_{\text{stab}}$, $L \in \mathscr{L}_{\text{stab}}$ and potentially any $L_0 \in \mathscr{L}_0$ if direct term is allowed, the system (1) controlled with either (2) or (3) is stable.

With the dynamics in (1) with either the control in (2) or (3) in mind, for every $K \in \mathscr{K}_{\text{stab}}$, $L \in \mathscr{L}_{\text{stab}}$ and possibly every $L_0 \in \mathscr{L}_0$, we define the linear quadratic cost function, when the process is in stationarity,

$$J(K, L, L_0) = \mathbf{E}\begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T \underbrace{\begin{bmatrix} Q_x & Q_{xu} \\ Q_{xu}^T & Q_u \end{bmatrix}}_{Q} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}, \tag{4}$$

where Q is positive semi-definite and Q_u is positive definite. In case no direct term is allowed we instead write $J(K, L)$. Defining $\tilde{x} = x - \hat{x}$, the

resulting closed loop system becomes

$$\begin{bmatrix} x(t+1) \\ \tilde{x}(t+1) \end{bmatrix} = A_{\text{cl}} \begin{bmatrix} x(t) \\ \tilde{x}(t) \end{bmatrix} + \mathcal{L} \begin{bmatrix} w(t) \\ e(t) \end{bmatrix}, \tag{5}$$

where

$$\begin{aligned} A_{\text{cl}} &= \begin{bmatrix} A - BK & BK \\ 0^{n\times n} & A - LC \end{bmatrix}, \\ \mathcal{K} &= \begin{bmatrix} I & 0^{n\times n} \\ -K & K \end{bmatrix}, \\ \mathcal{L} &= \begin{bmatrix} I & 0^{n\times p} \\ I & -L \end{bmatrix} \end{aligned} \tag{6}$$

if no direct term is allowed or otherwise

$$\begin{aligned} A_{\text{cl}} &= \begin{bmatrix} A - BK & B(K - L_0 C) \\ 0^{n\times n} & A - LC \end{bmatrix}, \\ \mathcal{K} &= \begin{bmatrix} I & 0^{n\times n} \\ -K & K - L_0 C \end{bmatrix}, \\ \mathcal{L} &= \begin{bmatrix} I & -BL_0 \\ I & -L \end{bmatrix}. \end{aligned} \tag{7}$$

The cost in (4) can be expressed using either of the solutions to the following Lyapunov equations

$$P_{\text{cl}} = A_{\text{cl}} P_{\text{cl}} A_{\text{cl}}^T + \mathcal{L} R \mathcal{L}^T, \tag{8}$$

$$S_{\text{cl}} = A_{\text{cl}}^T S_{\text{cl}} A_{\text{cl}} + \mathcal{K}^T Q \mathcal{K}. \tag{9}$$

Then we have that

$$\begin{aligned} J(K, L, L_0) &= \operatorname{tr}\left(P_{\text{cl}} \cdot \mathcal{K}^T Q \mathcal{K}\right) + \operatorname{tr}\left(L_0 R_e L_0^T\right) \\ &= \operatorname{tr}\left(S_{\text{cl}} \cdot \mathcal{L} R \mathcal{L}^T\right) + \operatorname{tr}\left(L_0 R_e L_0^T\right), \end{aligned} \tag{10}$$

where the last term disappears in both expressions when there is no direct term allowed.

Now, the objective for the control synthesis is then to find the admissible stabilizing feedback, predictor and possibly the L_0 matrix that minimizes $J(K, L)$ or $J(K, L, L_0)$. It should be noted that the structured optimal controller does not have to satisfy this structure, see for example [Swigart and Lall, 2011]. Hence, the aim of this paper is to find the optimal structured controller parametrized by (2) or (3).

2. Separate Synthesis Scheme

Finding the ordinary optimal LQG controller involves solving two separate problems, finding the optimal state feedback and finding the optimal state predictor. The separation principle allows us to combine these solutions into the optimal LQG controller. With this in mind, in this section we propose to treat the structured LQG problem in a similar way, that is, finding a structured state feedback and a structured state predictor separately and then combine them giving the controller structure in (2).

In this section we show how to iteratively determine a structured state predictor in a way that reduces a given cost. At the same time, a bound on the degree of suboptimality of the current predictor is determined, which is the ratio between the current and the optimal cost. This measure evaluates the current matrix and for example allows us to decide at what degree of suboptimality to terminate the iterative procedure. Determining the structured state feedback is done in a similar fashion and only the results are given. A more involved discussion of determining the structured state feedback can be found in [Mårtensson and Rantzer, 2011b, Mårtensson and Rantzer, 2010].

2.1 State Predictor Synthesis

In this section we do not allow a direct term in the resulting controller and hence the gain $L_0 = 0$ is omitted from the calculations.

Letting $u \equiv 0$ in (1), we consider an observer on the form

$$\hat{x}(t+1) = A\hat{x} + L(y(t) - C\hat{x}(t)). \tag{11}$$

The dynamics equation for the error $\tilde{x} = x - \hat{x}$ is given by the equation

$$\tilde{x}(t+1) = (A - LC)\tilde{x}(t) + w(t) - Le(t). \tag{12}$$

It is a well-known fact that there is a matrix L (without any structural restrictions) which minimizes the cost $\mathbf{E}\,(a^T\tilde{x})^2$ for any a. The solution to this problem is given by

$$\begin{aligned} L &= (APC^T + R_{we})(R_e + CPC^T)^{-1}, \\ P &= (A - LC)P(A - LC)^T + \begin{bmatrix} I & -L \end{bmatrix} R \begin{bmatrix} I & -L \end{bmatrix}^T. \end{aligned}$$

The solution is obtained by solving the resulting discrete time algebraic Riccati equation.

In the same spirit we define the performance of the observer, for a given a, by

$$J_o(L, a) = \mathbf{E}\,(a^T\tilde{x})^2, \tag{13}$$

where $\tilde{x}$ satisfies (12). This performance is only defined for matrices L such that $A - LC$ is stable. The objective is to find a $L \in \mathscr{L}_{\text{stab}}$, which minimizes (13). The following proposition shows how to determine the gradient to J_o with respect to L.

PROPOSITION 1
Given the system (12) and a matrix L such that $A - LC$ is stable, the gradient to J_o with respect to L is

$$\nabla_L J_o = 2\mathcal{A}\left[LR_e - R_{we} - (A - LC)PC^T\right], \tag{14}$$

where

$$\mathcal{A} = (A - LC)^T \mathcal{A}(A - LC) + aa^T, \tag{15}$$

$$P = (A - LC)P(A - LC)^T + R_L \tag{16}$$

and $R_L = R_w - LR_{we}^T - R_{we}L^T + LR_eL^T$. □

Proof. We use the fact that the cost function $J_o(L, a) = \text{tr}(Paa^T)$. In order to find the gradient to this expression, we will determine the differential dP with respect to L. For simplicity, make the following denotations

$$A_L = A - LC,$$
$$M = \left(LR_e - R_{we} - A_L PC^T\right).$$

Now, differentiating (16) it is easily seen that dP satisfies the following Lyapunov equation

$$dP = A_L dP A_L^T + M dL^T + dLM^T.$$

This means that

$$dP = \sum_{t=0}^{\infty} A_L^t (M dL^T + dLM^T)(A_L^T)^t.$$

Finally, using the expression for dP we have that

$$dJ_o = \text{tr}(dPaa^T) = \text{tr}\left(2\sum_{t=0}^{\infty} A_L^t M dL^T (A_L^T)^t aa^T\right) = 2\text{tr}\left(\mathcal{A}M dL^T\right).$$

We conclude the proof by using the fact that $dZ = \text{tr}(Y \cdot dX^T) \Rightarrow \nabla_X Z = Y$ for matrices X^T, Y of size $n \times p$. □

The result in Proposition 1 is an intermediate result in the search for a tractable method for large-scale systems. To calculate the gradient (14) in Proposition 1, the Lyapunov equations (15) and (16) need to be determined. When faced with large-scale systems it is not possible to solve these Lyapunov equations due to time and memory limitations. In the following proposition we show how to replace these solutions to the Lyapunov equations by introducing adjoint (or dual) state variables. The benefit is that now only two systems have to simulated. For certain families of systems, for example for sparse systems, the computational time and memory requirements for simulation of them scales linearly with the size of the system, see Section 4 for more details. Therefore this approach produces a scalable method, thus better suited for such large-scale system.

PROPOSITION 2
Given a matrix L such that $A - LC$ is stable, define the systems

$$\bar{x}(t+1) = (A - LC)^T \bar{x}(t), \tag{17}$$
$$\lambda(t-1) = (A - LC)\lambda(t) + R_L \bar{x}(t) \tag{18}$$

with initial and final condition $\bar{x}(0) = a$ and $\lim_{t\to\infty} \lambda(t) = 0$. Then

$$\nabla_L J_o = 2\left(\sum_{t=0}^{\infty} \bar{x}(t)\left(\bar{x}(t)^T(LR_e - R_{we}^T) - \lambda(t)^T C^T\right)\right). \tag{19}$$

□

Proof. With the previously introduced notations we have that

$$\lambda(t) = \sum_{j=t+1}^{\infty} A_L^{j-t-1} R_L \bar{x}(j) = \sum_{j=0}^{\infty} A_L^j R_L (A_L^T)^{j+1} \bar{x}(t) = P A_L^T \bar{x}(t).$$

Hence

$$\sum_{t=0}^{\infty} \bar{x}(t)\lambda(t)^T = \sum_{t=0}^{\infty} \bar{x}(t)\bar{x}(t)^T A_L P = \mathcal{A} A_L P.$$

Using this result in (14) proves the proposition. □

When the gradient for a certain L has been calculated it can be used to update that predictor matrix to reduce the cost. Since we impose a structure on L, the gradient $\nabla_L J_o$ needs to be projected to the subspace $\mathscr{L}$ defining this structure. This will also be a descent direction of $J_o(L, a)$.

To understand this fact, consider the restriction of J_o on $\mathscr{L}$. The gradient of this function is exactly the projection of $\nabla_L J_o$ on $\mathscr{L}$.

Using Proposition 2 we can iteratively change the matrix L to decrease the cost J_o. Instead of simulating the systems for an infinite time, the sums in (19) are truncated to end at some final time t_{final}. Now, the system (17) is simulated from time $t = 0$ to $t = t_{\text{final}}$ in increasing time direction. Thereafter the adjoint states in (18) are simulated from $t = t_{\text{final}}$ to $t = 0$ in decreasing time direction. The procedure is summarized into the following algorithm.

ALGORITHM 1
Consider a system (1) with the observer in (11) where $L \in \mathscr{L}_{\text{stab}}$. To find a local minimizer to (13), start with $L^{(0)} \in \mathscr{L}_{\text{stab}}$ and for each $\tau \geq 0$,

1. Let $\bar{x}(0) = a$ and simulate the states of (17) for times $t = 0, \ldots, t_{\text{final}}$.

2. Simulate the adjoint states of system (18) for times $t = 0, \ldots, t_{\text{final}}$ in decreasing time with $\lambda(t_{\text{final}}) = 0$.

3. Calculate an approximation of $\nabla_L J_o$

$$\nabla_L J_o \approx 2 \sum_{t=0}^{t_{\text{final}}} \bar{x}(t)\big(\bar{x}(t)^T (L R_e - R_{we}^T) - \lambda(t)^T C^T\big)$$

   and project the approximation on the admissible set of predictor matrices

$$G = \text{proj}_{\mathscr{L}} \left(\nabla_L J_o\right)$$

4. Update the predictor matrix in the direction of the projected gradient

$$L^{(\tau+1)} = L^{(\tau)} - \gamma_\tau G$$

   for some step length γ_τ.

5. Increase τ with 1 and goto 1.

□

Remark 1
In step 4 of the algorithm, the predictor matrix L is updated in the direction of the projected gradient with some step length. It is of concern how to pick an appropriate step length. If we pick one that is too small the algorithm will require unnecessary many iterations to converge and if we pick a too long step length we potentially end up with a destabilizing L. In a way, the step length is a design parameter of the algorithm, but there are approaches to estimate a suitable step length given a direction. One way is to simulate (1) with the $L = L^{(\tau)} - \gamma_\tau G$. If this L does not lead to a decreased cost, the step length is scaled by some factor < 1. □

As posed, Algorithm 1 does not include a stopping criterion. One way is simply to specify the number of iterations to complete before stopping. This approach implies that the computation time for the complete algorithm will be known beforehand. On the other hand, there will be no guarantees of the performance the resulting predictor matrix will give. The following section shows a procedure to use the previously determined variables to determine a bound of the degree of suboptimality the current predictor matrix gives rise to. By specifying the degree of suboptimality required of the sought predictor matrix, the bound can function as a stopping criterion.

2.2 Determining the Degree of Suboptimality

When finding the optimal observer without any restriction on the structure of the matrix L, there is a closed form to determine L. When we introduce restrictions in the structure of L, there is no general formula for finding the matrix. The minimization problem is not even guaranteed to be convex. Since the underlying method of Algorithm 1 is a descent direction method, we can only know that a locally optimal solution is reached. If we can find a value $\beta \geq 1$ such that we can verify the inequality

$$J_o(L, a) \leq \beta J_o(L_{\text{opt}}, a), \tag{20}$$

we would know that the performance of the current L is within a factor β of the performance of the optimal solution L_{opt}. In other words, if we have a way of verifying that a β close to 1 is such that (20) holds, then we know that even though the current L might not be optimal, at least we will not be able to find another L reducing the cost much more. Introduce the truncated version of the cost function by

$$\bar{J}_o(L, a, t_{\text{final}}) = \sum_{t=0}^{t_{\text{final}}} \bar{x}(t)^T R_L \bar{x}(t), \tag{21}$$

where $\bar{x}$ satisfies (17). The following theorem determines the degree of suboptimality by looking at the dual function to the constrained optimization problem.

THEOREM 1
If $\beta \geq 1$ is such that for a given sequence of adjoint variables $\lambda(t)$, with $\lambda(t_{\text{final}}) = 0$

$$\bar{J}_o(L, a, t_{\text{final}}) \leq \beta \min_{\substack{\bar{x},\xi \\ \bar{x}(0)=a}} \sum_{t=0}^{t_{\text{final}}} \Bigg(\begin{bmatrix} \bar{x}(t) \\ \xi(t) \end{bmatrix}^T R \begin{bmatrix} \bar{x}(t) \\ \xi(t) \end{bmatrix} + 2\lambda(t)^T \left(\bar{x}(t+1) - A^T \bar{x}(t) - C^T \xi(t) \right) \Bigg). \quad (22)$$

Then the following bound can be verified

$$\bar{J}_o(L, a, t_{\text{final}}) \leq \beta \bar{J}_o(L_{\text{opt}}, a, t_{\text{final}}), \quad (23)$$

where

$$L_{\text{opt}} = \arg\min_L \bar{J}_o(L, a, t_{\text{final}}).$$

□

Proof. Assume that for $\beta \geq 1$ and a given sequence $\lambda(t)$, that (22) holds. Then we have that

$$\begin{aligned}
\bar{J}_o(L_{\text{opt}}, a, t_{\text{final}}) &= \begin{cases} \min\limits_{L,\bar{x}} \sum\limits_{t=0}^{t_{\text{final}}} \begin{bmatrix} \bar{x}(t) \\ -L\bar{x}(t) \end{bmatrix}^T R \begin{bmatrix} \bar{x}(t) \\ -L\bar{x}(t) \end{bmatrix} \\ \text{subject to: } \bar{x}(t+1) = (A - LC)^T \bar{x}(t) \\ \qquad\qquad \bar{x}(0) = a \end{cases} \\
&\geq \begin{cases} \min\limits_{\bar{x},\xi} \sum\limits_{t=0}^{t_{\text{final}}} \begin{bmatrix} \bar{x}(t) \\ \xi(t) \end{bmatrix}^T R \begin{bmatrix} \bar{x}(t) \\ \xi(t) \end{bmatrix} \\ \text{subject to: } \bar{x}(t+1) = A^T \bar{x}(t) + C^T \xi(t) \\ \qquad\qquad \bar{x}(0) = a \end{cases} \\
&\geq \min_{\substack{\bar{x},\xi \\ \bar{x}(0)=a}} \sum_{t=0}^{t_{\text{final}}} \Bigg(\begin{bmatrix} \bar{x}(t) \\ \xi(t) \end{bmatrix}^T R \begin{bmatrix} \bar{x}(t) \\ \xi(t) \end{bmatrix} + 2\lambda(t)^T \left(\bar{x}(t+1) - A^T \bar{x}(t) - C^T \xi(t) \right) \Bigg),
\end{aligned}$$

where the second inequality comes from introducing the dual variables $\lambda(t)$. Hence, the inequality gives that if (22), holds, then so must (23). □

To determine β an adjoint trajectory has to be chosen. That we choose the adjoint of (18) can be motivated by the following. Let $\mathcal{H}(\bar{x}, \xi, \lambda)$ be the objective function of the minimization problem in the last inequality of the previous proof. For a saddle point of $\mathcal{H}$ we must have that

$$\begin{aligned} 0 &= \nabla_{\bar{x}(t)}\mathcal{H} = 2(R_w\bar{x}(t) + R_{we}\xi(t) + \lambda(t-1) - A\lambda(t)), \\ 0 &= \nabla_{\xi(t)}\mathcal{H} = 2(R_{we}^T\bar{x}(t) + R_e\xi(t) - C\lambda(t)). \end{aligned}$$

We get (18) by $\nabla_{\bar{x}(t)}\mathcal{H} + L\nabla_{\xi(t)}\mathcal{H} = 0$ and replacing $\xi(t) = L^T\bar{x}(t)$.

2.3 State Feedback Synthesis

To determine the structured state feedback matrix, the dynamics in (1) is slightly modified by letting $w(t) = 0$ and introducing an initial condition. Hence, we consider the similar system

$$x(t+1) = Ax(t) + Bu(t), \qquad x(0) = x_0. \tag{24}$$

The cost we use will now be

$$J_c(K, x_0) = \sum_{t=0}^{\infty} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T \underbrace{\begin{bmatrix} Q_x & Q_{xu} \\ Q_{xu}^T & Q_u \end{bmatrix}}_{Q} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}. \tag{25}$$

The corresponding relations for the state feedback matrix are given in the following propositions. The proofs are omitted and can be found in [Mårtensson and Rantzer, 2011b].

Proposition 3
Given the system (1) and a stabilizing K, let the adjoint states λ be defined by the backwards iteration

$$\begin{aligned} \lambda(t-1) = (A - BK)^T\lambda(t) \\ - (Q_x - Q_{xu}K - K^TQ_{xu}^T + K^TQ_uK)x(t), \end{aligned} \tag{26}$$

where $x(t)$ are the states of (1), with $\lim_{t\to\infty} \lambda(t) = 0$. Then

$$\nabla_K J_c = 2\left(\sum_{t=0}^{\infty} \left(-Q_u u(t) - Q_{xu}^T x(t) + B^T\lambda(t)\right) x(t)^T\right). \tag{27}$$

□

THEOREM 2
If $\alpha \geq 1$ is such that for a given sequence of adjoint variables $\lambda(t)$, with $\lambda(t_{\text{final}}) = 0$

$$\bar{J}_c(K, x_0, t_{\text{final}}) \leq \alpha \min_{\substack{x,u \\ x(0)=x_0}} \sum_{t=0}^{t_{\text{final}}} \left(\begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} + 2\lambda(t)^T \left(x(t+1) - Ax - Bu(t) \right) \right), \quad (28)$$

then

$$\bar{J}_c(K, x_0, t_{\text{final}}) \leq \alpha \bar{J}_c(K_{\text{opt}}, x_0, t_{\text{final}}), \quad (29)$$

where

$$K_{\text{opt}} = \arg\min_K \bar{J}_c(K, x_0, t_{\text{final}}).$$

□

2.4 Combined Degree of Suboptimality

For a given predictor matrix L the solution P of the Lyapunov equation in (16) can be used to determine the performance of the estimator. Similarly, for a given feedback matrix K, the solution S to the Lyapunov equation

$$S = (A - BK)^T S (A - BK) + Q_K, \quad (30)$$

where $Q_K = Q_x - Q_{xu}K - K^T Q_{xu}^T + K^T Q_u K$, can be used to determine the performance of the state feedback controller. In Theorem 1 and Theorem 2 we find a way to determine a bound of the degree of suboptimality for each problem in the given directions a and x_0, respectively. If the process is repeated in different directions and similar values of the degrees of suboptimality is acquired, this is an indication that $S \leq \alpha S_{\text{opt}}$ and $P \leq \beta P_{\text{opt}}$, where S_{opt} and P_{opt} solves the corresponding discrete-time algebraic Riccati equations. Used separately, these bounds tell us that the state feedback and the state estimator are within a factor of α and β of their optimal values, respectively. However, it does not yet ensure us that when the feedback matrix K is combined with the predictor matrix L into an output feedback controller, what the combined performance is? More importantly, if we have obtained bounds α and β close to 1, will the combined controller also have good performance. This is the question that will be investigated in this section. We define the sets of α-suboptimal feedback matrices and β-suboptimal predictor matrices, where for each K and L, the matrices S and P satisfy (30) and (16), respectively:

$$\kappa_\alpha = \{K \mid A - BK \text{ is stable and } S \leq \alpha S_{\text{opt}}\},$$
$$\iota_\beta = \{L \mid A - LC \text{ is stable and } P \leq \beta P_{\text{opt}}\}.$$

With these sets we define the worst case cost of the resulting output feedback controller for given degrees of suboptimality α and β by

$$\mathcal{J}(\alpha,\beta) = \sup_{\substack{K\in\kappa_\alpha \\ L\in\iota_\beta}} J(K,L). \tag{31}$$

We will show that the value of $\mathcal{J}(\alpha,\beta)$ is close to the optimal value in a neighborhood of $(1,1)$. We start by stating the following lemmas.

Lemma 1
Let $A(x) \in \mathbb{R}^{n\times n}$ and $Q(x) \in \mathbb{R}^{n\times n}$ be continuous in the parameter $x \in D$, where D is the open domain of the parameter, and $A(x)$ stable for all $x \in D$. Then $P(x)$, determined uniquely by

$$P(x) = A(x)P(x)A(x)^T + Q(x), \tag{32}$$

is continuous in x. □

Proof. Let $T(x) = I - A(x) \otimes A(x)$, where $\otimes$ denotes the Kronecker product. Since $A(x)$ is stable for all $x \in D$, the matrix $T(x)$ is invertible for all $x \in D$. In for example [Lancaster and Rodman, 1995] an equivalent formulation of (32) can be found

$$T(x) \cdot \mathrm{vec}(P(x)) = \mathrm{vec}(Q(x)),$$

where $\mathrm{vec}(X)$ is the vectorization of the matrix X for which the columns of X are stacked on top of each other. Now, since inversion of matrices is a continuous transformation $\mathrm{vec}(P(x))$, and thus, $P(x)$ is continuous in x. □

Lemma 2
Let $\alpha_i > 1$ for $i \geq 1$ be a sequence of reals converging to 1. For any sequence K_i of feedback matrices controlling (24) such that $S_i \leq \alpha_i S_{\mathrm{opt}}$, then $K_i \to K_{\mathrm{opt}}$. Moreover, the set κ_α is compact for all $\alpha \geq 1$. □

Proof. For any i, let $dS_i = S_i - S_{\mathrm{opt}}$ and $dK_i = K_i - K_{\mathrm{opt}}$. Then

$$\begin{aligned}(S_{\mathrm{opt}} + dS_i) = (A - BK_{\mathrm{opt}} - BdK_i)^T &(S_{\mathrm{opt}} + dS_i)(A - BK_{\mathrm{opt}} - BdK_i) \\ &+ Q_x + (K_{\mathrm{opt}} + dK_i)^T Q_u (K_{\mathrm{opt}} + dK_i) \\ &- Q_{xu}(K_{\mathrm{opt}} + dK_i) - (K_{\mathrm{opt}} + dK_i)^T Q_{xu}^T.\end{aligned}$$

Expanding this expression and using the Lyapunov equation

$$\begin{aligned}S_{\mathrm{opt}} = (A - BK_{\mathrm{opt}})^T &S_{\mathrm{opt}}(A - BK_{\mathrm{opt}}) \\ &+ Q_x + K_{\mathrm{opt}}^T Q_u K_{\mathrm{opt}} - Q_{xu}K_{\mathrm{opt}} - K_{\mathrm{opt}}^T Q_{xu}^T\end{aligned}$$

and that $Q_u K_{\text{opt}} - Q_{xu}^T - B^T S_{\text{opt}}(A - BK_{\text{opt}}) = 0$ (by the relation for the optimal feedback matrix) we get that

$$\begin{aligned} dS_i &= (A - BK_i)^T dS_i (A - BK_i) + dK_i^T (Q_u + B^T S_{\text{opt}} B) dK_i \\ &\geq dK_i^T (Q_u + B^T S_{\text{opt}} B) dK_i, \end{aligned} \tag{33}$$

where the inequality comes from using that $dS_i \geq 0$. Now, if dK_i does not converge to 0 then we easily understand that neither dS_i would converge to 0. But this contradicts the fact that $S_i \leq \alpha_i S_{\text{opt}}$ for all i. Hence $dK_i \to 0$ and $K_i \to K_{\text{opt}}$.

It is also easily realized from (33) that the set κ_α is bounded. To show that the set is closed, let $Y_i \in \kappa_\alpha$ be a converging sequence in $\mathbb{R}^{p \times n}$, converging to Y. By Lemma 1 with $A(X) = (A - BX)^T$, $Q(X) = Q_x - Q_{xu}X - X^T Q_{xu}^T + X^T Q_u X$ and replacing P with S, $S(X)$ is continuous implying that $S(Y_i) \to S(Y)$. Hence, since $S(Y_i) \leq \alpha S_{\text{opt}}$, $S(Y) \leq \alpha S_{\text{opt}}$ implying that $Y \in \kappa_\alpha$. □

Remark 2
Lemma 2 can naturally be changed to the equivalent estimation formulation, that is, if $P_i \leq \beta_i P_{\text{opt}}$ for $\beta_i \to 1$, then $L_i \to L_{\text{opt}}$. The set ι_β is also compact. □

With this lemma we are able to prove the following theorem.

Theorem 3
The function $\mathcal{J}(\alpha, \beta)$ is continuous in the point $(\alpha, \beta) = (1, 1)$. □

Proof. Let $(\alpha_i, \beta_i) \geq (1, 1)$, for $i \geq 1$, be a sequence converging to $(1, 1)$. By Lemma 2, κ_{α_i} and ι_{β_i} are compact. Hence, there exists $K_i \in \kappa_{\alpha_i}$ and $L_i \in \iota_{\beta_i}$ such that $J(K_i, L_i) = \mathcal{J}(\alpha_i, \beta_i)$. Also $(K_i, L_i) \to (K_{\text{opt}}, L_{\text{opt}})$. By Lemma 1 P_{cl} in (8) is continuous in K and L. Hence, $P_{\text{cl}}^{(i)}$, defined by K_i and L_i, converges to $P_{\text{cl}}^{\text{opt}}$. This means that

$$\mathcal{J}(\alpha_i, \beta_i) = \text{tr}\left(P_{\text{cl}}^{(i)} \cdot \mathcal{K}_i^T Q \mathcal{K}_i\right) \to \text{tr}\left(P_{\text{cl}}^{\text{opt}} \cdot \mathcal{K}_{\text{opt}}^T Q \mathcal{K}_{\text{opt}}\right) = \mathcal{J}(1, 1).$$

Hence, $\mathcal{J}$ is continuous in the point $(1, 1)$.

□

Remark 3
Theorem 3 tells us that in a neighborhood of $(1, 1)$ the value of $\mathcal{J}(\alpha, \beta)$ is close to the optimal value $\mathcal{J}(1, 1)$. Hence, if we determine K and L with the corresponding bounds of the degree of suboptimality α and β close to 1, we can expect that the resulting performance of the output feedback controller is close to the optimal value. □

REMARK 4
For any stabilizing K and L, let

$$P_{\text{cl}} = \begin{bmatrix} P_x & P_{x\tilde{x}} \\ P_{x\tilde{x}}^T & P_{\tilde{x}} \end{bmatrix}.$$

Then by (8) $P_{\tilde{x}} = P$ for P in (16) and

$$\begin{aligned} P_x = (A - BK)P_x(A - BK)^T + BKP_{\tilde{x}}K^TB^T \\ + (A - BK)P_{x\tilde{x}}K^TB^T + BKP_{x\tilde{x}}^T(A - BK)^T + R_w, \end{aligned} \quad (34)$$

meaning that $P_x = \sum_{k=0}^{\infty}(A - BK)^k M((A - BK)^T)^k$ where M is the last terms of the right hand side of (34). Now, using (10) we have that

$$J(K, L) = \text{tr}\,(P_x Q_K) + 2\text{tr}\left(P_{x\tilde{x}}\left(K^TQ_{xu}^T - K^TQ_uK\right)\right) + \text{tr}\left(P_{\tilde{x}}K^TQ_uK\right),$$

where $Q_K = Q_x - Q_{xu}K - K^TQ_{xu}^T + K^TQ_uK$. Using the expression for P_x, we can rewrite the first trace of the right hand side to get

$$\text{tr}\,(P_x Q_K) = \text{tr}\left(\sum_{k=0}^{\infty}((A - BK)^T)^k Q_K(A - BK)^k M\right) = \text{tr}\,(SM)$$

where S satisfies (30). This leads us to the expression for J

$$\begin{aligned} J(K, L) = 2\text{tr}\left(P_{x\tilde{x}}K^T\left(B^TS(A - BK) + Q_{xu}^T - Q_uK\right)\right) \\ + \text{tr}\left(PK^T(Q_u + B^TSB)K\right) + \text{tr}\,(SR_w). \end{aligned} \quad (35)$$

From (35) we realize that

$$J(K_{\text{opt}}, L) \leq \beta\text{tr}\left(P_{\text{opt}}K_{\text{opt}}^T(Q_u + B^TS_{\text{opt}}B)K_{\text{opt}}\right) + \text{tr}\,(S_{\text{opt}}R_w),$$

that is, $J(1, \beta)$ is bounded from above by a linear expression. An analogous result can be given for $J(\alpha, 1)$ being bounded from above by a linear expression. Using a Taylor expansion this gives us an indication of the behaviour of $J(\alpha, \beta)$ in a neighborhood of $(1, 1)$. □

3. Simultaneous Synthesis Scheme

As previously mentioned, when calculating the ordinary LQG controller, the problem is solved using the separation principle by separating it into finding the optimal state feedback matrix and finding the optimal predictor matrix. However, when not considering the optimal matrix gains, the separation principle does not apply. Hence, if we are comparing two pairs of matrices, (K_1, L_1) and (K_2, L_2), separately K_1 could perform better than K_2, that is $J_c(K_1) < J_c(K_2)$, and similarly for L_1 and L_2, but when combined the total LQG cost (4) is less for (K_2, L_2) than for (K_1, L_1). This fact leads us to treat the synthesis procedure simultaneously.

3.1 Without Direct Term

In this section we do not allow a controller with direct term. Hence, the matrices in system (5) are the ones in (6), where $K \in \mathscr{K}_{\text{stab}}$ and $L \in \mathscr{L}_{\text{stab}}$. With the solutions to the Lyapunov equations in (8) and (9) and the expression in (10) for the cost J, the gradient of J with respect to both K and L can be determined.

PROPOSITION 4
Consider the system (5) where K and L are stabilizing and the cost J defined in (4). The gradients of J with respect to both K and L are

$$\begin{aligned} \nabla_K J &= 2(M_{11} + M_{22}), \\ \nabla_L J &= 2N_{22}, \end{aligned}$$

with

$$\begin{aligned} M &= \left(\begin{bmatrix} -B^T & 0^{m\times n} \\ B^T & 0^{m\times n} \end{bmatrix} S_{\text{cl}} A_{\text{cl}} + \begin{bmatrix} Q_u K - Q_{xu}^T & -Q_u K \\ Q_{xu}^T - Q_u K & Q_u K \end{bmatrix} \right) P_{\text{cl}}, \\ N &= S_{\text{cl}} \left(A_{\text{cl}} P_{\text{cl}} \begin{bmatrix} 0^{n\times p} & 0^{n\times p} \\ 0^{n\times p} & -C^T \end{bmatrix} + \begin{bmatrix} 0^{n\times p} & -R_{we} \\ 0^{n\times p} & LR_e - R_{we} \end{bmatrix} \right), \end{aligned} \tag{36}$$

where S_{cl} and P_{cl} are defined by (9) and (8), M_{11} and M_{22} are the diagonal blocks of M partitioned in blocks of equal size

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$$

and similar for N_{22}, the lower diagonal block of N. □

Proof. To determine the gradient of J with respect to K we begin by determining a Lyapunov equation for the differential dS_{cl},

$$dS_{\text{cl}} = A_{\text{cl}}^T dS_{\text{cl}} A_{\text{cl}} + \widehat{M} + \widehat{M}^T,$$

where

$$\widehat{M} = \begin{bmatrix} dK^T & 0^{n\times m} \\ 0^{n\times m} & dK^T \end{bmatrix} \left(\begin{bmatrix} -B^T & 0^{m\times n} \\ B^T & 0^{m\times n} \end{bmatrix} S_{\mathrm{cl}} A_{\mathrm{cl}} + \begin{bmatrix} Q_u K - Q_{xu}^T & -Q_u K \\ Q_{xu}^T - Q_u K & Q_u K \end{bmatrix} \right).$$

This leads to the expression for dS_{cl}

$$dS_{\mathrm{cl}} = \sum_{k=0}^{\infty} (A_{\mathrm{cl}}^T)^k (\widehat{M} + \widehat{M}^T) A_{\mathrm{cl}}^k$$

and using that $dJ = \mathrm{tr}\left(dS_{\mathrm{cl}} \cdot \mathcal{L} R \mathcal{L}^T\right)$ we get

$$\begin{aligned} dJ &= 2\mathrm{tr}\left(\sum_{k=0}^{\infty} (A_{\mathrm{cl}}^T)^k \widehat{M} A_{\mathrm{cl}}^k \cdot \mathcal{L} R \mathcal{L}^T \right) = 2\mathrm{tr}\left(\widehat{M} \sum_{k=0}^{\infty} A_{\mathrm{cl}}^k \mathcal{L} R \mathcal{L}^T (A_{\mathrm{cl}}^T)^k \right) \\ &= 2\mathrm{tr}\left(\widehat{M} P_{\mathrm{cl}} \right). \end{aligned}$$

The expression for $\nabla_K J$ is derived by using the relation about differentials

$$dZ = \mathrm{tr}\left(\begin{bmatrix} dX^T & 0 \\ 0 & dX^T \end{bmatrix} Y \right) \implies \nabla_X Z = Y_{11} + Y_{22},$$

for $Y = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix}$ where Y_{11}, Y_{12}, Y_{21} and Y_{22} all have the appropriate sizes.

The expression for $\nabla_L J$ is derived analogously. □

Remark 5

From these relations we see that some care has to be taken when choosing initial K and L in the coming update scheme. When $Q_{xu} = 0$, if $K = 0$ and $L = 0$ we easily get from (8) and (9) that

$$S_{\mathrm{cl}} = \begin{bmatrix} S_{11} & 0 \\ 0 & 0 \end{bmatrix} \text{ and } P_{\mathrm{cl}} = \begin{bmatrix} P_{11} & P_{11} \\ P_{11} & P_{11} \end{bmatrix},$$

for some S_{11} and P_{11}. This means that

$$M = \begin{bmatrix} -B^T S_{11} A P_{11} & -B^T S_{11} A P_{11} \\ B^T S_{11} A P_{11} & B^T S_{11} A P_{11} \end{bmatrix},$$

resulting in that $\nabla_K J = 0$. Hence, we will get stuck if we use this gradient as a descent direction. This is in fact a local maximum. Similar reasoning holds when we inspect $\nabla_L J$. □

As in Proposition 1, to calculate these gradients involve the solution of two Lyapunov equations, which is not possible when dealing with large-scale systems. In the similar way as previously we introduce adjoint variables to circumvent this issue. The exception in this section is that instead of introducing initial condition, we will continue to use the stochastic representation of the system.

PROPOSITION 5
Consider the systems

$$\begin{aligned} x(t+1) &= A_{\mathrm{cl}}x(t) + \mathcal{L}v(t), \\ \lambda(t-1) &= A_{\mathrm{cl}}^T\lambda(t) - \mathcal{K}^T\mathcal{Q}\mathcal{K}x(t) \end{aligned} \tag{37}$$

and

$$\begin{aligned} \bar{x}(t+1) &= A_{\mathrm{cl}}^T\bar{x}(t) + \mathcal{K}^T\bar{v}(t), \\ \bar{\lambda}(t-1) &= A_{\mathrm{cl}}\bar{\lambda}(t) - \mathcal{L}R\mathcal{L}^T\bar{x}(t), \end{aligned} \tag{38}$$

where $v(t) \in \mathcal{N}(0,R)$ and $\bar{v}(t) \in \mathcal{N}(0,Q)$ and $\lim_{t\to\infty}\lambda(t) = \lim_{t\to\infty}\bar{\lambda}(t) = 0$.

The matrices in (36) equals (with the same size of the zeros)

$$M = 2\mathbf{E}\left(\begin{bmatrix} B^T & 0 \\ -B^T & 0 \end{bmatrix}\lambda(t) + \begin{bmatrix} Q_uK - Q_{xu}^T & -Q_uK \\ Q_{xu}^T - Q_uK & Q_uK \end{bmatrix}x(t)\right)x(t)^T, \tag{39}$$

$$N = 2\mathbf{E}\,\bar{x}(t)\left(\bar{\lambda}(t)^T\begin{bmatrix} 0 & 0 \\ 0 & C^T \end{bmatrix} + \bar{x}(t)^T\begin{bmatrix} 0 & -R_{we} \\ 0 & LR_e - R_{we} \end{bmatrix}\right) \tag{40}$$

and thus, the gradients of J with respect to both K and L are

$$\begin{aligned} \nabla_K J &= 2(M_{11} + M_{22}) \\ \nabla_L J &= 2N_{22} \end{aligned}$$

□

Proof. We prove the relation for M and $\nabla_K J$ and the relations for N and $\nabla_L J$ follow analogously. We have that

$$\begin{aligned} \lambda(t) &= -\sum_{j=t+1}^{\infty}(A_{\mathrm{cl}}^T)^{j-t-1}\mathcal{K}^T\mathcal{Q}\mathcal{K}x(j) \\ &= -\sum_{j=0}^{\infty}(A_{\mathrm{cl}}^T)^j\mathcal{K}^T\mathcal{Q}\mathcal{K}A_{\mathrm{cl}}^{j+1}x(t) + \Psi\{(v(t),v(t+1),\ldots)\}, \end{aligned}$$

where Ψ is the linear operator on the sequence $(v(t),v(t+1),\ldots)$ coming from expressing all $x(t+j)$, $j > 0$, in $x(t)$ and $(v(t),v(t+1),\ldots)$ and

summing these with matrix factors according to the sum. Since $x(t)$ and $(v(t), v(t+1), \ldots)$ are uncorrelated we have that

$$\mathbf{E}\,\lambda(t)x(t)^T = -\mathbf{E}\sum_{j=0}^{\infty}(A_{\mathrm{cl}}^T)^j \mathcal{K}^T Q \mathcal{K} A_{\mathrm{cl}}^{j+1} x(t)x(t)^T = -S_{\mathrm{cl}}A_{\mathrm{cl}}P_{\mathrm{cl}},$$

since $P_{\mathrm{cl}} = \mathbf{E}\,x(t)x(t)^T$. Fitting this into (36) gives the expression for M and thus the expression for $\nabla_K J$ follows. □

As previously we need to approximate the gradients by only simulating the systems in (37) and (38) in order to find approximations of the cross-covariances. The final time in the simulations will be denoted by t_{final}. We summarize the procedure of updating the control matrices to reduce the cost in the following algorithm.

ALGORITHM 2

Consider the system (5) with $K \in \mathscr{K}_{\mathrm{stab}}$ and $L \in \mathscr{L}_{\mathrm{stab}}$. To find an approximation to a local minimizer to (4), start with $K^{(0)} \in \mathscr{K}_{\mathrm{stab}}$ and $L^{(0)} \in \mathscr{L}_{\mathrm{stab}}$ and for each $\tau \geq 0$,

1. Simulate the systems in (37) and (38) for some realization of the noise v and $\bar{v}$.
2. Calculate approximations of M and N in (39) and (40) by calculating approximations of the cross-covariances with the determined state trajectories.
3. Determine approximations for the gradients $\nabla_K J$ and $\nabla_L J$.
4. Project the approximations on the admissible set of feedback matrices and predictor matrices,

$$G_K = \mathrm{proj}_{\mathscr{K}}\left(\nabla_K J\right),$$
$$G_L = \mathrm{proj}_{\mathscr{L}}\left(\nabla_L J\right).$$

5. Update the matrices in the direction of the projected gradients

$$K^{(\tau+1)} = K^{(\tau)} - \gamma_\tau^K G_K,$$
$$L^{(\tau+1)} = L^{(\tau)} - \gamma_\tau^L G_L$$

for some step lengths γ_τ^K and γ_τ^L.

6. Increase τ with 1 and goto 1).

□

3.2 With Direct Term

In this section we consider controllers with direct term, meaning that the system matrices for the system in (5) are given in (7).

It is easily seen that the expressions for the gradients of J with respect to both K and L are as in Proposition 4 with the system matrices in (7). Hence, it is only necessary to determine an expression for the gradient of J with respect to L_0.

PROPOSITION 6
Consider the system (5) with the system matrices in (7) and where K and L are stabilizing and the cost J defined in (4). The gradient of J with respect to L_0 is

$$\nabla_{L_0} J = 2(U_{12} + V_{22} + W_{12} + L_0 R_e),$$

where

$$\begin{aligned} U &= -\begin{bmatrix} B^T & 0^{m\times n} \\ 0^{m\times n} & 0^{m\times n} \end{bmatrix} S_{\mathrm{cl}} A_{\mathrm{cl}} P_{\mathrm{cl}} \begin{bmatrix} 0^{n\times p} & 0^{n\times p} \\ 0^{n\times p} & C^T \end{bmatrix}, \\ V &= -\mathcal{Q}\mathcal{K}P_{\mathrm{cl}} \begin{bmatrix} 0^{n\times p} & 0^{n\times p} \\ 0^{n\times p} & C^T \end{bmatrix}, \\ W &= -\begin{bmatrix} B^T & 0^{m\times n} \\ 0^{m\times n} & 0^{m\times n} \end{bmatrix} S_{\mathrm{cl}} \mathcal{L} R. \end{aligned} \tag{41}$$

□

Proof. We begin be determining the differential of S_{cl} with respect to L_0

$$dS_{\mathrm{cl}} = A_{\mathrm{cl}}^T dS_{\mathrm{cl}} A_{\mathrm{cl}} + \widehat{M} + \widehat{M}^T,$$

where

$$\widehat{M} = -\begin{bmatrix} 0^{n\times n} & 0^{n\times n} \\ C^T dL_0^T B^T & 0^{n\times n} \end{bmatrix} S_{\mathrm{cl}} A_{\mathrm{cl}} - \begin{bmatrix} 0^{n\times n} & 0^{n\times m} \\ 0^{n\times n} & C^T dL_0^T \end{bmatrix} \mathcal{Q}\mathcal{K},$$

leading to the expression

$$dS_{\mathrm{cl}} = \sum_{k=0}^{\infty} (A_{\mathrm{cl}}^T)^k (\widehat{M} + \widehat{M}^T) A_{\mathrm{cl}}^k.$$

Determining the differential of J we get

$$\begin{aligned} dJ = \mathrm{tr}\left(dS_{\mathrm{cl}} \cdot \mathcal{L} R \mathcal{L}^T\right) + 2\mathrm{tr}\left(dL_0^T L_0 R_e\right) \\ + 2\mathrm{tr}\left(\begin{bmatrix} 0^{n\times m} & 0^{n\times n} \\ dL_0^T & 0^{p\times n} \end{bmatrix} \begin{bmatrix} -B^T & 0^{m\times n} \\ 0^{n\times n} & 0^{n\times n} \end{bmatrix} S_{\mathrm{cl}} \mathcal{L} R\right). \end{aligned}$$

We simplify the first term of this expression in the same way as in the proof of Proposition 4, and use that

$$\begin{bmatrix} 0^{n\times n} & 0^{n\times n} \\ C^T dL_0^T B^T & 0^{n\times n} \end{bmatrix} = \begin{bmatrix} 0^{n\times p} & 0^{n\times p} \\ 0^{n\times p} & C^T \end{bmatrix} \begin{bmatrix} 0^{p\times m} & 0^{p\times m} \\ dL_0^T & 0^{p\times m} \end{bmatrix} \begin{bmatrix} B^T & 0^{m\times n} \\ 0^{m\times n} & 0^{m\times n} \end{bmatrix}$$

and get that

$$\begin{aligned} \operatorname{tr}\left(dS_{\mathrm{cl}} \cdot \mathcal{L}R\mathcal{L}^T\right) &= 2\operatorname{tr}\left(\widehat{M}P_{\mathrm{cl}}\right) \\ &= -2\operatorname{tr}\left(\begin{bmatrix} 0^{p\times m} & 0^{p\times m} \\ dL_0^T & 0^{p\times m} \end{bmatrix} \begin{bmatrix} B^T & 0^{m\times n} \\ 0^{m\times n} & 0^{m\times n} \end{bmatrix} S_{\mathrm{cl}} A_{\mathrm{cl}} P_{\mathrm{cl}} \begin{bmatrix} 0^{n\times p} & 0^{n\times p} \\ 0^{n\times p} & C^T \end{bmatrix}\right) \\ &\quad - 2\operatorname{tr}\left(\begin{bmatrix} 0^{p\times n} & 0^{p\times m} \\ 0^{p\times n} & dL_0^T \end{bmatrix} Q\mathcal{K}P_{\mathrm{cl}} \begin{bmatrix} 0^{n\times p} & 0^{n\times p} \\ 0^{n\times p} & C^T \end{bmatrix}\right). \end{aligned}$$

Now, the expression for $\nabla_{L_0} J$ is derived be using the relations

$$\begin{aligned} dZ = \operatorname{tr}\left(\begin{bmatrix} 0 & 0 \\ dX^T & 0 \end{bmatrix} Y\right) &\implies \nabla_X Z = Y_{12}, \\ dZ = \operatorname{tr}\left(\begin{bmatrix} 0 & 0 \\ 0 & dX^T \end{bmatrix} Y\right) &\implies \nabla_X Z = Y_{22} \end{aligned}$$

for $Y = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix}$ where Y_{11}, Y_{12}, Y_{21} and Y_{22} all have the appropriate sizes. □

In Section 3.1 we saw how to get rid of the Lyapunov solutions S and P by making some approximations. The expressions in (41) resemble the ones in (36) and we use the same approach to rewrite the expression for $\nabla_{L_0} J$.

Proposition 7
Consider the systems in (37) and (38) with the system matrices in (7). Then the matrices in (41) equal

$$\begin{aligned} U &= \begin{bmatrix} B^T & 0 \\ 0 & 0 \end{bmatrix} \mathbf{E}\, \frac{1}{2}\left(\lambda(t)x(t)^T + \bar{x}(t)\bar{\lambda}(t)^T\right) \begin{bmatrix} 0 & 0 \\ 0 & C^T \end{bmatrix}, \\ V &= -Q\mathcal{K}\mathbf{E}\, x(t)x(t)^T \begin{bmatrix} 0 & 0 \\ 0 & C^T \end{bmatrix}, \\ W &= -\begin{bmatrix} B^T & 0 \\ 0 & 0 \end{bmatrix} \mathbf{E}\, \bar{x}(t)\bar{x}(t)^T \mathcal{L}R, \end{aligned} \tag{42}$$

where the sizes of the zeros are analog from Proposition 6. □

Proof. The theorem follows simply from using the relations derived in the proof of Proposition 5. □

REMARK 6
The factor $\frac{1}{2}$ in the expression for M can obviously be replaced to be any affine combination of the two terms in the sum. □

Now, Algorithm 2 can be updated with the expressions in (42) to include an update of L_0. Of course the system matrices will now be the ones found in (7).

4. Scalable Synthesis Schemes

The synthesis schemes presented in the previous sections rely on simulation of some systems. Certain structural constraints of the controller matrices in K, L and L_0 with some additional structure on the matrices in (1) may be exploited to allow these systems to be simulated even when the state space is huge. In such cases the schemes can be applied to find controllers to large-scale systems. One such structure that can be exploited is when the system matrices are sparse, for example when considering distributed systems. Such systems and how to exploit their structure will briefly be described in this section. For a more detailed analysis, see [Mårtensson and Rantzer, 2011b].

In a distributed system the states are partitioned into subsystems or agents, according to some property, for example that each partition corresponds to a distinct component of the complete system. The agents that directly affect an agent i through the dynamics matrix, that is, the block $A_{ik} \neq 0$, are called neighbors of agent i. Essential is that each agent only has a few neighbors, implying that the dynamics matrix A is sparse. We also assume that each agent has a distinct set of inputs and outputs, implying that both matrices B and C will be block-diagonal. A natural assumption for the structure of the controller gains is that an agent may only communicate with its neighbors, that is, it may only use measurements of neighbors to form estimates of its own states and only estimates of neighboring states to form its control input. This translates to the sets

$$\begin{aligned}
\mathscr{K} &= \{K \mid K_{ik} = 0 \text{ if } i \text{ and } k \text{ are not neighbors}\},\\
\mathscr{L} &= \{L \mid L_{ik} = 0 \text{ if } i \text{ and } k \text{ are not neighbors}\},\\
\mathscr{L}_0 &= \{L_0 \mid L_{0ik} = 0 \text{ if } i \text{ and } k \text{ are not neighbors}\}.
\end{aligned}$$

With these structural constraints, the closed loop matrices $A - BK$ and $A - LC$ have the same sparsity structure as A. Hence, the assumption that each agent only has a few neighbors guarantees that both the computational time and the memory requirement to simulate the systems (17)

and (24) scales linearly with the number of agents. Similarly, simulation of the system (26) is also scalable if the weights for the cost Q_x, Q_u and Q_{xu} are block-diagonal, meaning that the cost can be separated into costs for each agent. The same holds for system (18), if R_w, R_e and R_{we} are block-diagonal, that is, the noise coming into different agents are independent. This means that for distributed systems, the separate synthesis scheme described in Section 2 will be scalable with respect to the size of the system if the average number of neighbors for each agent is constant. With the same assumptions, simulation of the systems in (37) and (38) is also scalable, implying that the simultaneous synthesis scheme is scalable. This shows that both schemes will be applicable to large-scale distributed systems. A numerical example of the scalability of the method can be found in [Mårtensson and Rantzer, 2011b].

5. Numerical Examples

This example focuses on a system that is randomly generated. The system consists of 50 agents, each agent with 5 states, meaning a total of 250 states. The adjacency matrix is randomly generated, to give all agents 4 neighbors and a connected graph. The interconnection graph is assumed to be undirected, thus giving a sparse symmetric adjacency matrix. This means that the ratio of non-zero elements in the dynamics matrix A will be 10% of the total number of elements. For each agent i, the specifications of how its subsystem is generated, are given below.

- The part of dynamics matrix A_{ii} will be uniformly drawn and scaled to make the largest magnitude of the eigenvalues equal to 0.8.
- The neighbors affect the agent by $A_{ik} = b \cdot c^T$ where b and c are uniformly drawn from $[-0.4, 0.4]^5$.
- It will have 3 control inputs. The part of control matrix B_i will be uniformly drawn from $[-1, 1]^{5\times 3}$.
- It will have 2 measurements. The part of measurement matrix C_i will be uniformly drawn from $[-1, 1]^{2\times 5}$.

The resulting system is verified to be stable, with spectral radius $\rho(A) = 0.94$.

The remaining parameters of the design method are set to $Q = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$ and $R = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$, where I has the appropriate size in each block.

5.1 Separate Synthesis

This section presents the results of when the separate synthesis scheme is applied to the system. Initially the feedback matrix $K = 0$ and the predictor matrix $L = 0$. The number of update iterations of the scheme is set to 500, where the simulation time $t_{\text{final}} = 100$ in each iteration. A constant update step length $\gamma_\tau = 2.5 \cdot 10^{-3}$ is used. The result of a simulation of the scheme is shown in Figure 1. The curves denoted α and β correspond to the degree of suboptimality of the separate state feedback problem and state estimation problem, respectively. That is,

$$\alpha_i = \arg\min_{\alpha} \{S_i \leq \alpha S_{\text{opt}}\},$$
$$\beta_i = \arg\min_{\beta} \{P_i \leq \beta P_{\text{opt}}\}.$$

Since the system matrices are of moderate size, it is possible to compute the cost (4). The degree of suboptimality of the complete output feedback controller is shown by the curve denoted φ, that is,

$$\varphi = \frac{J(K^{(i)}, L^{(i)})}{J(K_{\text{opt}}, L_{\text{opt}})}.$$

In the figure, we see that both the degree of suboptimality of the state feedback, α, and the degree of suboptimality of the state estimator, β, approach 1. At the same time the degree of suboptimality of the complete output feedback, φ, approaches 1. The curve of φ is always less than both α and β in this example, meaning that the bounds α and β are actually conservative bounds of the degree of suboptimality φ.

5.2 Simultaneous Synthesis

Now, the simultaneous synthesis scheme is applied to the system. By Remark 5 the matrices K and L initially can not be 0, they are randomly generated in a small neighborhood around 0. The number of update iterations of the scheme is set to 500, where the simulation time $t_{\text{final}} = 200$ in each iteration. The step lengths are set equal, that is, $\gamma_\tau^K = \gamma_\tau^L$, and are changed to guarantee that with the current noise realization, the cost decreases. Figure 2 shows the result of a simulation of the scheme applied to the system. The degree of suboptimality is denoted φ and is calculated in the same way as in the previous section.

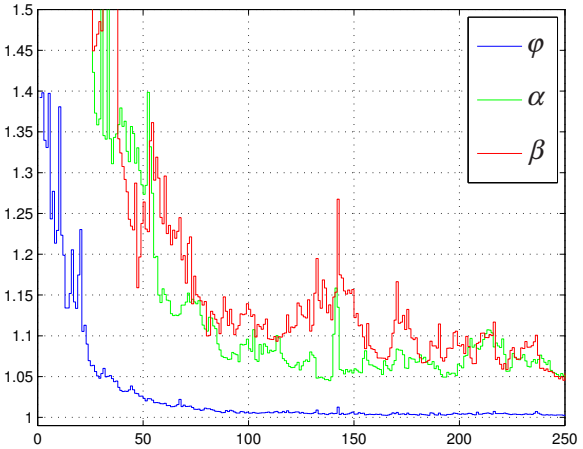


Figure 1. The result of applying the separate synthesis scheme. The degree of suboptimality of the state feedback and state estimation is given by the curve α and β, respectively. The degree of suboptimality of the complete output feedback is shown by the curve φ.

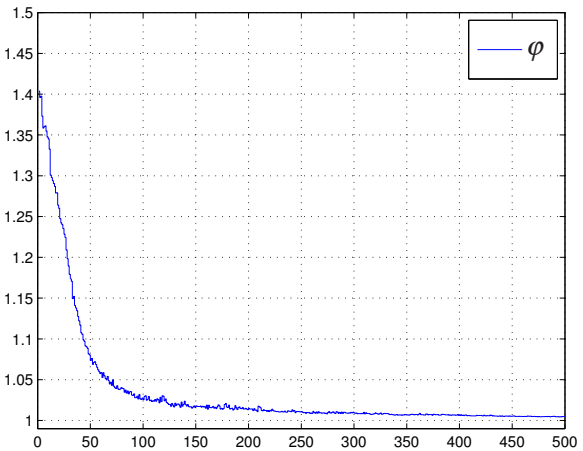


Figure 2. The result of applying the simultaneous synthesis scheme. The degree suboptimality of the complete output feedback is shown by the curve φ.

6. Conclusions and Future Works

6.1 Conclusions

In this paper we have studied a methodology to calculate structured LQG controllers suitable for large-scale systems. The underlying idea is to iteratively update the matrices of the controller in a descent direction for each matrix in order to improve the LQG performance. The descent directions are determined by simulating systems related to the original one. If the structure of the system and the controller matrices allow for efficient simulation, as in the case of sparse matrices, it is shown that the schemes are tractable for large-scale systems.

It is shown how to both separately determine suboptimal state feedback and state predictor matrices, or simultaneously simulate the complete system with the output feedback controller to find the descent update directions. Also, in case of the simultaneous scheme it is possible to treat both the case of controllers with and without direct term.

When using the separate scheme it is possible to use the trajectories determined in the update procedure, to determine a bound of the degree of suboptimality of the current matrix gains. This bound validates the performance of the controllers and can be used as a stopping criterion for the iteration process.

6.2 Future Works

The calculation for the bound of the degree of suboptimality for the separate scheme relies on the deterministic nature of the setup. A possibility to handle stochastic noise when determining the bounds should be explored. This could also prove fruitful for finding a way to determine a bound of the degree of suboptimality for the simultaneous scheme.

An example of a large-scale system is the large deformable mirrors presented in for example [Heimsten, 2011]. The system has more than 10000 states and the system matrices are of a sparse nature. Hence, this system is suitable for the methods presented here. Possible improvements compared to the controller given in [Heimsten, 2011] will be investigated.

References

Amodei, L. and J. Buchot (2010): "An invariant subspace method for large-scale algebraic Riccati equation." *Applied Numerical Mathematics*, **60:11**, pp. 1067–1082.

Åström, K. J. (2006): *Introduction to Stochastic Control Theory*. Dover, New York.

Bamieh, B., F. Paganini, and M. A. Dahleh (2002): "Distributed control of spatially invariant systems." *IEEE Transactions on Automatic Control*, **47:7**, pp. 1091–1107.

Bamieh, B. and P. G. Voulgaris (2005): "A convex characterization of distributed control problems in spatially invariant systems with communication constraints." *Systems & Control Letters*, **54:6**, pp. 575–583.

Benner, P. and H. Faßbender (2011): "On the numerical solution of large-scale sparse discrete-time Riccati equations." *Advances in Computational Mathematics*, **35:2**, pp. 119–147.

Benner, P., J.-R. Li, and T. Penzl (2008): "Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems." *Numerical Linear Algebra with Applications*, **15**, November, pp. 755–777.

Beseler, J., J. H. Chow, and K. D. Minto (1992): "A feedback descent method for solving constrained LQG control problems." *Proceedings of the 1992 American Control Conference*, **29**, pp. 1044–1048.

Bini, D., B. Iannazzo, and B. Meini (2011): *Numerical Solution of Algebraic Riccati Equations*. Society for Industrial and Applied Mathematics.

Bryson, A. E. and Y.-C. Ho (1975): *Applied Optimal Control*. Hemisphere Pub. Corp.

Gattami, A. (2006): "Generalized linear quadratic control theory." In *Proceedings of the 45th IEEE Conference on Decision and Control*. San Diego, CA.

Geromel, J. and J. Bernussou (1982): "Optimal decentralized control of dynamic systems." *Automatica*, **18:5**, pp. 545–557.

Harn, Y. and R. Kosut (1993): "Optimal low-order controller design via 'LQG-like' parametrization." *Automatica*, **29:6**, pp. 1377–1394.

Heimsten, R. (2011): *Study of a Large Deformable Mirror Concept*. PhD thesis, Department of Astronomy and Theoretical Physics, Lund University, Sweden.

Hodzic, M., R. Krtolica, and D. Siljak (1983): "A stochastic inclusion principle." *The 22nd IEEE Conference on Decision and Control*, **22**, pp. 17–22.

Hodzic, M. and D. Siljak (1985): "Estimation and control of large sparse systems." *Automatica*, **21:3**, pp. 277–292.

Ikeda, M., D. Siljak, and K. Yasuda (1983): "Optimality of decentralized control for large-scale systems." *Automatica*, **19:3**, pp. 309–316.

Khan, U. A. and J. M. F. Moura (2008): "Distributing the Kalman filters for large-scale systems." *IEEE Transactions on Signal Processing*, **56:10**, pp. 4919–4935.

Lancaster, P. and L. Rodman (1995): *Algebraic Riccati Equation*. Oxford University Press.

Langbort, C., R. S. Chandra, and R. D´Andrea (2004): "Distributed control design for systems interconnected over an arbitrary graph." *IEEE Transactions on Automatic Control*, **49:9**, pp. 1502–1519.

Lasdon, L. S. (2002): *Optimization Theory for Large Systems*. Dover Publications Inc.

Lasiecka, I. and A. Tuffaha (2009): "Riccati theory and singular estimates for a Bolza control problem arising in linearized fluid structure interaction." *Systems and Control Letters*, **58:7**, pp. 499–509.

Li, L. and F. Paganini (2006): "LMI relaxation to Riccati equations in structured $\mathcal{H}_2$ control." In *Proceedings of the 2006 American Control Conference*, pp. 644–649.

Mårtensson, K. and A. Rantzer (2010): "Sub-optimality bound on a gradient method for iterative distributed control synthesis." In *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems*. Budapest, Hungary.

Mårtensson, K. and A. Rantzer (2011a): "A scalable modularized synthesis method for distributed Kalman filters." In *Proceedings of the 18th IFAC World Congress*. Milano, Italy.

Mårtensson, K. and A. Rantzer (2011b): "Synthesis of structured controllers for large-scale systems." *IEEE Transactions on Automatic Control*. Submitted.

Olfati-Saber, R. (2007): "Distributed Kalman filtering for sensor networks." In *Proc. of the 46th IEEE Conference on Decision and Control*.

Rantzer, A. (2006): "A separation principle for distributed control." In *Proceedings of the 45th IEEE Conference on Decision and Control*. San Diego, CA.

Rao, B., H. Durrant-Whyte, and J. Sheen (1993): "A fully decentralized multi-sensor system for tracking and surveillance." *Int. Journal of Robotics Research*, **12:1**, pp. 20–44.

Rao, X., K. Gallivan, and P. Van Dooren (2000): "Riccati equation-based stabilization of large scale dynamical systems." In *Proceedings of the 39th IEEE Conference on Decision and Control*, vol. 5, pp. 4672–4677.

Rotkowitz, M. and S. Lall (2002): "Decentralized control information structures preserved under feedback." In *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 1, pp. 569–575.

Rotkowitz, M. and S. Lall (2006): "A characterization of convex problems in decentralized control." *IEEE Transactions on Automatic Control*, **51:2**, pp. 274–286.

Spanos, D., R. Olfati-Saber, and R. Murray (2005): "Distributed sensor fusion using dynamic consensus." In *Proc. of the 16th IFAC World Congress*. Prague, Czech Republic.

Staffans, O. (1996): "On the discrete and continuous time infinite-dimensional algebraic Riccati equations." *Systems and Control Letters*, **29:3**, pp. 131–138.

Stankovic, S., X.-B. Chen, M. Matausek, and D. Siljak (1999): "Stochastic inclusion principle applied to decentralized automatic generation control." *International Journal of Control*, **72:3**, pp. 276–288.

Stipanovic, D., G. Inalhan, R. Teo, and C. Tomlin (2004): "Decentralized overlapping control of a formation of unmanned aerial vehicles." *Automatica*, **40:8**, pp. 1285–1296.

Swigart, J. and S. Lall (2011): "Optimal controller synthesis for a decentralized two-player system with partial output feedback." In *Proceedings of the 2011 American Control Conference*, pp. 317–323.

Todorov, E. and W. Li (2005): "A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems." *American Control Conference*, **1**, pp. 300–306 vol. 1.

Witsenhausen, H. S. (1968): "A counterexample in stochastic optimum control." *SIAM Journal on Control*, **6:1**, pp. 138–147.

Zhou, K., J. Doyle, and K. Glover (1996): *Robust and Optimal Control*. Prentice-Hall.

Paper III

A Scalable Method for Continuous-Time Distributed Control Synthesis

Karl Mårtensson and Anders Rantzer

Abstract

In this paper a synthesis method for distributed controllers for continuous time distributed systems, is discussed. The systems considered consists of subsystems interconnected in a graph structure. This graph represents a communication structure of the system and hence governs the structure of the admissible controller, meaning that distributed controllers are considered. The objective of the synthesis is to obtain such admissible controllers that optimize a given performance. The method is scalable with respect to the size of the system and is therefore suitable for large-scale systems.

Distributed controllers are suboptimal with respect to centralized ones and it is desirable to measure the amount of performance degradation. Using the variables of the synthesis scheme, it is shown how to determine such a measure of suboptimality.

 To appear in *Proceedings of the 2012 American Control Conference*, Montréal, Canada, 2012.

1. Introduction

Decision making when the decision makers have access to different information concerning underlying uncertainties has been studied since the late 1950s [Marschak, 1955, Radner, 1962]. The subject is sometimes called team theory, sometimes decentralized or distributed control. The theory was originally static, but work on dynamic aspects was initiated by Witsenhausen [Witsenhausen, 1968], who also pointed out a fundamental difficulty in such problems. Some special types of team problems were solved in the 1970's [Sandell and Athans, 1974, Ho and Chu, 1972], but the problem area has recently gain renewed interest. Spatial invariance was exploited in [Bamieh *et al.*, 2002, Bamieh and Voulgaris, 2005], conditions for closed loop convexity were derived in [Rotkowitz and Lall, 2002, Rotkowitz and Lall, 2006] and methods using linear matrix inequalities were given in [Langbort *et al.*, 2004, Rantzer, 2006, Gattami, 2006].

In this paper we will focus on finding a solution to the linear quadratic regulator (LQR) problem for systems in continuous time. The method for finding the centralized solution for this problem has been known for a long time. However, when considering large-scale systems conventional methods for finding this solution are no longer tractable. The reason lies in that the computational time and memory requirements scales as $O(n^3)$ and $O(n^2)$, respectively, for these methods, see for example [Bini *et al.*, 2011]. Methods for large-scale systems needs to exploit some structure in the problem. How to take advantage of the structure of a system with sparse dynamics matrices is presented in [Benner *et al.*, 2008]. Here it is also assumed that the number of input signals is a lot less than the number of states. The resulting controllers will approximately solve the centralized control problem. When considering distributed systems, they usually have a communication constraint, meaning that a subsystem does not have access to the full state vector. With the use of the augmented Lagrangian method, iterative schemes to find structured state feedback matrices minimizing a quadratic cost, are treated in [Wenk and Knapp, 1980, Lin *et al.*, 2011]. The methods are initialized with the centralized solution for the problem, and recursively the solution approaches a structured feedback matrix. The methods hinge on the solutions of Lyapunov equation and the solution of the centralized problem, thus not applicable to large-scale systems. In [Mårtensson and Rantzer, 2010] an iterative distributed control synthesis scheme for discrete-time systems is considered. We will follow a similar approach when developing the theory in this paper. The synthesis method operates by iteratively updating the controller in a descent direction of the LQR performance. This direction is determined by simulating the system and the corresponding adjoint system. When the system matrices are sparse it is realized that this produces a

scalable method, hence suitable for large-scale systems. Since distributed controllers are suboptimal compared to centralized solutions, it is desirable to have a measure of suboptimality. We show how to use the variables of the synthesis algorithm to determine a bound of suboptimality of the current controller.

Section 2 contains a description of the distributed systems considered and the notations used in the paper are defined. In section 3 the method for updating the control laws using descent directions to the cost function is presented. In section 4 the theory for finding the suboptimality bound to the previously mentioned method, is formulated. An example is given in section 5, showing the described methodology.

2. Problem Formulation

The systems treated in this paper are continuous time, linear time invariant systems

$$\dot{x}(t) = Ax(t) + Bu(t), \quad x(0) = x_0, \tag{1}$$

where $x(t) \in \mathbb{R}^m$, $u(t) \in \mathbb{R}^p$ and $x_0 \in \mathcal{N}(0, \sigma)$. We will assume that the system is distributed, a property that will be explained by a graph associated to the system. The vertices $v_1, v_2, \ldots, v_n$ of the graph represent subsystems or *agents* of the complete system. Hence, the vertices can be thought as a partition of all of the states. If the states are rearranged such that the states of each subsystem are next to each other, we write $x = [\, x_1^T \quad x_2^T \quad \ldots x_n^T \,]^T$ where x_i are the states of subsystem i. The sparsity structure of the system is now defined by the edges of the graph. The collection of all edges is denoted by $\mathcal{E}$, where $(i, j) \in \mathcal{E}$ if there is an edge from v_i to v_j. By convention we assume that $(i, i) \in \mathcal{E}$ for all i. We call two distinct subsystems neighbors if there is an edge between the corresponding vertices. The edges describe the sparsity of the dynamics matrix and its structure is restricted by

$$A_{ij} = 0 \quad \text{if} \quad (j, i) \notin \mathcal{E},$$

(throughout the paper the subscript will denote blocks of the intended matrices corresponding to the subsystems). Hence, the dynamics of a subsystem is only directly affected by the states of the subsystem and its neighbors. The subsystems will also be assumed to have a distinct set of control signals, i.e. each control signal affects only one subsystem directly. This assumption is translated to assuming that the matrix B is block-diagonal, i.e. $B = \text{diag}(B_1, B_2, \ldots, B_n)$.

With these initial definitions, an example of the setup is given in Figure 1.

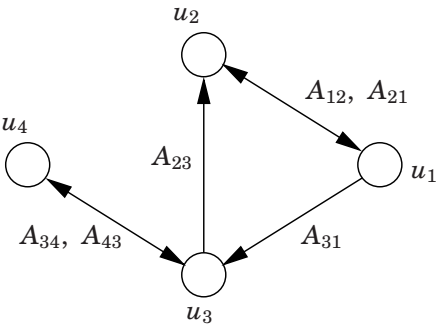


Figure 1. Graphical representation of a distributed system. The arrows shows how each agent directly affects the others. The set $\mathcal{E} = \{(1,1),(2,2),(3,3),(4,4),(1,2),(2,1),(1,3),(3,2),(3,4),(4,3)\}$

The system (1) is controlled using state feedback $u(t) = -Kx(t)$. The graph also imposes a communication constraint on the system and the admissible controllers are only those where each subsystem uses only the states of itself and its neighbors to determine its control input. This restriction translates to a structure in the feedback matrix according to

$$K_{ij} = 0 \quad \text{if} \quad (j,i) \notin \mathcal{E}.$$

The subspace of admissible controllers is denoted by

$$\mathcal{K} = \{K \mid \forall K \text{ such that } K_{ij} = 0 \text{ if } (j,i) \notin \mathcal{E}\}.$$

The set of admissible *stabilizing* controllers is denoted by

$$\mathcal{K}_{\text{stab}} = \{K \mid \forall K \in \mathcal{K} \text{ and } K \text{ is stabilizing}\}.$$

With the restrictions of the matrices A, B and K, the closed loop dynamics matrix $A - BK$ has the same structure as A, and hence satisfies the structural constraint that the graph gives. That the closed loop dynamics matrix still is sparse will prove valuable later on when looking into the complexity of the coming method.

3. Iterative Distributed Control Synthesis

The objective of the control synthesis is to determine a feedback matrix that minimizes some performance. The performance that is considered is

the commonly known LQR cost for continuous time systems

$$J(K, x_0) = \int_0^\infty x(t)^T Q_x x(t) + u(t)^T Q_u u(t) dt, \tag{2}$$

where $x(t)$ satisfies the dynamics equation (1) and $u(t) = -Kx(t)$. The weights Q_x and Q_u are assumed to be block-diagonal meaning that it is possible to separate the cost into costs for each subsystem. For all stabilizing K, the cost (2) can be determined by solving certain Lyapunov equation. Specifically, we have that $J(K, x_0) = \mathrm{tr}\left((Q_x + K^T Q_u K) X_0\right) = \mathrm{tr}\left(P x_0 x_0^T\right)$ where X_0 and P are the solutions to the following Lyapunov equations, respectively.

$$(A - BK) X_0 + X_0 (A - BK)^T + x_0 x_0^T = 0, \tag{3}$$

$$(A - BK)^T P + P(A - BK) + Q_x + K^T Q_u K = 0. \tag{4}$$

With these solutions an expression for the gradient of J with respect to K can also be determined.

PROPOSITION 1
Given the system (1) and a stabilizing K, the gradient of the cost function (2) with respect to K is

$$\nabla_K J = 2(Q_u K - B^T P) X_0. \tag{5}$$

□

Proof. For simpler expression we define the matrices

$$A_K = A - BK,$$
$$M = Q_u K - B^T P.$$

By differentiating (4) with respect to K we get the following Lyapunov equation

$$A_K^T dP + dP A_K + dK^T M + M^T dK = 0.$$

The integral solution to this equation is

$$dP = \int_0^\infty e^{t A_K^T} (dK^T M + M^T dK) e^{t A_K} dt.$$

Now, since $dJ = \mathrm{tr}\left(dP x_0 x_0^T\right)$, we get

$$\begin{aligned} dJ &= \mathrm{tr}\left(\int_0^\infty e^{t A_K^T} (dK^T M + M^T dK) e^{t A_K} x_0 x_0^T dt \right) \\ &= 2 \cdot \mathrm{tr}\left(dK^T M \int_0^\infty e^{t A_K} x_0 x_0^T e^{t A_K^T} dt \right) \\ &= 2 \cdot \mathrm{tr}\left(dK^T M X_0 \right). \end{aligned}$$

By using the relation about differentials

$$dZ = \mathrm{tr}\left(dX^T \cdot Y\right) \implies \nabla_X Z = Y,$$

relation (5) is verified. □

In order to calculate the gradient of J with respect to K using the result of Proposition 1, the Lyapunov equations (3) and (4) needs to be solved. If large-scale systems are considered it is not possible to solve these equations in a reasonable time. Hence, for a scalable method these solutions we need to find an expression that does not rely on these solutions. In the next proposition adjoint variables are introduced and it is shown how to use them to get rid of X_0 and P from (5).

THEOREM 1
Given the system (1) and a stabilizing K, let the adjoint states λ be defined by

$$-\dot{\lambda}(t) = (A - BK)^T \lambda(t) - (Q_x + K^T Q_u K)x(t), \tag{6}$$

where $x(t)$ are the states of (1) and $\lim_{t\to\infty} \lambda(t) = 0$. Then

$$\nabla_K J = 2 \int_0^\infty \left(-Q_u u(t) + B^T \lambda(t)\right) x(t)^T dt. \tag{7}$$

□

Proof. By denoting $Q_K = Q_x + K^T Q_u K$, the adjoint states can be determined

$$\lambda(t) = \int_t^\infty -e^{(s-t)A_K^T} Q_K x(s) ds.$$

With this expression we can rewrite to following

$$\begin{aligned}
\int_0^\infty \lambda(t)x(t)^T dt &= \int_0^\infty \int_t^\infty -e^{(s-t)A_K^T} Q_K x(s) ds \cdot x(t)^T dt \\
&= \int_0^\infty \int_t^\infty -e^{(s-t)A_K^T} Q_K e^{(s-t)A_K} x(t) ds \cdot x(t)^T dt \\
&= -PX_0.
\end{aligned}$$

This relation and that $Q_u K X_0 = -Q_u \int_0^\infty u(t)x(t)^T dt$ fitted into (5) gives the desired result. □

REMARK 1
The dynamical system for the adjoint variables is stable when considering time going from the future backwards, i.e. from $t = \infty$ to $t = 0$. Hence, it is simulated in the backwards time direction. □

The gradient gives a direction in which the feedback matrix K can be updated in to decrease the cost $J(K, x_0)$. Though, since we impose a structure on K, the gradient $\nabla_K J$ needs to be projected to the subspace $\mathcal{K}$ defining this structure. This projected gradient will also produce a descent direction of $J(K, a)$. To understand this, consider the restriction of J on $\mathcal{K}$. The gradient of this function is exactly the projection of $\nabla_K J$ on $\mathcal{K}$.

In order to get a tractable algorithm, the time for simulating the states in (1) and (6) must be truncated to some finite time t_{final}. The truncation implies that an approximation of the gradient in Theorem 1 will be determined. The algorithm of iteratively updating the feedback matrix is given below.

ALGORITHM 1
Consider the system (1) with control $u(t) = -Kx(t)$ where $K \in \mathcal{K}_{\text{stab}}$. To find a local minimizer to (2), start with $K^{(0)} \in \mathcal{K}_{\text{stab}}$ and for each $\tau \geq 0$

1. Simulate the states $x(t)$ of (1) with control $u(t) = -K^{(\tau)}x(t)$ for times $t \in [0, t_{\text{final}}]$.
2. Simulate the adjoint states $\lambda(t)$ of (6) with for times $t \in [0, t_{\text{final}}]$ in the backwards time direction with $\lambda(t_{\text{final}}) = 0$.
3. For all agents i and all j such that $(j, i) \in \mathcal{E}$

   I) Calculate

$$G_{ij} = 2 \int_0^{t_{\text{final}}} \left(-[Q_u]_i u_i(t) + B_i^T \lambda_i(t) \right) x_j(t)^T dt.$$

   II) Update the feedback matrix

$$K_{ij}^{(\tau+1)} = K_{ij}^{(\tau)} - \gamma G_{ij},$$

   for some step length γ.

4. Increase τ with 1 and goto 1).

□

Remark 2
As previously mentioned the closed loop matrix $A-BK$ follow the sparsity pattern described by the graph associated with the distributed system. Examining the matrix $Q_x + K^T Q_u K$ we find that it also has a distributed structure related to the graph. Hence, if the distributed system is sparse we understand that, using a sparse ODE solver, Algorithm 1 benefits from this structure. In fact, the scheme is linear in complexity when considering systems consisting of subsystems with the same average state space size and number of neighbors. This can be compared to solving Lyapunov equations which in standard implementation requires $O(n^3)$ flops. This means that the alternate version of Algorithm 1 where the gradient instead is determined through (5) would not be tractable for large-scale systems. □

Remark 3
In order to approximate the gradient in Algorithm 1 a final time t_{final} needs to be determined to ensure that the approximation is still a descent direction. For any descent direction D, $\operatorname{tr}\left(\nabla_K J^T \cdot D\right) < 0$ must hold. Letting G be the truncated gradient and $H = \nabla_K J - G$. Then G is a descent direction if $\operatorname{tr}\left((G+H)^T G\right) < 0$, that is $\operatorname{tr}\left(G^T G\right) < \operatorname{tr}\left(H^T G\right)$. Since $\operatorname{tr}\left(G^T G\right)$ can be determined, a valid final time would be one for which it is possible to determine a bound on H in order for the inequality to hold. A strategy could be to analyse the decrease in the state trajectory to find such bound. This is an issue that needs further attention. □

4. Suboptimality Bound

Solving the ordinary LQR control problem is a well-studied problem and has a tractable solution when the system is of moderate size. But when we introduce restrictions in the structure of the feedback matrix, there is no general formula for finding the optimal one. The minimization problem is not even guaranteed to be convex. The underlying method in Algorithm 1 is a descent method, thus we can not guarantee that the globally optimal structured feedback matrix is ever attained. We only know that a locally optimal solution is reached. A measure of the suboptimality in each iteration step of the Algorithm 1, is $\alpha \geq 1$ such that

$$J(K, x_0) \leq \alpha J(K_{\text{opt}}, x_0), \tag{8}$$

where $K_{\text{opt}} = \arg\min_K J(K, x_0)$. The value of α tells us that the cost of the feedback matrix, $J(K, x_0)$, is within a factor α of the optimal value.

If there is a way to verify that an α close to 1 must satisfy (8), then even though K might not be the optimal feedback matrix, we will not find one that reduces the cost greatly compared to this one. Hence, the suboptimality bound can be used as a stop criterion for Algorithm 1. That is when the bound goes below a given value we consider that the current feedback matrix is satisfactory and return it from the algorithm.

To determine the suboptimality bounds, we start by define the truncated cost

$$\hat{J}(K, x_0, t_{\text{final}}) = \int_0^{t_{\text{final}}} x(t)^T Q_x x(t) + u(t)^T Q_u u(t) dt, \tag{9}$$

where $x(t)$ satisfies (1) and $u(t) = -Kx(t)$. The following theorem gives us a suboptimality bound telling us that in the time interval $[0, t_{\text{final}}]$ we are within a factor of α of the optimal solution on this interval.

THEOREM 2
If $\alpha \geq 1$ is such that for a given trajectory of adjoint (or dual) variables $\lambda(t)$, with $\lambda(t_{\text{final}}) = 0$

$$\begin{aligned}\hat{J}(K, x_0, t_{\text{final}}) \leq \alpha \min_{\substack{x,u \\ x(0)=x_0}} \int_0^{t_{\text{final}}} \Big[& x(t)^T Q_x x(t) \\ & + u(t)^T Q_u u(t) + 2\lambda(t)^T \left(\dot{x}(t) - Ax - Bu(t)\right) \Big] dt,\end{aligned} \tag{10}$$

then

$$\hat{J}(K, x_0, t_{\text{final}}) \leq \alpha \hat{J}(K_{\text{opt}}, x_0, t_{\text{final}}), \tag{11}$$

where

$$K_{\text{opt}} = \arg\min_K \hat{J}(K, x_0, t_{\text{final}}).$$

□

Proof. Assume that α is such that for a given trajectory of $\lambda(t)$, (10) holds.

We have that

$$\begin{aligned}
\hat{J}(K_{\text{opt}}, x_0, t_{\text{final}}) &= \begin{cases} \min\limits_{K,x} \int_0^{t_{\text{final}}} x(t)^T (Q_x + K^T Q_u K) x(t) dt \\ \text{subject to: } \dot{x}(t) = (A - BK)x(t) \\ \qquad\qquad x(0) = x_0 \end{cases} \\
&\geq \begin{cases} \min\limits_{x,u} \int_0^{t_{\text{final}}} x(t)^T Q_x x(t) + u(t)^T Q_u u(t) dt \\ \text{subject to: } \dot{x}(t) = Ax(t) - Bu(t) \\ \qquad\qquad x(0) = x_0 \end{cases} \\
&\geq \min_{\substack{x,u \\ x(0)=x_0}} \int_0^{t_{\text{final}}} \Big(x(t)^T Q_x x(t) + u(t)^T Q_u u(t) \\
&\qquad\qquad + 2\lambda(t)^T (\dot{x}(t) - Ax(t) - Bu(t)) \Big) dt,
\end{aligned}$$

where the second inequality comes from introducing dual variables. Hence, if (10) holds, so must (11). □

The Theorem 2 gives a method to evaluate the expected performance an updated feedback matrix will give to the system. We only have to choose the adjoint or dual variables. The name suggest that we choose the adjoint variables defined by (6). To motivate this choice, we could refer to Pontryagin's maximum principle. The motivation comes from examining the Hamiltonian

$$\begin{aligned}
\max_{\lambda} \min_{x,u} \int_0^{t_{\text{final}}} \Big(& x(t)^T Q_x x(t) + u(t)^T Q_u u(t) \\
& \underbrace{+ 2\lambda(t)^T (\dot{x}(t) - Ax(t) - Bu(t)) \Big)}_{\mathcal{H}(x,u,\lambda)} dt
\end{aligned} \tag{12}$$

from Theorem 2. We let the objective function be denoted by $\mathcal{H}(x, u, \lambda)$. To find a saddle point for $\mathcal{H}$ then

$$\begin{aligned}
0 &= \nabla_{x(t)} \mathcal{H} = 2(Q_x x(t) - \dot{\lambda}(t) - A^T \lambda(t)), \\
0 &= \nabla_{u(t)} \mathcal{H} = 2(Q_u u(t) - B^T \lambda(t)).
\end{aligned}$$

We get (6) by $\nabla_{x(t)} \mathcal{H} + K^T \nabla_{u(t)} \mathcal{H} = 0$ and replacing $u(t) = -Kx(t)$.

To show that it is actually possible to solve the minimization program in (10) we give the following proposition.

PROPOSITION 2
The value of the minimization program in (10) can be determined by

$$-\int_0^{t_{\text{final}}} \Big[\left(\dot{\lambda}(t) + A^T\lambda(t)\right)^T Q_x^{-1} \left(\dot{\lambda}(t) + A^T\lambda(t)\right) + \lambda(t)^T B Q_u^{-1} B^T \lambda(t) \Big] dt - 2\lambda(0)^T x_0.$$

□

Proof. Introduce $f = Q_x x(t) - \dot{\lambda}(t) - A^T\lambda(t)$ and $g = Q_u u(t) - B^T\lambda(t)$. The integral in the objective in (10) is manipulated:

$$\begin{aligned}
&\int_0^{t_{\text{final}}} \Big[x(t)^T Q_x x(t) + u(t)^T Q_u u(t) + 2\lambda(t)^T \left(\dot{x}(t) - Ax - Bu(t)\right) \Big] dt \\
&= \int_0^{t_{\text{final}}} \Big[x(t)^T Q_x x(t) + u(t)^T Q_u u(t) \\
&\qquad - 2\dot{\lambda}(t)^T x(t) - \lambda(t)^T \left(Ax + Bu(t)\right) \Big] dt + 2\Big[\lambda(t)^T x(t)\Big]_{t=0}^{t_{\text{final}}} \\
&= \int_0^{t_{\text{final}}} \Big[f^T Q_x^{-1} f + g^T Q_u^{-1} g - \lambda(t)^T B Q_u^{-1} B^T \lambda(t) \\
&\qquad - \left(\dot{\lambda}(t) + A^T\lambda(t)\right)^T Q_x^{-1} \left(\dot{\lambda}(t) + A^T\lambda(t)\right) \Big] dt - 2\lambda(0)^T x_0,
\end{aligned}$$

where the first equality comes from partial integration and the second equality from completing the squares. When minimizing the last expression we understand that choosing $x(t)$ and $u(t)$ to make $F = G = 0$ minimizes the integral (the only point on the trajectory of $x(t)$ that can not be chosen is $x(0)$ but this point does not change the value of the integral). □

5. Example

In this example we consider a small-scale system. The reason for not using a large-scale system is to be able to verify the actual cost of the feedback matrices by determining the corresponding Lyapunov solution. The system consists of 10 subsystems, each with one state, connected in a linear fashion as shown in Figure 2. This structure of the graph results

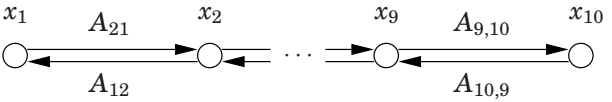


Figure 2. Graphical representation of the system in the example. The arrows shows how each agent affects the others.

in a tri-diagonal dynamics matrix. In this example its value is

$$A = \begin{bmatrix} -4 & 1 & & & & & & & & \\ 1 & -4 & -1 & & & & & & & \\ & -2 & -4 & 1 & & & & & & \\ & & -1 & -8 & 8 & & & & & \\ & & & 1 & -2 & 2 & & & & \\ & & & & -1 & -9 & -1 & & & \\ & & & & & -2 & -5 & -2 & & \\ & & & & & & -1 & -8 & -2 & \\ & & & & & & & 7 & -10 & -1 \\ & & & & & & & & -1 & -4 \end{bmatrix}$$

and the remaining entries equal zero. The matrix B is diagonal and given by

$$B = \operatorname{diag}\left(\left[-1,\, 2,\, 6,\, -3,\, -6,\, -1,\, 3,\, -4,\, 7,\, 2\right]\right).$$

The weights $Q_x = Q_u = I$ and $x_0 \in \mathcal{N}(0, I)$.

The system is initially stable (the largest real part of the eigenvalues of A is -1.18), meaning that we can start Algorithm 1 with $K^{(0)} = 0$. The algorithm is used for 100 iterations where the systems are simulated between times $[0, 10]$ in each iteration. The method for estimating suboptimality bounds is performed in each update iteration. Also, the actual suboptimality is determined by solving the appropriate Lyapunov equation with the current feedback matrix K. The result of a simulation is given in Figures 3-4.

In Figure 3 the estimated suboptimal bound is denoted by α and shown in blue. A first remark is that in the first iteration we get a negative value of the suboptimality bound. This is due to the fact that the minimization program in (10) is not guaranteed to give a positive value. In case a negative value is obtained nothing can be said about the suboptimality with this method. Though, as we get closer to the optimal feedback matrix, the adjoint trajectory will approach the optimal (with respect to (12)) and the inequalities in the proof of Theorem 2 will almost be equal implying that we can expect a positive value from (10).

When positive, the suboptimality bound is always larger than 1 which is natural. In the same figure denoted by α_{exact} and shown in green, is the true suboptimality determined by

$$\alpha_{\text{exact}} = \frac{J(K^{(k)}, x_0)}{J(K_{\text{opt}}, x_0)}.$$

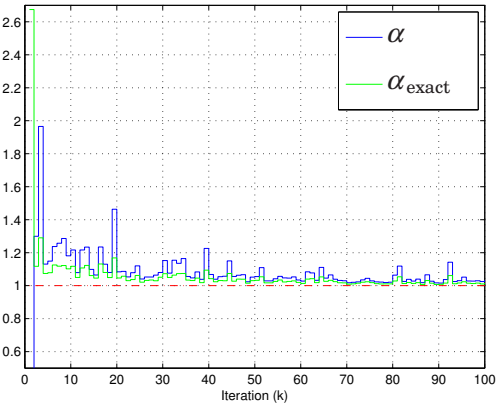


Figure 3. Plots of the estimated sub-optimality using the described method and the exact suboptimality.

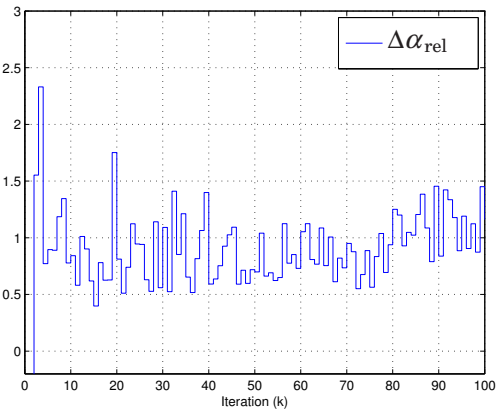


Figure 4. Plot of the relative difference between the estimated and the exact suboptimality.

As expected the suboptimality bound (when positive) is also always larger than the true suboptimality. As the true suboptimality approaches 1, that is the cost with the feedback matrix approaches the optimal cost, the suboptimality bound also approaches 1.

In Figure 4 the relative difference between the suboptimality bound

and the true suboptimality is shown. The relative difference is determined by

$$\Delta\alpha_{\text{rel}} = \frac{\alpha - \alpha_{\text{exact}}}{\alpha_{\text{exact}} - 1}.$$

As can be seen in the figure, the relative difference is in most iteration below 1.5 meaning that the suboptimality bound is not more than a factor 1.5 more from the true suboptimality (when the true suboptimality is measured as the distance to 1).

6. Conclusions and Future Work

6.1 Conclusions

In this paper a scalable method for doing synthesis of distributed controllers for linear, continuous time, distributed systems. The objective is to obtain a linear, structured controller which minimizes the LQR cost. The method works in an iterative fashion where, in each iteration, a descent direction (with respect to the cost) is determined by simulating the system and the corresponding adjoint system. The controller is then updated by a step in that direction. The fact that the method relies on simulation of distributed, sparse system shows that it is scalable with respect to the number of subsystems.

In each iteration the trajectories are also used to determine a bound of the suboptimality of the current controller. This bound gives qualitative information of the current controller, and can for example be used as a stopping criteria for the synthesis method.

6.2 Future Works

How to determine a valid final time in the simulations to guarantee that the approximated gradient still is a descent direction, needs further attention. For more discussion, see Remark 3.

We will work on connecting the state feedback synthesis with the similar observer synthesis to get a method for output feedback synthesis. We will for example look at what happens to the suboptimality bounds in this case.

7. Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agree-

ment number 224428, acronym CHAT. This work was also supported by the Linnaeus Grant LCCC from the Swedish Research Council.

References

Bamieh, B., F. Paganini, and M. A. Dahleh (2002): "Distributed control of spatially invariant systems." *IEEE Transactions on Automatic Control*, **47:7**, pp. 1091–1107.

Bamieh, B. and P. G. Voulgaris (2005): "A convex characterization of distributed control problems in spatially invariant systems with communication constraints." *Systems & Control Letters*, **54:6**, pp. 575–583.

Benner, P., J.-R. Li, and T. Penzl (2008): "Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems." *Numerical Linear Algebra with Applications*, **15**, November, pp. 755–777.

Bini, D., B. Iannazzo, and B. Meini (2011): *Numerical Solution of Algebraic Riccati Equations*. Society for Industrial and Applied Mathematics.

Gattami, A. (2006): "Generalized linear quadratic control theory." In *Proceedings of the 45th IEEE Conference on Decision and Control*. San Diego, CA.

Ho, Y.-C. and K.-C. Chu (1972): "Team decision theory and information structures in optimal control problems–Part I." *IEEE Transactions on Automatic Control*, **17:1**, pp. 15–22.

Langbort, C., R. S. Chandra, and R. D´Andrea (2004): "Distributed control design for systems interconnected over an arbitrary graph." *IEEE Transactions on Automatic Control*, **49:9**, pp. 1502–1519.

Lin, F., M. Fardad, and M. Jovanovic (2011): "Augmented lagrangian approach to design of structured optimal state feedback gains." *IEEE Transactions on Automatic Control*, **56:12**, pp. 2923–2929.

Marschak, J. (1955): "Elements for a theory of teams." *Management Sci.*, **1**, pp. 127–137.

Mårtensson, K. and A. Rantzer (2010): "Sub-optimality bound on a gradient method for iterative distributed control synthesis." In *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems*. Budapest, Hungary.

Radner, R. (1962): "Team decision problems." *Ann. Math. Statist.*, **33:3**, pp. 857–881.

Rantzer, A. (2006): "A separation principle for distributed control." In *Proceedings of the 45th IEEE Conference on Decision and Control*. San Diego, CA.

Rotkowitz, M. and S. Lall (2002): "Decentralized control information structures preserved under feedback." In *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 1, pp. 569–575.

Rotkowitz, M. and S. Lall (2006): "A characterization of convex problems in decentralized control." *IEEE Transactions on Automatic Control*, **51:2**, pp. 274–286.

Sandell, N. R. and M. Athans (1974): "Solution of some nonclassical LQG stochastic decision problems." *IEEE Transactions on Automatic Control*, **19:2**, pp. 108–116.

Wenk, C. and C. Knapp (1980): "Parameter optimization in linear systems with arbitrarily constrained controller structure." *IEEE Transactions on Automatic Control*, **25:3**, pp. 496–500.

Witsenhausen, H. S. (1968): "A counterexample in stochastic optimum control." *SIAM Journal on Control*, **6:1**, pp. 138–147.

Paper IV

Synthesis of Structured State Feedback Controllers for a Large Deformable Mirror

Karl Mårtensson and Rikard Heimsten

Abstract

Replacing relay optics located late in the optical chain, by a deformable mirror in the telescope itself, improves the resolution of the obtained image. Implementing deformable mirrors, especially in extremely large telescopes, introduces another problem. The models needed to describe these mirrors have a very large state space and have a large number of actuators and sensors. For such systems it is far from trivial to design optimal controllers. In this paper we implement a method of finding optimal feedback matrices for a quadratic cost. The obtained controller is used to make the mirror track changes in the phase of the incoming light. The performance of the controller is for example measured in its ability to track both Zernike polynomials and phase screens, a model of the added phase by the atmosphere. In these measures the controller exhibits promising results.

1. Introduction

When observing light from distant stars and planets with ground-based telescopes, it is desirable to correct for the rapidly changing optical distortions introduced by the Earth's atmosphere. The turbulence in the atmosphere affects the phase of the incoming wavefront, which results in a blurred image. Adaptive optics (AO) is a technique used to increase the resolution of optical systems. The objective of such a system is to counteract the distortion in the wavefronts, thus enhancing the image quality. An AO system includes three basic components; a wavefront sensor, a control system, and one or more corrective elements; often implemented as deformable mirrors. By changing the shape of the mirror to correct for the aberrations in the wavefront of the incoming light, the image quality is increased. For an introduction to adaptive optics, see for example [Andersen and Enmark, 2011, Hardy, 1998, Roddier, 1999]. Traditionally, the AO systems consist of a small deformable mirror with piezoelectric actuators located late in the optical chain. For example, [von Bokern *et al.*, 1992, Paschall and Anderson, 1993] shows how to design controllers for such deformable mirrors using linear quadratic Gaussian (LQG) technique. In the procedures they use a system describing the 14 first Zernike modes (excluding the piston mode) to approximate the atmospheric turbulence.

Integrating the deformable mirror into the telescope design, thus going from tens of millimeters to a few meters in diameter, will avoid lossy relay optics and make the telescope more compact. Large deformable mirrors have successfully been integrated into the Multi Mirror Telescope [Wildi *et al.*, 2003] and the Large Binocular Telescope [Quiros-Pacheco *et al.*, 2010]. Large deformable mirrors are generally implemented as thin shells of a glass ceramic with hundreds or thousands of actuators on the back. Often, there are also local deflection sensors on the back of the mirror, resulting in a multiple input multiple output system.

When choosing to implement large deformable mirrors a problem of scalability arises. The size of the systems describing the mirror becomes of such an order that many of the conventional methods are not applicable to design the controllers. As in most large-scale optimization problems, it is necessary to find and take advantage of structures in the dynamics matrices [Lasdon, 2002]. In [Miller and Grocott, 1999] the deformable mirror is modeled by the use of circulant matrices. It is then possible to completely decouple the system and find a LQG controller for each decoupled subsystem. Another scheme that relies on decoupling can be found in [Andersen *et al.*, 2006, Heimsten, 2011]. The actuators are collected in families and the force distributions for different mirror deflections are determined to form a decoupling matrix. A decentralized controller for this

decoupled system can now be designed. In [Fraanje *et al.*, 2010] the mirror is modeled by identical subsystems interconnected in a sparse graph structure. In this setup it is shown how to obtain a distributed controller for a mirror of any size. An LQG design method for a hybrid model is suggested in [Looze, 2009]. The structure of the model is used to reduce the dimension of the involved Riccati equations. [Roux *et al.*, 2004] shows how to design minimal variance controllers for both AO and multiconjugate AO (MCAO) systems, for which there are a larger number of degrees of freedom than classical AO. [Petit *et al.*, 2005] deals with the problem of wavefront reconstruction, open and closed loop optimal control for MCAO systems. A thorough analysis and validation of LQG controllers for MCAO systems is performed in [Petit *et al.*, 2009]. Another experimental validation of a $\mathcal{H}_2$ optimal control strategy is given in [Hinnen *et al.*, 2007].

In this paper we will apply a scheme to determine a state feedback matrix for the linear quadratic regulator (LQR) control problem for large-scale systems, presented in [Mårtensson and Rantzer, 2011, Mårtensson and Rantzer, 2010]. The model of the mirror developed in [Heimsten, 2011] has sparse dynamics matrices and it will be the model used in this paper. The control synthesis method takes advantage of sparsity structures in the dynamics matrix of the system to determine a sparse feedback matrix that minimizes the LQR cost. This is accomplished by determining gradients to the cost function and updating the feedback matrix in these directions. The gradients are determined through a combination of the trajectories of simulations of the mirror from a non-zero initial state and its adjoint system. This updating process will create a sequence of feedback matrices with decreasing cost. The advantage with this method compared to conventional methods solving the LQR problem is that the computation time only grows linearly with the size of the system, thus making it applicable on large-scale systems. Also, we do not have to assume any other structures on the system except for that it is sparse. Since the obtained controller is distributed, it can be implemented on separate CPUs, which will increase robustness and flexibility.

The paper is organized as follows: In Section 2 the continuous model of the deformable mirror is derived. Since the method to design controllers in this paper relies on the system being posed in discrete time, we also show how we discretize the model to retain its sparsity structure. The general method used to design the controllers for the mirror is described in Section 3. We give the parameters and the setup of using the synthesis method to compute controllers for the mirror model in Section 4. Some implementation issues are also discussed here. In Section 5 the performance of the controllers derived in the previous section are investigated. Finally we give some conclusion and future work in Section 6.

1.1 Mathematical Notation

The set of real numbers is denoted by $\mathbb{R}$, real vectors of dimension n by $\mathbb{R}^n$ and real $n \times m$ matrices by $\mathbb{R}^{n\times m}$. The zero matrix with the appropriate size will be referred to as 0. When a partition of a vector or a matrix has been defined, subscripts will refer to that partition. For example, for $x \in \mathbb{R}^n$ then $x_i \in \mathbb{R}^{n_i}$ refers to the ith partition of x and $A \in \mathbb{R}^{n\times m}$ then $A_{i,j} = A_{ij} \in \mathbb{R}^{n_i \times m_j}$ refers to the i,jth partition of A. When the components of the vectors or matrices are ordered with respect to the partition, x_i or A_{ij} equivalently means the ith or i,jth *block* of x or A, respectively. For a matrix valued function $f : \mathbb{R}^{n\times m} \to \mathbb{R}$ we define the differential df as the part of $f(X + dX) - f(X)$ that is linear in dX, or in other words, the linearized part of f. The gradient of f with respect to X is denoted $\nabla_X f$ and means

$$\nabla_X f = \begin{bmatrix} \frac{\partial f}{\partial X_{1,1}} & \frac{\partial f}{\partial X_{1,2}} & \cdots & \frac{\partial f}{\partial X_{1,m}} \\ \frac{\partial f}{\partial X_{2,1}} & \frac{\partial f}{\partial X_{2,2}} & \cdots & \frac{\partial f}{\partial X_{2,m}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial X_{n,1}} & \frac{\partial f}{\partial X_{n,2}} & \cdots & \frac{\partial f}{\partial X_{n,m}} \end{bmatrix}$$

For a pair of $(A, B) \in \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times p}$ we say that the matrix $K \in \mathbb{R}^{p\times n}$ stabilizes (A, B) if $A - BK$ has all its eigenvalues in the unit circle. A pair (A, B) is said to be stabilizable if such K exists.

We use the notation $\mathcal{N}(\mu, \sigma^2)$ for Gaussian noise with mean μ and variance σ^2.

2. Modeling the Deformable Mirror

The dynamic behavior of large thin-plate deformable mirrors can be modeled mathematical either by partial differential equations based on the principle of virtual work or by linear differential equations based on Newton's laws using the method of finite elements. The later is the most common discretization technique in structural mechanics. Finite element analysis is formed by setting the inertia forces minus the force related to friction and elasticity equal to the external force for the nodes of the finite elements [Reddy, 2006]. This analysis gives a model on the form

$$\mathcal{M}\ddot{\boldsymbol{\xi}} + \mathcal{C}\dot{\boldsymbol{\xi}} + \mathcal{K}\boldsymbol{\xi} = \boldsymbol{F} \tag{1}$$

where $\mathcal{M}$ is the mass matrix, $\mathcal{C}$ the damping matrix, $\mathcal{K}$ the stiffness matrix, $\boldsymbol{F}$ a time-dependent force vector for the external forces, and $\boldsymbol{\xi}$

Parameter	Definition	Value
E	Young's modulus	63×109 Pa
ρ	density	2.23×103 kg/m^3
ν	Poisson ratio	0.2
h	DM thickness	2 mm
r	radius of the mirror	0.5 m

Table 1. Parameter values for a 1 m deformable mirror used for the case study.

a vector holding translational and angular displacements. The finite element model of the deformable mirror was derived using the software COMSOL Multiphysics, where $\mathcal{M}$ and $\mathcal{K}$ are determined. How the matrix $\mathcal{C}$ is determined will be described below.

The case study is concerned with a 1 m faceplate made of borosilicate. The 2 mm thick mirror is constrained at the inner rim, which has a diameter of 5 mm. The physical parameters are given in Table 1. Actuators with an actuator pitch of 45 mm, are placed in a Cartesian topology, resulting in a total of 372 actuator nodes. The actuators are modeled as ideal systems, meaning that we assume that the actuators does not have any dynamics. This paper will not focus on state reconstruction and thus we will assume that all states of all nodes are available when controlling the system.

The spatially discretized model of the mirror (1) obtained after the finite element analysis in this case study, has 6128 discretization points or nodes. The matrix $\mathcal{C}$ is determined by Rayleigh damping model, that is, letting $\mathcal{C} = \mu\mathcal{M} + \lambda\mathcal{K}$ for the parameters μ and λ. How the parameters μ and λ are chosen will be explained in the following discussion. When determining the matrices $\mathcal{M}$ and $\mathcal{K}$ through COMSOL Multiphysics, we also obtain the norm of the eigenvalues of the first 2519 stable modes of (1), which are independent of $\mathcal{C}$. For each mode, letting ω_i be the norm of the eigenvalue with corresponding eigenvector v_i, we have that $\mathcal{K}v_i = \omega_i^2\mathcal{M}v_i$. Without loss of generality, we assume mass normalization of the eigenvectors such that $v_i^T\mathcal{M}v_i = 1$. If we would assign the dampings ζ_i to every mode i, $\mathcal{C}$ must be chosen such that the relations $v_i^T\mathcal{C}v_i = 2\omega_i\zeta_i$ hold. With the Rayleigh damping model this implies that $\mu + \lambda\omega_i^2 = 2\omega_i\zeta_i$. Given the modal dampings ζ_i, it may not possible to satisfy the equations for all i. Instead, we will choose the parameters μ and λ to minimize the average distance to a damping of $\zeta_i = 0.02$ for the first 2519 modes, while not letting any damping be less than 0.005, that is, $\zeta_i \geq 0.005$. This

optimization program will, in this case study, give us the parameters

$$\mu = 33.6$$
$$\lambda = 7.44 \cdot 10^{-7}$$

which gives an average modal damping of $\zeta = 0.025$.

Each node of the discretized system has 6 degrees of freedom, 3 translational and 3 angular. However, in this paper only z translations are of interest when controlling the mirror. It turns out that it is possible to reduce the system and only keep 3 degrees of freedom in every node. With this reduction, the state $\boldsymbol{\xi}$ will contain 1 translational and 2 angular displacements for every node, that is, $\boldsymbol{\xi} = [\, \xi_1^T \quad \xi_2^T \quad \dots \quad \xi_n^T \,]^T$ where $\xi_i = [\, z_i \quad \phi_i^x \quad \phi_i^y \,]^T$. The distribution of nodes is visualized in Figure 1. The set of nodes where the actuators are placed is denoted Act and is represented by the red dots in the figure. The force $\boldsymbol{F}$ exerted by the actuators only directly influences the actuator nodes, thus for $\boldsymbol{F} = [\, F_1^T \quad F_2^T \quad \dots \quad F_n^T \,]^T$, where $F_i = [\, f_i \quad 0 \quad 0 \,]^T$, the force f_i can only be non-zero if $i \in Act$. Introduce the matrix $\widetilde{B} = \left[\, \widetilde{B}_1 \quad \widetilde{B}_2 \quad \dots \quad \widetilde{B}_{|Act|} \,\right]$, with $\widetilde{B}_i = \boldsymbol{e}_{3(a_i-1)+1}$ for all $a_i \in Act$, where $\boldsymbol{e}_k$ is the Cartesian unit vector with a 1 in position k. We are now able to introduce an unrestricted control signal $u \in \mathbb{R}^{|Act|}$ by letting $\boldsymbol{F} = \widetilde{B}u$.

We write the system (1) as

$$\underbrace{\begin{bmatrix} I & 0 \\ 0 & \mathcal{M} \end{bmatrix}}_{\mathcal{E}} \begin{bmatrix} \dot{\boldsymbol{\xi}} \\ \ddot{\boldsymbol{\xi}} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & I \\ -\mathcal{K} & -\mathcal{C} \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} \boldsymbol{\xi} \\ \dot{\boldsymbol{\xi}} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \widetilde{B} \end{bmatrix}}_{\mathcal{B}} u$$

This dynamical system is now discretize for a given discretization sampling period Δt. In this case study the sampling period will be set to $\Delta t = 2 \cdot 10^{-4}$ s. The states of the discretized system are denoted by $x(k) = \left[\, \boldsymbol{\xi}(k\Delta t)^T \quad \dot{\boldsymbol{\xi}}(k\Delta t)^T \,\right]^T$. We will use an average between forward Euler and semi-implicit Euler, that is, an average between the difference equations

$$\mathcal{E}x(k+1) = (\mathcal{E} + \Delta t\mathcal{A})\, x(k) + \Delta t\mathcal{B}u(k)$$
$$(\mathcal{E} - \Delta t\mathcal{A})\, x(k+1) = \mathcal{E}x(k) + \Delta t\mathcal{B}u(k)$$

This approach will give us the system

$$\underbrace{\begin{bmatrix} I & -\frac{\Delta t}{2} I \\ \frac{\Delta t}{2}\mathcal{K} & \mathcal{M} + \frac{\Delta t}{2}\mathcal{C} \end{bmatrix}}_{E} x(k+1) = \underbrace{\begin{bmatrix} I & \frac{\Delta t}{2} I \\ -\frac{\Delta t}{2}\mathcal{K} & \mathcal{M} - \frac{\Delta t}{2}\mathcal{C} \end{bmatrix}}_{A} x(k) + \underbrace{\begin{bmatrix} 0 \\ \Delta t\widetilde{B} \end{bmatrix}}_{B} u(k) \tag{2}$$

The discretized system will have 36768 states and still 372 control inputs.

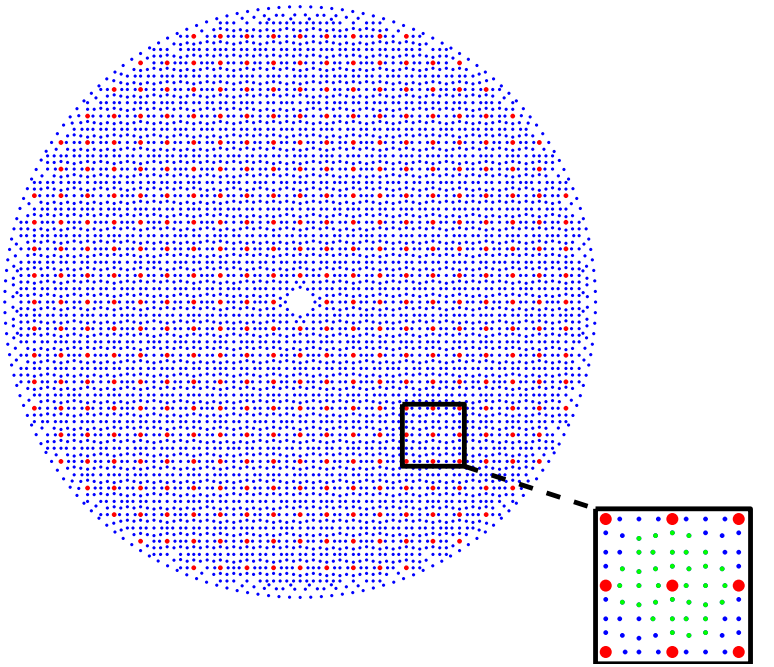


Figure 1. The distribution of the nodes on the mirror. The red nodes symbolize the positions of the actuators. In the zoomed box, the green nodes shows the neighbors to the middle actuator.

3. Control Synthesis Scheme for Descriptor Systems

In [Mårtensson and Rantzer, 2011, Mårtensson and Rantzer, 2010] a control synthesis scheme for large-scale linear, discrete time systems is presented. The objective of the scheme is to find a sparse state feedback matrix that minimizes a quadratic cost function. The procedure is to, given an initial stabilizing feedback matrix, determine a descent direction to the cost function and change the feedback matrix in this direction. This is repeated for number of steps to iteratively reduce the cost. The advantage of the scheme is that when considering sparse systems, the descent directions can be determined efficiently even for large-scale systems. We will show how to generalize this scheme to systems on descriptor form in this section. The reason to keep the system on descriptor form is that both the matrices E and A may have some structure, for example they may be sparse. But the dynamics matrix $E^{-1}A$ for the equivalent system

not on descriptor form looses this structure. For large-scale systems it is not even possible to calculate this matrix due to memory requirement.

Consider the linear, discrete time system on descriptor form

$$\begin{aligned} Ex(k+1) &= Ax(k) + Bu(k), \quad x(0) = x_0 \\ u(k) &= -Kx(k) \end{aligned} \tag{3}$$

where E is non-singular and K is stabilizing. We will assume that the admissible feedback matrices are restricted to a given set $\mathscr{K}$. This set can for example describe the sparsity structure that the feedback matrices are required to satisfy. The objective of the synthesis scheme is to find an admissible feedback matrix that minimizes the quadratic cost function

$$J(K, x_0) = \sum_{k=0}^{\infty} x(k)^T Q_x x(k) + u(k)^T Q_u u(k) \tag{4}$$

where $x(k)$ and $u(k)$ satisfy (3).

In the following proposition we show how to determine the gradient of J with respect to K, by determining the solutions to two Lyapunov equations. It should be noted that it is not possible to determine these solutions when considering large-scale systems. The reason is that computational time and memory requirements for solving a Lyapunov equation, scales as $O(n^3)$ and $O(n^2)$, respectively, see for example [Bini *et al.*, 2011]. The result is only an intermediate step in finding a tractable method for determining the gradient. We introduce the notation $A_K = A - BK$ and $Q_K = Q_x + K^T Q_u K$.

PROPOSITION 1
The gradient of J in (4) with respect to K, for a stabilizing K, is

$$\nabla_K J = 2 \cdot (Q_u K - B^T S A_K) X_0 \tag{5}$$

where X_0 and S satisfy the following Lyapunov equations

$$EX_0E^T = A_K X_0 A_K^T + Ex_0 x_0^T E^T \tag{6}$$

$$E^T S E = A_K^T S A_K + Q_K \tag{7}$$

□

Proof. Initially, it should be noted that $X_0 = \sum_{k=0}^{\infty} x(k)x(k)^T$ for $x(k)$ satisfying (3). Also, the value of $J(K, x_0) = \mathrm{tr}\left(SEx_0x_0^TE^T\right)$.

Now, we will determine a Lyapunov equation for the differential dS.

$$E^T dS E = A_K^T dS A_K + M + M^T$$

where $M = dK^T(Q_u K - B^T S A_K)$. This means that

$$dP = \sum_{t=0}^{\infty} (E^{-T} A_K^T)^t E^{-T} (M + M^T) E^{-1} (A_K E^{-1})^t$$

With the use of this relation, we can determine an expression for dJ

$$\begin{aligned} dJ &= \operatorname{tr}\left(dP \cdot E x_0 x_0^T E^T\right) \\ &= 2 \cdot \operatorname{tr}\left(\sum_{k=0}^{\infty} (E^{-T} A_K^T)^k E^{-T} M E^{-1} (A_K E^{-1})^k \cdot E x_0 x_0^T E^T\right) \\ &= 2 \cdot \operatorname{tr}\left(\sum_{k=0}^{\infty} E^{-T} (A_K^T E^{-T})^k M (E^{-1} A_K)^k E^{-1} \cdot E x_0 x_0^T E^T\right) \\ &= 2 \cdot \operatorname{tr}\left(M \sum_{k=0}^{\infty} (E^{-1} A_K)^k x_0 x_0^T (A_K^T E^{-T})^k\right) \\ &= 2 \cdot \operatorname{tr}\left(dK^T (Q_u K - B^T P A_K) X_0\right) \end{aligned}$$

Now, we use the relation about differentials, $dZ = \operatorname{tr}(dX^T \cdot Y) \Rightarrow \nabla_X Z = Y$ for matrices X^T, Y of size $n \times p$. □

The calculation in Proposition 1 relies on the solutions of (6) and (7). As mentioned, when considering large-scale systems, it is not tractable to find these solutions. In the following proposition we show how they can be replaced with the use of the trajectories of (3) and the adjoint system. The advantage is that now we only have to simulate two systems in order to determine the gradient of the cost function.

PROPOSITION 2
Define the adjoint states $\lambda(k)$ with the following dynamics

$$E^T \lambda(k-1) = A_K^T \lambda(k) - Q_K x(k), \quad \lim_{k \to \infty} \lambda(k) = 0 \tag{8}$$

where $x(k)$ is the trajectory of (3). Then the gradient of J with respect to K is

$$\nabla_K J = 2 \left(\sum_{k=0}^{\infty} \left(Q_u K x(k) + B^T \lambda(k)\right) x(k)^T \right) \tag{9}$$

□

Proof. We determine an expression for $\lambda(k)$ for any $t \geq 0$

$$\begin{aligned}\lambda(k) &= -\sum_{i=k+1}^{\infty} (E^{-T}A_K^T)^{i-k-1}E^{-1}Q_K x(i) \\ &= -\sum_{i=0}^{\infty}(E^{-T}A_K^T)^i E^{-T}Q_K(E^{-1}A_K)^{i+1}x(k)\end{aligned}$$

This gives us that

$$\begin{aligned}\sum_{k=0}^{\infty}\lambda(k)x(k)^T &= -\sum_{k=0}^{\infty}\sum_{i=0}^{\infty}(E^{-T}A_K^T)^i E^{-T}Q_K E^{-1}(A_K E^{-1})^i A_K x(k)x(k) \\ &= -SA_K X_0\end{aligned}$$

Fitting this into (5) and using that $X_0 = \sum_{k=0}^{\infty} x(k)x(k)^T$ gives (9). □

To be able to simulate the system (3) and (8) we need to choose a final time t_{final}. This means that the sum in (9) is truncated to a finite sum and thus we get an approximation of the gradient. In order to get a good approximation we need to choose t_{final} large enough to get the remaining part of the sum to be negligible.

4. Control Synthesis for the Deformable Mirror

In this section we will apply the synthesis method described in Section 3 on the deformable mirror model (2) given in Section 2. Firstly, we will find a suboptimal feedback matrix that reduces the influence of a non-zero initial state. This will automatically give us a controller that also reduces the effects of process noise. This feedback matrix is used to initialize another feedback matrix that will control the mirror model augmented with integrator states. The reason to introduce integrator states is to be able to track a given set-point.

4.1 Initial State Regulation

In this section we will find a structured feedback matrix K for the system (2). We assume that each actuator is only allowed to use measurements from nodes close to it, denoted its neighbors, see the green dots in the zoomed square of Figure 1 for an example. This setup imposes a structure on the feedback matrix. The structure implies that each row K_i, used for determining the control signal for actuator i, will be restricted.

For each K_i, only the elements corresponding to the states of that actuator's neighbors are allowed to be non-zero. More formally, K has to belong to the set $\mathscr{K}$

$$\mathscr{K} = \{K \mid K_{ij} = 0 \text{ if } j \text{ is not a neighbor of actuator } i\}$$

With this setup the maximum number of non-zero elements of K is 86538 or 0.63% of the total number of elements. An implication of this structure is that it is much faster to determine the control signal for every actuator compared to if all elements were allowed to be non-zero.

The method in Section 3 requires us to choose initial states for the mirror model system. By simulating the mirror in open loop and examining the size of the components of each node, we decide that we choose $x_0 \in \mathcal{N}(0, X_0)$, where X_0 is diagonal and the diagonal of each block X_0^i is

$$[\,10^{-7} \quad 2\cdot 10^{-6} \quad 2\cdot 10^{-6} \quad 10^{-3} \quad 10^{-2} \quad 10^{-2}\,]$$

The objective is to determine a feedback matrix that suppresses noise in the translational states of the mirror, that is, the z-states. This objective is reflected in the choice of the weight Q_x of (4), assigning a large weight on the z components of each node. We let Q_x be diagonal, where the diagonal for each node is $[\,10^6 \quad 10^{-2} \quad 10^{-2} \quad 10^{-4} \quad 10^{-14} \quad 10^{-14}\,]$. The weight for the control Q_u is chosen to be a diagonal matrix with all diagonal elements set to 10^{-6}.

The model of the mirror has integrator states, which means that if it is left uncontrolled it is not asymptotically stable. In order to apply the synthesis method, we need to find an initial stabilizing feedback matrix, $K^{(0)}$. A rudimentary strategy to find $K^{(0)}$ is to try a decentralized P-controller, that is, the only non-zero element in each row i of $K^{(0)}$ will be the element corresponding to the state of that actuators translational displacement, z. By simulation we find that one possible choice is to let all these non-zero elements equal $2.5 \cdot 10^2$.

From the initial stabilizing feedback matrix, $K^{(0)}$, we update the feedback matrix according to the following steps

ALGORITHM 1
In iteration step $\tau \geq 0$, to update the current feedback matrix $K^{(\tau)}$

1. Choose $x_0 \in N(0, X_0)$ and simulate system (2) for 1000 time samples.

2. Simulate the adjoint system (8) with $\lambda(1000) = 0$.

3. Let $\psi(k) = Q_u K x(k) + B^T \lambda(k)$ and determine the descent direction G by letting $G_{ij} = \sum_{k=0}^{1000} \psi_i(k) x_j(k)^T$ for all actuators i and the corresponding neighboring nodes j. The remaining elements of G are set to 0.
4. Let $J_{\text{est}}(K, x_0) = \sum_{k=0}^{1000} x(k)^T Q_x x(k) + u(k)^T Q_u u(k)$, where $x(k)$ satisfies (2) with $u(k) = -Kx(k)$. Now, let $\gamma_\tau = 1.5\gamma_{\tau-1}$ if $J_{\text{est}}(K^{(\tau)} - \gamma_\tau G, x_0) < J_{\text{est}}(K^{(\tau)}, x_0)$. Otherwise we decrease γ_τ with a factor of 5 until $J_{\text{est}}(K^{(\tau)} - \gamma_\tau G, x_0) \leq J_{\text{est}}(K^{(\tau)}, x_0)$.
5. Let $K^{(\tau+1)} = K^{(\tau)} - \gamma_\tau G$.
6. Increase τ with 1 and goto step 1.

□

Remark 1
Step 4 of the algorithm requires some explanation. In this step we choose the step length for the descent direction. If the step length in the previous iteration does not lead to a decrease of the cost it is reduced with a factor 5 until the cost is reduced. Since G is a descent direction it is obvious that the process of reducing the step length will terminate. Hence, this ensures us that we get a reduction of the cost in each iteration. But if the step length is much shorter than necessary the convergence of the algorithm will be slow. Hence, if it is possible to reduce the cost with a step length of 1.5 of the previous step length, the step length is increased by this factor. The reason for not doing a line search is for each evaluation of the cost we need to simulate the system. The simulation of the system is the time consuming part of the algorithm and we would like to keep the simulations at a minimum. Since the gradient calculation also relies on simulations, it requires the similar amount of time as determining the cost. Hence, we rather calculate a new descent direction instead of finding the optimal step length in the current direction at the expense of making many cost evaluations. □

A problem with gradient methods when solving optimization problems is that if the level sets of the cost function are flat in some directions, the decision variables in each iteration tend to jump back and forth without reducing the cost by much. This means that poorly scaled decision variables heavily affect the number of iterations to converge to the optimal value. For small scaled optimization problems a remedy is to determine

the Hessian and use Newton's method. This approach is not possible in the control synthesis for the mirror for a simple reason. The Hessian is isomorphic to a square matrix with the side $372 \cdot 36768 \approx 1.4 \cdot 10^7$, a huge matrix. To find a way to handle the scaling problem, we will use the fact that the nodes of the mirror are homogeneous with respect to the types of components of the node. We will assume that for a gradient the step that is possible to take for the ith component in all nodes is similar. Given a direction G in which we will update the feedback matrix K, let G_i be the projection of G onto the subspace

$$\{K \mid K_{:,b} = 0 \text{ unless } b \equiv i \pmod 6\}$$

Hence, in this subspace only component i in every node is allowed to be non-zero. Now, we find an approximation of the maximal step length γ_i in which $\gamma_i G_i$ reduces the cost. This is done by starting at an initial γ_i. If the cost is reduced with this initial step, while γ_i reduces the cost it is increased by some factor. Otherwise, if the initial step does not reduce the cost, γ_i is decreased by some factor until it is a reducing step length. Letting H be the a diagonal matrix where each node has diagonal $[\gamma_1 \quad \gamma_2 \quad \gamma_3 \quad \gamma_4 \quad \gamma_5 \quad \gamma_6]$, the descent direction $G \cdot H$ is better scaled than G. Typically, the vector γ is in the order of $[10^9 \quad 10^5 \quad 10^5 \quad 10^3 \quad 10^0 \quad 10^0]$ for each i. This means that without the scaling the first component of each node will not change since the step length will be governed by the last 2. The calculations of the scaling matrix requires a number of evaluations of the cost function, which makes it computationally demanding. However, it does not change much in between iterations and thus, does not have to be recalculated often to obtain the benefits of introducing it. For this reason, we will only recalculate the scaling matrix every 50th iteration.

In Algorithm 1 the number of iterations that the feedback matrix is updated is set to be $\tau_{\max} = 1000$. All calculations in one iteration requires on average 1 minute to perform (with a Linux PC with a 3 GHz Intel Core i7 processor and 4 GB memory). This results in a total computation time of approximately 16.6 hours. It should be noted that 70% of the computation time is spent on calculating $E^{-1}x$ and $E^{-T}\lambda$ in the simulation phase. The inversion is accomplished by precomputing sparse LU-factorizations of E and E^T and using them to perform the inversions. This computational time should be compared to the expected computational time required by conventional methods for finding the optimal full feedback matrix. The computational complexity of these methods are $O(n^3)$. We determine the computation time for solving the LQR problem for the mirror model where we remove a number of nodes. By removing different number of nodes, we can extrapolate the obtained computational times to estimate the time required to solve the LQR problem for the full mirror model, if we dis-

regard the problem with memory requirement. This procedure gives an estimated computation time of 81 days.

4.2 Set-Point Tracking

In the previous section we found a stabilizing feedback matrix to minimize the influence of a non-zero initial state. The actual objective when designing a controller for a deformable mirror is to be able to follow the atmospheric distortion, or in other words follow a reference trajectory. To accomplish the reference tracking, we will introduce integral action by including integrator states in the mirror model

$$x_i^{\text{int}}(k) = \sum_{j=0}^{k} r_i(j) - z_i(j)$$

where $r_i(k)$ is the reference for the translational states z_i for the nodes i that are to have integral action. However, it is not possible to introduce integrator states for every node. The reason is simply that the extended system would not be stabilizable. If the extended system was stabilizable it would be possible to, for any reference signal r, determine a control signal that would make the z components equal r in stationarity. Such a stationary control signal can be determined by solving

$$\min_{x,u} \|r - x_z\|$$
$$\text{subject to: } Ex = Ax + Bu$$

It turns out that for most reference signals the optimal norm does not equal zero implying that it is impossible to reach that reference in stationarity.

Instead we will only introduce integrator states for the actuator nodes. The extended dynamical system becomes

$$\begin{bmatrix} E & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k+1) \\ x^{\text{int}}(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C_{\text{ref}} & I \end{bmatrix} \begin{bmatrix} x(k) \\ x^{\text{int}}(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ I \end{bmatrix} r(k)$$

where C_{ref} is the partial identity matrix picking out only the z states for the actuator nodes. The extended system requires us to determine an extended set of admissible feedback matrices $\mathcal{K}_{\text{ext}}$. This is done by extending the previous set $\mathcal{K}$

$$\mathcal{K}_{\text{ext}} = \{[K \quad K^{\text{int}}] \mid K \in \mathcal{K}, K_{\text{int}} \in \mathcal{K}_{\text{int}}\}$$

The set $\mathcal{K}_{\text{int}}$ allows each actuator to use the integrator states of approximately 20 of the closest actuators. The number of non-zero elements of

K_{ext} is 93648, that is, 7110 more than the feedback matrix for the system without integral action.

The distribution of the initial state x_0 is kept as in the previous section, while the initial state of the integrators are always set to 0. The weight Q_x and Q_u for the states x and the control u, respectively, are also kept the same. The weight for the integrator states is set to $Q_x^{\text{int}} = 10^6 I$.

We will now run Algorithm 1 on the extended system. As in the previous section we need to initialize the algorithm with a stabilizing feedback matrix, $K_{\text{ext}}^{(0)} = [\, K_0 \quad K_0^{\text{int}} \,]$. We use the previously determined feedback matrix to initialize K_0. To find an initial K_0^{int} we apply the same technique as previously and let only the integrator state for actuator i influence the control of actuator i. One possible choice for these non-zero elements is to set them to -10.

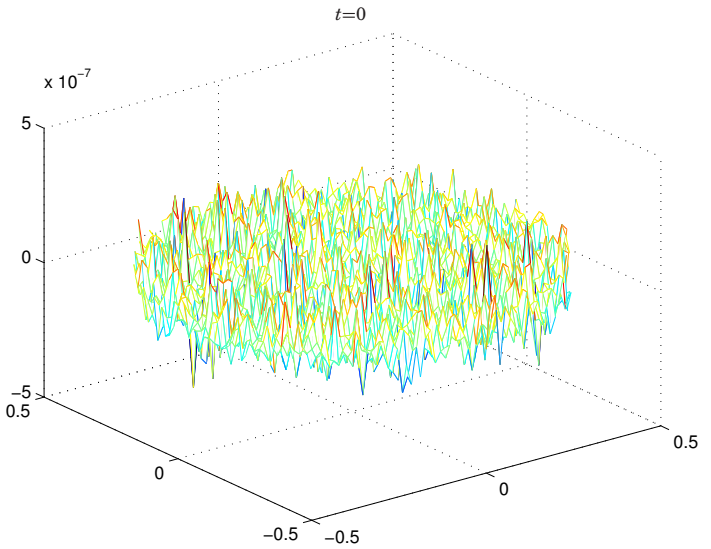
The number of iterations is still set to be $\tau_{\max} = 1000$. The inversion of the extended E matrix is essentially the same as previously, since it is block diagonal with the upper block equal to the original E and the lower block as the identity. Since this was the major time consuming operation, the computational time is hardly changed.

5. Evaluation of the Controllers

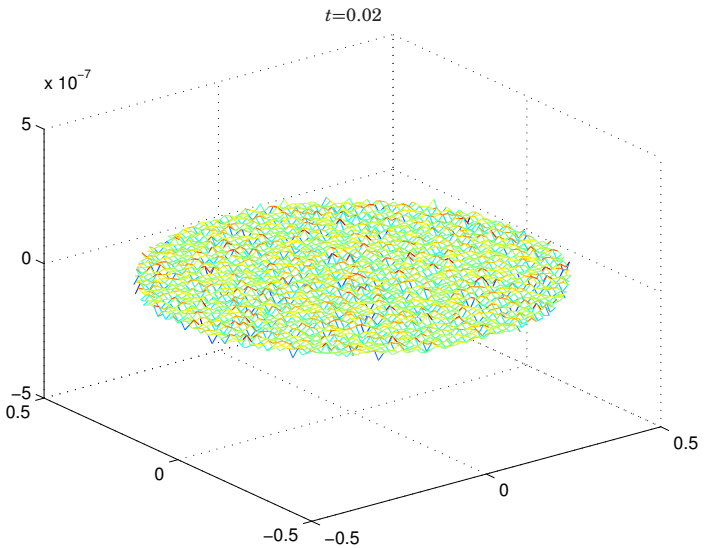
In this section feedback matrices determined in Section 4 are to be evaluated with respect to different performance measures. These measures will involve the LQR cost function, but also measures used when evaluating controllers for adaptive optics.

5.1 Feedback Matrix for Initial State Regulation

In this section we will evaluate the feedback matrix determined in Section 4.1. First we will show the suppression of the initial state when using the feedback matrix. We pick an initial state $x_0 \in \mathcal{N}(0, X_0)$ and simulate the mirror model (2) with the control $u(k) = -Kx(k)$ for 0.2 seconds, that is, 1000 time samples. In Figure 2 we find the initial state and the state after 0.02 seconds, 100 time samples, of the z component of each node. We can tell that the initial noise caused by the initial state is reduced significantly. The state cost, that is, $x(k)^T Q_x x(k) + u(k)^T Q_u u(k)$, is determined for all times during the simulation. In Figure 3 we show the time evolution of the state cost for $0 \leq k \leq 200$. It should be noted that the cost is plotted in logarithmic scale. In the plotted time interval the ratio between the maximal and minimal state cost is $3 \cdot 10^{-3}$. When examining the state cost trajectory for the complete time interval $0 \leq k \leq 1000$ it turns out that this ratio is $1.7 \cdot 10^{-11}$. This motivates that we truncate the cost after 1000 samples.



(a) The initial shape of the mirror.



(b) The shape of the mirror after 0.02 seconds.

Figure 2. The initial shape and the shape after 0.02 seconds when controlling the mirror with the obtained feedback matrix.

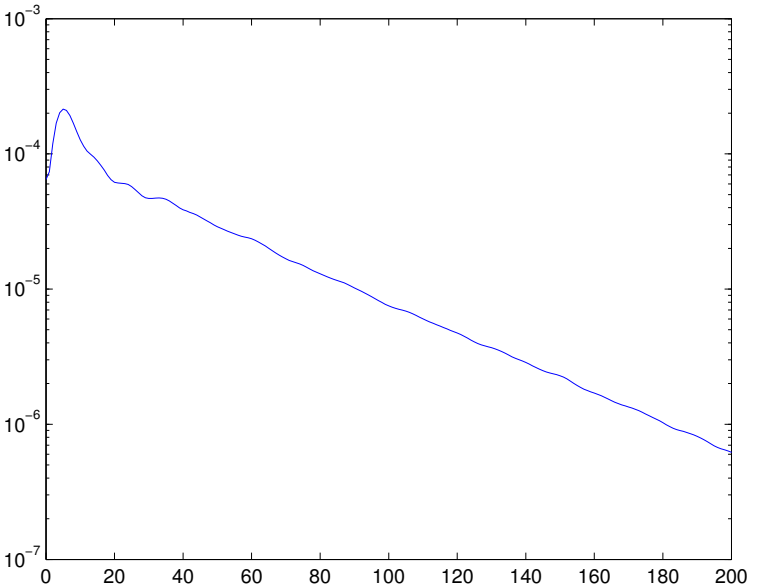


Figure 3. The trajectory of the step cost $x(k)^T Q_x x(k) + u(k)^T Q_u u(k)$ for the time samples in the interval $[0, 200]$ when controlling the mirror with the obtained feedback matrix *without* integral action.

While running Algorithm 1 when determining the feedback matrix used in this section, the feedback matrix in every 20th iteration is saved. For each of these matrices we will determine the LQR performance to show how the cost is changed when the algorithm runs. The LQR cost will be estimated by simulating the system for 1000 time samples from 100 different initial states, and determining the average of the truncated cost function

$$J_{\text{trunc}}(K, x_0) = \sum_{k=0}^{1000} x(k)^T Q_x x(k) + u(k)^T Q_u u(k)$$

over these initial states. The same initial states are used to determine the cost for all feedback matrices. The resulting cost for each matrix normalized with the smallest cost, is found in Figure 4. Only a zoomed part of the plot is shown to indicate the changes in the final feedback matrices. In the full plot the initial feedback matrix has a normalized cost of more

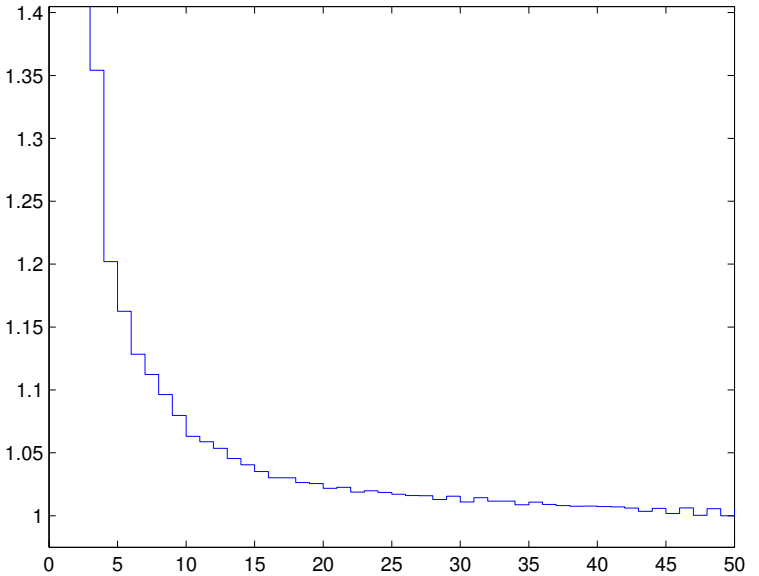


Figure 4. The change in the estimated cost of the feedback matrices in the synthesis process. The estimated cost for every 20th feedback matrix is plotted. We can see that the cost that the feedback matrices seem to converge.

than 12. Hence the final feedback matrix reduces the LQR cost by more than a factor 12. We see that the cost seems to converge meaning that continuing the update process in Algorithm 1 will probably not reduce the cost much more.

5.2 Feedback Matrix for Set-Point Tracking

We will now evaluate the feedback matrix for the extended system that was introduced in Section 4.2. We will define the reference trajectory $r_i(k)$ by the value we would like the z component in node i to have in time k. The control signal will now be determined by $u(k) = K_{\text{ext}}(R(k) - x_{\text{ext}}(k))$ where $R(k)$ is the collected reference for all the states, that is, each element of $R(k)$ equals 0 unless it corresponds to the z-component of each agent i, for which it equals $r_i(k)$.

First we examine the cost when starting from an initial state and controlling all states to 0, that is, $r_i(k) \equiv 0$ for all nodes i. The result of a simulation is given in Figure 5. The plot shows the step cost for that time, that is, $x(k)^T Q_x x(k) + u(k)^T Q_u u(k)$ for each k. We can see that in 200 time

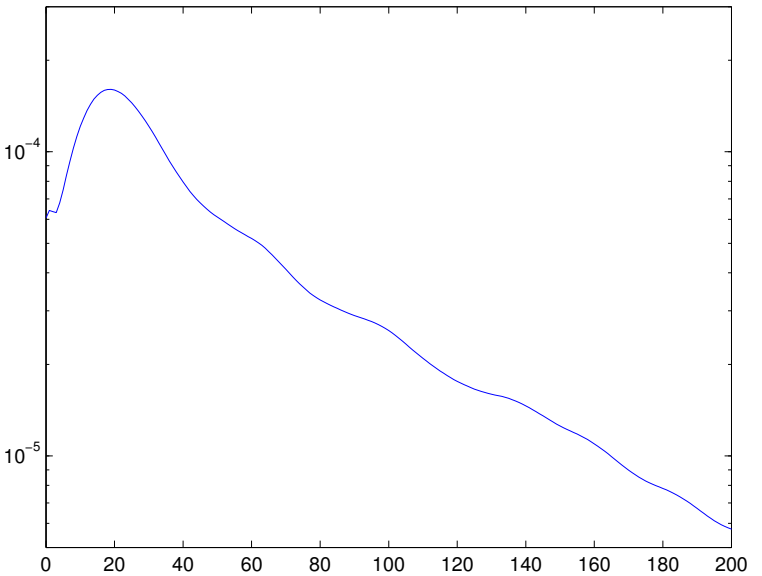


Figure 5. The trajectory of the step cost $x(k)^T Q_x x(k) + u(k)^T Q_u u(k)$ for the time samples in the interval $[0, 200]$ when controlling the mirror with the obtained feedback matrix *with* integral action.

steps the step cost is reduced by a factor of $3.6 \cdot 10^{-2}$ from the largest value. It turns out that in 1000 samples this ratio is $9.3 \cdot 10^{-7}$. As previously, this motivates us to truncate the cost after 1000 time steps. Similarly to the evaluation of the feedback matrices without integral action, every 20th feedback matrix is saved when running Algorithm 1. We determine the truncated cost for each of these matrices by simulating the system for 1000 time steps for 100 different initial states. The same initial states are used when determining the cost for each feedback matrix. The improvement of the cost function is shown in Figure 6.

When a wavefront enters the aperture of the telescope, the optimal scenario is that it has the same phase at any two points. As light travels through the atmosphere the wavefronts is subjected to the turbulence of the atmosphere. This causes the phase of the wavefront to be distorted and when it eventually reaches the telescope it is significantly out of phase, see Figure 7. Hence, if left uncorrected the phase at different points of the aperture will be noticeably different. The objective when controlling

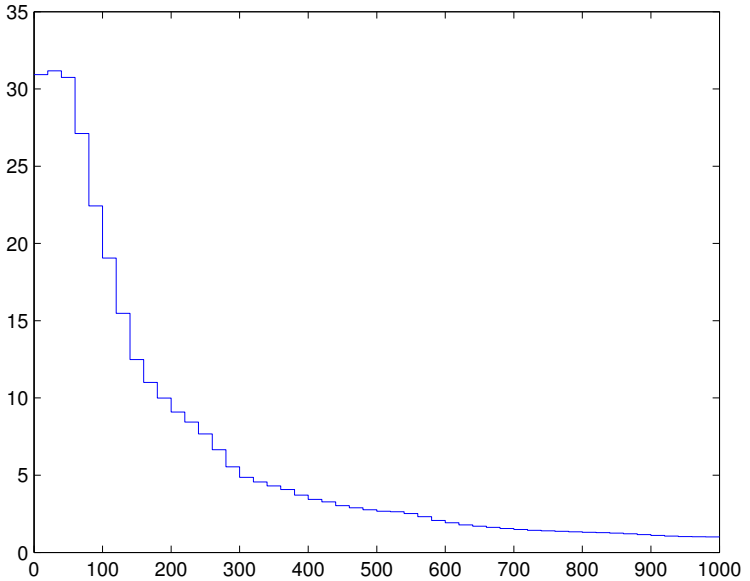


Figure 6. The change in the estimated cost of the feedback matrices with integral action in the synthesis process. The estimated cost for every 20th feedback matrix is plotted. We can see that the cost that the feedback matrices seem to converge.

the deformable mirror is to change the shape of the mirror such that the wavefront enters the aperture with the same phase again. If the mirror matches the shape of the incoming wavefront, the optical system is limited by diffraction and not by the atmosphere. The distortion of circular wavefronts with radius R is usually described by using the orthogonal Zernike polynomials as a basis [Noll, 1976], that is, the wavefronts take the form

$$\phi(r,\theta;t) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} a_{nm}(t) Z_n^m(r/R,\theta)$$

where $a_{nm}(t)$ is the time varying coordinate for each Zernike polynomial. The Zernike polynomials are given by

$$Z_n^m(\rho,\theta) = \mathcal{R}_n^m(\rho)\cos(m\theta)$$
$$Z_n^{-m}(\rho,\theta) = \mathcal{R}_n^m(\rho)\sin(m\theta)$$

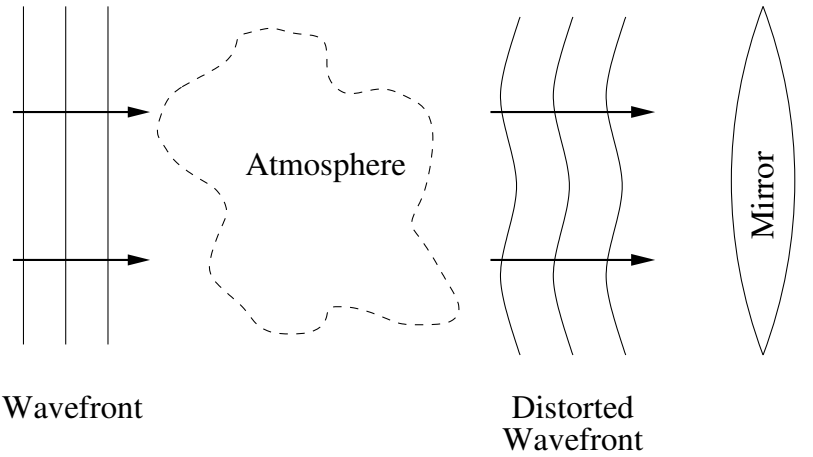


Figure 7. A schematic diagram of the distortion of wavefronts. Initially the wavefronts are in phase, shown by the straight lines. As they pass through the atmosphere they are distorted, meaning that in a cross section of the light, the phase is not equal in different locations of the cross section. This is represented by the wavy lines.

for $0 \leq \rho \leq 1$ and $0 \leq \theta \leq 2\pi$ and where the radial function

$$\mathcal{R}_n^m(\rho) = \sum_{k=0}^{(n-m)/2} \frac{(-1)^k (n-k)!}{k!((n+m)/2-k)!((n-m)/2-k)!} \rho^{n-2k}$$

for $0 \leq m \leq n$ and $n \equiv m \pmod 2$. The polynomials $Z_n^m(\rho, \theta)$ are referred to as even, while $Z_n^{-m}(\rho, \theta)$ are referred to as odd. It turns out that the low order polynomials hold the most of the phase distortion information of light through the atmosphere. Hence, a controller of the mirror should be able to mimic the shape of the low order Zernike polynomials in a reasonable time in order to correct the time varying distortion in a satisfactory way. A guideline is that the mirror should be able to mimic the shapes in 0.02 seconds. Since the radius of the mirror is 0.5 meters we will control the mirror to the shapes of $Z_n^m(2r, \theta)$. To evaluate the control to a shape, the root mean square error (RMS) is computed. In Figure 8 we show the time evolution of this ratio when controlling the mirror from 0 initial state to the Zernike polynomial $10^{-6} \cdot Z_4^2(2r, \theta)$. The plot shows that in 0.02 seconds the RMS value stabilizes implying that the mirror has reached a stationary shape. To show the performance of the controller for the low order Zernike polynomials, we will examine the response to the 27 first Zernike polynomials excluding the constant z-translation, that is,

for all Zernike polynomials with $1 \leq n \leq 6$. For each polynomial we initialize all states of the mirror to be 0 and simulate the controlled system for 0.02 seconds. This will show how well the controller is able to reduce each polynomial in a reasonable time. The blue dots in Figure 9 represent the resulting ratio between the RMS of the uncontrolled mirror and the resulting RMS of the controlled mirror. For each n, the ratio of the even and odd polynomial with the same m are plotted next to each other. For the first few polynomials the ratio is around 100. As the Zernike polynomial becomes more complex, the ratio drops down to a value around 10. These values should be related to the optimal ratios that can be obtained in stationarity. These optimal stationary RMS ratios can be determined by solving the quadratic program for each Zernike polynomial

$$\min_{x,u} \|r - x_z\|$$
$$\text{subject to: } Ex = Ax + Bu$$

where x is the full state vector, x_z is the part of the state vector corresponding to z-translations and r is the value of the Zernike polynomial in each node. The RMS of the obtained x_z will give us the optimal RMS ratio. The resulting ratios are shown in red in Figure 9. We actually see that the apparent low ratios in the last polynomials are not more that a factor 1.85 from the optimal values.

Remark 2
It should be noted that for some n, the seemingly similar polynomials $Z_n^m(\rho, \theta)$ and $Z_n^{-m}(\rho, \theta)$ give different RMS ratio. This is explained by the asymmetric configuration of the actuators. For example, the area around $x = 0.5$ and $y = 0$ is hard to control, thus potentially giving large errors. In the area around $x = y = \frac{1}{2\sqrt{2}}$ there are more actuators making it easier to control this area. □

The atmospheric disturbance can be modeled in the time domain using phase screens. A phase screen $P_\lambda(t, x, y)$ describes the phase added to a wavefront for a given wavelength λ, as it passes through the atmosphere at time t in the spatial coordinates (x, y). Given the wind velocity vector (v_x, v_y) and assuming frozen turbulence, the phase screen has the following property

$$P_\lambda(t, x, y) = P_\lambda(s, x + v_x(s - t), y + v_y(s - t))$$

for all times $t, s \in \mathbb{R}$ and all points $(x, y) \in \mathbb{R}^2$. This property means that the phase screen can be thought of as a continuous, static matrix moving in the wind direction. The values of the matrix is the amount of phase distortion that is to be added to the wavefront. Hence, only the shape of the

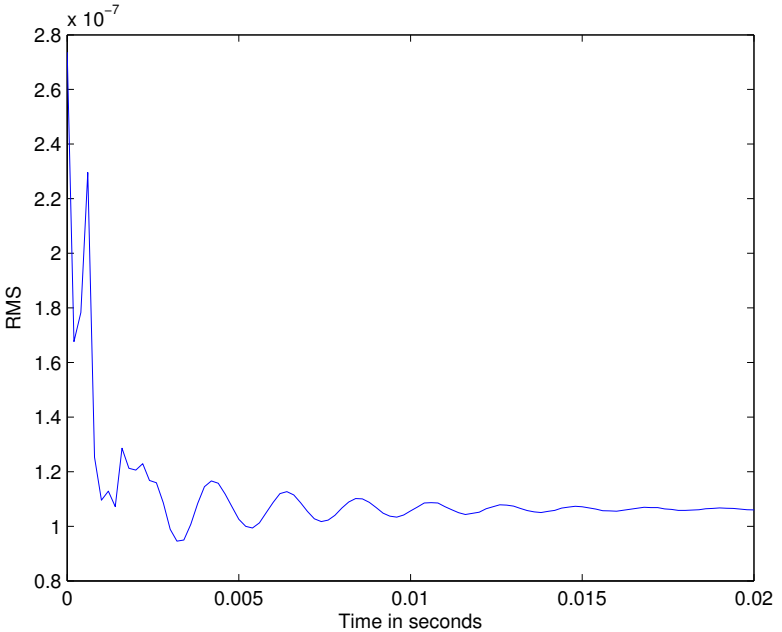


Figure 8. The RMS of the error in the z component when controlling the mirror to the shape of $Z_4^2(2r,\theta)$. In the time interval the RMS converges to its stationary value, implying that the controller is able to reduce this mode in a reasonable time.

phase screen at time 0 needs to be determined. The phase screen computed here is based upon Kolmogorov turbulence, described in [Hardy, 1998]. By gridding the spatial coordinates for a part of the phase screen we get a matrix of a section of the phase screen where the entries are the phase distortion in the grid points at time 0. To determine the phase distortion for a coordinate (x, y) within the boundaries of the matrix, interpolations of the grid points is used. The phase screen is now swept over the mirror in the wind velocity vector. The purpose of the controller is to make the mirror follow the shape of the phase screen. A measure of the performance commonly used in adaptive optics is the Strehl ratio, given by

$$S = e^{-(2\pi\epsilon(k)/\lambda)^2}$$

where $\epsilon(k)$ is the RMS at time k and λ is the given wavelength. The Strehl ratio is thus in the interval $(0, 1]$. An acceptable level of the Strehl ratio is 0.8 or higher. To measure the atmosphere a wavefront sensor is used, for

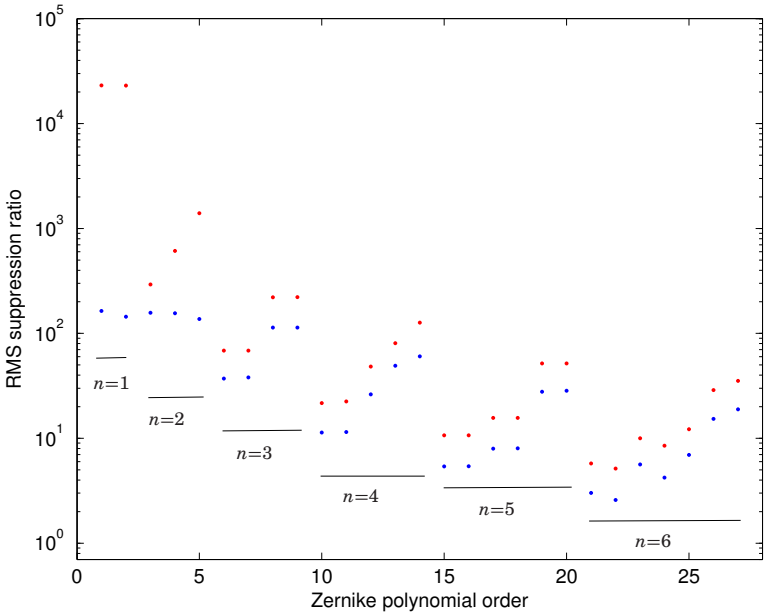


Figure 9. The suppression ratios of the RMS values when controlling the mirror to the first 27 Zernike polynomials. The values of the ratios when using the feedback controller is given by the blue dots. In red we find the optimal stationary suppression ratios.

example a Shack–Hartmann wavefront sensor. It is not possible to measure the aberration in every node. We will assume perfect measurements of the phase screen only in the actuator nodes. To determine the value for the remaining nodes we use interpolation of the measurements from the actuator nodes. For the nodes outside of the convex hull of the actuator nodes, we will assume that we also have measurements of imaginary nodes in the grid of actuators, in a sense extrapolated actuator nodes. Also, we assume that the sampling time of the wavefront sensor is 2 ms. In between the sampling time we will hold the reference to the mirror constant, thus not using any technique to predict the change in the phase screen. When simulating the mirror following the phase screen, the actual phase screen is updated in the sampling time of the system. Hence, the error will be determined by the difference of the mirror and the phase screen. The result of a simulation of the mirror following a phase screen in a 0.02 second window was performed and the resulting Strehl ratio is

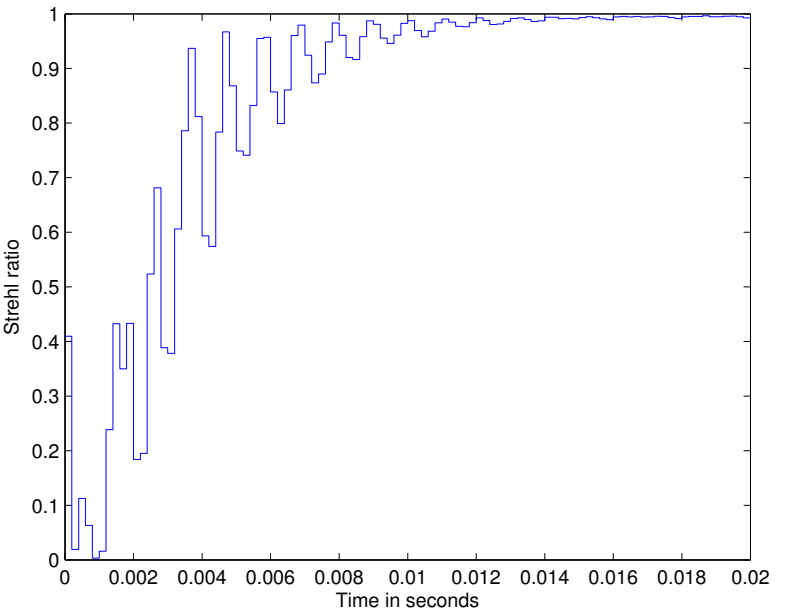


Figure 10. The Strehl ratio obtained when running a phase screen over the mirror for a time interval of $[0, 0.02]$ seconds. Initially it starts at a low Strehl ratio due to the initial position of the mirror. As time progresses the ratio approaches the maximal possible value of 1. The oscillatory behavior of the trajectory can be explained by the longer sampling time of the wavefront sensor.

given in Figure 10. The initial state of the mirror is 0, giving the transient behavior in the beginning of the time interval. After about 1 ms the Strehl ratio converges to a value close to 1. It should be noted this extremely good Strehl ratio will probably drop when some sources of error are introduced that have been neglected in this paper. For example, it will not be possible to measure all states of all nodes. When a Kalman filter is to estimate the states, the performance will naturally drop. Also, the dynamics of actuators and future sensors have not been considered here.

6. Conclusions and Future Works

6.1 Conclusions

In this paper we have demonstrated how to determine structured state feedback controllers for a large deformable mirror for adaptive optics. The synthesis method for the controllers relies on previous work [Mårtensson and Rantzer, 2011, Mårtensson and Rantzer, 2010]. The mirror is discretized into nodes with 3 degrees of freedom and then modelled by a discrete time system where the structure of system matrices follow the interconnection graph of the nodes. Similar, the admissible feedback matrices must also follow this structure, which will result in sparse matrices. Using the synthesis method, a feedback matrix to suppress process noise is determined. The feedback matrix is evaluated by simulating the system from non-zero initial states and calculating the LQR cost. Also, a feedback matrix for set point tracking is determined. The evaluation of this matrix involve the tracking of Zernike polynomials and a phase screen. A promising Strehl ratio of a value close to 1 is exhibited when tracing the phase screen. It should be kept in mind that a few sources of error has been disregarded. For example, the states of all nodes are assumed to be measured. When introducing sensors in only a few nodes, the value of the Strehl ratio will probably drop.

6.2 Future Works

In this paper we present initial work for finding an LQG controller for deformable mirrors. There are still a number off issues that needs to be looked into.

- We assume measurements of every state in every node. When introducing positions sensors in a subset of the nodes, an state estimator needs to be implemented. In [Mårtensson and Rantzer, 2012] it is shown how to determine structured Kalman filters in a similar fashion as the structured feedback matrix, in order to obtain a structured output feedback controller.

- We have neglected the influence of dynamics in the actuators and the future sensors. With the high sampling frequency for the discrete time model used in this paper these components may show a dynamic nature.

- The robustness of the future LQG controller should be examined. In this paper the performance of the controller is evaluated using the same model of the mirror. The evaluation could be made with a

continuous model of the mirror building on a modal representation of the system.

References

Andersen, T. and A. Enmark (2011): *Integrated Modeling of Telescopes*. Springer.

Andersen, T., O. Garpinger, M. Owner-Petersen, F. Bjoorn, R. Svahn, and A. Ardeberg (2006): "Novel concept for large deformable mirrors." *Opt. Eng.*, **45**.

Bini, D., B. Iannazzo, and B. Meini (2011): *Numerical Solution of Algebraic Riccati Equations*. Society for Industrial and Applied Mathematics.

Fraanje, R., P. Massioni, and M. Verhaegen (2010): "A decomposition approach to distributed control of dynamic deformable mirrors." *International Journal Of Optomechatronics*, **4**.

Hardy, J. (1998): *Adaptive Optics for Astronomical Telescopes*. Oxford University Press.

Heimsten, R. (2011): *Study of a Large Deformable Mirror Concept*. PhD thesis, Department of Astronomy and Theoretical Physics, Lund University, Sweden.

Hinnen, K., M. Verhaegen, and N. Doelman (2007): "Exploiting the spatiotemporal correlation in adaptive optics using data-driven $\mathcal{H}_2$-optimal control." *J. Opt. Soc. Am. A*, **24:6**, pp. 1714–1725.

Lasdon, L. S. (2002): *Optimization Theory for Large Systems*. Dover Publications Inc.

Looze, D. (2009): "Linear-quadratic-Gaussian control for adaptive optics systems using a hybrid model." *J. Opt. Soc. Am. A*, **26:1**, pp. 1–9.

Mårtensson, K. and A. Rantzer (2010): "Sub-optimality bound on a gradient method for iterative distributed control synthesis." In *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems*. Budapest, Hungary.

Mårtensson, K. and A. Rantzer (2011): "Synthesis of structured controllers for large-scale systems." *IEEE Transactions on Automatic Control*. Submitted.

Mårtensson, K. and A. Rantzer (2012): "Synthesis of structured output feedback controllers for large-scale systems." *Automatica*. Submitted.

Miller, D. and S. Grocott (1999): "Robust control of the multiple mirror telescope adaptive secondary mirror." *Opt. Eng.*, **38**, pp. 1276–1287.

Noll, R. (1976): "Zernike polynomials and atmospheric turbulence." *J. Opt. Soc. Am.*, **66:3**, pp. 207–211.

Paschall, R. and D. Anderson (1993): "Linear quadratic Gaussian control of a deformable mirror adaptive optics system with time-delayed measurements." *Applied Optics*, **32**, pp. 6347–6358.

Petit, C., J.-M. Conan, C. Kulcsár, and H.-F. Raynaud (2009): "Linear quadratic Gaussian control for adaptive optics and multiconjugate adaptive optics: experimental and numerical analysis." *J. Opt. Soc. Am. A*, **26:6**, pp. 1307–1325.

Petit, C., J.-M. Conan, C. Kulcsár, H.-F. Raynaud, T. Fusco, J. Montri, and D. Rabaud (2005): "Optimal control for multi-conjugate adaptive optics." *Comptes Rendus Physique*, **6:10**, pp. 1059–1069.

Quiros-Pacheco, F., L. Busoni, G. Aqapito, S. Esposito, E. Pinna, A. Puqlisi, and A. Riccardi (2010): "First light AO (FLAO) system for LBT: performance analysis and optimization." *Proceedings of the SPIE*, **7736**.

Reddy, J. (2006): *An Introduction to the Finite Element Method*. McGraw-Hill.

Roddier, F. (1999): *Adaptive Optics in Astronomy*. Camebridge University Press.

Roux, B. L., J.-M. Conan, C. Kulcsár, H.-F. Raynaud, L. Mugnier, and T. Fusco (2004): "Optimal control law for classical and multiconjugate adaptive optics." *J. Opt. Soc. Am. A*, **21:7**, pp. 1261–1276.

von Bokern, M., R. Paschall, and B. Welsh (1992): "Modal control for an adaptive optics system using LQG compensation." *Computers & Electrical Engineering*, **18**, pp. 421–433.

Wildi, F. P., G. Brusa, M. Lloyd-Hart, L. M. Close, and A. Riccardi (2003): "First light of the 6.5 m MMT adaptive optics system." *Proceedings of the SPIE*, **5169**.