Diffusion Distributed Kalman Filtering with Adaptive Weights

Federico Cattivelli Ali H. Sayed

Department of Electrical Engineering
University of California, Los Angeles, CA 90095
Emails: {fcattiv, sayed}@ee.ucla.edu

***Abstract*— We study the problem of distributed Kalman filtering, where a set of nodes are required to collectively estimate the state of a linear dynamic system from their measurements. In diffusion Kalman filtering strategies, neighboring state estimates are linearly combined using a set of scalar weights. In this work we show how to optimally select the weights, and propose an adaptive algorithm to adapt them using local information at every node. The algorithm is fully distributed and runs in real time, with low processing complexity. Our simulation results show performance improvement in comparison to the case where fixed, non-adaptive weights are used.**

I. Introduction

We study the problem of distributed Kalman filtering, where a set of nodes are required to collectively estimate the state of a linear dynamic system from their individual measurements. The performance of the state estimation procedure will depend heavily on the collaboration strategy employed. In a centralized solution, all nodes send their measurements to a fusion center, which uses a conventional Kalman filtering algorithm to obtain the optimal state estimate, and then sends the resulting estimate back to every node. This strategy may require large amounts of energy for communications and has the potential for a critical failure point at the central node.

Distributed strategies are an attractive alternative, since they are in general more robust than centralized solutions, may require fewer communications, and allow parallel processing. Distributed Kalman filtering was proposed before in the context of diffusion estimation in [1], [2], and in the context of average consensus in [3], [4], [5]. Our focus is on diffusion Kalman filtering , where nodes communicate only with their neighbors, and no fusion center is present. Diffusion strategies are robust to node and link failure and are flexible for ad-hoc deployment and topology changes.

In diffusion strategies, the estimates available within the neighborhood of a certain node are linearly combined using a set of scalar weights. In this work we propose an adaptive algorithm to adapt these weights, based on the data statistics. The algorithm is fully distributed and runs in real time, with low processing complexity. The algorithm is based on previous work on diffusion LMS [6]. We extend the methodology of [6] to the Kalman filtering case, and apply the weight adaptation strategy to our diffusion Kalman filtering algorithm proposed in [1].

This material was based on work supported in part by the National Science Foundation under awards ECS-0601266 and ECS-0725441.

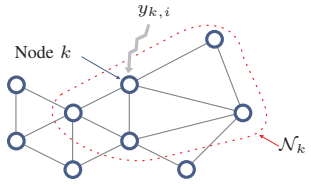

Fig. 1. At time i, every node k in the network obtains a measurement $y_{k,i}$.

II. Background

A. Data model and Problem Formulation

Consider a state-space model of the form:

$$\begin{aligned} \boldsymbol{x}_{i+1} &= F_i \boldsymbol{x}_i + G_i \boldsymbol{n}_i \\ \boldsymbol{y}_i &= H_i \boldsymbol{x}_i + \boldsymbol{v}_i \end{aligned} \tag{1}$$

where $\boldsymbol{x}_i \in \mathbb{C}^M$ and $\boldsymbol{y}_i \in \mathbb{C}^{pN}$ denote the state and measurement vectors of the system, respectively, at time i, and M, N and p are positive integers. The signals $\boldsymbol{n}_i$ and $\boldsymbol{v}_i$ denote state and measurement noises, respectively, and are assumed to be zero-mean, uncorrelated and white, with covariance matrices denoted by

$$\mathrm{E} \begin{bmatrix} n_i \\ v_i \end{bmatrix} \begin{bmatrix} n_j \\ v_j \end{bmatrix}^* = \begin{bmatrix} Q_i & 0 \\ 0 & R_i \end{bmatrix} \delta_{ij} \tag{2}$$

where the operator * denotes complex conjugate transposition and δ_{ij} is the Kronecker delta. The initial state $\boldsymbol{x}_0$ is assumed to be zero-mean, with covariance matrix $\Pi_0 > 0$, and is uncorrelated with $\boldsymbol{n}_i$ and $\boldsymbol{v}_i$, for all i. We further assume that $R_i > 0$. The case where $\mathrm{E}\, \boldsymbol{n}_i \boldsymbol{v}_i^* \neq 0$ can always be transformed into an equivalent problem of the form (2) as explained in [7]. The cases where $\mathrm{E}\, \boldsymbol{x}_0 \neq 0$ or when (1) has a deterministic can be handled similarly.

B. Diffusion Kalman Filtering

Consider the case where we have N nodes spatially distributed over some region. We say two nodes are connected if they can communicate directly to each other. A node is always connected to itself. The set of nodes connected to node k is the

neighborhood of node k, and is denoted by $\mathcal{N}_k$. It is assumed that at time i, every node k collects a measurement $\boldsymbol{y}_{k,i} \in \mathbb{C}^p$ according to model (1) as follows:

$$\boldsymbol{y}_{k,i} = H_{k,i}\boldsymbol{x}_i + \boldsymbol{v}_{k,i} \qquad k = 1, ..., N \tag{3}$$

The process is shown schematically in Fig. 1. Model (1) is related to (3) by stacking the N measurements across all nodes at time i as follows:

$$\begin{array}{rcl} \boldsymbol{y}_i & = & \mathrm{col}\{\boldsymbol{y}_{1,i}, \ldots, \boldsymbol{y}_{N,i}\} \\ H_i & = & \mathrm{col}\{H_{1,i}, \ldots, H_{N,i}\} \\ \boldsymbol{v}_i & = & \mathrm{col}\{\boldsymbol{v}_{1,i}, \ldots, \boldsymbol{v}_{N,i}\} \end{array} \tag{4}$$

We further assume that the measurement noises $\boldsymbol{v}_{k,i}$ are spatially uncorrelated, i.e.,

$$\mathrm{E}\, \boldsymbol{v}_{k,i}\boldsymbol{v}_{l,j}^* = R_{k,i}\delta_{i,j}\delta_{k,l}$$

where $R_{k,i} > 0$ for all k, i.

The objective in distributed Kalman filtering implementations is for every node k in the network to compute an estimate of the unknown state $\boldsymbol{x}_i$, while sharing data only with its neighbors $\{l \in \mathcal{N}_k\}$. We will denote the estimates of $\boldsymbol{x}_i$ obtained by node k and based on local observations up to time j as $\hat{x}_{k,i|j}$.

An algorithm for distributed Kalman filtering, where all nodes communicate only with their neighbors was proposed in [1], and is known as *diffusion Kalman filter*. The diffusion KF algorithm and its variants require the introduction of a $N \times N$ matrix C with real, non-negative entries $c_{l,k}$ satisfying:

$$\mathbb{1}^* C = \mathbb{1}^* \qquad c_{l,k} = 0 \text{ if } l \notin \mathcal{N}_k \qquad c_{l,k} \geq 0 \tag{5}$$

where $\mathbb{1}$ is a column vector with unit entries of size N. We call C the *diffusion* matrix, since it governs the diffusion process, and plays an important role in the steady-state performance of the network. The diffusion Kalman filtering is obtained by adding a diffusion step consisting of a convex combination of neighboring estimates after a conventional Kalman filtering measurement update (see [1] for details). The algorithm is shown below for convenience.

Algorithm 1: Diffusion Kalman filter

Consider a state-space model as in (1) and a diffusion matrix as in (5). Start with $\hat{x}_{k,0|-1} = 0$ and $P_{k,0|-1} = \Pi_0$ for all k, and at every time instant i, compute at every node k:

Step 1: Incremental Update:

$\psi_{k,i} \leftarrow \hat{x}_{k,i|i-1}$
$P_{k,i} \leftarrow P_{k,i|i-1}$
for every neighboring node $l \in \mathcal{N}_k$, repeat:
$R_e \leftarrow R_{l,i} + H_{l,i}P_{k,i}H_{l,i}^*$
$\psi_{k,i} \leftarrow \psi_{k,i} + P_{k,i}H_{l,i}^* R_e^{-1}\left[y_{l,i} - H_{l,i}\psi_{k,i}\right]$
$P_{k,i} \leftarrow P_{k,i} - P_{k,i}H_{l,i}^* R_e^{-1} H_{l,i}P_{k,i}$
end

Step 2: Diffusion Update:

$\hat{x}_{k,i|i} \leftarrow \sum_{l\in\mathcal{N}_k} c_{l,k}\psi_{l,i}$
$P_{k,i|i} \leftarrow P_{k,i}$
$\hat{x}_{k,i+1|i} = F_i\hat{x}_{k,i|i}$
$P_{k,i+1|i} = F_iP_{k,i|i}F_i^* + G_iQ_iG_i^*$

Algorithm 1 requires that at every instant i, nodes communicate to their neighbors their measurement matrices $H_{k,i}$, the covariance matrices $R_{k,i}$, and the measurements $y_{k,i}$ for the incremental update, and their intermediate estimates $\psi_{k,i}$ for the diffusion update. It is important to note that the matrices $P_{k,i|i}$ and $P_{k,i|i-1}$ do *not* represent the covariance of the state estimation errors $\tilde{x}_{k,i|i}$ and $\tilde{x}_{k,i|i-1}$ as in the conventional Kalman filter, since the diffusion update is not taken into account in the recursions for these matrices. Exact expressions for the new covariances of the state error estimates are provided in [1].

III. Adaptive Weights

The aim of this work is to design the weighting coefficients $c_{l,k}$ in the diffusion update of Algorithm 1. The objective is to compute a set of weights that adapt to changes in the data statistics. Moreover, we seek solutions that achieve this adaptation by using information locally available at every node. In this way, the algorithm is fully distributed in the sense that it does not require access to global information.

A. Problem Formulation

Let $\boldsymbol{\psi}_{k,i}$ denote the intermediate estimate available at node k and time i at the end of the incremental update (see Algorithm 1), and let $\boldsymbol{\Psi}_i$ denote the $M \times N$ matrix obtained by stacking these intermediate estimates row-wise, i.e.,

$$\boldsymbol{\Psi}_i = [\boldsymbol{\psi}_{1,i} \;\; \boldsymbol{\psi}_{2,i} \;\; \ldots \;\; \boldsymbol{\psi}_{N,i}]$$

Let A denote an $N \times N$ matrix with individual entries $\{c_{l,k}\}$, and let c_k denote the k-th column of A. The objective is for every node k to obtain a set of coefficients $\{c_{l,k}\}_{l=1,\ldots,N}$ that solve the following optimization problem:

$$\begin{array}{ll} \underset{c_k}{\text{minimize}} & \mathrm{E}\,\|\boldsymbol{x}_i - \boldsymbol{\Psi}_i c_k\|^2 \\ \text{subject to} & c_{l,k} = 0 \text{ if } l \notin \mathcal{N}_k \quad \text{and} \quad \mathbb{1}^T c_k = 1 \end{array} \tag{6}$$

Problem (6) minimizes the norm of the error between the desired state at time i, and a linear combination of the intermediate estimates $\boldsymbol{\psi}_{k,i}$. The restriction that $c_{l,k} = 0$ if $l \notin \mathcal{N}_k$ ensures that node k will only need to access the estimates from its neighbors $l \in \mathcal{N}_k$. Thus, access to information outside of the neighborhood is not required. We also ensure that the coefficients add up to one through the constraint $\mathbb{1}^T c_k = 1$.

We now remove the constraint $c_{l,k} = 0$ if $l \notin \mathcal{N}_k$ from (6). Let n_k denote the degree of node k (i.e., the number of nodes connected to node k including itself), and let $\{k_1, \ldots, k_{n_k}\}$ denote the indexes of the neighbors of node k. We define a matrix S_k as follows

$$S_k = [e_{k_1} \ldots e_{k_{n_k}}] \qquad (N \times n_k)$$

where e_l denotes the l-th column of an $N \times N$ identity matrix. Thus, we can rewrite (6) as follows

$$\begin{array}{ll} \underset{b_k}{\text{minimize}} & \mathrm{E}\,\|\boldsymbol{x}_i - \boldsymbol{\Psi}_{k,i} b_k\|^2 \\ \text{subject to} & \mathbb{1}^T b_k = 1 \end{array} \tag{7}$$

where $b_k \in \mathbb{R}^{n_k}$ is a vector containing the non-zero entries of c_k, i.e., $c_k = S_k b_k$, and $\boldsymbol{\Psi}_{k,i} = \boldsymbol{\Psi}_i S_k$.

B. Lagrange minimization

The cost in (7) can be minimized by forming the Lagrangian

$$\begin{aligned} L(b_k,\lambda_k) &= \mathrm{E}\,\boldsymbol{x}_i^*\boldsymbol{x}_i - 2\mathrm{Re}\{\mathrm{E}\,\boldsymbol{x}_i^*\boldsymbol{\Psi}_{k,i}\}b_k + \\ & \quad b_k^T\mathrm{Re}\{\mathrm{E}\,\boldsymbol{\Psi}_{k,i}^*\boldsymbol{\Psi}_{k,i}\}b_k + 2\lambda_k(\mathbb{1}^T b_k - 1) \end{aligned}$$

Let

$$q_{k,i} \triangleq \mathrm{Re}\{\mathrm{E}\,\boldsymbol{\Psi}_{k,i}^*\boldsymbol{x}_i\} \qquad Q_{k,i} \triangleq \mathrm{Re}\{\mathrm{E}\,\boldsymbol{\Psi}_{k,i}^*\boldsymbol{\Psi}_{k,i}\}$$

Differentiating the above expression and setting the gradient to zero, we obtain:

$$-2q_{k,i}^T + 2b_k^T Q_{k,i} + 2\lambda_k\mathbb{1}^T = 0$$

Assuming $\mathrm{E}\,\boldsymbol{\Psi}_{k,i}^*\boldsymbol{\Psi}_{k,i} > 0$, we have

$$b_k^{\mathrm{opt}} = Q_{k,i}^{-1}[q_{k,i} - \lambda_k^{\mathrm{opt}}\mathbb{1}] \tag{8}$$

and using the constraint $\mathbb{1}^T b_k = 1$, we obtain

$$\lambda_k^{\mathrm{opt}} = \frac{\mathbb{1}^T Q_{k,i}^{-1} q_{k,i} - 1}{\mathbb{1}^T Q_{k,i}^{-1}\mathbb{1}} \tag{9}$$

C. Gradient-Descent Solution

Equations (8) and (9) provide a way of computing the optimal set of weights at every iteration, provided that the correlations $\mathrm{E}\,\boldsymbol{\Psi}_{k,i}^*\boldsymbol{x}_i$ and $\mathrm{E}\,\boldsymbol{\Psi}_{k,i}^*\boldsymbol{\Psi}_{k,i}$ are known. These quantities can be computed provided the observation model at every node in the network is known [1] and at the expense of several computations. However, in distributed implementations, these quantities are not known globally, and computations need to be minimized. Thus, in this section we seek an adaptive implementation where the nodes adapt their weights relying solely on local data available within their neighborhoods, while at the same time attempting to keep the computational complexity low. Again, our procedure is inspired on [6]. We first derive a gradient-descent solution to problem (6) and then modify the iteration to introduce adaptivity.

Let $\mathcal{P}_{V_k}$ denote the projection from $\mathbb{R}^{n_k}$ onto $V_k = \{x \in \mathbb{R}^{n_k} : \mathbb{1}^T x = 1\}$, which is given by [6]:

$$\mathcal{P}_{V_k}(a) = \left(I_{n_k} - \frac{1}{n_k}\mathbb{1}\mathbb{1}^T\right)a + \frac{1}{n_k}\mathbb{1} \qquad \text{for all } a \in \mathbb{R}^{n_k}$$

and let

$$A_k \triangleq I_{n_k} - \frac{1}{n_k}\mathbb{1}\mathbb{1}^T$$

Then we can rewrite (7) as:

$$\underset{b_k}{\text{minimize}}\ \mathrm{E}\,\|\boldsymbol{x}_i - \boldsymbol{\Psi}_{k,i}\mathcal{P}_{V_k}(b_k)\|^2 \tag{10}$$

The gradient of the cost in (10) is given by:

$$\nabla^*(b_k) = -2A_k q_{k,i} + 2A_k Q_{k,i}\mathcal{P}_{V_k}(b_k) \tag{11}$$

Gradient-descent techniques minimize convex functions of the form (10) through a set of iterations of the form [8], [9]:

$$b_{k,i+1} = b_{k,i} - \frac{\mu_{k,i}}{2}\nabla^*(b_{k,i})$$

Thus, we propose minimizing (10) using a gradient-descent algorithm as follows:

$$b_{k,i+1} = b_{k,i} + \mu_{k,i}A_k[q_{k,i} - Q_{k,i}\mathcal{P}_{V_k}(b_{k,i})] \tag{12}$$

Notice that if $b_{k,i}$ satisfies $\mathbb{1}^T b_{k,i} = 1$, we have $\mathcal{P}_{V_k}(b_{k,i}) = b_{k,i}$ and therefore our iteration becomes

$$b_{k,i+1} = b_{k,i} + \mu_{k,i}A_k[q_{k,i} - Q_{k,i}b_{k,i}] \tag{13}$$

Recall that the desired weights can be recovered through $c_{k,i+1} = S_k b_{k,i+1}$.

D. Adaptive Solution

In order make iteration (13), we introduce the following approximations:

$$\begin{aligned} q_{k,i} &= \mathrm{Re}\{\mathrm{E}\,\boldsymbol{\Psi}_{k,i}^*\boldsymbol{x}_i\} \approx \mathrm{Re}\{\Psi_{k,i-1}^*\hat{x}_{k,i-1|i}\} \\ Q_{k,i} &= \mathrm{Re}\{\mathrm{E}\,\boldsymbol{\Psi}_{k,i}^*\boldsymbol{\Psi}_{k,i}\} \approx \mathrm{Re}\{\Psi_{k,i-1}^*\Psi_{k,i-1}\} \end{aligned}$$

where

$$\Psi_{k,i} = \Psi_i S_k \quad \text{and} \quad \Psi_i = [\psi_{1,i}\ \ \psi_{2,i}\ \ \ldots\ \ \psi_{N,i}]$$

Recall that $\psi_{k,i}$ is the intermediate estimate available at node k after the incremental update, and this estimate is communicated to all neighbors of node k. Using the above approximations into (13), we obtain the adaptive weights algorithm provided below:

Adaptive Weights for Diffusion Kalman Filtering

Start with $b_{k,-1} \in \mathbb{R}^{n_k}$ for every node k such that $\mathbb{1}^T b_{k,-1} = 1$. Then, for $i \geq 0$, repeat:

$$\left\{\begin{array}{rl} b_{k,i+1} = & b_{k,i} + \mu_{k,i}A_k \times \\ & \quad \mathrm{Re}\left\{\Psi_{k,i-1}^*(\hat{x}_{k,i-1|i} - \Psi_{k,i-1}b_{k,i})\right\} \\ c_{k,i+1} = & S_k b_{k,i+1} \end{array}\right. \tag{14}$$

One possible choice for the step-size $\mu_{k,i}$ that will guarantee $c_{k,i}$ will have non-negative entries for all k and i is

$$\begin{aligned} \eta_{k,i+1} &= \max\{0, b_{k,i+1}\} \\ c_{k,i+1} &= \tfrac{1}{\mathbb{1}^T\eta_{k,i+1}}S_k\eta_{k,i+1} \end{aligned}$$

where the above maximum is entry-wise.

Algorithm 2: Diffusion Kalman filter with Adaptive Weights

Consider a state-space model as in (1) and a diffusion matrix as in (5). Start with $\hat{x}_{k,0|-1} = 0$ and $P_{k,0|-1} = \Pi_0$ for all k, and at every time instant i, compute at every node k:

Step 1: Incremental Update: same as in Algorithm 1

Step 2: Diffusion Update:

$\hat{x}_{k,i|i} \leftarrow \sum_{l\in\mathcal{N}_k} c_{l,k,i}\psi_{l,i}$
$P_{k,i|i} \leftarrow P_{k,i}$
$\hat{x}_{k,i+1|i} = F_i\hat{x}_{k,i|i}$
$P_{k,i+1|i} = F_i P_{k,i|i}F_i^* + G_i Q_i G_i^*$

Step 3: Adapt the weights $\{c_{l,k,i}\}$ using (14)

IV. Simulations

In order to illustrate the performance of Algorithms 2, we present a simulation example in Figs. 2-4. We consider the problem of estimating and tracking the position of a projectile. We assume that the projectile is in proximity of an adaptive network, where the sensors obtain noisy measurements of the position of the projectile.

The state x of the system is a vector of dimension 6, formed by stacking the 3-D velocity and 3-D position of the object, which evolves as follows:

$$\underbrace{\begin{bmatrix}\dot{\boldsymbol{v}}\\ \dot{\boldsymbol{d}}\end{bmatrix}}_{\dot{\boldsymbol{x}}} = \underbrace{\begin{bmatrix}0 & 0\\ I_3 & 0\end{bmatrix}}_{\Phi}\underbrace{\begin{bmatrix}\boldsymbol{v}\\ \boldsymbol{d}\end{bmatrix}}_{\boldsymbol{x}} + \underbrace{\begin{bmatrix}\begin{bmatrix}0\\0\\-g\end{bmatrix}\\ 0\end{bmatrix}}_{c}$$

where g is the gravity constant (we use $g = 10$). The above model is discretized (see [2] for details) by denoting $\boldsymbol{w}_i = \boldsymbol{x}(i\delta)$, to obtain:

$$\begin{aligned}\boldsymbol{w}_{i+1} &= F\boldsymbol{w}_i + G_i\boldsymbol{n}_i + u\\ \boldsymbol{z}_{k,i} &= H_{k,i}\boldsymbol{w}_i + \boldsymbol{v}_{k,i}\end{aligned} \tag{15}$$

where

$$F \triangleq I + \delta\Phi \qquad \text{and} \qquad u \triangleq [\delta I + \delta^2\Phi/2]c$$

and $\boldsymbol{z}_{k,i}$ are the individual measurements obtained by node k at time i, $\boldsymbol{n}_i$ accounts for modeling errors, and $\boldsymbol{v}_{k,i}$ is the measurement noise at node k. Model (15) can be formulated in the same form as 1 by defining the zero-mean quantities $\boldsymbol{x}_i = \boldsymbol{w}_i - \mathrm{E}\, w_i$ and $\boldsymbol{y}_{k,i} = \boldsymbol{z}_{k,i} - \mathrm{E}\, z_{k,i}$.

We assume that every node measures the position of the unknown object in either the two horizontal dimensions, or a combination of one horizontal dimension and the vertical dimension. Thus, individual nodes do not have direct measurements of the position in the three dimensions. The assignment of which pair is observable by every node is done at random, but taking care that for every neighborhood, there is at least one node of each type. Therefore, we have, $H_{k,i} = [0 \quad \mathrm{diag}([1\ 1\ 0])]$ for the case where only the horizontal dimensions are observed, or $H_{k,i} = [0 \quad \mathrm{diag}([1\ 0\ 1])]$ for the case where one horizontal dimension and the vertical dimension are observed.

In our experiment we use a network with $N = 10$ nodes, with topology shown in Fig. 2. The values of the parameters are $\delta = 0.1$, $G_i = I$,$Q_i = (0.001)I$ and $R_{k,i} = PR_0P^T$ with $R_0 = 0.5 \times \mathrm{diag}[1 \quad 4 \quad 7]$ and P being a permutation matrix, chosen at random for every node. The expected value of the initial state is $\mathrm{E}x_0 = [20\ 4\ 16\ 0.1\ 0.1\ 0.1]^T$ and its covariance $P_0 = I$. The results were averaged over 100 independent experiments over the same network topology.

We consider two simulation scenarios with different noise profiles. In the first scenario, the measurement noise covariance $R_{k,i}$ is the same for every node k. We also assume that this covariance matrix does not change with time. This situation is depicted in the bottom left plot of Fig. 2. In the second scenario, we have very dissimilar noise conditions

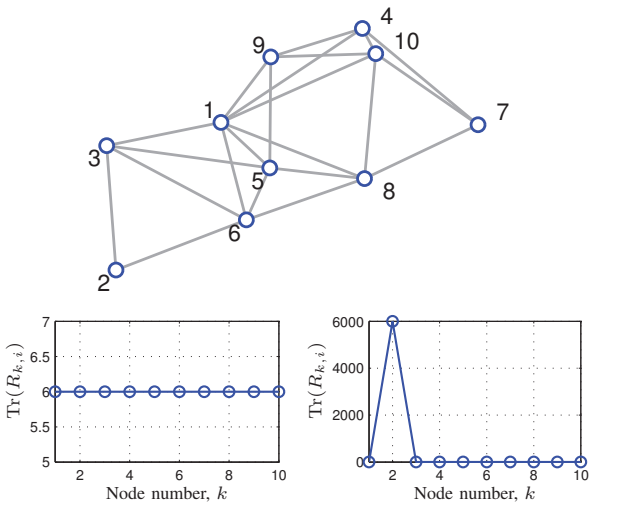

Fig. 2. Network topology (top), and trace of observation noise covariances for scenario 1 (bottom, left) and scenario 2 (bottom, right).

at different nodes. For instance, the trace of the covariance matrices at nodes 2, 3 and and 6 are 1000 times higher (30dB lower SNR) than the rest of the nodes. Again we assume these covariance matrices do not change with time. This scenario is depicted in the bottom right plot of Fig. 2.

For Alg. 1, we used the relative-degree rule to select matrix C, where

$$c_{lk} = \begin{cases}\alpha_k|\mathcal{N}_l| & \text{if } l \in \mathcal{N}_k\\ 0 & \text{otherwise}\end{cases}$$

and n_k denotes the degree of node k (i.e., the number of neighbors including itself), and α_k is a normalization parameter chosen such that $\mathbb{1}^*C = \mathbb{1}^*$. For Alg. 2, the initial weights were chosen uniform, i.e., equal for every neighbor, and the step-size was $\mu_{k,i} = 0.01$.

Fig. 3 shows the transient and steady-state performance for Scenario 1. The performance is measured in therms of the *network* Mean-Square Deviation (MSD), defined for node k at time i as:

$$\mathrm{MSD}_{k,i} = \mathrm{E}||\boldsymbol{x}_i - \hat{\boldsymbol{x}}_{k,i|i}||^2$$

The *network* MSD is defined as the average over all nodes.

$$\mathrm{MSD}_i^{\mathrm{network}} = \frac{1}{N}\sum_{k=1}^{N}\mathrm{MSD}_{k,i|i}$$

The top plot of Fig. 3 shows the network MSD as a function of time, for different algorithms, while the bottom plot shows the steady-state MSD at every node. The algorithm denoted "Local" allows nodes to exchange only measurements with their neighbors, but there is no diffusion process (i.e., $C = I$). Once a node has all the measurements from its neighborhood, it can run a conventional Kalman filtering update with this data. We also show the diffusion KF (Alg. 1), and the proposed diffusion KF with adaptive weights (Alg. 2). Also shown for comparison is the centralized case, where it is assumed that a fusion center has access to all measurements across the nodes,

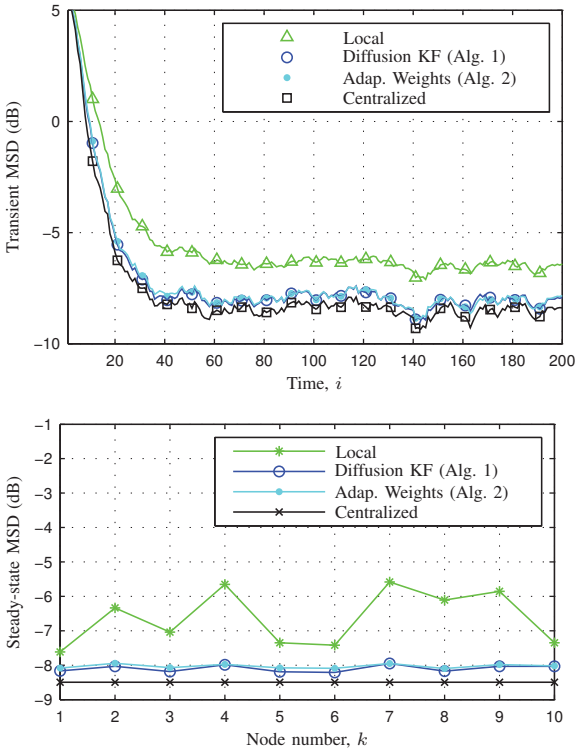

Fig. 3. Transient network MSD as a function of time (top) and steady-state MSD as a function of node number (bottom) for Scenario 1, where all nodes have similar measurement noise profiles.

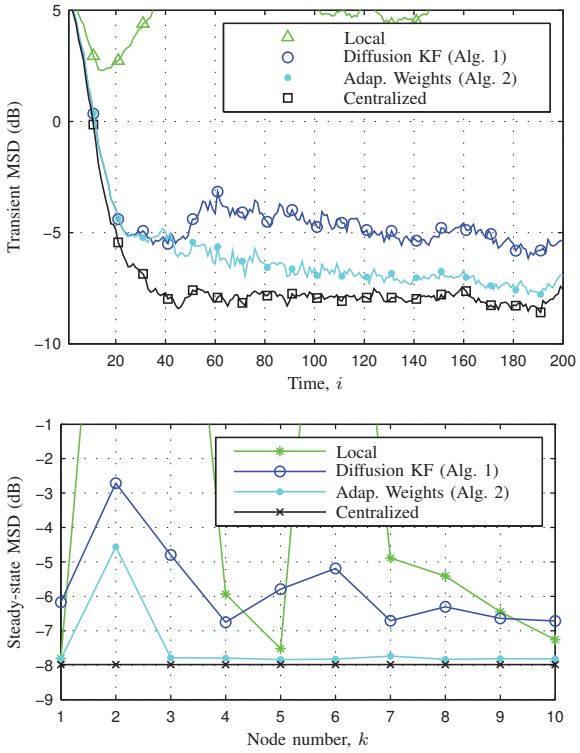

Fig. 4. Transient network MSD as a function of time (top) and steady-state MSD as a function of node number (bottom) for Scenario 2, where all nodes have very dissimilar measurement noise profiles.

and uses them to produce the optimal Kalman filter estimate. It can be observed from Fig. 3 that both the Diffusion KF (Alg. 1) and the proposed variant with adaptive weights (Alg. 2) have a very similar performance, which is also close to the global performance in this case.

We now resort to Scenario 2, where the measurement noise profiles are very different at different nodes. Again, in Fig. 3 we show the transient performance as a function of time (top) and the steady-state performance for each node (bottom). We now observe that the proposed diffusion KF with adaptive weights (Alg. 2) considerably outperforms the non-adaptive diffusion KF (Alg. 1), and obtains a performance close to the centralized solution. This behavior can be explained intuitively as follows. Recall that in Scenario 2, nodes 2, 3 and 6 have a 30dB lower SNR than the rest of the nodes. Thus, the local estimates produced by these nodes will be very noisy, as depicted by the green curve in the bottom plot of Fig. 3. The reason is that local estimates rely on local measurements, which are noisy in this case. Thus, in the diffusion step of Alg. 1, where these local estimates are combined in a convex manner, less weight should be given to these noisy local estimates, which is what our proposed Alg. 2 does. This behavior allows Alg. 2 to achieve increased performance.

V. Conclusions

We discussed optimal choices for the combination weights in diffusion Kalman filtering, and proposed an adaptive variant that can be computed in real time and in a distributed manner. The proposed algorithm has been observed to outperform existing techniques for the case of dissimilar noise profiles across nodes.

References

[1] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion strategies for distributed Kalman filtering: formulation and performance analysis," in *Proc. Cognitive Information Processing*, Santorini, Greece, June 2008.

[2] F. S. Cattivelli and A. H. Sayed, "Diffusion mechanisms for fixed-point distributed Kalman smoothing," in *Proc. EUSIPCO*, Lausanne, Switzerland, August 2008.

[3] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *Proc. 46th IEEE Conf. Decision and Control*, New Orleans, LA, December 2007.

[4] I. Schizas, S. I. Roumeliotis, G. B. Giannakis, and A. Ribeiro, "Anytime optimal distributed Kalman filtering and smoothing," in *Proc. IEEE Workshop on Statistical Signal Process.*, Madison, WI, August 2007, pp. 368–372.

[5] U. A. Khan and J. M. F. Moura, "Distributing the Kalman filter for large-scale systems," *IEEE Trans. on Signal Processing*, vol. 56, no. 10, Part 1, pp. 4919–4935, October 2008.

[6] N. Takahashi, I. Yamada, and A. H. Sayed, "Diffusion least-mean squares with adaptive combiners," in *Proc. ICASSP*, Taipei, Taiwan, April 2009.

[7] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*, Prentice Hall, NJ, 2000.

[8] A. H. Sayed, *Fundamentals of Adaptive Filtering*, Wiley, NJ, 2003.

[9] A. H. Sayed, *Adaptive Filters*, Wiley, NJ, 2008.