51st IEEE Conference on Decision and Control
December 10-13, 2012. Maui, Hawaii, USA

Accelerated Iterative Distributed Controller Synthesis with a Barzilai-Borwein Step Size

Frederik Deroo*[a], Michael Ulbrich[b], Brian D. O. Anderson[c] and Sandra Hirche[a]

Abstract— **Distributed control of large-scale dynamical systems poses a new challenge to the field of control driven by the technological advances of modern communication networks. A particular challenge is the distributed design of such control systems. Here, a distributed iterative controller synthesis method for continuous time linear systems using a gradient descent method is presented. One of the main contributions is the determination of the step size according to a distributed Barzilai-Borwein (BB) method. As the control objective, we treat the finite horizon linear quadratic cost functional. The gradient approach uses communication only with direct neighbors and is based on the forward simulation of the system states and the backwards simulation of adjoint states. The effectiveness of the approach is shown by means of numerical simulations.**

I. Introduction

Over the last decade, there has been extensive renewed interest in the field of the control of large-scale dynamical systems. Typical application examples are power systems, water distribution, transportation, or traffic systems. Because of the spatial distribution and the size of these systems, communication between all subsystems is generally not possible thus making centralized classical control approaches not applicable. Since the 1970s a common approach to deal with this problem is decentralized control where each subsystem uses only its own information for the control task [1] while the influence from other subsystems is usually regarded as unwanted disturbance. Motivated by the wide-spread use of modern communication networks more and more research is now being conducted in the field of distributed control where, instead of limiting each subsystem to only its own information, communication between specific – but not all – subsystems is allowed. The overall goal of distributed control is to improve performance compared to decentralized control while maintaining the low complexity compared to centralized (full information) control.

Interesting approaches for distributed controller synthesis are for example [2], [3] and [4], and they all have in common that centralized global knowledge about the system model is required for the control design. However, for large-scale interconnected systems, this assumption might not be valid for several reasons. First of all, the system might be simply too large to formulate a central, global model for the whole system. Second, even if one can assume that a system model is available, the system might be too large to be handled by a centralized synthesis method. Third, privacy might be an issue which means that even though agents might be willing to collaborate and communicate, they might not be willing to give away their whole dynamic model to every other agent in the interconnected system. Lastly, a centralized approach requires a lot of effort when there are changes in the network, e.g. when a node is added or removed. A distributed approach on the other hand could react only in those nodes that are immediately affected by these changes.

A very promising approach to deal with these problems is introduced in [5]–[7], where a distributed gradient descent method is used to iteratively determine a locally optimal linear feedback matrix which minimizes an LQR cost functional. It is based on the computation of adjoint states and uses simulated trajectories to compute the gradient. The agents only require a model for their local dynamics and exchange measurement information only with direct neighbors. However, the authors do not give practical details on a possible step size selection, even though the step size is a crucial part for every gradient method greatly influencing the convergence rate. Furthermore their algorithm is not guaranteed to converge. Additionally, the resulting feedback matrices are dependent on the initial condition of the state used during the design process. These disadvantages motivate the following works of this paper.

This paper presents an accelerated distributed gradient descent method to determine a distributed linear control. The key contributions of the paper are the following: (1) Importantly, the main contribution is the development of a distributed step size scheme for this gradient descent based on the Barzilai-Borwein step size rule, which vastly improves the performance of the algorithm compared to a constant step size. This step size is computed in a distributed fashion by using a consensus phase in each iteration of the gradient descent algorithm. Further, distributed tests of the Armijo rule guarantee convergence. (2) In order to gain independence of the controller with respect to the initial condition of the state trajectories needed to compute the gradient, we introduce averaging over the initial condition. (3) Treatment of the finite horizon LQR-case. (4) The effectiveness of the developments presented in the paper is validated through numerical simulations.

The actual computation of the gradient for the algorithm builds on the infinite horizon foundation given by [7] which is the extension of [6] to continuous-time systems. This is

*Corresponding author.
[a]F. Deroo and S. Hirche are with the Institute of Automatic Control Engineering, Technische Universität München, D-80290 München, Germany, fred.deroo@tum.de, hirche@tum.de
[b]M. Ulbrich, Chair of Mathematical Optimization, Department of Mathematics, Technische Universitt München, Boltzmannstr. 3, D-85747 Garching b. München, Germany, mulbrich@ma.tum.de
[c]B.D.O. Anderson is with The Australian National University and National ICT Australia, Canberra ACT 2600 Australia. brian.anderson@anu.edu.au

motivated by the fact that many practical applications deal with continuous-time systems.

The remainder of the paper is organized as follows. In Section II, the problem formulation is presented. Section III shows the algorithm to determine the feedback matrix of a distributed controller. The main results, namely the usage of a Barzilai-Borwein step size, are introduced in Section IV. A numerical example is given in Section V, and the paper concludes with a summary in Section VI.

Notation

Given a matrix $A \in \mathbb{R}^{m\times n}$ with columns a_i, we can give a vectorized version of the matrix by associating the vector

$$\text{vec}(A) = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \in \mathbb{R}^{nm\times 1}.$$

The scalar product of two vectors $a, b \in \mathbb{R}^n$ is denoted by $\langle a, b\rangle$. A (block-)diagonal combination of n vectors or matrices M_i is denoted by $\text{diag}(M_1, ..., M_n)$. The term $A\bullet B$ denotes the Frobenius inner product of two matrices A,B which is defined as $\text{trace}(AB^T)$. The partial derivative of a matrix A with respect to a matrix B in the direction described by the matrix C ($\frac{\partial A}{\partial B}C$) is denoted by $A_B(C)$.

II. Problem formulation

We consider an interconnected large-scale linear system consisting of N subsystems. It is assumed that each agent's control signal can only directly influence the respective agent, but that each agent's state can be influenced by other agents' states. Thus, the dynamics of subsystem i can be written as

$$\dot{x}_i(t) = A_{ii}x_i(t) + B_iu_i(t) + \sum_{\substack{j=1\\ j\neq i}}^{N} A_{ij}x_j(t),\ i = 1, ..., N,$$

where $x_i \in \mathbb{R}^{n_i}$, $u_i \in \mathbb{R}^{m_i}$, $A_{ii} \in \mathbb{R}^{n_i\times n_i}$, $A_{ij} \in \mathbb{R}^{n_i\times n_j}$ and $B_i \in \mathbb{R}^{n_i\times m_i}$.

In order to define the set of neighbors of a subsystem i, we consider the directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ associated with the matrix A. The vertex set $\mathcal{V}$ is given by the set of subsystems $\mathcal{V} = \{1, ..., N\}$, and the edge set $\mathcal{E}$ contains the edge $(j, i) \in \mathcal{E}$ iff $A_{ij} \neq 0$. This means an edge $(j, i) \in \mathcal{E}$ iff subsystem i is influenced directly by the states of agent j. We define the set of neighboring nodes to node i as

$$\mathcal{N}_i = \{j|(i, j) \in \mathcal{E} \text{ or } (j, i) \in \mathcal{E}\}.$$

Additionally, we define the set of influenced nodes of node i as $\mathcal{I}_i = \{j|(i, j) \in \mathcal{E}\}$. By concatenation of the subsystems' states, the overall interconnected system can be written compactly as the continuous-time time-invariant linear system

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

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the input and $x_0 \in \mathbb{R}^n$ is the initial condition. The matrix A consists of the blocks A_{ij}, and B has the form $\text{diag}(B_i)$.The agents form a partition of the states of the system, so it must hold that $\sum_{i=1}^{N} n_i = n$.

The goal is to design a constant linear feedback $u(t) = -Kx(t)$ minimizing the following cost functional[1]

$$J(x, u) = \int_0^T x^T(t)Qx(t) + u^T(t)Ru(t)\text{d}t. \tag{2}$$

The symmetric weighting matrices $Q \in \mathbb{R}^{n\times n}$ (positive semidefinite) and $R \in \mathbb{R}^{m\times m}$ (positive definite) are assumed to be block-diagonal with the block-dimension corresponding to the respective subsystem size. Under this assumption, the cost functional is separable for each agent

$$J(x, u) = \sum_{i=1}^{N} \int_0^T x_i^T(t)Q_ix_i(t) + u_i^T(t)R_iu_i(t)\text{d}t.$$

As a results the optimization problem to find the optimal feedback is not coupled in the cost but only in the constraint to satisfy the underlying dynamics.

Assumption 1: The feedback matrix K is constrained to have a distributed structure where communication between subsystems is only allowed among neighbors, so the block $K_{ij} \neq 0$ only if $j \in \mathcal{N}_i$.

Note that this structure does not in general imply convexity of the optimization problem with respect to the controller parameters and that it is not related to any conditions for convexity like quadratic invariance [8]. The assumption stems from the idea that physically linked systems are likely to have a communication connection as well, see e.g. power line communication in power systems. We define the communication graph of the controller as $\mathcal{G}_c$ which is the undirected version of the graph $\mathcal{G}$. It is furthermore assumed that the communication signals are not quantized, and are noise- and delay-free.

III. Control Synthesis

In this section, we first present a distributed gradient descent method to solve the finite horizon LQR optimal control problem to achieve a structured linear state feedback matrix using only local information. Since the approach depends on the initial condition of the state x_0, an extension is presented in the second subsection in order to circumvent this dependency.

A. Gradient descent method

We consider the cost functional (2) where x satisfies (1) and $u = -Kx$. Then the following proposition holds.

Proposition 1: Considering system (1), the gradient of the cost functional (2) with respect to the entry of the feedback matrix K_{ij} is given by

$$(\nabla_K J)_{ij} = (2(RKX_0(T) - B^T X_P(T)))_{ij} \tag{3}$$

[1]Given the structure in K imposed below, it may be that non-constant, e.g. periodic $K(t)$, could achieve a lower value, but this is not considered. Note also that the optimizing K is x_0 dependent, but this point is explored further below

where $X_0(T)$ and $X_P(T)$ satisfy

$$\dot{X}_0(t) = A_K X_0(t) + X_0 A_K^T + x_0 x_0^T, \quad (4)$$
$$X_0(0) = 0,$$
$$X_P(T) = \int_0^T P(t) e^{A_K t} x_0 x_0^T e^{A_K^T t} \mathrm{d}t, \quad (5)$$

and where $P(t)$ is the solution to the matrix Lyapunov differential equation

$$-\dot{P}(t) = Q_K + P(t)A_K + A_K^T P(t),\ P(T) = 0, \quad (6)$$

where $A_K = A - BK$ and $Q_K = Q + K^T R K$.

Proof: Differentiating (6) with respect to the block K_{ij}, we get

$$-\dot{P}_{K_{ij}}(H_{ij})(t) = A_K^T P_{K_{ij}}(H_{ij})(t) + P_{K_{ij}}(H_{ij})(t) A_K + M(t) + M^T(t),$$

where $M(t) = (K^T R - P(t)B) K_{K_{ij}}(H_{ij})$. The solution of this can be given as

$$P_{K_{ij}}(H_{ij})(t) = \int_t^T e^{A_K^T(\tau - t)} (M(\tau) + M^T(\tau)) e^{A_K(\tau - t)} \mathrm{d}\tau.$$

The value of the cost functional (2) is $J = \mathrm{tr}(P(0)x_0 x_0^T)$. Using the solution of the Lyapunov differential equation (4), in addition to (5), we get

$$\begin{aligned} &\mathrm{tr}\left(P_{K_{ij}}(H_{ij})(0) x_0 x_0^T\right) \\ =&\mathrm{tr}\left(2K_{K_{ij}}(H_{ij})^T \left(RKX_0(T) - B^T X_P(T)\right)\right). \end{aligned} \quad (7)$$

This concludes the proof. ∎

This formulation of the gradient is not very interesting by itself since the solution of Lyapunov equations requires global information. So the question is if the (i,j)th block entry (with $j \in \mathcal{N}_i$) of the gradient can be computed by agent i with information only provided by its neighbors $\mathcal{N}_i$. Using the idea of dynamic dual decomposition, as presented in [9], this becomes possible by introducing adjoint states.

Proposition 2: Consider system (1). The gradient of the cost functional (2) with respect to the (i,j)th entry of the feedback matrix is given by

$$\begin{aligned} (\nabla_K J)_{ij} = &-2R_i \int_0^T u_i(t) x_j(t)^T \mathrm{d}t \\ &- B_i^T \int_0^T \lambda_i(t) x_j(t)^T \mathrm{d}t, \end{aligned} \quad (8)$$

where

$$\dot{\lambda}(t) = -A_K^T \lambda(t) - 2Q_K x(t),$$

where $\lambda(T) = 0$.

Proof: Using the solution of the Lyapunov differential equation of X_0 (4), the first term in (3) $(RKX_0(T))_{ij}$ can be rewritten as

$$\left(RK \int_0^T x(t)x(t)^T \mathrm{d}t\right)_{ij} = -R_i \int_0^T u_i(t) x_j(t)^T \mathrm{d}t. \quad (9)$$

If nodes i and j are neighbors, node i can compute this term using information only about the state of its neighbor j, thus local communication suffices.

The solution of (6) is given by

$$P(t) = \int_t^T e^{A_K^T(\tau - t)} Q_K e^{A_K(\tau - t)} \mathrm{d}\tau,$$

We can plug this into (5) to obtain

$$\begin{aligned} X_P(T) = \int_0^T \left(\int_t^T e^{A_K^T(\tau - t)} Q_K e^{A_K(\tau - t)} \mathrm{d}\tau \right) \\ \times e^{A_K t} x_0 x_0^T e^{A_K^T t} \mathrm{d}t. \end{aligned} \quad (10)$$

Also, we can give a solution for $\lambda(\tau)$ as

$$\lambda(t) = 2 \int_t^T e^{A_K^T(\tau - t)} Q_K x(\tau) \mathrm{d}\tau. \quad (11)$$

Plugging this into (10) we see that

$$X_P(T) = \frac{1}{2} \int_0^T \lambda(t) x^T(t) \mathrm{d}t,$$

∎

Now, the following algorithm can be applied.

Algorithm 1:

1) Simulate the states $x_i(t)$ of system (1) for the finite horizon T by communicating the states between neighboring nodes using a distributed ODE solver.
2) Simulate the adjoint states $\lambda_i(t)$ for the same horizon T with $\lambda(T) = 0$ in the backwards direction by communicating adjoint states between neighboring nodes.
$$\begin{aligned} \dot{\lambda}_i(t) = &\sum_{j \in \mathcal{I}_i} -(A_{K,ji})^T \lambda_j(t) - 2Q_i x_i(t) \\ &+2 \sum_{j \in \mathcal{I}_i} (K_{ji}^{(k)})^T R_j u_j(t). \end{aligned}$$
3) Every agent calculates the respective entries of the gradient by
$$\begin{aligned} \nabla_K J_{ij}^{(k)} = &-2R_i \int_0^T u_i(t) x_j(t)^T dt \\ &- B_i^T \int_0^T \lambda_i(t) x_j(t)^T dt. \end{aligned}$$
4) For each neighboring agent j, update
$$K_{ij}^{(k+1)} = K_{ij}^{(k)} - \gamma_k \nabla_K J_{ij}^{(k)}.$$
with a scalar step length γ_k, independent of i, j.
5) If $||(\nabla_K J)_{ij}^{(k)}|| < \epsilon$, stop. Otherwise, increase k and go back to 1.

The algorithm makes it clear that no global model knowledge is necessary to obtain the optimal feedback and that only neighbors need to communicate their respective states, co-states and inputs.

Our result is quite similar to the one in [7] where the infinite horizon is treated. However, their computation of the gradient for the infinite horizon case involves an approximation using finite state trajectories. It turns out now that their algorithm is not just an approximation of the infinite horizon case but in fact identical to the solution to the finite horizon case.

The biggest difference between the finite horizon formulation presented here and the finite time approximation of the

infinite horizon problem given in [7] is that the infinite horizon formulation has the formal requirement of an initial stabilizing feedback since otherwise the cost functional and the infinite horizon adjoint states are not defined. This may be difficult to find given the distributed setting of the problem, and is not necessary in the finite horizon formulation.

Note that the optimization problem is in general non-convex so the gradient descent method might only result in a local optimum.

Remark 1: The setup in this paper and the structure of the closed-loop system matrix $(A - BK)$ also allows optimal tracking to minimize the cost functional

$$J = \int_0^T u^T(t)Ru(t) + (x(t) - r(t))^T Q(x(t) - r(t))\mathrm{d}t,$$

where $r(t)$ is the desired reference trajectory. The optimal input for tracking [10] is given by

$$u(t) = -Kx(t) - R^{-1}B^T b(t)$$

where K is determined by Algorithm 1 and where $b(t)$ is the solution of

$$-\dot{b}(t) = (A - BK)^T b(t) + Qr(t),\ b(T) = 0.$$

Clearly, this differential equation can also be solved backwards given the presented information exchange topology and thus, the optimal tracking input can be computed distributedly without global knowledge.

B. Securing independence from the initial condition x_0

A closer look at Algorithm 1 reveals that two initial conditions need to be selected (Note that this is also true for the algorithm presented in [7]). The first is K_0 which initializes the actual decision variables, i.e. the entries of the feedback matrix K. The second is x_0 which is used to simulate the state trajectories in order to determine the gradient in a distributed fashion. This second initial condition, unrelated to the decision variables, is unwanted and we want to gain independence of it for two reasons. First, we would like to find the optimal feedback matrix independent of a specific state initial condition because we generally do not know the actual initial condition of the process in advance during the controller design, and because it might be quite different from the one used in the design algorithm. Secondly, a systematic approach to pick this one specific initial condition is not obvious.

Another, more model related point can be made. Since the agents are confined to models of their own dynamics and have no global model, all the information necessary for the controller design has to be extracted from the simulation data. This makes it important that all states are sufficiently excited using the initial condition of the state trajectory. By using just one specific, fixed x_0 as in the previous section, only a limited direction of the system behavior could be excited or the coupling structure of the system could prohibit the spreading of the signal. Imagine that x_0 happens to be only a unit base vector, such that only one state of one agent is excited by the initial condition. If the system is large, nodes that do not have a direct coupling to the excited node and are relatively far away will probably get very little information about the system dynamics and thus will not be able to determine an appropriate controller for every possible excitation of the system. Naturally, the resulting controller will work well for the specific x_0 but this is usually not desired when designing a feedback controller.

In order to get rid of this dependence and to ensure sufficient dynamical excitation of the system, we propose an averaging approach. Therefore we make the following assumption about the initial condition of the state x_0. Note that this assumption is only used for the initial condition of the simulated trajectories in the design process. This is not related to the actual initial condition of the online process and therefore does not change the original problem.

Assumption 2: The initial condition x_0 is a random variable, uniformly distributed on the surface of the n-dimensional unit sphere with expected value $\mathrm{E}[x_0 x_0^T] = \frac{1}{n}I$, where I is the identity matrix.

The cost functional (2) has to be changed to the following

$$J(x,u) = \mathrm{E}\left[\int_0^T x^T(t)Qx(t) + u^T(t)Ru(t)\mathrm{d}t\right], \quad (12)$$

where E represents the expected value with respect to the initial condition x_0. The value of this cost functional can then be given by $J = \frac{1}{n}\mathrm{trace}(P(0))$. Thus, the cost functional is independent of the initial condition of the state x_0.

Proposition 3: Considering system (1), and given Assumption 2, the gradient of the cost functional (12) with respect to the block entry of the feedback matrix K_{ij} is given by

$$(\nabla_K J)_{ij} = 2((RKX_0(T) - B^T X_P(T)))_{ij} \quad (13)$$

where the formulas $X_0(T)$, $X_P(T)$ and $P(t)$ are described in Proposition 1, with the difference that $\frac{1}{n}I$ replaces $x_0 x_0^T$ according to Assumption 2.

Proof: The proof is identical to the proof of Proposition 1 except that the product $x_0 x_0^T$ is replaced by the matrix $\frac{1}{n}I$. ∎

Because the distributed computation in Algorithm 1 depends on the simulation of states using an initial condition, we use the fact that $I = \sum_{m=1}^n e_m e_m^T$, where e_i is the i-th unit base vector and we recognize that Eqs. (4) and (5) are linear in $x_0 x_0^T$. This enables us to distribute the computation and to write $X_0(T)$ as

$$X_0(T) = \frac{1}{n}\sum_{m=1}^n \int_0^T x_m(\tau)x_m^T(\tau)\mathrm{d}\tau, \quad (14)$$

where $x_m(t)$ is the simulated state trajectory based on the initial condition e_m. We then define the adjoint state $\dot{\lambda}_m(t) = -A_K\lambda_m(t) - 2Q_K x_m(t)$ and with that we define the whole gradient as

$$(\nabla_K J)_{ij} = \frac{1}{n}\left(2RK\sum_{m=1}^n \int_0^T x_m(\tau)x_m(\tau)^T d\tau - B^T\sum_{m=1}^n \int_0^T \lambda_m(\tau)x_m(\tau)^T d\tau\right)_{ij}.$$

With that result, a new algorithm can be presented.

Algorithm 2:
1) Simulate the states $x_{m,i}(t)$ of System (1) for a finite horizon T by communicating the states between neighboring nodes for every initial condition e_m with $m = 1, ..., n$, and using a distributed ODE solver.
2) Simulate the adjoint states $\lambda_{m,i}(t)$ for the same horizon T in the backwards direction (with $\lambda_m(T) = 0$) by communicating adjoint states between neighboring nodes.
$$\dot{\lambda}_{m,i}(t) = \sum_{j \in \mathcal{I}_i} -(A_{K,ji})^T \lambda_{m,j}(t) - 2Q_i x_{m,i}(t) + 2 \sum_{j \in \mathcal{I}_i} (K_{ji}^{(k)})^T R_j u_{m,j}(t).$$
3) Every agent calculates the respective entries of the gradient by
$$(\nabla_K J)_{ij}^{(k)} = -\frac{1}{n} \left(2R_i \sum_{m=1}^{n} \int_0^T u_{m,i}(\tau) x_{m,j}(\tau)^T d\tau + B_i^T \sum_{m=1}^{n} \int_0^T \lambda_{m,i}(\tau) x_{m,j}(\tau)^T d\tau \right).$$
4) For each neighboring agent j, update
$$K_{ij}^{(k+1)} = K_{ij}^{(k)} - \gamma_k \nabla_K J_{ij}^{(k)}$$
with a suitable step length γ_k
5) If $||(\nabla_K J)_{ij}^{(k)}|| < \epsilon$, stop. Otherwise, increase k and go back to 1.

Using this algorithm, the resulting controller is independent of the initial condition of the state at the cost of requiring some global knowledge about the total number of states n in the system so that the agents know how many simulations they have to run. Also, there has to be some protocol that determines which unit base vector is used at what time as the initial condition.

We see that making use of Assumption 2 ensures the point we made earlier because in the algorithm, it leads to the simulation of the system with every unit base vector as an initial condition, thus exciting every state. Thus, the resulting controller will be optimal given the maximum possible amount of information about the system dynamics, without actually knowing the system model. It is important to note that this does not help to overcome the non-convexity of the problem with respect to the controller parameters.

IV. Step size selection

Since the gradient descent method is generally slow, the selection of a good step size is important. This is difficult in the presented setup because of the distributed nature of the solution of the problem. A reasonable method to determine the step size is the Barzilai-Borwein step size rule because it only needs first order information which is computed anyways when using a gradient method. So in this section, we first present the method to distributedly determine the Barzilai-Borwein step size. Afterwards, we present a method to distributedly check a condition that the step size must satisfy in order to guarantee convergence.

A. Barzilai-Borwein stepsize

The distributed setting of the problem makes finding a good step size γ_k for the algorithm presented above difficult. The straightforward approach of doing a line search to find the optimal γ_k is not applicable. In addition, the exact line search involves the solution of an optimization problem in every iteration step and can be computationally expensive. The easiest choice for a step size for the algorithm presented above is a constant step size. However, this choice does not guarantee convergence to a local optimum and it is rather slow. A different method for the selection of the step size is presented in [11] which has since been called "Barzilai-Borwein"-method. Applied to the presented problem, the Barzilai-Borwein method gives the step size with
$$\gamma_k = \frac{\langle \Delta\text{vec}(K), \Delta\text{vec}(K) \rangle}{\langle \Delta\text{vec}(K), \Delta\text{vec}(\nabla_K J) \rangle} \tag{15}$$
where $\Delta\text{vec}(K) = \text{vec}(K^{(k)}) - \text{vec}(K^{(k-1)})$ and $\Delta\text{vec}(\nabla_K J) = \text{vec}((\nabla_K J)^{(k)}) - \text{vec}((\nabla_K J)^{(k-1)})$. This computation requires additional storage because the feedback matrix and the gradient for the current and the last iteration are necessary.

This step size, however, cannot be computed in a distributed fashion using the formula from (15). In [12], a method is presented to compute the BB-step size distributedly in two steps. In the first step, each agent will use its own entries of the feedback and gradient matrix to determine an estimate of the BB-step size γ_k, and in the second step a distributed consensus algorithm will give the value of (15). As a first step, each node $i, i = 1, ...n$ initializes the two scalar values
$$\rho_i(k(0)) = \left\langle \Delta\text{vec}(K_i^T), \Delta\text{vec}(K_i^T) \right\rangle, \tag{16}$$
and
$$\psi_i(k(0)) = \left\langle \Delta\text{vec}(K_i^T), \Delta\text{vec}((\nabla_K J)_i^T) \right\rangle. \tag{17}$$
This means that every agent uses its own respective row(s) of the feedback and gradient matrix to compute the local parameters $\rho_i(k(0))$ and $\psi_i(k(0))$ corresponding to the iteration k. Then the agents start the following consensus iterations during which information exchange is necessary:
$$\rho_i(k(t+1)) = W_{ii}\rho_i(k(t)) + \sum_{j \in \mathcal{N}_i} W_{ij}\rho_j(k(t)), \tag{18}$$
$$\psi_i(k(t+1)) = W_{ii}\psi_i(k(t)) + \sum_{j \in \mathcal{N}_i} W_{ij}\psi_j(k(t)). \tag{19}$$
Here, W is a symmetric, non-negative with strictly positive diagonal entries, doubly stochastic matrix, compatible with graph $\mathcal{G}_c$. A common choice for W is according to the so-called Metropolis rule [12], [13]. This leads to the following proposition [12].

Proposition 4: If graph $\mathcal{G}_c$ is connected, then
$$\lim_{t \to \infty} \frac{\rho_i(k(t))}{\psi_i(k(t))} = \gamma_k, \quad \text{for all } i = 1, ..., n. \tag{20}$$

See [12] for the proof.

The consensus phase stops when the relative difference of $\alpha_i := \frac{\rho_i(k(t))}{\psi_i(k(t))}$ between consecutive iterations t and $(t-1)$ falls below a pre-specified threshold.

Remark 2: If the system topology contains isolated systems that cannot participate in the consensus (i.e. $\mathcal{G}_c$ is not connected), it might happen that the difference of the respective entries of the feedback matrices or the gradient matrices tends to zero, thus yielding an α_i which tends to infinity or not a number. In that case, it is necessary that this isolated node uses a fixed step size as its own step size while the other nodes can continue with the BB-step size.

Remark 3: It is quite obvious that this improved step size can also be used in the algorithm in [6].

B. Convergence

The possible step size choices discussed so far do not guarantee convergence. In order to guarantee convergence of the gradient method to a stationary point, the step size γ_k needs to satisfy the so-called Armijo rule [14] which is stated as follows

$$J(K^{(k)} + \gamma_k s^{(k)}) - J(K^{(k)}) \leq \alpha\gamma_k \text{vec}(\nabla_K J(K^{(k)}))^T s^{(k)} \tag{21}$$

where $\alpha \in (0,1)$, γ_k is initially the BB step size and where in our case $s^{(k)} = -\text{vec}(\nabla_{K^{(k)}} J(K^{(k)}))$. It can be shown that this condition is always satisfied for sufficiently small γ_k.

Since both the left and the right hand side of (21) are separable for each agent, all agents can compute their respective summands in the term

$$\begin{aligned}\Phi^{(k)} := &\sum_{i=1}^{N} J(K_i^{(k)} - \gamma_k \nabla_{K_i} J(K_i^{(k)})) - J(K_i^{(k)}) \\ &+ \alpha\gamma_k ||\text{vec}(\nabla_{K_i} J(K_i^{(k)}))||_2^2.\end{aligned} \tag{22}$$

Then, a consensus phase is used to determine the average of this term. The consensus is designed correspondingly to the one in the previous section. After reaching consensus, each agent can check whether this term is smaller or equal to 0. If that is not the case, the step size needs to be reduced until the condition is satisfied. Each test with a new step size requires a new consensus phase.

V. Numerical example

In this section, we present two numerical examples to illustrate the contributions of this paper.

A. Advantage of averaged initial condition

As mentioned in Section III-B, usually, when we are interested in finding an optimal controller, we want optimality independently of the actual initial condition of the state. In the following, we want to investigate the effect of computing the optimal feedback using an averaged initial condition according to Algorithm 2 and compare it to the result when we use the finite horizon version of Algorithm 1. To do that, we apply the algorithms to 500 randomly created stable systems. All systems have $n = 10$ states and $m = 10$ inputs ($B = I$), and the time horizon T is set to 10 s. The entries of the A matrices are picked randomly and also a number of off-diagonal entries are randomly set to zero to create a distributed structure. On average, 44 entries of the A matrices are nonzero. The threshold for the consensus phase in the BB step size determination is set to 10^{-3}, as is the threshold to stop the overall algorithm. The initial condition for Algorithm 1 is picked randomly. We then compare the actual cost using the resulting controllers from both controllers. Note that for fairness reasons and because we want optimality independently of the initial condition, we use a different randomly picked initial condition to compute the cost than the one that was used to compute the gradient according to Algorithm 1.

It turns out that the controller resulting from Algorithm 2 produces on average only 71.4% of the cost produced by the controller resulting from Algorithm 1. Of course, if the same initial condition is used for the cost trajectory as for the computation of the controller, Algorithm 1 performs better. However, on average the cost is only 1.5% higher when the averaged initial condition is used to compute the controller. This shows the superiority of Algorithm 2 clearly and motivates its usage.

B. Efficiency of Barzilai-Borwein step size

In order to demonstrate the effectiveness of the new step size rule, the algorithm (Algorithm 2) is again applied to 500 stable systems. All the system and simulation parameters are identical to the previous section. We compare the number of iterations needed with the BB step length and with a constant step length of $\gamma = 1$. For both step size methods, the Armijo rule from the previous subsection is used to guarantee convergence. The comparison shows that the BB step size is better for every example. On average, the algorithm needs 16 steps with the BB step size and 113 with the constant step size. We also compute the relative number of iterations ($\frac{\text{iterations BB}}{\text{iterations constant step length}}$), and on average less than 20% of iterations are needed. When comparing the actual resulting feedback matrices, the average maximum difference between the entries is also quite small, being less than 2%. Obviously, the number of consensus iterations also needs to be taken into account when considering the overall effort of the algorithm. The maximum number of consensus iterations in these simulations is 450 while the average number is 127. This number is taken over all outer iterations of the gradient descent. But when we keep in mind that the consensus iterations are far less demanding than the gradient descent iterations the computational and communication effort is drastically reduced, even when considering the effort of the consensus. Over all iterations, checking the Armijo rule requires 261 consensus iterations on average for the BB step size, while 2039 are necessary for the constant step size.

The results are compactly summarized in Table I.

A visualization for the convergence speed can be the evolution of the cost. To visualize this, we compute the cost for each example system for every iteration. The costs are then normalized with respect to the final optimal value and the

TABLE I

COMPARISON BETWEEN BB STEP SIZE AND CONSTANT STEP LENGTH

$\max\left(\frac{\text{iterations BB}}{\text{iterations constant step length}}\right)$ [%]	29.09
$\min\left(\frac{\text{iterations BB}}{\text{iterations constant step length}}\right)$ [%]	3.58
$\text{Mean}\left(\frac{\text{iterations BB}}{\text{iterations constant step length}}\right)$ [%]	16.03
Average difference in result [%]	1.81
Maximum number of consensus iterations for BB step size	450
Minimum number of consensus iterations for BB step size	53
Average number of consensus iterations for BB step size	127

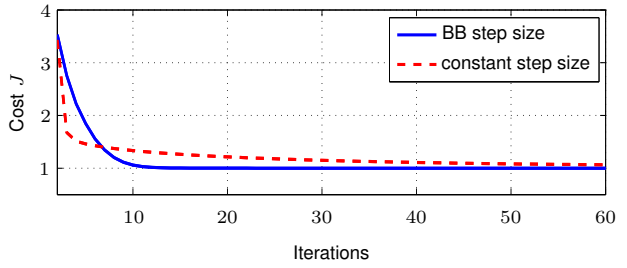

Fig. 1. Evolution of the averaged (500 random systems) normalized cost over the iterations

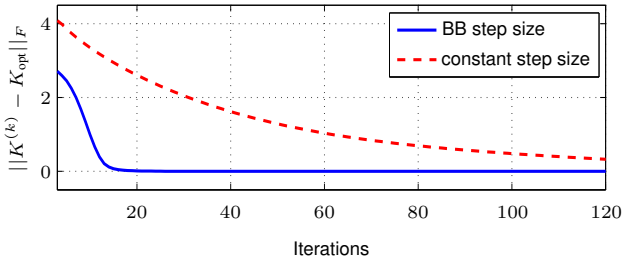

Fig. 2. Evolution of the averaged (500 random systems) Frobenius norm of $K^{(k)} - K_{\text{opt}}$ over the iterations

average is taken. The results for the BB step size and the constant step size are shown in Figure 1. Again, it becomes clear that convergence is achieved much faster with the BB step size.

Additionally, we also plot the Frobenius norm of the difference between the final resulting K_{opt} and $K^{(k)}$ ($||K^{(k)} - K_{\text{opt}}||_F$) for each iteration in Figure 2. Here too, it can be seen very well that the BB step size rule performs much better than a constant step size.

In principle, it is possible that the feedback matrix is not stabilizing in every iteration which might lead to numerical problems since the computation of the gradient is based on simulations of the system. However, since the Armijo rule makes sure that the cost is reduced from one iteration to the next, this problem is not likely to occur and never caused any issues in the presented simulation results.

VI. CONCLUSIONS

This paper introduces an accelerated distributed gradient descent method to determine a distributed linear controller. The approach uses a Barzilai-Borwein step size which can be determined using only information from neighbors in a consensus phase. Also, a method is presented to check the Armijo rule distributedly to determine step sizes that guarantee convergence. The computation of the search direction is based on the simulation of the trajectories of the system states and the adjoint states, and independence of the initial condition of the system state for these simulations is achieved. The effectiveness of the new step size is shown through numerical simulations and the computation effort is reduced significantly.

VII. ACKNOWLEDGEMENTS

The work of Frederik Deroo, Sandra Hirche, and Michael Ulbrich is supported by the German Research Foundation (DFG) within the Priority Program SPP 1305 “Control Theory of Digitally Networked Dynamical Systems”. The work of B. D. O. Anderson is supported by NICTA, which is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program, and the Australian Research Councils Discovery Project DP- 110100538.

REFERENCES

[1] D. D. Siljak, *Large-Scale Dynamic Systems: Stability and Structure*. North-Holland, 1978.

[2] C. Langbort, R. Chandra, and R. D’Andrea, “Distributed control design for systems interconnected over an arbitrary graph,” *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1502–1519, 2004.

[3] P. Shah and P. A. Parrilo, “H2-optimal decentralized control over posets: A state space solution for state-feedback,” in *Proc. 49th IEEE Conference on Decision and Control (CDC)*, 2010.

[4] A. S. M. Vamsi and N. Elia, “Design of distributed controllers realizable over arbitrary directed networks,” in *Proc. 49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 4795–4800.

[5] K. Martensson and A. Rantzer, “Gradient methods for iterative distributed control synthesis,” in *Proc. 48th IEEE Conference on Decision and Control (CDC) held jointly with 28th Chinese Control Conference*, 2009, pp. 549–554.

[6] ——, “Sub-optimality bound on a gradient method for iterative distributed control synthesis,” in *Proc. 19th International Symposium on Mathematical Theory of Networks and Systems*, 2010.

[7] ——, “A scalable method for continuous-time distributed control synthesis,” in *Proc. American Control Conf. (ACC)*, 2012, pp. 6308–6313.

[8] M. Rotkowitz and S. Lall, “A characterization of convex problems in decentralized Control,” *IEEE Transactions on Automatic Control*, vol. 51, no. 2, pp. 274–286, 2006.

[9] A. Rantzer, “Dynamic dual decomposition for distributed control,” in *Proc. American Control Conf. (ACC)*, 2009, pp. 884–888.

[10] B. D. O. Anderson and J. B. Moore, *Linear Optimal Control*. Prentice-Hall, 1971.

[11] J. Barzilai and J. M. Borwein, “Two-point step size gradient methods,” *IMA Journal of Numerical Analysis*, vol. 8, no. 1, pp. 141–148, 1988.

[12] G. Calafiore, L. Carlone, and M. Wei, “A distributed gradient method for localization of formations using relative range measurements,” in *Proc. IEEE Int. Symposium on Computer-Aided Control System Design (CACSD)*, 2010, pp. 1146–1151.

[13] L. Xiao, S. Boyd, and S. Lall, “A scheme for robust distributed sensor fusion based on average consensus,” in *Proc. Fourth International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005, pp. 63–70.

[14] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999.