Nonlinear Average Consensus

Leonidas Georgopoulos† and Martin Hasler†

†School of Computer and Communication Sciences, Ecole Polytechnique Federale de Lausanne
Lausanne, Switzerland
Email: leonidas.georgopoulos@epfl.ch, martin.hasler@epfl.ch

Abstract—We present an algorithm for nonlinear consensus in complex networks. Our motivation draws from analysis on the algorithm based on a weighted linear update protocol. Comparison of the asymptotic with early convergence rate encourages an alternative algorithm which exploits both stages.

1. Introduction

We consider a discrete time dynamical network composed of a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ and a first order linear or non-linear dynamical system associated with each vertex v_i on the vertex set $\mathcal{V}$. The dynamics should be assigned in such a manner that consensus is reached asymptotically. This means that the states $\mathbf{x}_i(t)$, $i = \{1, 2, 3 \dots n\}$ converge to one and the same value. In addition , we require that this value is the mean of the initial state. i.e. $\mathbf{x}_i(t) \underset{t\to\infty}{\longrightarrow} \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i(0),\ \forall i = \{1, 2, \dots n\}$

This problem has been studied in the literature [4],[6],[7]. In addition we desire that convergence takes place as fast as possible. For this purpose the interaction coefficients between the vertices, i.e. the weights on the edges can be optimized [1]. However, the optimization criteria commonly used is the asymptotic exponential speed of convergence. $r = \underset{t\to\infty}{lim} \left(\frac{\|\mathbf{x}(t)-\mathbf{x}^*\|}{\|\mathbf{x}(0)-\mathbf{x}^*\|}\right)^{1/t}$. We claim that this might not be the most pertinent criterion over all applications.

Our work is mainly targeted in wireless sensor networks. The graph is a geometrical graph where the vertices correspond to the devices and the edges to the wireless connections. The state at time zero represents a physical quantity of which the elements are noisy measurements from each sensor. Thus, the consensus algorithm determines after convergence the maximum likelihood estimate of this value , assuming the noise is white. In many applications, the estimate does not have to be extremely precise. Thus, the consensus algorithm is stopped long before convergence. Therefore, it may not be the best choice to maximize the asymptotic convergence rate but to examine specifically the initial transient dynamics.

To demonstrate the difference we have represented the time evolution of the mean standard deviation

$$E[\sigma(\mathbf{x}(t))] = E\left[\left(\frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i(t)-\mu)^2\right)^{1/2}\right]$$

from consensus as a function of time, using a linear algorithm, cf. equation 1, where the expectation value E is with respect to a random choice of the initial states according to the distribution $\mathcal{N}(0,1)$, in figure 1.

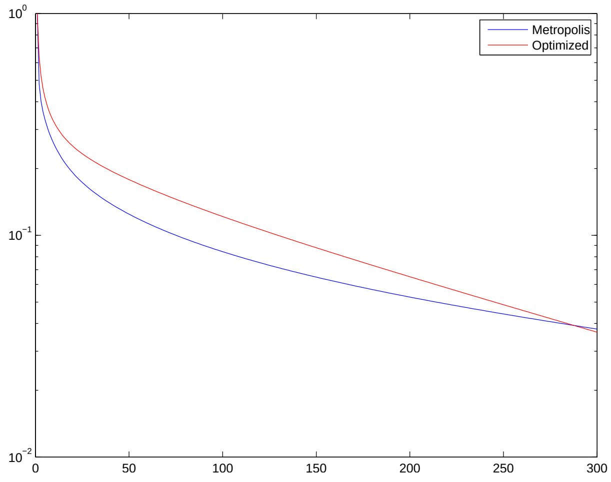

Figure 1: **Comparison of mean standard deviation** $E[\sigma(\mathbf{x}(t))]$. Red line: Using weights of convex optimization. Blue line: Weights using the Metropolis-Hastings rule.

The red line corresponds to the output of the algorithm when the weights on the edges where obtained after convex optimization with respect to the asymptotic speed of convergence. The blue line corresponds to the same graph but this time the edge weights have been assigned incorporating the Metropolis-Hastings rule [6], where d_i is the degree of vertex i.

$$[\mathbf{W}]_{ij} = \begin{cases} \frac{1}{max\{d_i,d_j\}} & i \neq j \\ 1 - \sum_{k=1}^{n}[\mathbf{W}]_{ik} & i = j \end{cases}$$

Clearly, asymptotically the optimized weights lead to faster convergence rate. However, the MH-weights have initially a faster decrease of the expected mean standard deviation from consensus. For this graph, up to precision 10^{-1} it is more advantageous to use the MH-weights than the optimal weights. In fact the MH-weights are also much easier to determine since the knowledge required of the graph is much less. We expect this effect to be more pronounced, the larger the graph is.

2. Linear Consensus

The linear consensus algorithm as presented in [3] consists of a simple vertex-local update equation.

$$\mathbf{x}_i(t+1) = \sum_j w_{ij}\mathbf{x}_j(t) \tag{1}$$

where $\mathbf{x}_i \in \mathbb{R}$ is the state variable defined on the vertex and $w_{ij} \in \mathbb{R}_+$ are weights on the edges of the graph. Specifically, $w_{ij} \neq 0$ when vertices v_i and v_j are connected by an edge on the graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ and $w_{ij} = 0$ otherwise. A global update equation can be readily retrieved in matrix form. Hence, equation (1) becomes,

$$\mathbf{x}(t+1) = \mathbf{W}\mathbf{x}(t) \tag{2}$$

where the state defined on the graph is denoted by $\mathbf{x} \in \mathbb{R}^n$ with n being the cardinality vertex set. The elements of matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ correspond to the weights on the edges of the graphs such as $[\mathbf{W}]_{ij} = w_{ij}$.

Suppose that matrix $\mathbf{W}$ satisfies the conditions below

$$\mathbf{W}^T = \mathbf{W}\,, \mathbf{W}\mathbf{1} = \mathbf{1}\,, \rho(\mathbf{W} - \mathbf{1}\mathbf{1}^T/n) < 1 \tag{3}$$

where $\mathbf{1} = (1,1,1,\dots,1)^T \in \mathbb{R}^n$ and $\rho(\cdot)$ denotes the spectral radius. Due to the presence of the eigenvector $\mathbf{1}$ the dynamical system has an infinity of fixed points of the form $\mathbf{x}^* = \alpha\mathbf{1}$ where $\alpha \in \mathbb{R}$. The convergence of the solutions of (2) to a fixed point is enforced due to condition $\rho(\mathbf{W} - \mathbf{1}\mathbf{1}^T/n) < 1$ and that fixed point under conditions in (3) is such that $\alpha = \mathbf{1}^T\mathbf{x}(0)/n$. The reader is directed to [7] for further details on the subject.

2.1. Asymptotic Phase

The dynamical system (2) will converge to the average of the initial state

$$\mathbf{x}_i(t) \underset{t\to\infty}{\longrightarrow} \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i(0),\ \forall\, i = \{1,2,\dots n\} \tag{4}$$

with exponential speed of convergence. The exponent is time-dependent and is determined by the initial state and the eigenvalues of $\mathbf{W}$. The eigenvalues satisfy.

$$1 = \lambda_1 > \lambda_2 \geq \lambda_3 \dots \geq \lambda_n > -1$$

Therefore the exponential speed of convergence is at least

$$|\log\min\{1-\lambda_2, 1+\lambda_n\}| \tag{5}$$

The latter is as well the asymptotic exponential speed of convergence as $t \to \infty$.

2.2. Transient Phase

Initially, the exponential speed of convergence is much higher than its asymptotic value. This is due to the fact that the components corresponding to the eigenvalues $1-\lambda_i$, where $i = \{3,4,\dots,n\}$, diminish rapidly. Therefore, in early stages of the process those eigenvalues contribute and essentially determine the speed of convergence. Their impact vanishes later on and finally, only the spectral gap, equation (5), is important as $t \to \infty$. This transient effect is rather evident in larger networks where the number of the eigenvalues is large. Therefore, the collapse of the components, in these networks, requires more iterations to complete.

Assume that the initial state of each vertex is sampled from a normal distribution $\mathbf{x}_i \sim \mathcal{N}(\alpha,\sigma^2)$, $\forall i = \{1,2,\dots n\}$. Therefore the initial state on the graph is sampled from a normal distribution $\mathbf{x} \sim \mathcal{N}(\alpha\mathbf{1},\sigma^2\mathbf{I})$. The weight matrix is a diagonalizable matrix according to $\mathbf{W} = \mathbf{S}\mathbf{D}\mathbf{S}^{-1}$. Since it is a symmetric matrix, its eigenvectors form an orthonormal basis $\mathbf{S}^T\mathbf{S} = \mathbf{I}$. Hence, the state $\mathbf{x}$ can be decomposed as $\mathbf{y} = \mathbf{S}^T\mathbf{x}$. Subsequently, the components also follow a normal distribution $\mathbf{y} \sim \mathcal{N}(\mathbf{S}^T\alpha\mathbf{1}, \mathbf{S}^T\sigma^2\mathbf{I}\mathbf{S})$, which leads to $\mathbf{y} \sim \mathcal{N}(\alpha\mathbf{e}_1, \sigma^2\mathbf{I})$, where $\mathbf{e}_1$ is the eigenvector corresponding to the 1 eigenvalue. It holds that $\|\mathbf{x}(t) - \mathbf{1}(\mathbf{x}(0)^T\mathbf{1})/n\|^2 = \|(\mathbf{W} - \mathbf{1}\mathbf{1}^T/n)^t\mathbf{x}(0)\|^2$. Let define the quantity $\Phi(t) = \frac{1}{n-1}\|(\mathbf{W} - \mathbf{1}\mathbf{1}^T/n)^t\mathbf{x}(0)\|^2$. Since $\mathbf{y}(t+1) = \mathbf{D}\mathbf{y}(t)$ then one can readily show that the update rule that follows from the subsequent application of map (2) onto some initial state $\mathbf{x}(0)$ is given by,

$$\Phi(t) = \frac{1}{n-1}\mathbf{y}(0)^T\tilde{\mathbf{D}}^{2t}\mathbf{y}(0) \tag{6}$$

where $\mathbf{y}(0) = \mathbf{S}^T\mathbf{x}(0)$ and $\tilde{\mathbf{D}} = diag\{0,\lambda_2,\lambda_3,\dots,\lambda_n\}$. Thereafter, the expectation may be obtained by $E[\Phi(t)] = \frac{1}{n-1}Tr[\tilde{\mathbf{D}}^{2t}]E[\mathbf{y}(0)^T\mathbf{y}(0)]$ which leads to

$$E[\Phi(t)] = \frac{1}{n-1}\sum_{i=2}^{n}\lambda_i^{2t}\sigma^2 \tag{7}$$

3. Nonlinear

Our intention is to have both a high exponential speed in the transient phase and in the asymptotic phase by using a nonlinear function. The trick is to leave the system with appropriate linear weights at the asymptotic phase while modulating them appropriately during the transient phase.

The proposed local update rule can be summarized in the equation below.

$$\mathbf{x}_i(t) \mapsto w_{ii}\mathbf{x}_i(t) + \sum_j w_{ij}f(u_{ij}(t)) \tag{8}$$

Where $u_{ij} = \mathbf{x}_j - \mathbf{x}_i$ and the nonlinear function in the summation can be any $\mathcal{C}^1$ continuous function which satisfies $f(0) = 0, f(-u) = f(u), \frac{df}{du} > 0$. This family of functions for consensus has been mentioned in [5] but is focused on the continuous time case. We provide the following theorem which suffices for the convergence of the non-linear dynamical system.

Theorem 1. *Suppose that* $\mathbf{W} \in \mathbb{R}^{n\times n}$ *is a doubly stochastic matrix. Let* f *be an odd, increasing scalar function* $f : \mathbb{R} \to \mathbb{R}$, *with a bounded first order derivative* $0 \leq \frac{df}{dx} \leq 1$. *Assume* $\rho(\mathbf{W} - \mathbf{1}\mathbf{1}^T/n) < 1$, *then the evolution of the discrete dynamical system* $\mathbf{x}(t+1) = \mathcal{A}(\mathbf{x}(t))\mathbf{x}(t)$ *converges according to:*

$$\mathbf{x}_i(t) \underset{t\to\infty}{\longrightarrow} \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i(0),\ \forall i = \{1, 2, \dots n\}$$

The proof follows.

Proof. Equation (8) can be written

$$\mathbf{x}_i(t+1) = w_{ii}\mathbf{x}_i(t) + \sum_{j\in\beta_i} w_{ij}\frac{f(u_{ij}(t))}{u_{ij}}u_{ij}(t)$$

as long as $\mathbf{x}_j \neq \mathbf{x}_i$.

Let define the matrix $[\mathcal{A}]_{ij} = w_{ij}\frac{f(\mathbf{u}_{ij})}{\mathbf{u}_{ij}}[\mathbf{A}]_{ij}$, $\forall i \neq j$ and $[\mathcal{A}]_{ii} = 1 - \sum_{j=1}^{n}[\mathcal{A}]_{ij}$, where $\mathbf{A}$ is the adjacency matrix. When $\mathbf{u}_{ij} = 0$ the matrix element is defined as $[\mathcal{A}]_{ij} = \lim_{\mathbf{u}_{ij}\to 0}\frac{f(\mathbf{u}_{ij})}{\mathbf{u}_{ij}}$.

The matrix $\mathcal{A}$ is a function of $f(u_{ij})$. Since u_{ij} is a function of $\mathbf{x}$ then $\mathcal{A}$ is a function of $\mathbf{x}$, as well. Subsequently, we are lead to the following global update equation.

$$\mathbf{x}(t+1) = \mathcal{A}(\mathbf{x}(t))\mathbf{x}(t) \tag{9}$$

The matrices $\mathcal{A}$ and $\mathbf{W}$ can be written $\mathcal{A}(\mathbf{x}) = \mathbf{I} - \mathcal{L}(\mathbf{x})$ and $\mathbf{W} = \mathbf{I} - \mathbf{L}$. Where $\mathbf{L}$ is the weighted graph laplacian, [2], and $\mathcal{L}(\mathbf{x})$ the corresponding weighted graph laplacian of the nonlinear system. The time index is omitted for simplicity where it is not necessary.

Let μ_k and λ_k are the eigenvalues of $\mathcal{L}(\mathbf{x})$ and $\mathbf{L}$, respectively. Therefore the eigenvalues of the two matrices, $\mathcal{A}$ and $\mathbf{A}$, are $1 - \mu_k$ and $1 - \lambda_k$, respectively. Due to the fact that these matrices are symmetric their eigenvalues are ordered as $\lambda_1 \leq \lambda_2 \leq, \dots, \lambda_n$ and $\mu_1 \leq \mu_2 \leq, \dots \mu_n$.

The eigenvalues of λ_k and μ_k are increasing functions of $[\mathbf{W}]_{ij}$, $[\mathcal{A}]_{ij}$ respectively. This follows from

$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \sum_{i=1}^{n}\sum_{j=1}^{n}[\mathbf{W}]_{ij}(x_j - x_i)^2$$

Hence $\mathbf{x}^T\mathbf{L}\mathbf{x}$ is a positive increasing function of w_{ij}. Since both $\mathbf{L}$ and $\mathcal{L}(\mathbf{x})$ are symmetric the Courant-Fischer theorem holds. This extends to $\mathcal{A}$ and therefore we need only the conditions on $\mathbf{W}$. □

A function that satisfies the conditions of theorem 1 and modulates the weights appropriately is

$$f(u) = tanh(\theta_1 u)\theta_2 \tag{10}$$

where parameters $\{\theta_1, \theta_2\} \in \mathbb{R}_+$ are assigned such that

$$\theta_1\theta_2 \leq 1 \tag{11}$$

We are going to verify in the next section by simulation that this function performs better when compared with the linear algorithm.

4. Validation

Our experiments have been focused on geometric graphs. These were mainly selected due to their relation with the application at hand, wireless sensor networks.

We have randomly generated 100 different connected graphs with fixed vertex set size, $|\mathcal{V}| = 100$, and varying edge set size $|\mathcal{E}|$. The weights on the edges of each graph were assigned using convex optimization. For each of these graphs we have generated 1000 different initial states sampled from a normal distribution $\mathcal{N}(0, 1)$, resulting in a dataset of 100000 different trials. For each of these initial state/graph combinations the linear algorithm and the non-linear algorithm have been simulated, equations (2) and (8) respectively, for 300 iterations. To accommodate the discussion here on, we refer to each of these simulations as epoch.

Two error measures are used to validate our approach. The first, is the mean value over all epochs of the standard deviation , $\sigma(\mathbf{x}^\eta(t))$, at iteration t, within an epoch η. The upper index $(\cdot)^\eta$ denotes the epoch. Suppose that the state follows a normal distribution over epochs then the variance follows a χ^2-distribution. Therefore, the expectation of the standard deviation should be the mean over the sample set. The measure can be summarized in the formula below,

$$E[\sigma(\mathbf{x}(t))] = \frac{1}{K-1}\sum_{\eta=1}^{K}\sigma(\mathbf{x}^\eta(t)) \tag{12}$$

where K is the total number of epochs. The latter allows to validate the precision of the algorithms for this specific graph-set. The time evolution of this error measure is plotted in figure 2.

Let us define the temporal mean the error measure, equation (12), over an entire epoch, $Q(\eta) = 1/T\sum_{t=1}^{T}E[\sigma(\mathbf{x}^\eta(t))]$, where T is the total number of iterations. Subsequently we may define the second measure as its mean over a class of epochs C where the number of edges on the graph is k. This provides a measure on the performance of an algorithm for a class of graphs in the graph-set.

$$\bar{Q}(k) = \sum_{\eta\in C} Q(\eta) \tag{13}$$

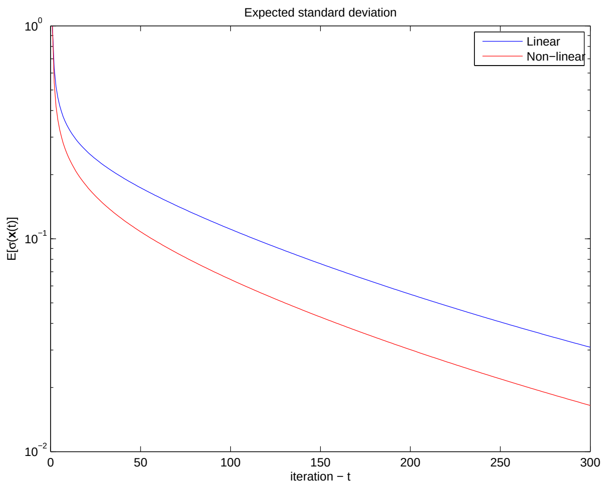

Figure 2: **Comparison of expected standard deviation $E[\sigma(\mathbf{x}(t))]$**, equation (12). The nonlinear algorithm performs better during the transient phases, not later than iteration 50, and then assimilates the asymptotic speed. This allows for an overall increase in precision in comparison to the linear. Blue line: Linear Algorithm, equation (2). Red line: Non-linear Algorithm, equation (8).

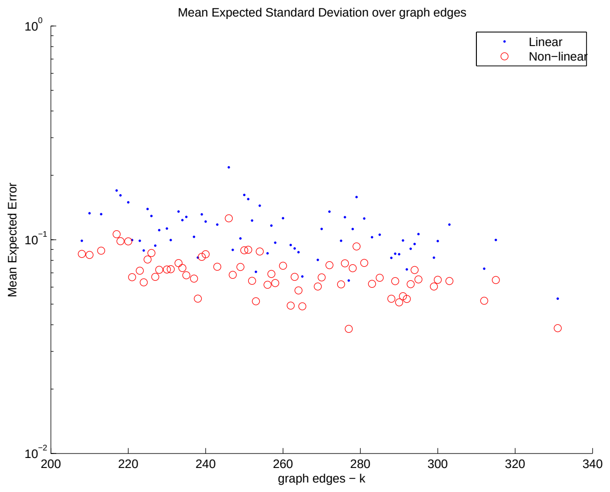

Figure 3: **Expected overall temporal precision** $E[Q(k)]$. The performance measure in equation (13) is plotted against the number of edges on the graph. The non-linear algorithm demonstrates better performance across the entire range. The non-linear algorithm benefits from the transient phase in comparison to the linear. Blue line: Linear Algorithm, equation (2). Red line: Non-linear Algorithm, equation (8).

This is plotted in figure 3 against the number of edges on the graph.

Our claim in section 3 that the non-linear algorithm benefits from both phases is verified in figure 2. Moreover, an examination of figure 3 leads to the conclusion that the improvement is uniform with respect to the number of edges on the graph. This encourages our postulation that the nonlinear algorithm has an overall better performance against the linear. This conclusion holds at least within the domain of graphs simulated in this section.

5. Conclusion

Our principal purpose has been twofolds herein. First, to establish the importance of the transient phase for the linear algorithm. Second, to demonstrate that an alternative approach with a non-linear function can benefit from the transient phase. The presented non-linear algorithm gains from both the transient and the asymptotic phase. This results in overall better convergence rate, at least for the class of graphs we have validated against.

Acknowledgments

This work has been funded by the WINSOC Project, a Specific Targeted Research Project (contract number 0033914) funded by the INFSO DG of the European Commission within the RTD activities of the Thematic Priority Information Society Technologies. 9

References

[1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[2] Chris Godsil and Gordon Royle. *Algebraic Graph Theory*, volume 207 of *Graduate Texts in Mathematics*. Springer, April 2001.

[3] Nancy A. Lynch. *Distributed Algorithms (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 1st edition.

[4] R. Olfati-Saber and R.M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *Automatic Control, IEEE Transactions on*, 49(9):1520–1533, Sept. 2004.

[5] R. O. Saber and R. M. Murray. Consensus protocols for networks of dynamic agents. In *American Control Conference, 2003. Proceedings of the 2003*, volume 2, pages 951–956, 2003.

[6] Lin Xiao and S. Boyd. Fast linear iterations for distributed averaging. volume 5, pages 4997–5002 Vol.5, Dec. 2003.

[7] Lin Xiao, Stephen Boyd, and Seung-Jean Kim. Distributed average consensus with least-mean-square deviation. *J. Parallel Distrib. Comput.*, 67(1):33–46, 2007.