Distributed Kalman Filter Using Fast Polynomial Filter

A. Abdelgawad  
The Center for Advanced Computer Studies,  
University of Louisiana at Lafayette,  
Lafayette, LA 70504, USA  
ama1916@cacs.louisiana.edu

M. Bayoumi  
The Center for Advanced Computer Studies,  
University of Louisiana at Lafayette,  
Lafayette, LA 70504, USA  
mab@cacs.louisiana.edu

Abstract— Distributed estimation algorithms have received a lot of attention in the past few years, particularly in the fusion framework of Wireless Sensor Network (WSN). Distributed Kalman Filter (DKF) for WSN is one of the most fundamental distributed estimation algorithms for scalable wireless sensor fusion. In the literature, most of DKF methods rely on consensus filter algorithms. The convergence rate of such distributed consensus algorithms is slow and typically depends on the network topology and the weights given to the edges between neighboring sensors. In this paper, we propose a DKF based on polynomial filter to accelerate the distributed average consensus in the static network topologies. The main contribution of the proposed methodology is to apply a polynomial filter on the network matrix that will shape its spectrum in order to increase the convergence rate by minimizing its second largest eigenvalue. The simulation results show that the proposed algorithm increases the convergence rate of DKF by 4 times compared to the standard iteration. The proposed methodology can contribute in the real time WSN’s applications.

I. INTRODUCTION

The problem of decentralized Kalman filter was addressed by several researchers [1], [2]. Decentralized Kalman filter requires a complete network with all-to-all links. This solution is not scalable for large-scale sensor networks due to its $O(n^2)$ communication complexity ($n$ is the number of sensors/nodes). Whenever, Wireless Sensor Network has finite battery lifetime and thus limited computing and communication capabilities, so decentralized Kalman filter is not applicable in WSN. In particular, it is preferred to avoid decentralized Kalman filter which requires communication between all nodes. Thus, distributed Kalman filter is preferred where only communication with neighboring nodes is required.

Advantages with multisensory data fusion in wireless sensor networks typically include increased flexibility and more robustness, as more than one unit is performing the same task. In recent years, increasing emphasis has been placed on distributed fusion where several fusion nodes exist in the network. The advantages of such distributed fusion architecture are: higher robustness due to redundancy of fusion nodes and lower processing load at each fusion node. Consensus algorithms have proven to be effective tools for performing network-wide distributed computation tasks. Consensus filters allows the network to agree on the value of a particular computation. Consensus filters can be also used independently for DKF. The role of this consensus filter is to perform distributed fusion of sensor measurements that is necessary for implementation of a scalable Kalman filter. Consensus problems and their special cases have been the subject of intensive studies by several researchers [3], [4], [5]. Low-pass and high-pass consensus filters are also developed to calculate the average of their inputs in sensor networks [6], [7]. The convergence rate of such distributed consensus algorithms depends on the network topology and the weights given to the edges between neighboring sensors. In this paper, we applied a polynomial filter on the network matrix that will shape its spectrum in order to increase the convergence rate by minimizing its second largest eigenvalue.

II. NETWORK REPRESENTATIONS

In our simulation, we consider a static network, where the link’s state does not changes over the time. We assume the network at any arbitrary iteration $t$ as an undirected graph $G = \{V,E\}$ with the set of nodes $V = \{1,2,...,n\}$ and $E$ is the edge set at iteration $t$. $E \subseteq E'$, where $E' \subseteq V \times V$ is employed to show the interaction between the nodes in a network and it is exist if and only if sensor $i$ can communicate with sensor $j$. The neighbors of the node $i$ are denoted by the set $\{N_i = j \in V : (i, j) \in E\}$.

III. CONSENSUS FILTER

To reach a consensus on a graph, each sensor $i$ reports a scalar value $x_0(i) \in \mathbb{R}$. The vector of initial values on the network $x_0$ is denoted by:

$$x_0 = [x_0(1), x_0(2), ..., x_0(n)]^T \in \mathbb{R}^n$$ (1)
The average of the initial values of the sensors defined as:

\[ \text{Avg}_0 = \frac{1}{n} \sum_{i=1}^{n} x_0(i) \quad (2) \]

However, one rarely has a complete view of the network. The problem of distributed averaging therefore becomes typically to compute \( \text{Avg} \) at each sensor by distributed linear iterations. The distributed linear iterations of the network can be defined in the following form:

\[ x_{t+1}(i) = W_{ii}x_t(i) + \sum_{j \in N_i} W_{ij}x_t(j) \quad (3) \]

Where \( i = 1, \ldots, n \), and \( x_t(j) \) represents the value computed by sensor \( j \) at iteration \( t \). \( W_{ij} \) denotes the edge weights of \( G \). Since each sensor communicates only with its direct neighbors, setting \( W_{ij} = 0 \) for \( j \notin N_i \), the above iteration can be written in vector form as

\[ x_{t+1} = Wx_t \quad (4) \]

\( W \) is the weight matrix corresponding to the graph \( G \) of iteration \( t \). \( W \) is symmetric, so its eigenvalue as arranges as: \( 1 = \lambda_1 (W) \geq \lambda_2 (W) \ldots \geq \lambda_n (W) \). \( \lambda_2 (W) \), the second largest eigenvalue is a measure of performance/speed of consensus algorithm. The iterative relation given by eq. (4) can be written as:

\[ x_t = \left( \prod_{i=0}^{t-1} W \right) x_0 \quad (5) \]

The distributed linear iteration (4) converges to the average if and only if:

\[ \lim_{t \to \infty} W = \frac{11^T}{n} \quad (6) \]

\( \mathbf{1} \) is the vector of ones. From (4) and (6), we can find:

\[ \lim_{t \to \infty} x_t = \frac{11^T}{n} x_0 = \text{Avg}_0 \mathbf{1} \quad (7) \]

The convergence rate of (4) depends on the magnitude of the second largest eigenvalue \( \lambda_2 \). The asymptotic convergence factor is defined as:

\[ r_{\text{asym}} (W) = \rho \left( W - \frac{11^T}{n} \right) \quad (8) \]

\( \rho(\cdot) \) is the spectral radius of a matrix. Also the per-step convergence factor as follows:

\[ r_{\text{step}} (W) = \left\| W - \frac{11^T}{n} \right\|_2 \quad (9) \]

Since \( W \) is symmetric, the asymptotic convergence factor coincides with the per-step convergence factor, which implies that eq. (8) and eq. (9) are equivalent.

We used a polynomial filter on the spectrum of \( W \) in order to impact the magnitude of \( \lambda_2 (W) \) that mainly drives the convergence rate i.e. the smaller the value of \( \lambda_2 (W) \), the faster the convergence rate. Denote by \( p_m(\lambda) \) the polynomial filter of degree \( m \) that is applied on the spectrum of \( W \),

\[ p_m(W) = \alpha_0 I + \alpha_1 W + \ldots + \alpha_m W^m \quad (10) \]

In the implementation level, working on \( p_m(W) \) implies a periodic update of the current sensor’s value with a linear combination of its previous values. To see why this is true, we observe that:

\[ x_{t+m+1} = p_m(W)x_t = \alpha_0 x_t + \alpha_1 Wx_t + \ldots + \alpha_m W^m x_t \]

\[ = \alpha_0 x_t + \alpha_1 Wx_{t+1} + \ldots + \alpha_m W^m x_{t+m+1} \quad (11) \]

Each sensor typically applies polynomial filter for distributed consensus. The \( \alpha_m \)'s are computed off-line assuming that \( W \) and respectively \( E[W] \) are known a priori. The goal is finding the polynomial that leads to the fastest convergence of linear iteration described in eq. (4), for a given weight matrix \( W \) and a certain degree \( m \). The optimal polynomial is the one that minimizes the second largest eigenvalue of \( W \). In term of asymptotic convergence factor, we need to solve an optimization problem where the optimization variables are the \( m+1 \) polynomial coefficients \( \alpha_0, \alpha_1, \ldots, \alpha_m \) and the objective function is the spectral radius of \( W - \frac{11^T}{n} \). We need to solve the following optimization problem:

\[ \text{Minimize } \rho \left( \sum_{i=0}^{m} \alpha_i W^i \right), \text{ where } \alpha \in \mathbb{R}^{m+1} \]

\[ \text{Subject to } \left( \sum_{i=0}^{m} \alpha_i W^i \right) \mathbf{1} = \mathbf{1} \quad (12) \]

The Linear Matrix Inequality (LMI) for eq. (12) is equivalent to a set of \( m \) polynomial inequalities in \( W \), i.e., the leading principal minors of \( \alpha \) must be positive. To solve this optimization problem, we use an auxiliary variable \( f \) to bind the objective function, and then we express the spectral radius constraint as a linear matrix inequality. Thus, we need to solve the following optimization problem.

\[ \text{Minimize } f, \text{ where } f \in \mathbb{R}^{m+1} \]

\[ \text{Subject to } -fI \leq \sum_{i=0}^{m} \alpha_i W^i - \frac{11^T}{n} \leq fI, \quad \left( \sum_{i=0}^{m} \alpha_i W^i \right) \mathbf{1} = \mathbf{1} \quad (13) \]
Since $W$ is symmetric, $W = \sum_{i=0}^{m} \alpha_i W^i$ will be symmetric as well. Hence, the constraint $W 1 = 1$ is sufficient to ensure that $W$ will be also a left eigenvector of $W$.

The above optimization problem is equivalent to a semi-definite program (SDP). SDPs are in fact a special case of cone programming and can be efficiently solved by interior point methods. The solution is computed efficiently in practice, where the SDP only has a moderate number of $m+2$ unknowns.

IV. KALMAN FILTER

Kalman filters are based on linear dynamical systems discretized in the time domain. They are modeled on a Markov chain built on linear operators perturbed by Gaussian noise. The state of the system is represented as a vector of real numbers. At each discrete time increment, a linear operator is applied to the state to generate the new state, with some noise mixed in, and optionally some information from the controls on the system if they are known. Then, another linear operator mixed with more noise generates the visible outputs from the hidden state.

Let's consider a sensor network with $n$ sensors that are interconnected via an undirected graph. Let us describe the model of a process as such:

$$x_{k+1} = A_k x_k + B_k u_k + w_k \quad k \geq 0 \quad (14)$$

$$z_k = C_k x_k + v_k \quad k \geq 0 \quad (15)$$

Where $z_k \in \mathbb{R}^{np}$ represents the vector of $p$-dimensional measurements obtained via $n$ sensors, $w_k$ and $v_k$ are assumed to be $m \times l$ and $p \times l$ zero-mean white noise processes, respectively. The process $v_i$ is called measurement noise and $w_i$ is called process noise.

The above equations have several variables:
- $A$, $B$, $C$ are matrices
- $k$ is the time index
- $x$ is the state system
- $u$ is the input to the system
- $z$ is the measurement output
- $w$ and $v$ are process and measurement noise respectively.

Additionally, $x_0 \in \mathbb{R}^m$ is the zero-mean initial state of the process with covariance matrix $P_0$, and is assumed to be uncorrelated with $u_k$ and $v_k$.

Building upon the underlying dynamic system model, and introducing measurements $z_k = \{z_0, z_1, \ldots, z_k\}$, we define the information matrix to be the inverse of the state covariance matrix. We describe below the Kalman filter iterations in the information form.

$$M_k^{-1} = P_k^{-1} + C_k^* R_k^{-1} C_k \quad (16)$$

$$K_k = M_k C_k^* R_k^{-1} \quad (17)$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - C_k \hat{x}_{k|k-1}) \quad (18)$$

$$P_{k+1} = A_k M_k A_k^* + B_k Q_k B_k^* \quad (19)$$

$$\hat{x}_{k+1|k} = A_k \hat{x}_{k|k} + B_k u_k \quad (20)$$

Where $Q$ is the covariance of the process noise $w_k$ and $R$ is covariance of the measurement noise $v_k$.

V. DISTRIBUTED KALMAN FILTER

The Kalman filter is not only result in identical state equations, but also outperforms the central Kalman alternative in terms of computational cost. We will begin by rewriting our $z_k = C_k x_k + v_k$ sensing model equation, which again is essentially equating two $np \times l$ vectors. Recall that these vectors were stacked with the information obtained at each individual sensor. Intuitively so, in the distributed scenario we will consider each individual sensor one at a time, producing the following equation:

$$z_k(k) = C_k(k) + v_k(k) \quad (21)$$

Each sensor calculates the state estimate using a local micro-Kalman filter as following:

$$M_k^{-1} = P_k^{-1} + C_k^* R_k^{-1} C_k \quad (22)$$

$$K_k = M_k C_k^* R_k^{-1} \quad (23)$$

$$P_{k+1} = A_k M_k A_k^* + B_k Q_k B_k^* \quad (24)$$

The local estimate $\hat{x}_{k|k|l}^{\text{local}}$ is formed by the predicted regional estimate $\hat{x}_{k|k-1}^{\text{reg}}$ and the local measurement $z_k$.

$$\hat{x}_{k|k}^{\text{local}} = \hat{x}_{k|k-1}^{\text{reg}} + K_k (z_k - C_k \hat{x}_{k|k-1}^{\text{reg}}) \quad (25)$$

The sensor nodes exchange their estimates over the communication channel and combine the estimates in the neighboring nodes $N_i$.

$$\hat{x}_{k|k}^{\text{reg}} = \sum_{j \in N_i} W_{ij} \hat{x}_{k|k}^{\text{local}} \quad (26)$$

We calculate $W_{ij}$ according to the fast polynomial consensus in Sec. III. Then, each node predicts the regional estimate $\hat{x}_{k+1|k}^{\text{reg}}$ as follow:

$$\hat{x}_{k+1|k}^{\text{reg}} = A \hat{x}_{k|k}^{\text{reg}} + B_k u_k \quad (27)$$

The main advantage of the proposed methodology is to shape the network spectrum using polynomial filter in order to increase the convergence rate by minimizing its second largest
Moreover, the proposed methodology reduces the required communication bandwidth by exchanging estimates between the neighbors' nodes instead of exchanging measurements and covariance matrix.

VI. SIMULATION RESULTS

In this section we provide the simulation results of the proposed distributed Kalman filter. First, we compare the performance of SDP polynomial with the standard iterative method. We provide simulation results for different network sizes, where varies from 50 to 150 with step 10. In particular, we are trying to show how fast the proposed polynomial filter compared with the standard iteration for the distributed Kalman filter. We measure the average number of iterations needed by each method to reach the desired level of absolute error across different network sizes with a fixed tolerance $\delta = 10^{-3}$. This provides an estimate on the average convergence time achieved by each method over 50 random experiments. Figure 1 shows that the polynomial filtering method is faster than the standard iteration by 4 times in average. Moreover, the improvement of polynomial filter methods on the convergence rate is increased in larger networks.

![Figure 1. Convergence time for different network sizes](image)

We also provide the output performance of the proposed DKF versus the Central Kalman Filter (CKF). We consider a network of $n = 100$ sensors that are distributed randomly with a topology shown in Figure 2. The values used for the system defined in eq. (14) are:

$$
A = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix},
B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}
$$

In the simulations, a heterogeneous network is used. Literally, heterogeneous means different in some aspect. In our discussion of heterogeneity we refer to different kinds of nodes based on their measurement model defined in eq. (15) i.e. different $C$ matrix. In our simulations, half of the nodes have class $C_1$ while the other half has class $C_2$. The two different $C$ matrices used are:

$$
C_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix},
C_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}
$$

The Simulation time is 10 seconds with sampling time $T_s = 0.01$ second and initial value as below:

$$
X_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix},
P_0 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}
$$

The estimation obtained from a CKF, shown in figure 3, will be our reference to compare the proposed DKF performance. Figure 4 shows the squared estimation error for the proposed DKF at node 9 compared with the CKF squared error. Apparently, the proposed distributed and the central Kalman filters provide almost the same estimates and that is can be shown clearly in figure 5 which shows the average Mean Square Error (MSE) for all the nodes versus the MSE of the CKF.

![Figure 2. Network topology of $n = 100$ sensors](image)

![Figure 3. Estimation obtained through the CKF (xch) and the real signal (x)](image)
VII. CONCLUSIONS

We addressed the DKF problem by reducing it into a dynamic consensus problem in terms of weighted average estimates matrix that can be viewed as data fusion problem. We have presented a Distributed Kalman Filter based on polynomial filter to accelerate the distributed average consensus in the static network topologies. The proposed algorithm, performs closely to the central filter, and also reduces the filter complexity at each node, by reducing the dimension of the data. The simulation results show that the proposed algorithm increases the convergence rate of DKF by 4 times compared to the standard iteration.

REFERENCES


