Optimal Polynomial Filtering for Accelerating Distributed Consensus

Effrosyni Kokiopoulou, Pascal Frossard
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Laboratory - LTS4
CH - 1015 Lausanne, Switzerland
{effrosyni.kokiopoulou,pascal.frossard}@epfl.ch

Dimitra Gkorou
Dept. Comp. Eng. & Informatics
University of Patras,
26500 Patras, Greece.
gkorou@ceid.upatras.gr

***Abstract*—In the past few years, the problem of distributed consensus has received a lot of attention, particularly in the framework of ad hoc sensor networks. Most methods proposed in the literature attack this problem by distributed linear iterative algorithms, with asymptotic convergence of the consensus solution. It is known that the rate of convergence depends on the second largest eigenvalue of the weight matrix. In this paper, we propose the use of polynomial filtering in order to accelerate the convergence rate. The main idea of the proposed methodology is to apply a polynomial filter that will shape the spectrum of the weight matrix by minimizing its second largest eigenvalue and therefore increase the convergence rate. We formulate the computation of the optimal polynomial as a semi-definite program (SDP) that can be efficiently and globally solved. We provide simulation results that demonstrate the validity and effectiveness of the proposed scheme in both fixed and dynamic network topologies.**

I. Introduction and motivation

We consider the problem of accelerating distributed average consensus that has become recently particularly interesting in the context of ad hoc sensor networks. The distributed averaging problem has attracted a lot of research efforts in the field of wireless sensor networks (e.g., [1], [2], [3], [5] and references therein). It presents many applications for distributed estimation and for coordination of networks of autonomous agents.

Given the initial values at the sensors, the problem of distributed averaging is to compute their average *at each sensor* using distributed linear iterations. Each distributed iteration involves local communication among the sensors. In particular, each sensor updates its own local estimate of the average by a weighted linear combination of the corresponding estimates of its neighbors. The weights drive the importance of the measurements of its neighbors. This is equivalent to a matrix vector product of the network weight matrix W with the vector of the sensor values. If W satisfies some conditions, it can be shown that the average consensus solution can be reached by successive multiplications of W with the vector of initial sensor values. Furthermore, it can be shown [1] that the convergence rate depends on the second largest eigenvalue of W, $\lambda_2(W)$; that is, the smaller the $\lambda_2(W)$, the faster the convergence.

The main research direction so far focuses on the computation of the optimal weights that will yield the fastest convergence rate to the consensus solution [1], [2], [3], under the successive multiplications paradigm. In this work, we diverge from this paradigm and we allow the sensors to use their previous estimates in order to accelerate the convergence rate. The main idea is to use a matrix polynomial p applied on W, in order to shape its spectrum (see e.g. [7]). Given the fact that the convergence rate is driven by $\lambda_2(W)$, it is possible to impact on the convergence rate by careful design of the polynomial p. In the implementation viewpoint, working with $p(W)$ is equivalent to each sensor aggregating its value periodically using its previous estimates. Note that this is along the lines of our previous work [6], where we proposed the use of extrapolation methods for accelerating the convergence to the consensus solution. In particular, we have shown that the consensus solution is reached in a finite number of steps. Along the same lines, the authors in [8] view the distributed consensus problem as a linear dynamical system and they propose an approach which also converges in a finite number of steps. However, both methodologies assume that the network topology is fixed. On the contrary, the polynomial filtering approach that we propose does not make use of any such assumption, which makes it amenable to both fixed and dynamic network topologies. Moreover, filtering allows one to have more control on the convergence rate. We provide experimental results that verify the validity and the effectiveness of our method in both cases.

II. Background

We model the network as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V} = \{1, \ldots, n\}$ corresponding to sensors. An edge $(i, j) \in \mathcal{E}$ is drawn if and only if sensor i can communicate with sensor j. We denote the set of neighbors for node i as $\mathcal{N}_i = \{j |\ (i, j) \in \mathcal{E}\}$.

Initially, each sensor i reports a scalar value $x_0(i) \in \mathbb{R}$. We denote by $x_0 = [x_0(1), \ldots, x_0(n)]^\top \in \mathbb{R}^n$ the vector of initial values on the network. Denote by $\mu = \frac{1}{n}\sum_{i=1}^{n} x_0(i)$ the average of the initial values of the sensors. The problem of distributed averaging is to compute μ by distributed linear iterations *at each sensor*. In this work, we consider distributed linear iterations of the following form

$$x_{t+1}(i) = W_{ii}x_t(i) + \sum_{j \in \mathcal{N}_i} W_{ij}x_t(j), \qquad (1)$$

for $i = 1, \ldots, n$, where $x_t(j)$ represents the value computed by sensor j at iteration t. Since the sensors communicate in each iteration t, we assume that they are synchronized. The parameters W_{ij} denote the edge weights of $\mathcal{G}$ and $W_{ij} = 0$ when $(i, j) \notin \mathcal{E}$, which implies that each sensor communicates only with its direct neighbors. The above iteration can be compactly written in the following form

$$x_{t+1} = W x_t. \tag{2}$$

We call the matrix W that gathers the edge weights W_{ij}, as the weight matrix. We assume that W is symmetric, so we can arrange its (real) eigenvalues as follows:

$$\lambda_1(W) \geq \lambda_2(W) \geq \ldots \geq \lambda_{n-1}(W) \geq \lambda_n(W). \tag{3}$$

Denote also $W = Q \Lambda Q^\top$ its eigenvalue decomposition.

Note that the distributed linear iteration (2) converges to the average if and only if

$$\lim_{t \to \infty} W^t = \frac{\mathbf{1}\mathbf{1}^\top}{n}, \tag{4}$$

where $\mathbf{1}$ is the vector of ones. Indeed, notice that in this case $\lim_{t\to\infty} x_t = \lim_{t\to\infty} W^t x_0 = \frac{\mathbf{1}\mathbf{1}^\top}{n} x_0 = \mu \mathbf{1}$. It is well known that the convergence rate of (2) depends on the magnitude of the second largest eigenvalue $\lambda_2(W)$ [1]. We define the asymptotic convergence factor as

$$r_{\text{asym}}(W) = \sup_{x_0 \neq \mu \mathbf{1}} \lim_{t \to \infty} \left(\frac{\|x_t - \mu \mathbf{1}\|_2}{\|x_0 - \mu \mathbf{1}\|_2} \right)^{1/t}, \tag{5}$$

and the per-step convergence factor as follows

$$r_{\text{step}}(W) = \sup_{x_0 \neq \mu \mathbf{1}} \frac{\|x_{t+1} - \mu \mathbf{1}\|_2}{\|x_t - \mu \mathbf{1}\|_2}. \tag{6}$$

We provide a theorem from [1] that relates the spectrum of W to the convergence rate.

***Theorem** 1:* The equation (4) holds if and only if

$$\mathbf{1}^\top W = \mathbf{1}^\top \tag{7}$$

$$W \mathbf{1} = \mathbf{1} \tag{8}$$

$$\rho(W - \frac{\mathbf{1}\mathbf{1}^\top}{n}) < 1, \tag{9}$$

where $\rho(\cdot)$ denotes the spectral radius of a matrix. Furthermore,

$$r_{\text{asym}}(W) = \rho(W - \frac{\mathbf{1}\mathbf{1}^\top}{n}) \tag{10}$$

$$r_{\text{step}}(W) = \|W - \frac{\mathbf{1}\mathbf{1}^\top}{n}\|_2 \tag{11}$$

According to the above theorem, $\mathbf{1}$ is a left and right eigenvector of W associated with the eigenvalue one, and the magnitude of all other eigenvalues is strictly less than one. Also, the smaller the value of $\rho(W - \frac{\mathbf{1}\mathbf{1}^\top}{n})$ (or $\lambda_2(W)$), the faster the convergence rate. Note finally, that since W is symmetric, the asymptotic convergence factor coincides with the per-step convergence factor, which implies that the (10) and (11) are equivalent.

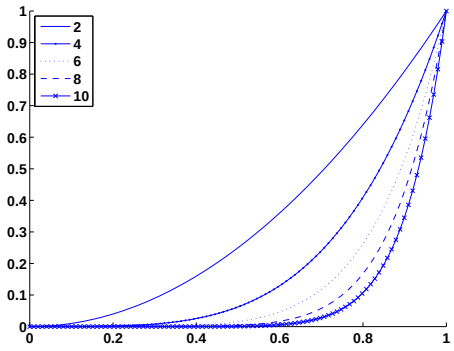


Fig. 1. Newton's polynomial of various degrees k.

III. Fast consensus with polynomial filtering

Motivated by Theorem 1, the authors in [1] compute the optimal weight matrix W that will bring the fastest convergence to the consensus solution, under the paradigm of successive multiplications. In this paper, we allow the sensors to use their previous iterates and we propose a methodology that can be applied on top of that and yield even faster convergence rates. Starting from a given (possibly optimal) weight matrix W, we propose the application of a polynomial filter on the spectrum of W in order to impact the magnitude of $\lambda_2(W)$. Denote by $p_k(\lambda)$ the polynomial filter of degree k that is applied on the spectrum of W,

$$p_k(\lambda) = \alpha_0 + \alpha_1 \lambda + \alpha_2 \lambda^2 + \ldots + \alpha_k \lambda^k. \tag{12}$$

Accordingly, the matrix polynomial is given as $p_k(W) = \alpha_0 I + \alpha_1 W + \alpha_2 W^2 + \ldots + \alpha_k W^k$. Observe now that

$$\begin{aligned} p_k(W) &= p_k(Q \Lambda Q^\top) \\ &= \alpha_0 I + \alpha_1 (Q \Lambda Q^\top) + \ldots + \alpha_k (Q \Lambda^k Q^\top) \\ &= Q p_k(\Lambda) Q^\top, \end{aligned} \tag{13}$$

which implies that the eigenvalues of $p_k(W)$ are simply the polynomial filtered eigenvalues of W i.e., $p_k(\lambda_i(W))$, $i = 1, \ldots, n$.

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

$$\begin{aligned} x_{t+k+1} &= p_k(W) x_t \\ &= \alpha_0 x_t + \alpha_1 W x_t + \ldots + \alpha_k W^k x_t \\ &= \alpha_0 x_t + \alpha_1 x_{t+1} + \ldots + \alpha_k x_{t+k}. \end{aligned} \tag{14}$$

Therefore, by the careful design of p_k one may impact the convergence rate dramatically. In what follows, we provide two different approaches for computing the coefficients of p_k.

A. Newton's interpolating polynomial

One possible approach for the design of the polynomial $p_k(\lambda)$ is to use Hermite interpolation. We assume that the

spectrum of W lies in an interval $[a, 1]$ and we impose smoothness constraints of p_k at the left endpoint a. In particular, the polynomial $p_k(\lambda) : [a, 1] \rightarrow \mathbb{R}$ that we seek, will be determined by the following constraints:

$$p_k(a) = 0,\ p_k^{(i)}(a) = 0,\ i = 1, \ldots, k-1 \quad (15)$$

$$p_k(1) = 1, \quad (16)$$

where $p_k^{(i)}(a)$ denotes the i-th derivative of $p_k(\lambda)$ evaluated at a. The computed polynomial will have degree equal to k. The coefficients of p_k that satisfies the above constraints can be computed by the Newton's divided differences table.

The intuition for designing such a polynomial is to dampen the smallest eigenvalues $\lambda_2, \ldots, \lambda_n$ of W (see (15)), while keeping the eigenvalue one intact (see (16)). Figure 1 shows the form of $p_k(\lambda)$ for $a = 0$ and different values of the degree k. The dampening is achieved by imposing smoothness constraints of the polynomial on the left endpoint of the interval. As k increases, the dampening of the small eigenvalues becomes more effective. The above polynomial design is mostly driven by intuitive arguments, which tend to obtain small eigenvalues for faster convergence. In the following section, we provide a technique for computing the optimal polynomial filter.

B. Optimal polynomial filtering using semi-definite programming

Algorithm 1 Polynomial Filtered Distributed Consensus

Input: polynomial coefficients $\alpha_0, \ldots, \alpha_{k+1}$, tolerance ϵ.
Output: average estimate $\bar{\mu}$.
Initialization:
  $\bar{\mu}_0 = x_0(i)$.
  Set the iteration index $t = 1$.
repeat
  if mod$(t, k+1)$ ==0 **then**
    $x_t(i) = \alpha_0 x_{t-k-1}(i) + \alpha_1 x_{t-k}(i) + \alpha_2 x_{t-k+1}(i) + \ldots + \alpha_k x_{t-1}(i)$ {polynomial filtered update}
    $x_t(i) = W_{ii} x_t(i) + \sum_{j \in \mathcal{N}_i} W_{ij} x_t(j)$.
  else
    $x_t(i) = W_{ii} x_{t-1}(i) + \sum_{j \in \mathcal{N}_i} W_{ij} x_{t-1}(j)$.
  end if
  Increase the iteration index $t = t + 1$.
  $\bar{\mu}_t = x_t(i)$.
until $\bar{\mu}_t - \bar{\mu}_{t-1} < \epsilon$

Let us consider now the following problem; given a weight matrix W and a certain degree k, what is the optimal polynomial that leads to the fastest convergence of linear iteration (2)? For notational ease, we call $W_p = p_k(W)$. According to Theorem 1, the optimal polynomial is the one that minimizes the second largest eigenvalue of W_p, i.e., $\rho(W_p - \frac{\mathbf{1}\mathbf{1}^\top}{n})$. Therefore, we need to solve an optimization problem where the optimization variables are the $k+1$ polynomial coefficients $\alpha_0, \ldots, \alpha_k$ and the objective function is the spectral radius of $W_p - \frac{\mathbf{1}\mathbf{1}^\top}{n}$.

We will use an auxiliary variable s to bound the objective function and then express the spectral radius constraint as a linear matrix inequality (LMI). Thus, we need to solve the following optimization problem.

Optimization problem: **OPT**
$\min_{s \in \mathbb{R}, \alpha \in \mathbb{R}^{k+1}} s$
subject to
$W_p = \alpha_0 I + \alpha_1 W + \alpha_2 W^2 + \ldots + \alpha_k W^k$,
$-sI \preceq W_p - \frac{\mathbf{1}\mathbf{1}^\top}{n} \preceq sI$,
$W_p \mathbf{1} = \mathbf{1}$.

Note that since W is symmetric, W_p will be symmetric as well. Hence, the condition $W_p \mathbf{1} = \mathbf{1}$ is sufficient to ensure that $\mathbf{1}$ will be also a left eigenvector of W_p. The spectral radius constraint,

$$-sI \preceq W_p - \frac{\mathbf{1}\mathbf{1}^\top}{n} \preceq sI$$

ensures that that all the eigenvalues of W_p other than the first one, are less or equal to s. Due to the LMI, the above optimization problem is a semi-definite program (SDP). SDPs are convex problems and can be efficiently and globally solved. In our case, the α_k's are computed off-line. Then, the i-th sensor applies polynomial filtering for distributed consensus, by implementing the Algorithm 1 (see also equations (13) and (14) for the equivalence of polynomial filtering and local update using the previous sensors' values).

IV. Experimental Results

In this section, we provide simulation results which show the effectiveness of the polynomial filtering methodology. First we introduce a weight matrix that has been extensively used in the distributed averaging literature. Suppose that $d(i)$ denotes the degree of the i-th sensor. It has been shown in [1], [2] that iterating with the Laplacian weight matrix defined below, leads to convergence to μ. Suppose that A is the adjacency matrix of $\mathcal{G}$ and D is a diagonal matrix which holds the vertex degrees. The Laplacian matrix is defined as $L = D - A$ and the Laplacian weight matrix is defined as

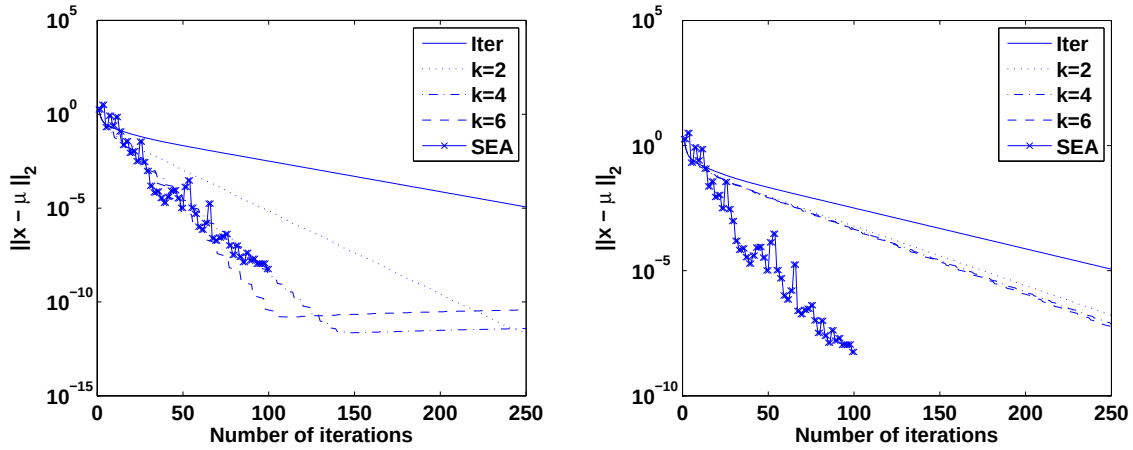
$$W = I - \gamma L, \quad (17)$$

where the scalar γ must satisfy $\gamma < 1/d_{\max}$.

The sensor networks are built using the random geographic graph model [9]. In particular, we place n nodes uniformly distributed on the 2-dimensional unit area. Two nodes are adjacent if their Euclidean distance is smaller than $r = \sqrt{\frac{\log(N)}{N}}$. It is known that this value of r guarantees connectedness of the graph with high probability [9].

Finally, the SDP programs for optimizing the polynomial filters are solved in Matlab using the SeDuMi [10] solver[1].

[1]Publically available at: `http://sedumi.mcmaster.ca/`



(a) SDP polynomial filtering

(b) Newton polynomial filtering

Fig. 2. Behavior of polynomial filtering for variable degree k on fixed topology using the Laplacian weight matrix.

A. Fixed network topology

We build a random network of $n = 50$ sensors and we use the Laplacian weight matrix for distributed averaging. We explore the behavior of the polynomial filtering methods under variable degree k from 2 to 6 with step 2. Fig. 2 shows the comparison of Newton's polynomial and the SDP optimal polynomial with the standard iterative method that is based on successive iterations of eq. (2). For the sake of completeness, we also provide the results of the scalar epsilon algorithm (SEA), which uses all previous estimates (see [6] for more details).

First, notice that polynomial filtering indeed accelerates the convergence of the standard method that is based on successive matrix-vector multiplications (solid line). Also, observe that the SDP polynomial outperforms Newton's polynomial, which is expected, since the former has been designed to optimize the convergence rate. Notice also, that the degree k governs the convergence rate, since larger k implies faster convergence. Note that the stagnation of the convergence process of the SDP polynomial filtering is due to the limited accuracy of the SDP solver.

Finally, we can see that the convergence rate is comparable for SEA and polynomial filtering. Note however that the former uses all previous values, in contrast to the latter that uses only the $k+1$ previous ones. Hence, the memory usage is smaller for polynomial filtering and also the process is smoother, which further permits easy extension to dynamic networks

B. Dynamic network topology

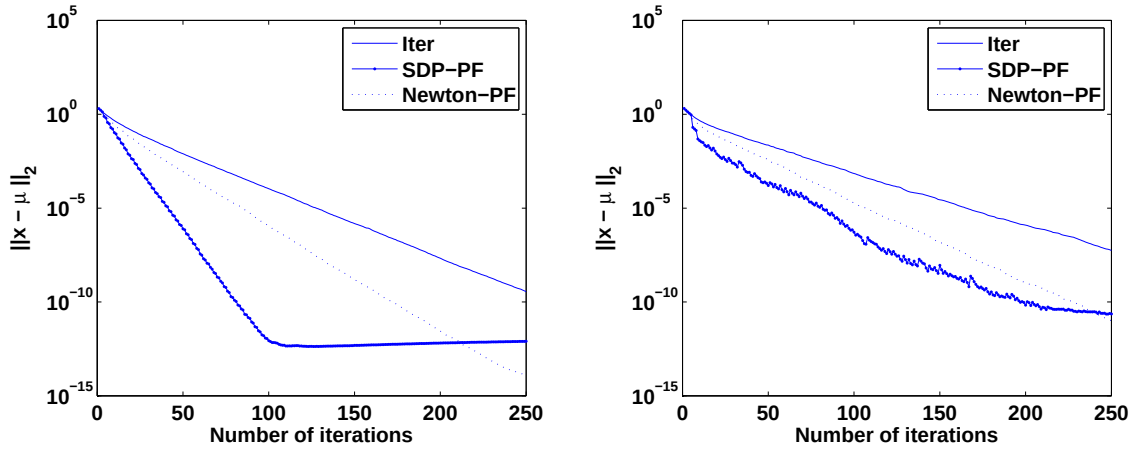
We model dynamic sensor networks using the model proposed in [4]. In particular, we assume that the network at any arbitrary iteration t is $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t))$, where $\mathcal{E}(t)$ denotes the edge set at iteration t. Since the network is dynamic, the edge set changes over the iterations. We assume that $\mathcal{E}(t) \subseteq \mathcal{E}^*$, where $\mathcal{E}^* \subseteq \mathcal{V} \times \mathcal{V}$ is the set of realizable edges. In other words, $\mathcal{E}(t) = \mathcal{E}^*$ if there is no link failure at iteration t. We also assume that each link fails independently of the other links according to a certain probability. Gathering all these probabilities together, we form a probability edge formation matrix P defined as follows

$$P_{ij} = \begin{cases} \text{prob of edge } (i,j) & \text{if } (i,j) \in \mathcal{E}^* \text{ and } i \neq j, \\ 0 & \text{otherwise.} \end{cases} \quad (18)$$

Thus, the edge set $\mathcal{E}(t)$ is a random subset of $\mathcal{E}^*$ according to the P matrix.

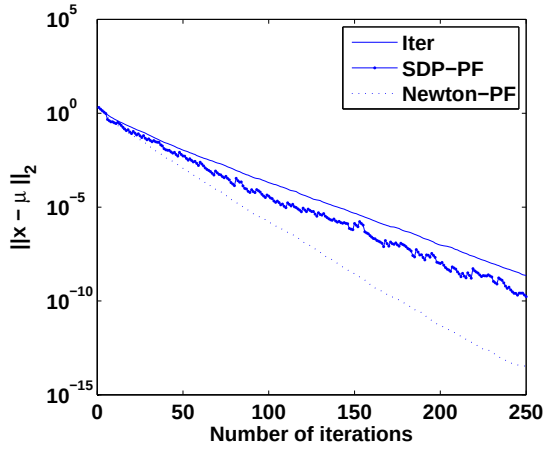
We build a sequence of random networks of $n = 50$ sensors and we assume that in each iteration the network changes with probability q, independently from the previous iterations. The authors in [4] show that the convergence rate depends on the second smallest eigenvalue of the average Laplacian matrix $\bar{L} = \bar{D} - \bar{A} = \bar{D} - P$. Hence, in the SDP formulation (see Sec III-B) we use $\bar{L}$ in order to form the weight matrix (17).

Fig. 3 shows the average performance of polynomial filtering using the median over 100 random experiments for some representative values of the degree k and the probability q. We do not report the performance of SEA, since it is not robust to changes of the network topology. Notice that when $k = 1$ (i.e., each sensor uses only its current value and the right previous one) polynomial filtering accelerates the convergence over the standard method. At the same time, it stays robust to network topology changes. Also, observe that in this case, the SDP polynomial outperforms Newton's polynomial. However, when $k = 2$, the roles between the two polynomial filtering methods change as the probability q increases. For instance, when $q = 0.8$, the SDP method even diverges. This is reasonable if we think that the coefficients of Newton's polynomial are computed using Hermite interpolation in a given interval. Thus, they are generic and they do not depend on the specific realization of underlying weight matrix.

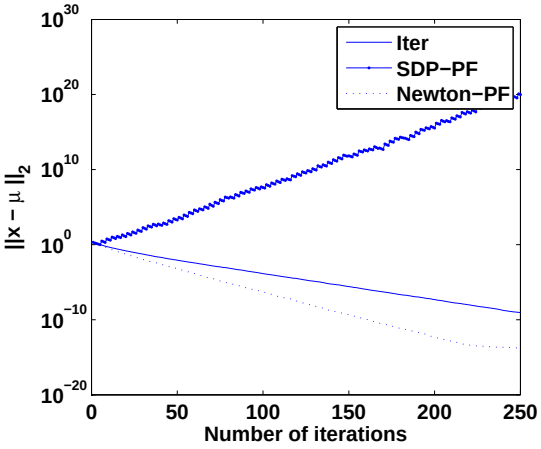


(a) $k = 1$, $q = 1$

(b) $k = 2$, $q = 0.1$



(c) $k = 2$, $q = 0.3$



(d) $k = 2$, $q = 0.8$

Fig. 3. Random network topology simulations. q denotes the probability that the network changes at each iteration.

V. Conclusions

In this paper, we proposed a polynomial filtering methodology in order to accelerate distributed average consensus. The main idea of polynomial filtering is to shape the spectrum of the polynomial weight matrix in order to minimize its second largest eigenvalue and subsequently increase the convergence rate. We proposed two different techniques to calculate the polynomial coefficients and the simulation results demonstrated the effectiveness of the proposed methodology.

References

[1] L. Xiao and S. Boyd, "Fast Linear Iterations for Distributed Averaging", *Systems and Control Letters*, February 2004.

[2] L. Xiao, S. Boyd and S. Lall, "A Scheme for Robust Distributed Sensor Fusion Based on Average Consensus", *Int. Conf. on Information Processing in Sensor Networks*, pp. 63-70, Los Angeles, April 2005.

[3] L. Xiao, S. Boyd and S. Lall, "Distributed Average Consensus with Time-Varying Metropolis Weights ", submitted to *Automatica*, June 2006.

[4] S. Kar and J. M. F. Moura, "Distributed Average Consensus in Sensor Networks with Random Link Failures", *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, April 2007.

[5] A. Olshevsky and J. Tsitsiklis, "Convergence Rates in Distributed Consensus and Averaging", *IEEE Conference on Decision and Control*, San Diego, CA, December 2006.

[6] E. Kokiopoulou and P. Frossard, "Accelarating Distributed Consensus Using Extrapolation", *IEEE Signal Processing Letters*, Vol. 14, Nr. 10, pp. 665-668, October 2007.

[7] E. Kokiopoulou and Y. Saad, "Polynomial filtering in Latent Semantic Indexing for Information Retrieval", *27th ACM-SIGIR Conference on Research and Development in Information Retrieval*, 2004.

[8] S. Sundaram and C. N. Hadjicostis, "Distributed Consensus and Linear Functional Calculation in Networks: An observability perspective", *6th Int. Conf. on Information Processing in Sensor Networks (IPSN)*, April 25-27, 2007.

[9] P. Gupta and P. R. Kumar, "The capacity of wireless networks", *IEEE Trans. on Information Theory*, 46(2):388-404, March 2000.

[10] J. F. Sturm, "Implementation of interior point methods for mixed semidefinite and second order cone optimization problems," *EconPapers 73*, August 2002, Tilburg University, Center for Economic Research.