Synthesis of Structured Controllers for Large-Scale Systems

Karl Mårtensson, Anders Rantzer, *Fellow, IEEE*

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.

I. Introduction

The theory of optimal linear quadratic control has been around for several decades and is well documented in the literature, for example [2], [9], [17]. 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 $\mathcal{O}(n^2)$ and the computational complexity is $\mathcal{O}(n^3)$, see [6], 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

The authors are with the Automatic Control Department, LTH, Lund University, Lund SE 223 63, Sweden (e-mail: karl@control.lth.se; rantzer@control.lth.se).

structure in the problem [19]. 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 [13], [20], [31], or when considering large interconnected systems. The papers [6], [5] 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. In [16], [8] solving large-scale algebraic Riccati equations using Arnoldi-based methods are discussed. By projecting the parameters onto Krylov subspaces, low rank approximations of the Riccati solutions can be determined. Another low rank approximation method based on stable invariant subspaces of the Hamiltonian is given in [1]. A Riccati based method for stabilization of large-scale systems can be found in [28].

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 [33] 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 [15]. 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 [30], [29], which transforms the problem of finding an optimal controller into a convex optimization program. The works [3], [4] 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 [21]. In [22], [23] $\mathcal{H}_2$ design of state feedback matrices subjected to structural constraints such that many entries are zero, are considered. Other methods which involve the use of linear matrix inequalities were presented in for example [11], [18], [27].

It is possible to investigate the solution of a linear quadratic control problem using Pontryagin's Maximum principle, by introducing adjoint states, [9], [14]. In [9] it was shown how the express the optimal control signal in terms of adjoint states. In [14], large-scale sparse continuous-time system were discretized to solve open-loop linear quadratic control problems. 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 [12] 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 [24], [25].

The paper is organized as follows. In Section II 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 III 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 IV. Two examples illustrating the methods are presented in Section V.

A. 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. Let dX denote an infinitesimal change of the variable X. 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.

The normal distribution with mean μ and variance σ^2 will be denoted $\mathcal{N}(\mu,\sigma^2)$.

II. 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 $u(t) \in \mathbb{R}^p$ for $t \geq 0$ and $x_0 \in \mathcal{N}(0,\sigma_{x_0}^2)$. 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.

A. 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 subset of $\mathscr{K}$ that consists of all 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. For the classical LQR problem with full feedback matrices, the initial condition x_0 does not influence the solution. However, when introducing structural constraints, generally the optimal feedback matrix differs for different initial conditions. Assuming that the initial condition is some stochastic variable, the cost

$$\mathbf{E}_{x_0}\left[J(K, x_0)\right] \tag{4}$$

will be the objective for the control synthesis for feedback matrices in $\mathscr{K}_{\text{stab}}$.

B. Analysis

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

$$X_0 = (A - BK)X_0(A - BK)^T + x_0 x_0^T \tag{5}$$

is given by

$$X_0 = \sum_{t=0}^{\infty} (A - BK)^t x_0 x_0^T ((A - BK)^T)^t$$

Inserting (1) and (2) into (3), we get an expression for the cost of the given system

$$\begin{aligned} J(K, x_0) &= \operatorname{tr}\left((Q_x - 2Q_{xu}K + K^T Q_u K) \sum_{t=0}^{\infty} x(t)x(t)^T \right) \\ &= \operatorname{tr}\left((Q_x - 2Q_{xu}K + K^T Q_u K) X_0 \right) \end{aligned}$$

Similarly, by denoting $Q_K = Q_x - Q_{xu}K - K^T Q_{xu}^T + K^T Q_u K$, the unique solution P to another Lyapunov equation

$$P = (A - BK)^T P (A - BK) + Q_K \tag{6}$$

can be expressed as

$$P = \sum_{t=0}^{\infty} ((A - BK)^T)^t Q_K (A - BK)^t$$

This solution P can be used to derive another expression for the cost function

$$J(K, x_0) = \sum_{t=0}^{\infty} x(t)^T Q_K x(t) = \operatorname{tr}\left(P x_0 x_0^T\right)$$

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

$$\nabla_K J = 2\left(Q_u K - Q_{xu}^T - B^T P(A - BK)\right) X_0, \tag{7}$$

where X_0 and P satisfy the Lyapunov equations (5) and (6).

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

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

Differentiating (6) 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 $dJ = \operatorname{tr}\left(dP x_0 x_0^T\right)$, we get that

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

By using the relation about differentials

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

the relation (7) is verified. □

The expression in (7) involves the solution of the Lyapunov equation (6). Finding the solution to a Lyapunov equation is a time-consuming operation for large-scale systems. It is well known that the adjoint states of a system are closely related to optimal control. In for example [9] 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 introduce the adjoint states in a way to get rid of the Lyapunov solution P in the expression for $\nabla_K J$. A benefit of this substitution is that the time complexity for finding the gradient is heavily reduced, which will later be shown.

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

$$\begin{aligned}\lambda(t-1) = (A-BK)^T\lambda(t) - \\ -(Q_x - Q_{xu}K - K^TQ_{xu}^T + K^TQ_uK)x(t)\end{aligned} \tag{8}$$

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

$$\nabla_K J = 2\left(\sum_{t=0}^{\infty}\left(-Q_u u(t) - Q_{xu}^T x(t) + B^T\lambda(t)\right)x(t)^T\right) \tag{9}$$

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

$$\begin{aligned}\lambda(j) &= -\sum_{k=j+1}^{\infty}(A_K^T)^{k-j-1}Q_Kx(k) = \\ &= -\sum_{k=0}^{\infty}(A_K^T)^kQ_KA_K^{k+1}x(j)\end{aligned}$$

Hence

$$\begin{aligned}\sum_{t=0}^{\infty}\lambda(t)x(t)^T &= -\sum_{t=0}^{\infty}\sum_{k=0}^{\infty}(A_K^T)^kQ_K(A_K)^{k+1}x(t)x(t)^T = \\ &= -PA_KX_0\end{aligned}$$

Fitting this into (7) and using that $X_0 = \sum_{t=0}^{\infty}x(t)x(t)^T$ gives the desired result. □

The gradient $\nabla_K J$ for a stabilizing admissible feedback matrix K is generally not in the space $\mathscr{K}$. This means that the gradient can not directly be used to update K since that $K-\gamma\nabla_K J \notin \mathscr{K}$ for any non-zero γ. Let us define the scalar product of elements of $\mathscr{K}$ as the scalar product of the vector representation of the elements. By orthogonally projecting the gradient onto $\mathscr{K}$,

denoted $\text{proj}_{\mathscr{K}}(\nabla_K J)$, we obtain an admissible descent direction. Now, $K - \gamma \cdot \text{proj}_{\mathscr{K}}(\nabla_K J)$ for a small enough γ will be an admissible feedback matrix with cost less than K.

C. General Algorithm

Instead of infinite time, we truncate the sum in (9), 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 (4), start with $K^{(0)} \in \mathscr{K}_{\text{stab}}$ and for each $\tau \geq 0$,

1) Simulate the states of (1) for some $x_0 \in \mathcal{N}(0, \sigma_{x_0}^2$ with control $u(t) = -K^{(\tau)}x(t)$ for times $t = [0, t_{\text{final}}]$.
2) Simulate the adjoint states of (8) 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: For a general constrained optimization problem the procedure of using projected gradients as the descent direction is commonly known as a projected gradient method. For more about projected gradient methods see for example [7], [10].

Remark 2: 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. By simulating (1) with the $K = K^{(\tau)} - \gamma_\tau G$ the cost can be evaluated. Hence, a line-search may be applied, for example Armijo's rule.

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, $\text{tr}\left(\nabla_K J^T \cdot D\right) < 0$ must hold. Letting G be the truncated gradient and $H = \nabla_K J - G$. Then G is a descent direction if $\text{tr}\left((G+H)^T G\right) < 0$, that is $\text{tr}\left(G^T G\right) < \text{tr}\left(H^T G\right)$. Since $\text{tr}\left(G^T G\right)$ 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.

Remark 4: The initialization of Algorithm 1 needs a stabilizing structured feedback matrix $K^{(0)}$. In case when (1) is stable $K^{(0)}$ can simply be chosen to be 0. However, for unstable systems it may not be a nontrivial problem to find such a stabilizing feedback matrix. For unstable system of a moderate size, methods to find structured stabilizing feedback matrices can for example be found in [32], [23].

As posed, there is no stopping criterion 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. In the end of the section it will be shown how to incorporate this bound in Algorithm 1 to get a stopping criterion.

D. 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

$$J(K, x_0) \leq \alpha J(K_{\text{opt}}, x_0), \tag{10}$$

where $K_{\text{opt}} = \underset{K}{\text{argmin}}\, J(K, x_0)$. 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 (10), 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

$$J(K, x_0, t_{\text{final}}) = \sum_{t=0}^{t_{\text{final}}} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \tag{11}$$

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. The theorem is a duality result of the original optimization problem. It should be noted that the theorem holds for any choice of adjoint variables $\lambda(t)$, but later on we show that a natural choice of the adjoint variables is determined by (8).

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

$$\begin{aligned} J(K, x_0, t_{\text{final}}) \leq \alpha \min_{\substack{x,u \\ x(0)=x_0}} \sum_{t=0}^{t_{\text{final}}} \Bigg(& \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} + \\ & + 2\lambda(t)^T \left(x(t+1) - Ax - Bu(t) \right) \Bigg) \end{aligned} \tag{12}$$

then α is a suboptimality bound, that is

$$J(K, x_0, t_{\text{final}}) \leq \alpha J(K_{\text{opt}}, x_0, t_{\text{final}}), \tag{13}$$

where

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

Proof. Assume that $\alpha \geq 1$ is such that for a given sequence of $\lambda(t)$, (12) holds. We have that

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

where the last inequality comes from introducing dual variables. The right hand side of this inequality is the dual function of the minimization problem on the left hand side. Now, it is well-known that the optimal value of a minimization problem is greater than or equal to any value of its dual function. Hence, if (12) holds, so must (13). □

Remark 5: If the matrix Q is not positive definite (that is, only positive semi-definite) it is easily realized the minimization program in (12) is not necessarily bounded. For any direction in $\begin{bmatrix} x(t)^T & u(t)^T \end{bmatrix}^T$ 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 $\begin{bmatrix} (\lambda(t-1)^T - \lambda(t)^T A) & \lambda(t)^T B \end{bmatrix}^T$ 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 (12) 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

$$\sum_{t=0}^{t_{\text{final}}} \Bigg(\begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} + \\ + 2\lambda(t)^T \left(x(t+1) - Ax(t) - Bu(t) \right) \Bigg) =$$

$$= \begin{bmatrix} x_0 \\ u(0) \end{bmatrix}^T Q \begin{bmatrix} x_0 \\ u(0) \end{bmatrix} - 2\lambda(0)^T (Ax_0 + Bu(0)) + \\ + \sum_{t=1}^{t_{\text{final}}} \Bigg(\begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} + \\ + 2\lambda(t-1)^T x(t) - 2\lambda(t)^T (Ax(t) + Bu(t)) \Bigg) =$$

$$= x_0^T Q_x x_0 - 2\lambda(0)^T A x_0 + F^T Q_u^{-1} F - f^T Q_u^{-1} f + \\ + \sum_{t=1}^{t_{\text{final}}} \left(G(t)^T Q^{-1} G(t) - g(t)^T Q^{-1} g(t) \right)$$

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 6: With similar reasoning as in Remark 5 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 $-\begin{bmatrix} x(t)^T & u(t)^T \end{bmatrix}^T$).

Theorem 1 does not specify the sequence of $\lambda(t)$ to choose to determine the suboptimality bound of a given feedback matrix. Any sequence giving giving an $\alpha \geq 1$ will actually do. Though, the name suggest that we choose the adjoint variables defined by (8). To motivate this choice, we could refer to Pontryagin's maximum principle. The motivation comes from examining

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

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

$$0 = \nabla_x(t)\mathcal{L} = 2(Q_x x(t) + Q_{xu} u(t) + \lambda(t-1) - A^T \lambda(t))$$
$$0 = \nabla_u(t)\mathcal{L} = 2(Q_{xu}^T x(t) + Q_u u(t) - B^T \lambda(t))$$

We get (8) by $\nabla_x(t)\mathcal{L} + K^T \nabla_u(t)\mathcal{L} = 0$ and replacing $u(t) = -Kx(t)$.

To include the suboptimality bound as a stopping criterion in Algorithm 1, only step 5 will be modified. Assume that a maximal degree of suboptimality $\alpha_{\max}$ is given. That is, Algorithm 1 should stop when it can be verified that $\alpha \leq \alpha_{\max}$. Then the step 5 should be modified to

5') Let

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

If $\dfrac{J(K, x_0, t_{\text{final}})}{M} \leq \alpha_{\max}$ then terminate, otherwise goto 1).

III. 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. Second, 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.

A. 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 $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ 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 $1 \leq i \leq n$. 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 $\mathcal{N}_i = \mathcal{N}_i^{\text{up}} \cup \mathcal{N}_i^{\text{down}}$.

B. 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 $x = \begin{bmatrix} x_1^T & x_2^T & \ldots & x_n^T \end{bmatrix}^T$, 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

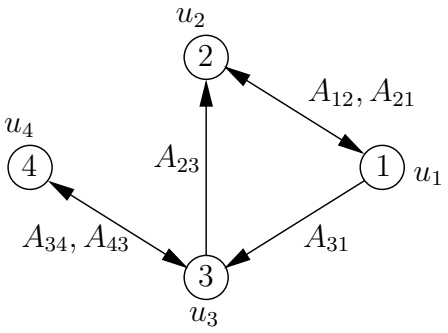

Fig. 1. Graphical representation of a distributed system. The arrows shows how each agent affects the others. The set $\mathcal{E} = \{(1,1),(2,2),(3,3),(4,4),(1,2),(2,1),(1,3),(3,2),(3,4),$
$(4,3)\}$

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,

$$B = \text{diag}(B_1, B_2, \ldots, B_n)$$

where B_i is the local B matrix for agent v_i.

Remark 7: 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

$$x_i(t+1) = A_{ii}x_i(t) + B_i u_i(t) + \sum_{k \in \mathcal{N}_i^{\text{up}}} A_{ik}x_k(t) \tag{14}$$

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

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

C. Problem Formulation

In Section II-A 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

$$\mathscr{K} = \{K \mid K_{ik} = 0 \text{ if } (v_i, v_k) \notin \mathcal{E},\ \forall i, k\} \tag{15}$$

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 8: 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.

D. 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 II-C 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 II-B a descent direction is obtained by projecting the gradient onto the subspace $\mathscr{K}$. With the structure of $\mathscr{K}$ in (15), the only part of the gradient that needs to be determined

is

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

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 (8), the adjoint state evolution of agent i is

$$\begin{aligned}\lambda_i(t-1) = &\left[(A-BK)^T\lambda(t)\right]_i - \\ &- \left[(Q_x - Q_{xu}K - K^TQ_{xu}^T + K^TQ_uK)x(t)\right]_i\end{aligned} \tag{16}$$

The first term of (16) simplifies to

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

The second term of (16) becomes

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

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

Algorithm 2: Consider a system (1) with control $u(t) = -Kx(t)$ where $K \in \mathscr{K}_{\text{stab}}$. 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 $t \in [0, t_{\text{final}}]$.
2) Simulate the adjoint states of (8) 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 scheme is modularized since the operations that are performed for each agent in the algorithm only needs information of its neighbors. The number of operations needed to determine the descent direction in each agent scales as $\mathcal{O}(|\mathcal{N}_i|)$. Assuming $|\mathcal{N}_i| \leq N_{\max} \ll n$ for all i, that is each agent only has a few neighbors, this implies that the total number of operations to determine the descent direction scales as $\mathcal{O}(n \cdot N_{\max})$.

Remark 9: 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.

E. 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 (12) 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.

IV. Real-Time Synthesis of Distributed Controllers

In Section II and III 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.

A. Restated System Model

We now consider the discrete time stochastic LTI system

$$x(t+1) = Ax(t) + Bu(t) + w(t) \tag{17}$$

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 III-B, 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 (17) are determined by the edges of $\mathcal{G}$ and so on.

In this stochastic setting the cost function will now be

$$J(K) = \mathbf{E} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \tag{18}$$

where $x(t)$ satisfies (17) and $u(t) = -Kx(t)$.

B. Analysis

Similar to the analysis in Section II we state the following two propositions, without proofs.

Proposition 4: Given the stationary stochastic process (17) where $u(t) = -Kx(t)$ for a stabilizing K and where w is white noise with covariance W. Then $J(K)$, defined by (18), has the gradient

$$\nabla_K J = 2\left[Q_u K - Q_{xu}^T - B^T P(A - BK)\right] X \tag{19}$$

where X and P satisfy the Lyapunov equations

$$X = (A - BK)X(A - BK)^T + W \tag{20}$$

$$\begin{aligned} P = {} & (A - BK)^T P(A - BK) + \\ & + Q_x - Q_{xu}K - K^T Q_{xu}^T + K^T Q_u K \end{aligned} \tag{21}$$

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

$$\begin{aligned} \lambda(t-1) = {} & (A - BK)^T \lambda(t) - \\ & -(Q_x - Q_{xu}K - K^T Q_{xu}^T + K^T Q_u K)x(t) \end{aligned} \tag{22}$$

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

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

C. Real-Time Scheme

Similar to Section III-D we use Proposition 5 to form an online scheme to update the feedback matrix K to improve the performance given in (18) while $K \in \mathscr{K}_{\text{stab}}$. 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 (17) with control $u(t) = -Kx(t)$ where $K \in \mathscr{K}_{\text{stab}}$. To iteratively improve the performance (18) 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 (22) for times $t = t_\tau, \dots, t_\tau + N$ 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).

V. 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.

A. 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

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

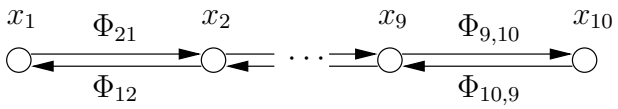

Fig. 2. Graphical representation of the system in Example V-A. The arrows shows how each agent affects the others.

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

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

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 $t = 0, \dots, 10$ 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 (12) 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 (12).

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

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

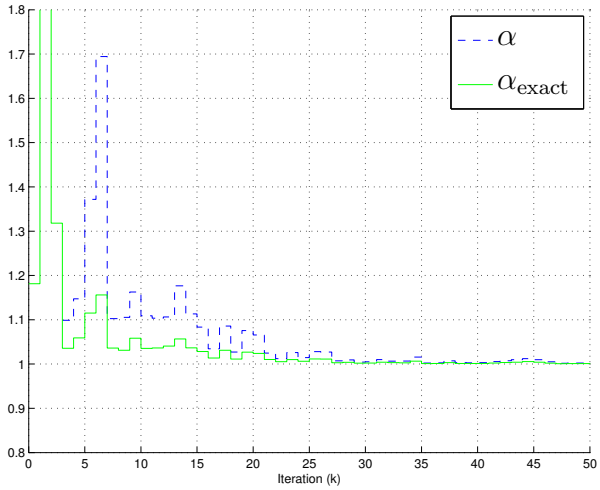

Fig. 3. Example V-A: Plots of the estimated suboptimality using the described method (in blue) and the exact suboptimality (in green).

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.

B. 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

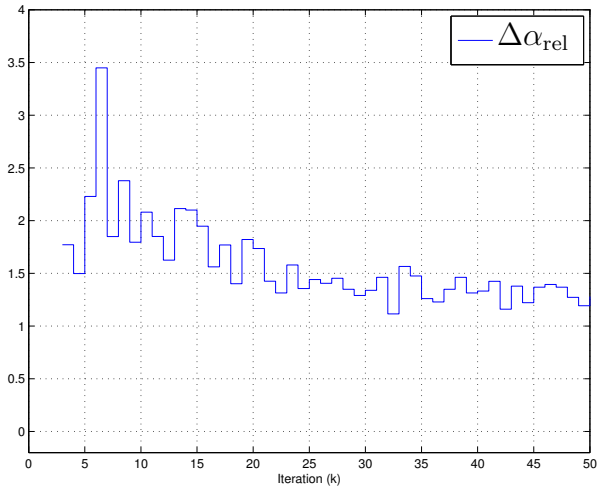

Fig. 4. Example V-A: Plot of the relative difference between the estimated and the exact suboptimality.

connected graph. The adjacency matrix will be symmetric, hence a undirected graph.

- 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 10000 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 will be $\gamma_\tau = 5 \cdot 10^{-4}$ 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

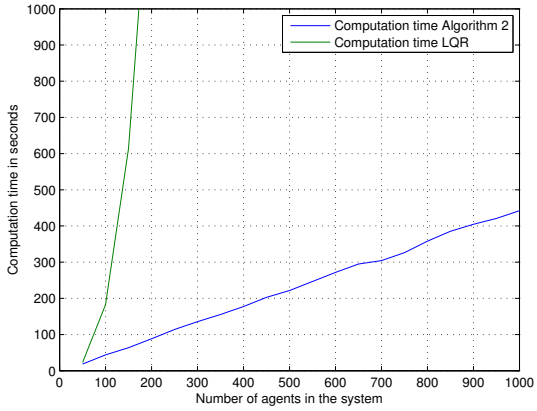

Fig. 5. Example V-B: Plot of the computation time with respect to the size of the system.

polynomial to the curve (since the computational complexity should be $\mathcal{O}(n^3)$), 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.

VI. Conclusions and Future Work

A. 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 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 evaluation of 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 criterion for the iteration process.

B. Future Work

In [26] 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 [13]. The system has more than 10000 states and the system matrices are of a sparse nature. In other words, the methods described in this paper are suitable for finding controllers for such deformable mirrors.

REFERENCES

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

[2] K. J. Åström. *Introduction to Stochastic Control Theory*. Dover, New York, 2006.

[3] B. Bamieh, F. Paganini, and M. A. Dahleh. Distributed control of spatially invariant systems. *IEEE Transactions on Automatic Control*, 47(7), July 2002.

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

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

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

[7] D. Bertsekas. On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Transactions on Automatic Control*, 21(2):174–184, April 1976.

[8] A. Bouhamidi, M. Heyouni, and K. Jbilou. Block arnoldi-based methods for large scale discrete-time algebraic riccati equations. *Journal of Computational and Applied Mathematics*, 236(6):1531–1542, 2011.

[9] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Hemisphere Pub. Corp., 1975.

[10] P. H. Calamai and J. J. Moré. Projected gradient methods for linearly constrained problems. *Math. Program.*, 39(1):93–116, October 1987.

[11] A. Gattami. Generalized linear quadratic control theory. In *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, CA, December 2006.

[12] J. C. Geromel and J. Bernussou. Optimal decentralized control of dynamic systems. *Automatica*, 18(5):545–557, 1982.

[13] R. Heimsten. *Study of a large deformable mirror concept*. PhD thesis, Department of Astronomy and Theoretical Physics, Lund University, Sweden, October 2011.

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

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

[16] K. Jbilou. An arnoldi based algorithm for large algebraic riccati equations. *Applied Mathematics Letters*, 19(5):437 – 444, 2006.

[17] P. Lancaster and L. Rodman. *Algebraic Riccati Equations*. Oxford University Press, 1995.

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

[19] L. S. Lasdon. *Optimization Theory for Large Systems*. Dover Publications Inc., 2002.

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

[21] L. Li and F. Paganini. LMI relaxation to Riccati equations in structured $\mathcal{H}_2$ control. *2006 American Control Conference*, pages 644–649, 2006.

[22] F. Lin, M. Fardad, and M. R. Jovanovic. On the optimal design of structured feedback gains for interconnected systems. In *Proceedings of the 48th IEEE Conference on Decision and Control*, Shanghai, China, 2009.

[23] F. Lin, M. Fardad, and M. R. Jovanovic. Augmented Lagrangian approach to design of structured optimal state feedback gains. *IEEE Transactions on Automatic Control*, 56(12):2923–2929, December 2011.

[24] K. Mårtensson and A. Rantzer. Gradient methods for iterative distributed control synthesis. In *Proceedings of the 48th IEEE Conference on Decision and Control*, Shanghai, China, 2009.

[25] K. Mårtensson and A. Rantzer. 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, July 2010.

[26] K. Mårtensson and A. Rantzer. A scalable modularized synthesis method for distributed Kalman filters. In *Proceedings of the 18th IFAC World Congress*, Milano, Italy, August 2011.

[27] A. Rantzer. A separation principle for distributed control. In *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, CA, December 2006.

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

[29] M. Rotkowitz and S. Lall. Decentralized control information structures preserved under feedback. In *Proceedings of the IEEE Conference on Decision and Control*, December 2002.

[30] M. Rotkowitz and S. Lall. A characterization of convex problems in decentralized control. *IEEE Trans. on Automatic Control*, 51(2):274–286, Feb 2006.

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

[32] C. Wenk and C. Knapp. Parameter optimization in linear systems with arbitrarily constrained controller structure. *IEEE Transactions on Automatic Control*, 25(3):496–500, June 1980.

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