A Scalable Modularized Synthesis Method for Distributed Kalman Filters

Mårtensson, Karl; Rantzer, Anders

2011

A Scalable Modularized Synthesis Method for Distributed Kalman Filters ⋆

Karl Mårtensson and Anders Rantzer *

* *The authors are with Automatic Control LTH, Lund University, Box 118, SE-221 00 Lund, Sweden, {karl, rantzer}@control.lth.se*

Abstract: This paper presents a scheme to construct distributed observers for a system consisting of agents interconnected in a graph structure. The scheme is an iterative procedure to improve the observers with respect to global performance. It is modular in the sense that each agent iterates using only local model information. As a consequence, the complexity of the scheme scales linearly with the size of the system. The resulting observers estimate states for each agent using only local measurements and model knowledge of its neighbors.
Distributed observers are suboptimal to centralized ones and it is desirable to measure the amount of performance degradation. We show how to use the variables of the synthesis scheme to also determine such a measure of the suboptimality.

1. INTRODUCTION

The classical Kalman filter problem has a well known solution where the filter gains are determined from the solution of a Riccati equation, see Kalman [1960] or Åström [2006]. However, for large-scale systems this solution may be intractable, for two main reasons. One is computational complexity. The other is that detailed models of all components may not be available in one place. The purpose of this paper is to address both these issues to develop a filter design scheme that is both scalable and modularized. To achive this, we will give up the idea of solving large-scale Riccati equations in detail, and instead design a distributed Kalman filter where each agent maintains estimates for its own internal states, based on a model of the local dynamics and the interaction with neighboring agents. In the distributed filter, each agent receives measurements from local sensors and exchanges information with its neighbors.

Decentralized Kalman filtering for data fusion in sensor networks was previously examined in Rao et al. [1993]. However, the methods presented there required all-to-all communication. The works Spanos et al. [2005], Olfati-Saber [2007] used consensus techniques to accomplish the data fusion. In Alriksson and Rantzer [2006] weighting matrices were computed which were used by each agent to linearly combine the sensor measurements and the neighbors state estimates to its own state estimate. Algorithms where each agent only determines a local state estimate of a large-scale, sparse system were presented in Khan and Moura [2008]. Here, neither the full state nor the full system model is known at any node of the system. A method using dual decomposition for distributed estimation is considered by Samar et al. [2007].

⋆ The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement number 224428, acronym CHAT. This work was also supported by the Linnaeus Grant LCCC from the Swedish Research Council.

The work in this paper relates to Mårtensson and Rantzer [2009, 2010]. In the first paper an distributed iterative learning control scheme was considered. The theory in the second paper treated an offline control synthesis scheme similar to the previous paper. In this paper we will use similar ideas to treat the distributed estimation problem. The synthesis scheme iteratively updates the observers to improve the performance. This is done by determining descent directions by calculating gradients via vector and matrix multiplications of the size of the state vector and dynamics matrix of each agent of the system. Hence, the result is an easily implementable procedure that is tractable for large-scale systems. Since distributed observers, i.e. observers with a structural constraint, are suboptimal compared to centralized solutions, a method of measuring the suboptimality of the intermediate observers is given. This method uses the same variables as the synthesis scheme. Hence, the method of finding suboptimality bounds is not more complex than the synthesis procedure.

In Section 2 we introduce the distributed systems considered. The notation used in the paper is also defined here. The theory behind the synthesis method is described in Section 3. We also show the procedure of updating the observer in this section. In Section 4 the theory for finding a bound of the suboptimality to the result of the previously mentioned method, is formulated. An example is given in section 5, showing the described methodology.

2. PROBLEM FORMULATION

2.1 Distributed Systems

Consider linear time-discrete systems

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

The noises w and e are assumed to be independent Gaussian with variance R_1 and R_2, respectively. The systems are restricted to have a distributed structure, described by an associated graph. The nodes (called agents) v_i,

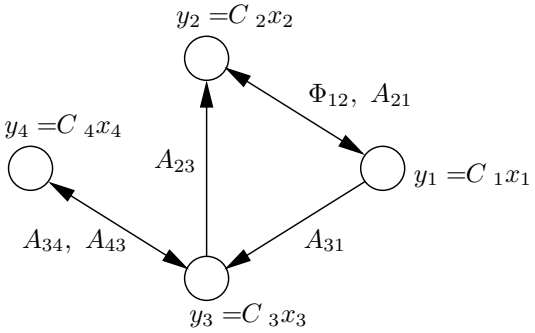

Fig. 1. Graphical representation of a distributed system. The arrows shows how each agent affects the others. The set $\mathcal{E} = \{(1,1),(2,2),(3,3),(4,4),(1,2),(2,1),(1,3),(3,2),(3,4),(4,3)\}$

$i = 1,\dots,n$, of the graph represent subsystems of the complete system, that is the agents are a partition of the states of the system. The variables in 1 corresponding to each agent v_i are denoted by x_i, y_i, w_i and e_i. The edges of the graph are represented by ordered pairs, (i,j). Let $\mathcal{E}$ be the collection of edges (by convention $(i,i) \in \mathcal{E},\ \forall i$). We call agents v_i and v_j neighbors if at least one of (i,j) and (j,i) are among the edges. An edge (i,j) means that agent v_i directly influence agent v_j through the dynamics of the system, that is

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

(throughout the paper, subscripts i,j on matrices will refer to blocks associated with agents i and j, respectively). Hence the dynamics matrix has a sparsity structure which resembles the graph structure of the distributed system. In the paper we assume that the output of each agent only depends on the states of that agent. This is represented by the C matrix being block-diagonal, that is,

$$C = \text{diag}(C_1,\dots,C_n),$$

where C_i is associated with agent v_i. One example of the complete setup is found in Figure 1.

We consider an observer to the system (1) on the form

$$\hat{x}(t+1) = A\hat{x} + K(y(t) - C\hat{x}(t))$$

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

$$\tilde{x}(t+1) = (A - KC)\tilde{x}(t) + w(t) - Ke(t) \tag{2}$$

Since we consider a distributed setup, we restrict each agent v_i to only use measurements from neighbors to calculate estimates $\hat{x}_i(t)$ of its states. This imposes constraints on the observer matrix K,

$$K_{ij} = 0 \quad \text{if} \quad (i,j),(j,i) \notin \mathcal{E} \tag{3}$$

With this restriction, the dynamics matrix $A - KC$ of the error $\tilde{x}$, satisfies the property that $[A - KC]_{ij} = 0$ unless agent i and j are neighbors. Denote

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

the subspace of matrices K satisfying the structural constraint.

3. ITERATIVE DISTRIBUTED OBSERVER SYNTHESIS

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

$$K = APC^T(R_2 - CPC^T)^{-1}$$
$$P = (A - KC)P(A - KC)^T + R_1 + KR_2K^T$$

The solution is easily obtained by solving the Riccati equation.

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

$$J(K,a) = \mathbf{E}\,(a^T\tilde{x})^2 \tag{4}$$

where $\tilde{x}$ satisfies (2). This performance is only defined for matrices K such that $A - KC$ is stable. The objective is to find a K that satisfies the restrictions in (3), which minimizes (4). The following proposition shows how to determine the gradient to J with respect to K.

Proposition 1. Given the system (2) and a matrix K such that $A - KC$ is stable, the gradient to J with respect to K is

$$\nabla_K J = 2\mathcal{A}\left[KR_2 - (A - KC)PC^T\right] \tag{5}$$

where

$$\mathcal{A} = (A - KC)^T\mathcal{A}(A - KC) + aa^T \tag{6}$$
$$P = (A - KC)P(A - KC)^T + R_1 + KR_2K^T \tag{7}$$

Proof. We use the fact that the cost function $J(K,a) = \text{tr}(Paa^T)$. In order to find the gradient to this expression, we will determine the differential dP with respect to K. For simplicity, let's make the following denotations

$$A_K = A - KC$$
$$M = \left(KR_2 - A_KPC^T\right)dK^T$$

Now, differentiating (7) it is easily seen that dP satisfies the following Lyapunov equation

$$dP = A_K dP A_K^T + M + M^T$$

This means that

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

Finally, using the expression for dP we have that

$$\begin{aligned}\text{tr}(dPaa^T) &= \text{tr}\left(2\sum_{t=0}^{\infty} A_K^t M(A_K^T)^t aa^T\right) \\ &= 2\text{tr}\left(\mathcal{A}\left(KR_2 - A_KPC^T\right)dK^T\right)\end{aligned}$$

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

The expression in (5) is not in a form appropriate for distributed calculations. We will show that by introducing adjoint (or dual) state variables, the expression can be rewritten to allow a distributed scheme.

Proposition 2. Given a matrix K such that $A - KC$ is stable, define the systems

$$\bar{x}(t+1) = (A - KC)^T\bar{x}(t) \tag{8}$$
$$\lambda(t-1) = (A - KC)\lambda(t) + (R_1 + KR_2K^T)\bar{x}(t) \tag{9}$$

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

$$\nabla_K J = 2\left(\sum_{t=0}^{\infty}\bar{x}(t)\bar{x}(t)^TKR_2 - \sum_{t=0}^{\infty}\bar{x}(t)\lambda(t)^TC^T\right) \tag{10}$$

Proof. Let's keep the notation $A_K = A - KC$ and introduce $R_K = R_1 + KR_2K^T$. Then

$$\lambda(t) = \sum_{j=t+1}^{\infty} A_K^{j-t-1} R_K \bar{x}(j) = \\ = \sum_{j=0}^{\infty} A_K^j R_K (A_K^T)^{j+1} \bar{x}(t)$$

Hence

$$\sum_{t=0}^{\infty} \bar{x}(t)\lambda(t)^T = \sum_{t=0}^{\infty}\sum_{j=0}^{\infty} \bar{x}(t)\bar{x}(t)^T A_K^{j+1} R_K (A_K)^T)^j = \\ = \mathcal{A} A_K P$$

Using this result in (5) proves the Proposition.

Since we impose a structure on K, the gradient $\nabla_K J$ needs to be projected to the subspace $\mathcal{K}$ defining this structure. This will be a descent direction of $J(K, a)$. To understand this, consider the restriction of J on $\mathcal{K}$. The gradient of this function is exactly the projection of $\nabla_K J$ on $\mathcal{K}$.

Using Proposition 2 we can iteratively change the matrix K to lower the cost J in a distributed manner. To determine each local direction to update the part of K which concerns agent i, the states of the agents are simulated for a time interval $[0, N]$. This is obviously done by only communicating states to neighboring agents. After this phase, the adjoint states $\lambda_i(t)$ are simulated for the same time interval $[0, N]$, but now backwards in time. Still, the structure of A_K and R_K insures that this can be done by only communicating with neighboring agents. Now, each agent can approximate the parts of (10) corresponding to neighboring agents and hence find a descent direction to change these parts of K. The method is formally described in the following Algorithm.

Algorithm 1. At iterate n, let $K^{(n)}$ is the current matrix and let the time interval for the simulation be $[0, N]$. To update $K^{(n)}$ in agent i:

(1) Let $\bar{x}(0) = a$ and simulate the states $x_i(t)$ of the system (8) for times $t = 0, \ldots, N$ by communicating states to and from neighboring agents.
(2) Simulate the adjoint states $\lambda_i(t)$ of system (9) for times $t = 0, \ldots, N$ backwards in time (with $\lambda(N) = 0$), by communicating adjoint states to and from neighboring agents

$$\lambda_i(t-1) = \sum_{j\in\mathcal{E}_i} (A - K^{(n)}C)_{ij}\lambda_j(t) \\ \left([R_1]_i \bar{x}_i(t) + \sum_{j\in\mathcal{E}_i} K_{ij}^{(n)} [R_2]_j K_{ji}^T \bar{x}(t) \right)$$

where $\mathcal{E}_i$ denotes the neighbors of agent i.
(3) For every neighboring agent j, calculate the approximation of the projected gradient by

$$G_{ij} = 2\left(\sum_{t=0}^{N} \bar{x}_i(t)[(K^{(n)})^T \bar{x}(t)]_j [R_2]_j \right. \\ \left. - \sum_{t=0}^{N} \bar{x}_i(t)\lambda_j(t) C_j^T \right)$$

(4) For each neighboring agent j, update

$$K_{ij}^{(n+1)} = K_{ij}^{(n)} - \gamma_n G_{ij}$$

for some step length γ_n.
(5) Increase n by one and go to 1) and repeat.

Remark 1. In the equations in Algorithm 1 the notation $[X]_j$ on a matrix or vector X, means the block in X corresponding to agent j and is used to e.g. separate from matrix subscripts.

Remark 2. In the equation in step 3, we find $[(K^{(n)})^T \bar{x}(t)]_j$. That this expression only uses local information to agent j follows easily from the structure of $K^{(n)}$.

An important property follows from the fact that the calculations only uses local information for each agent. This is that the complexity of the scheme is linear in the number of agents (or states) of the system. The structure of the matrices in the scheme implies that all matrix multiplications only are made on relatively small matrices compared to the full size of the matrices. This can be compared to solving Lyapunov equations which in standard implementation requires $\mathcal{O}(n^3)$ flops.

Another property is that introducing more agents to the system only changes the calculations for the agents which are to be neighbors to the new ones. Hence it does not involve much effort to add more agents to existing system.

4. SUBOPTIMALITY BOUND

4.1 Centralized Suboptimality Bound Calculations

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

$$J(K, a) \leq \alpha J(K_{\text{opt}}, a) \tag{11}$$

we would know that the performance of the solution K is within a factor α of the performance of the optimal solution K_{opt}. In other words, if we have a way of verifying that an α close to 1 is such that (11) holds, then we know that even though the current K might not be optimal, but at least we will not be able to find another K reducing the much more. For the remaining part of the paper let

$$J(K, a) = \sum_{t=0}^{N} \bar{x}(t)^T (R_1 + KR_2K^T)\bar{x}(t) \tag{12}$$

where $\bar{x}$ satisfies (8). This is a truncated version of the original cost function.

Theorem 3. If $\alpha \geq 1$ is such that for a given sequence of dual (or adjoint) variables $\lambda(t)$, with $\lambda(N) = 0$

$$J(K, a) \leq \alpha \min_{\substack{\bar{x}(t),\, \xi(t) \\ \bar{x}(0) = a}} \sum_{t=0}^{N} \Big[\bar{x}(t)^T R_1 \bar{x}(t) + \xi(t)^T R_2 \xi(t) \\ - 2\lambda(t)^T \left(\bar{x}(t+1) - A^T \bar{x}(t) + C^T \xi(t) \right) \Big] \tag{13}$$

then

$$J(K, a) \leq \alpha J(K_{\text{opt}}, a) \tag{14}$$

where

$$K_{\text{opt}} = \underset{K}{\text{argmin}} \min_{\bar{x}} J(K, a)$$

Proof. Assume that for $\alpha \geq 1$ and a given sequence $\lambda(t)$, that (13) holds. Then we have that

$$J(K_{\text{opt}}, a) = \begin{cases} \min\limits_{K, \bar{x}(t)} \sum\limits_{t=0}^{N} \bar{x}(t)^T (R_1 + K R_2 K^T) \bar{x}(t) \\ \text{subject to: } \bar{x}(t+1) = (A - KC)^T \bar{x}(t) \\ \qquad\qquad \bar{x}(0) = a \end{cases}$$

$$\geq \begin{cases} \min\limits_{\bar{x}(t), \xi(t)} \sum\limits_{t=0}^{N} \bar{x}(t)^T R_1 \bar{x}(t) + \xi(t)^T R_2 \xi(t) \\ \text{subject to: } \bar{x}(t+1) = A^T \bar{x}(t) - C^T \xi(t) \\ \qquad\qquad \bar{x}(0) = a \end{cases}$$

$$\geq \min_{\substack{\bar{x}(t), \xi(t) \\ \bar{x}(0) = a}} \sum_{t=0}^{N} \Big[\bar{x}(t)^T R_1 \bar{x}(t) + \xi(t)^T R_2 \xi(t) - 2\lambda(t)^T \left(\bar{x}(t+1) - A^T \bar{x}(t) + C^T \xi(t) \right) \Big]$$

where the second inequality comes from introducing the dual variables $\lambda(t)$. Hence, the inequality gives that if (13), holds, then so must (14).

The interpretation of the theorem is that over the time interval $[0, N]$ the cost can't be reduced by more than a factor $\frac{1}{\alpha}$ by changing K.

Theorem 3 gives a method to calculate a bound on the suboptimality α if we are given some variables $\lambda(t)$. This is done by evaluating $J(K, a) = a^T \lambda(0)$ for the given K, solving the minimization program in (13) and finally picking the smallest $\alpha \geq 1$ such that (13) holds. Now, we only need to choose $\lambda(t)$. As the name suggests, we will use the adjoint variables (9) which are determined in the algorithm for updating K. To motivate this choice of dual variables, we can refer to Pontryagin's maximum principle. Another motivation comes from examining

$$\max_{\lambda(t)} \min_{\bar{x}(t), \xi(t)} \sum_{t=0}^{N} \Big[\bar{x}(t)^T R_1 \bar{x}(t) + \xi(t)^T R_2 \xi(t) \underbrace{-2\lambda(t)^T (\bar{x}(t+1) - A^T \bar{x}(t) + C^T \xi(t))}_{\mathcal{L}(\bar{x}(t), \xi(t), \lambda(t))} \Big]$$

The minimization in the expression is exactly what shows up in the proof of Theorem 3. Hence, the maximization of this expression would give us $J(K_{\text{opt}}, a)$. Now, to find this saddle point, we differentiate the objective function $\mathcal{L}$

$$0 = \nabla_{\bar{x}(t)} \mathcal{L} = 2(R_1 \bar{x}(t) - \lambda(t-1) + A\lambda(t))$$
$$0 = \nabla_{\xi(t)} \mathcal{L} = 2(R_2 \xi(t) - C\lambda(t))$$

By considering $\nabla_{\bar{x}(t)} \mathcal{L} + K \nabla_{\xi(t)} \mathcal{L} = 0$ and substituting $\xi(t) = K^T \bar{x}(t)$ we get (9).

4.2 Distributed Suboptimality Bound Calculations

In Theorem 3 we have not yet introduced the structure of the matrix K. If this structure is considered, the minimization problem in the theorem can be split up to minimization problems for each agent of the system. This shows that the minimization problem actually can be solved in a distributed way, where each agent has its own minimization problem. Also, we understand that the minimization program has a complexity with a size *not* of the complete system but with a size of the separate subsystems. The following Proposition shows this.

Proposition 4. The minimization problem in Theorem 3 can be separated to minimization problems for each agent i by

$$\min_{\substack{\bar{x}_i(t), \xi_i(t) \\ \bar{x}_i(0) = a_i}} \sum_{t=0}^{N} \Big[\bar{x}_i(t)^T [R_1]_i \bar{x}_i(t) + \xi_i(t)^T [R_2]_i \xi_i(t) - 2\lambda_i(t)^T \left(\bar{x}_i(t+1) + C^T \xi_i(t) \right) - 2 \sum_{(i,j) \in \mathcal{E}} \lambda_j^T(t) A^T \bar{x}_i(t) \Big] \tag{15}$$

Proof. Summing (15) over all agent i in the system, gives the minimization problem in Theorem 3.

Since the cost $J(K, a)$ for a given matrix K equals $a^T \lambda(0)$, the left hand side of the inequality in Theorem 3, the cost J can be separated and evaluated for each agent. If we use this fact, we can rewrite Theorem 3 such that each agent determines its own suboptimality bound from only local measurements and information.

Corollary 5. If $\alpha_i \geq 1$ are such that for all agent i

$$a_i^T \lambda_i(0) \leq \alpha_i \min_{\substack{\bar{x}_i(t), \xi_i(t) \\ \bar{x}_i(0) = a_i}} \sum_{t=0}^{N} \Big[\bar{x}_i(t)^T [R_1]_i \bar{x}_i(t) + \xi_i(t)^T [R_2]_i \xi_i(t) - 2\lambda_i(t)^T \left(\bar{x}_i(t+1) + C^T \xi_i(t) \right) - 2 \sum_{(i,j) \in \mathcal{E}} \lambda_j^T(t) A^T \bar{x}_i(t) \Big]$$

then, letting $\alpha = \max\limits_i \alpha_i$,

$$J(K, a) \leq \alpha J(K_{\text{opt}}, a)$$

Proof. The proof follows easily from Theorem 3 and Proposition 4.

Obviously, the bounds that we get from Corollary 5 is greater than or equal to the bound we get from using Theorem 3. The bounds from Corollary 5 may even get much greater than the former. In order to make the distributed suboptimality bounds approach the centralized ones, we introduce slack variables d_{ij} in each agent i for each neighboring agent j. Now, each agent solves

$$a_i^T \lambda_i(0) \leq \alpha_i \Bigg(\min_{\substack{\bar{x}_i(t), \xi_i(t) \\ \bar{x}_i(0) = a_i}} \sum_{t=0}^{N} \Big[\bar{x}_i(t)^T [R_1]_i \bar{x}_i(t) + \xi_i(t)^T [R_2]_i \xi_i(t) - 2\lambda_i(t)^T \left(\bar{x}_i(t+1) + C^T \xi_i(t) \right) - 2 \sum_{(i,j) \in \mathcal{E}} \lambda_j^T(t) A^T \bar{x}_i(t) \Big] + \sum_{(i,j) \in \mathcal{E}} d_{ij} \Bigg)$$

with the restriction that $d_{ji} = -d_{ij}$. The variables d_{ij} will be used for all agents to have equal α_i. Using a consensus scheme, the agents change their d_{ij} to increase

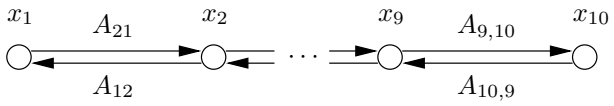

Fig. 2. Graphical representation of the system in the example. The arrows shows how each agent affects the others.

or decrease their suboptimality bound α_i. An important aspect when introducing the extra slack variables is that the minimization problem only has to be solved once for each agent. After that the variables d_{ij} are tuned in order to get consensus of the α_i.

5. EXAMPLE

The system

$$x(t+1) = Ax(t) + w(t)$$
$$y(t) = Cx(t) + e(t)$$

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

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

and with the remaining entries equal to zero. The noise $w(t)$ and $e(t)$ have variance $R_1 = I$ and $R_2 = I$, respectively. The output are the corrupted states, i.e. $C = I$. We wish to determine a Kalman filter matrix K to minimize the cost

$$J(K, a) = \mathbf{E}\,(a^T x)^2$$

where the constant a is picked form $\mathcal{N}(0, I)$. Since the maximal magnitude of the eigenvalues of A, $\rho(A) \approx 0.81$, we can initially set $K = 0$ when running Algorithm 1.

In Figures 3-4 shows the result of a simulation when using both Algorithm 1 and the results in Theorem 3 on the system. The number of update iterations in the Algorithm is set to be 50. Each update iteration consists of $N = 10$ time samples for which the systems (8)-(9) are simulated. In Figure 3, the plot denoted α_{exact} is the true suboptimality the Kalman filter K gives rise to, i.e.

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

Hence, the value for α_{exact} at each iteration is determined by solving the corresponding Lyapunov equation. In the figure we also plot α, the bound on the suboptimality calculated by using Theorem 3 (which is shown to be possible to do in a distributed way in Section 4.2). In Figure 4 we find the relative difference of α and α_{exact} of the plots in Figure 3, i.e.

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

In Theorem 3 it is not guarantee that the right hand side is always positive. In case the matrix K is far from the optimal, the result might be negative. In such cases it is not possible to find a positive α such that the inequality is satisfied. Iterations where this happens are omitted from the plots.

In Figure 3 we see that both α and α_{exact} approaches 1. This means that the cost when using matrix $K^{(n)}$ is approaching the optimal centralized Kalman filter solution. By analysing Figure 4 we see that the bound of the suboptimality is not more than a factor 3 from the actual suboptimality for most times. Also, the relative difference approaches a value of 1.4. Hence we understand that the bounds actually gives us valuable information of the current $K^{(n)}$.

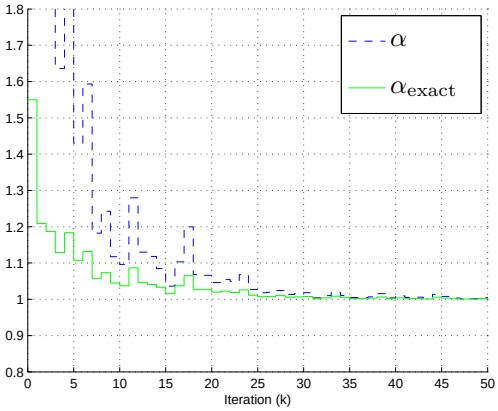

Fig. 3. Plots of the estimated suboptimality using the described method and the exact suboptimality.

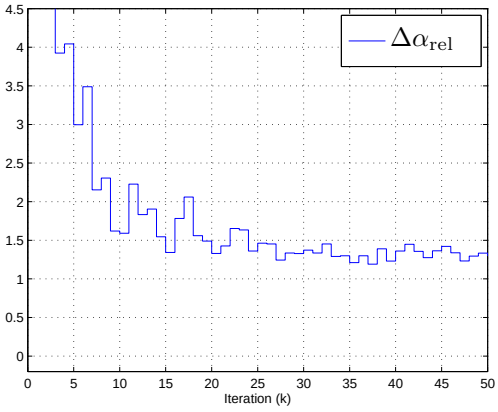

Fig. 4. Plot of the relative difference between the estimated and the exact suboptimality.

6. CONCLUSIONS AND FUTURE WORKS

6.1 Conclusions

By using adjoint states, we have seen a method to iteratively construct observers which will reduce a globally defined cost function. This is done by finding a descent direction to the cost function and change the observer matrix in that direction. The calculations in the scheme are shown to be possible to perform using only local information of the system for each agent, which results

in that the complexity of scheme is linear with respect to the number of agents.

Using the same adjoint variables that take part in the synthesis scheme, suboptimality bounds for the current observer can be determined. The suboptimality bounds give us an indication of the performance of the current observer and can function as a stopping criterion for the iterative synthesis scheme.

6.2 Future Works

We will work on connecting the state feedback synthesis with the observer synthesis to get a method for output feedback synthesis. We will for example look at what happens to the suboptimality bounds in this case.

REFERENCES

Peter Alriksson and Anders Rantzer. Distributed Kalman filtering using weighted averaging. In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*, Kyoto, Japan, July 2006.

Karl Johan Åström. *Introduction to Stochastic Control Theory*. Dover, New York, 2006. Reprint. Originally published by Academic Press 1970.

R.E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME–J.Basic Eng.*, 82 (2):35–45, 1960.

Usman A. Khan and Jos M. F. Moura. Distributing the kalman filters for large-scale systems. *IEEE Transactions on Signal Processing*, 56(10):4919–4935, Oct 2008.

Karl Mårtensson and Anders Rantzer. Sub-optimality bound on a gradient method for iterative distributed control synthesis. In *Proc. 19th International Symposium on Mathematical Theory of Networks and Systems*, Budapest, Hungary, July 2010.

Karl Mårtensson and Anders Rantzer. Gradient methods for iterative distributed control. In *Proceedings of 48th IEEE Conference on Decision and Control*, Shanghai, China, 2009.

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

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

Sikandar Samar, Stephen Boyd, and Dimitry Gorinevsky. Distributed estimation via dual decomposition. In *Proc. of the European Control Conference*, Kos, Greece, July 2007.

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