Accelerating Distributed Consensus Using Extrapolation

Effrosyni Kokiopoulou and Pascal Frossard
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Institute (ITS)
CH- 1015 Lausanne, Switzerland
Phone: +41 21 693 5655
{effrosyni.kokiopoulou,pascal.frossard}@epfl.ch

***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. In this paper, we propose the use of extrapolation methods in order to accelerate distributed linear iterations. The extrapolation methods are guaranteed to converge in a finite number of steps, upper bounded by the number of sensors. In particular, we show that the Scalar Epsilon Algorithm (SEA) can accelerate vector sequences produced by distributed linear iterations, with no communication overhead and without knowledge of the full network topology. We provide simulation results that demonstrate the validity and effectiveness of the proposed scheme.**

***Index Terms*— Distributed Linear Iterations, Average Consensus, Extrapolation, Sensor Networks**

EDICS Category: COM-NETW.

I. Introduction and motivation

We consider the problem of accelerating distributed linear iterations that has become recently particularly interesting in the context of ad hoc sensor networks. 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. We consider distributed linear iterations of the form

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

for $i = 1, \ldots, n$, where $x_t(j)$ represents the value computed by sensor j at iteration t. 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. We assume that the term $z(i)$ is constant (e.g., it could be a bias term). The above iteration can be compactly written in the following form

$$x_{t+1} = W x_t + z. \qquad (2)$$

This work has been partly supported by the Swiss National Science Foundation, under grant NCCR IM2.

The sequence $\{x_t\}$ is defined as the vector sequence, and the matrix W that gathers the edge weights W_{ij} is called the sequence generator. Under the assumption that the system given in Eq. (2) has a fixed point $s \in \mathbb{R}^n$, we are interested in the problem of computing s with a small number of iterations, and low communication among sensors.

We propose to accelerate the convergence rate of the distributed linear problem given in Eq. (2) by extrapolation [2]. Note that the problem of accelerating vector sequences was well studied in the 60's and 70's and it has resulted in a series of effective extrapolation methods like the minimum polynomial extrapolation (MPE) [3] and the scalar epsilon algorithm (SEA) [1]. These methods do not require explicit knowledge of the sequence generator and they act directly on the terms of the vector sequence. They assume first that the sequence has a fixed point s and exploit the fact that s is expressed as a linear combination of a few consecutive terms of the sequence. Second, the matrix W is considered to be fixed, which corresponds to a static network topology (albeit unknown). We investigate the applicability of the extrapolation methods in the framework of distributed sensor networks and we show that SEA is the most attractive, since it requires no communication overhead at all. It is important to note that the proposed methodology stays generic and may be applied to any kind of distributed linear iteration of the form of Eq. (2).

Extrapolation methods represent a completely different line of thought, compared to state-of-the-art solutions for distributed consensus problems. For example, the particular distributed averaging problem has attracted a lot of research efforts recently (e.g., [5], [6], [7], [8] and references therein), as it presents many applications on distributed estimation and on coordination of networks of autonomous agents. The main research directions generally consists in computing the optimal weights of the matrix W that will yield the fastest convergence rate for the iterative problem given in Eq. (2) (under some constraints on W and $z = 0$). Extrapolation methods represent a novel solution to distributed linear iteration problems with faster convergence and low communication costs, and do not require sophisticated methods for the choice of efficient weights in consensus averaging. We demonstrate the benefits of extrapolation methods in the case of distributed averaging with simulations of random sensor networks.

II. Extrapolation of distributed linear iterations

A. Extrapolation methods

We propose to accelerate the linear iteration (2) by extrapolation (see [2] and references therein). There are several extrapolation methods in the literature such as MPE, SEA, reduced rank extrapolation (RRE) [4], and the vector epsilon algorithm (VEA) [1]. The extrapolation methods are generally based on first- and second-order differences between consecutive terms of the vector sequence. For notational convenience, let us define

$$u_t = \Delta x_t = x_{t+1} - x_t \tag{3}$$

as well as the induced matrix

$$U = U_t = [u_0, u_1, \ldots, u_{t-1}]. \tag{4}$$

Notice that

$$u_{t+1} = W u_t = W^{t+1} u_0. \tag{5}$$

The *minimal polynomial* p of a matrix $W \in \mathbb{R}^{n\times n}$ with respect to a vector y, is the monic[1] polynomial in W of smallest degree k, such that

$$p(W)y = \sum_{i=0}^{k} c_i W^i y = 0,$$

where the c_j's, the coefficients of p. The degree k of the minimal polynomial p is called the *grade* of y with respect to W [9, ch.6]. Denote by

$$p(\lambda) = \sum_{j=0}^{k} c_j \lambda^j, \ c_k = 1, \tag{6}$$

the minimal polynomial of W with respect to u_0. All extrapolation methods make use of the following theorem, which we reproduce here for convenience.

***Theorem** 1:* ([2]) For any $k+1$ consecutive terms of the sequence starting from m, say $x_m, x_{m+1}, \ldots, x_{m+k}$, it holds that

$$\sum_{j=0}^{k} c_j x_{m+j} = (\sum_{j=0}^{k} c_j) s. \tag{7}$$

According to Theorem 1, the fixed point of iteration (2) is a linear combination of any $k+1$ consecutive terms of the sequence. It means that the extrapolation methods can be applied in a straightforward manner to distributed linear iteration problems.

The polynomial methods (e.g., MPE and RRE) represent well-known solutions to calculate the coefficients c_j's and eventually the fixed point s, according to the above theorem. For instance, in MPE, the computation of c_j's is achieved through the solution of a small $k \times k$ linear system, involving the matrix U, which is row-distributed among the sensors. One approach to work around this problem is to flood U among the sensors, but this involves much more communication cost than flooding the initial sensor values themselves. Since polynomial methods require a large amount of communication between sensors, due to the solution of a linear system, they are not amenable to distributed linear iterations in ad hoc networks. Therefore, we propose to use epsilon algorithms for extrapolation, which are described in details in the rest of this section.

[1]When the leading (highest degree) coefficient is 1, the polynomial is called monic.

Algorithm: **Epsilon Algorithm**
Input: Scalar sequence $\{x_t\}_{t=0}^{T-1}$.
Output: Epsilon array.
1. Initialization: $\epsilon_{-1}^{(t)} = 0, \ \ \epsilon_0^{(t)} = x_t, \ \ t = 0, 1, 2, \ldots, T-1$
2.1. **for** $k = 0, 1, 2, \ldots$
2.2. **for** $t = 0, 1, 2, \ldots, T-k-2$
$\epsilon_{k+1}^{(t)} = \epsilon_{k-1}^{(t+1)} + [\epsilon_k^{(t+1)} - \epsilon_k^{(t)}]^{-1}$
2.3 **end**
2.4 **end**

TABLE I
The Epsilon Algorithm.

B. Epsilon algorithms

The epsilon algorithms are based on the recursive computation of a triangular array, which is called the *epsilon array*. Each entry in the array is computed by three earlier ones by very simple arithmetic, which is ideal for sensor networks. For scalar sequences ($n = 1$), the *e-transform* is defined as

$$e_k(x_t) = \frac{\det \begin{pmatrix} x_t & x_{t+1} & \cdots & x_{t+k} \\ \Delta x_t & \Delta x_{t+1} & \cdots & \Delta x_{t+k} \\ & & \ddots & \\ \cdots & \cdots & \cdots & \cdots \\ \Delta x_{t+k-1} & \Delta x_{t+k} & \cdots & \Delta x_{t+2k-1} \end{pmatrix}}{\det \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \Delta x_t & \Delta x_{t+1} & \cdots & \Delta x_{t+k} \\ & & \ddots & \\ \cdots & \cdots & \cdots & \cdots \\ \Delta x_{t+k-1} & \Delta x_{t+k} & \cdots & \Delta x_{t+2k-1} \end{pmatrix}}. \tag{8}$$

It is known that the e-transform gives the exact limit s (i.e., $e_k(x_t) = s$, when k is the degree of p) of any scalar sequence x_t whose errors satisfy

$$x_t - s = \sum_{i=1}^{k} \alpha_i \lambda_i^t \ , \tag{9}$$

where α_i and λ_i are fixed scalars. The same condition applies for vector sequences ($n > 1$), where the α_i's become fixed vectors. It can be shown (see [2, p.208] for more details) that linear iterations of the form of Eq. (2) satisfy the above condition.

Evaluating the e-transform by direct application of Eq. (8) would be of little practical use (even for scalar sequences) due to the computation of large determinants. However, P. Wynn [1] discovered "with a remarkable burst of insight" [2] that the ratio of the determinants can be evaluated recursively without explicitly evaluating them and without matrix inversion. The resulting algorithm is called the scalar *epsilon algorithm* (EA), and can be compactly written as the following recursion,

$$\epsilon_{-1}^{(t)} = 0, \ \ \epsilon_0^{(t)} = x_t, \ \ t \geq 0 \tag{10}$$

$$\epsilon_{k+1}^{(t)} = \epsilon_{k-1}^{(t+1)} + [\epsilon_k^{(t+1)} - \epsilon_k^{(t)}]^{-1}, \ k \geq 0, \ t \geq 0. \tag{11}$$

It is exactly the above recursion that made the epsilon algorithm a practical method for the acceleration of scalar

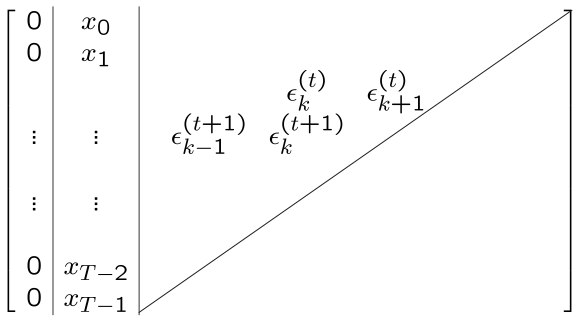


Fig. 1. The triangular form of the epsilon array.

sequences who satisfy the condition given in Eq. (9). The main steps of the epsilon algorithm are given in Table I, where T denotes the total number of available terms from the scalar sequence $\{x_t\}$. The notation $\epsilon_k^{(t)}$ denotes the entry of the epsilon array located in the t-th row and k-th column. Figure 1 depicts the triangular form of the epsilon-array and shows the entries that participate in the update of $\epsilon_{k+1}^{(t)}$, as well as their relative positions. Observe that during initialization the terms of the scalar sequence $x_0, \ldots, x_{T-1}$ are positioned in the first column of the array (see step 1 of the Algorithm).

Concerning the performance of the epsilon algorithm we have the following theorem.

***Theorem** 2:* (P. Wynn [1]) For each pair of nonnegative integers k and t, if the indicated quantities exist,

$$\epsilon_{2k}^{(t)} = e_k(x_t), \quad \epsilon_{2k+1}^{(t)} = 1/e_k(\Delta x_t). \tag{12}$$

In particular, the even numbered columns of the epsilon array evaluate the e-transform (8). Moreover, if $\{x_t\}$ satisfies (9) for some k, then it holds that $\epsilon_{2k}^{(t)} = s$, $\forall t$.

C. Epsilon algorithms for vector sequences

The epsilon algorithm presented above refers to scalar sequences x_t. Extending the algorithm to the case of vector sequences, calls for an appropriate interpretation of the inverse in Eq. (11). One approach is to interpret this inverse as the Samelson inverse

$$w^{-1} = w/\|w\|^2.$$

It yields the Vector Epsilon Algorithm (VEA) [1]. However, this version requires the knowledge of $\|w\|$ which again needs a small amount of communication among the sensors. A better approach is to consider the sequence of the i-th entries of the vector sequence, as an independent scalar sequence, so that the inverse in Eq. (11) can be interpreted as the reciprocal. This approach, when applied to vector sequences, is called the *scalar epsilon algorithm* (SEA) [1]. It is particularly interesting for sensor networks, since the i-th sensor's sequence terms are considered as an independent scalar sequence, which avoids any communication overhead. Interestingly, SEA works as good as the VEA algorithm in many cases [2], although it ignores the interdependencies among the different components of the vector sequence.

It is important to note that the parameter k does not need to be known a priori in practical cases, since the epsilon algorithm can be implemented incrementally. Adding a new term will add one more diagonal in the triangular epsilon array. Since k is the degree of the minimal polynomial of W, an upper bound of k is n, which is the total number of sensors in the network. Thus, the epsilon algorithm applied to scalar sequences is guaranteed to converge in a finite number of steps k, which is equal to n in the worst case. Since, SEA is a natural extension of it to the vector case, we expect a similar performance.

III. Distributed Averaging

A typical problem in distributed linear iterations is the distributed averaging problem, which is quite popular in ad hoc networks. Given the initial values x_0 of the sensors, the problem of distributed averaging amounts to computing in a distributed fashion, their average $\bar{x}_0$ in each sensor. The distributed averaging problem is a special case of the distributed linear iteration (2), where $z = 0$ and the matrix W satisfies

$$\lim_{t\to\infty} W^t = \frac{\mathbf{1}\mathbf{1}^\top}{n}. \tag{13}$$

Notice that in this case,

$$\lim_{t\to\infty} x_t = \lim_{t\to\infty} W^t x_0 = \bar{x}_0 \mathbf{1} = (\frac{1}{n}\sum_{i=1}^{n} x_0(i))\mathbf{1}. \tag{14}$$

In words, after convergence, the sensors share the mean value of their initial values x_0. To the best of our knowledge, all research efforts so far have focused on constructing the optimal weight matrix W, which will yield the fastest convergence of the linear iteration problem. For example, when $d(i)$ denotes the degree of the i-th sensor, it has been shown in [6] that iterating with the following matrices leads to convergence to $\bar{x}_0$.

- *Maximum-degree* weights. The maximum-degree weight matrix is

$$W_{ij} = \begin{cases} \frac{1}{n} & \text{if } (i,j) \in \mathcal{E}, \\ 1 - \frac{d(i)}{n} & i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

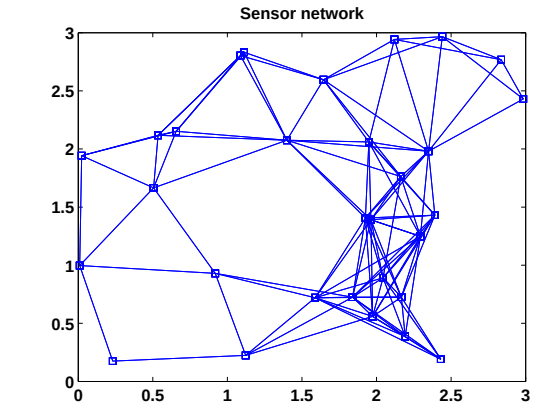
- *Metropolis* weights. The Metropolis weight matrix is

$$W_{ij} = \begin{cases} \frac{1}{1+\max\{d(i),d(j)\}} & \text{if } (i,j) \in \mathcal{E}, \\ 1 - \sum_{(i,k)\in\mathcal{E}} W_{ik} & i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

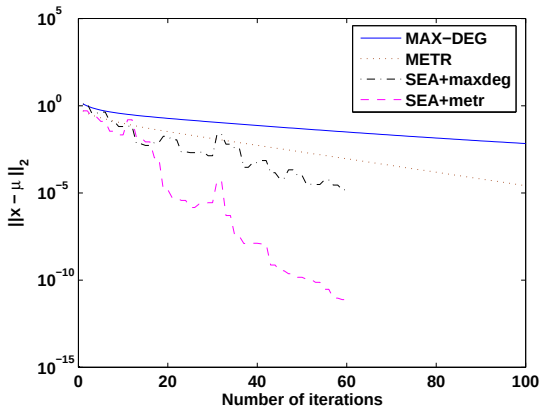
Alternatively, the work in [8] proposes three new algorithms for the distributed averaging problem where the dynamic topology algorithm has polynomial-time bound on the convergence time. The problem of determining the optimal matrix W is closely connected to the problem of finding the fastest mixing Markov chain on a graph, and can be formulated as a semidefinite program (SDP) [5].

We propose here a novel approach to the distributed averaging problem, and apply the SEA algorithm where each sensor i :

1) Computes the i-th entries of the vector sequence $x_0, \ldots, x_{T-1}$, by communicating only with its neighbors.



(a) Sensor network



(b) Convergence rate

Fig. 2. Simulation results on a sensor network with $n = 30$ sensors.

2) Applies the epsilon algorithm on its own scalar sequence, as described in Table I.
3) Picks the successive estimates of $\bar{x}_0$ in the leading entry of the even columns of the epsilon array.

Note that, at each step, the epsilon algorithm uses only three columns of the epsilon array. The memory requirements of the SEA algorithm are therefore limited to O(T), where T is the total number of available terms in the sequence. However, in the extreme case the storage requirement could be as high as O(n).

A. Simulation Results

We apply SEA algorithms to distributed averaging problems, and we compare its performance with state-of-the-art algorithms. We use a randomly generated sensor network with $n = 30$ nodes, uniformly distributed on the area $[0, 3] \times [0, 3]$ (see Figure 2(a)). For the construction of the graph, we consider two nearby nodes to be neighbors, if their Euclidean distance is smaller than r, where $r = 1$. The initial values $x_0(i)$ of the sensors are uniformly distributed in $[0, 1]$. We compare the proposed scheme to the convergence rate of the classical distributed averaging iterative method using the matrix W, with both max-degree weights (15) and Metropolis weights (16). We combine SEA with both max-degree and Metropolis weights. Figure 2(b) shows the simulated results, where "MAX-DEG" and "METR" denote the iterative method with max-degree and Metropolis weights respectively, and "SEA+maxdeg" and "SEA+metr" denote SEA with the corresponding weights. We test the SEA algorithms with increasing number of terms up to $2n$, since k cannot exceed n and therefore the size of the epsilon array should be at most $2n$ (see Theorem 2). We run the iterative methods up to a maximum number of iterations, which was set to 100. In the vertical axis, we plot the obtained error $\|x-\mu\|_2$, where x is the vector of the average estimations at the sensors and $\mu = \bar{x}_0\mathbf{1}$. SEA+maxdeg (dash-dot line) is to be compared to MAX-DEG (solid line) and SEA+metr (dashed line) is to be compared to METR (dotted). Observe that in SEA algorithms, the error does not decrease monotonically with respect to k. However, when k becomes equal to the grade of u_0 (and this may occur well before $k = n$), the approximation is quite good and superior to the one of state-of-the-art iterative methods. Thus, the SEA algorithms are more effective than the iterative methods and they converge in a small number of steps $k \ll n$.

IV. Conclusions

In this paper, we proposed the use of extrapolation techniques for accelerating distributed linear iterations that often appear in consensus problems in the context of ad hoc sensor networks. In particular, we proposed the use of the Scalar Epsilon Algorithm, which extrapolates the scalar sequence of each sensor, with no communication overhead. The simulation results demonstrated the effectiveness of the proposed scheme. We are currently investigating the extension of the methodology to the case of time-varying communication graphs.

V. Acknowledgements

The first author would like to thank Prof. Y. Saad for introducing her to the extrapolation methods while she was a graduate student at the University of Minnesota.

References

[1] P. Wynn, "Acceleration Techniques for Iterated Vector and Matrix Problems", *Math. Comp.*, vol. 16, pp. 301-322, 1962.
[2] D. Smith, W. Ford and A. Sidi, "Extrapolation Methods for Vector Sequences", *SIAM Review*, vol. 29(2), pp. 199-233, June 1987.
[3] S. Cabay and L.W. Jackson, "A Polynomial Extrapolation Method for Finding Limits and Antilimits of Vector Sequences", *SIAM J. Numer. Anal.*, vol. 13, pp. 734-752, 1976.
[4] R.P. Eddy, "Extrapolating to the limit of a vector sequence", *Information Linkage Between Applied Mathematics and Industry*, Academic Press, New York, pp. 387-396, 1979.
[5] L. Xiao and S. Boyd, "Fast Linear Iterations for Distributed Averaging", *Systems and Control Letters*, February 2004.
[6] 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.
[7] L. Xiao, S. Boyd and S. Lall, "Distributed Average Consensus with Time-Varying Metropolis Weights ", submitted to *Automatica*, June 2006.
[8] A. Olshevsky and J. Tsitsiklis, "Convergence Rates in Distributed Consensus and Averaging", *IEEE Conference on Decision and Control*, San Diego, CA, December 2006.
[9] Y. Saad, "Iterative methods for sparse linear systems", *SIAM*, 2nd edition, 2003.