Submitted, 2012 American Control Conference (ACC)
http://www.cds.caltech.edu/~murray/papers

Decentralised minimal-time dynamic consensus

Y. Yuan, J. Liu, R. M. Murray, and J. Gonçalves

***Abstract*— This paper considers a group of agents that aim to reach an agreement on individually measured time-varying signals by local communication. In contrast to static network averaging problem, the consensus we mean in this paper is reached in a dynamic sense. A discrete-time dynamic average consensus protocol can be designed to allow all the agents tracking the average of their reference inputs asymptotically. We propose a minimal-time dynamic consensus algorithm, which only utilises minimal number of local observations of randomly picked node in a network to compute the final consensus signal. Our results illustrate that with memory and computational ability, the running time of distributed averaging algorithms can be indeed improved dramatically using local information as suggested by Olshevsky and Tsitsiklis.**

I. Introduction

The central goal in multi-agent systems is to design a local control law in response to local information while ensuring that the desired global behaviour. Fuelled by applications in a variety of fields, [1], [2], [3], [4], [5], [6], [7], [8], [9], there has been a recent surge of interest in consensus dynamics. Broadly speaking, a consensus problem is one in which several spatially distributed agents or processors are seeking agreement upon a certain quantity of interest (for example, attitude, position, velocity, voltage, direction, temperature, and so on) but without recourse to a central coordinator or global communication. Well-known results [10], [11] give conditions to ensure that the state of each agent reaches the consensus value *asymptotically* using simple linear decentralised control law.

A large number of consensus problems studied so far are static, in the sense that the system is autonomous. However, the nature of decentralised control requires tight coordination among agents in a possibly dynamic environment. The dynamic average consensus problem arises in different contexts, such as formation control [12], distributed Kalman filtering [13] and load balancing [8]. These tasks require that all agents agree on the average of time-varying signals rather than the static average value [14], [15], [16], [17].

Similar to the static consensus problem, we are particularly interested in the following questions: first, what are the conditions on both the network and the signal such that the dynamic consensus is achieved? Second, how long does it take to reach such consensus? For the first question, some results are presented in [14], [15]. However, in all previous studies, the dynamic consensus signal is only reached asymptotically. This paper is motivated by the following

Ye Yuan and Jorge Gonçalves are with the Control Group, Department of Engineering, University of Cambridge. Jun Liu and Richard Murray are with Department of Control and Dynamical Systems, California Institute of Technology.

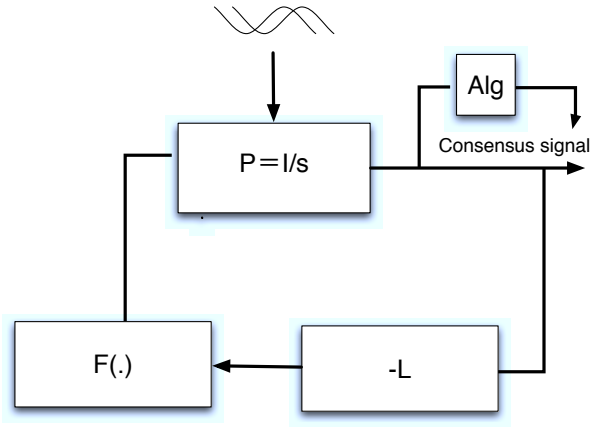


Fig. 1. Control diagram of dynamic consensus, the plant in consensus problem is usually a first order integrator. L is the Laplacian matrix defined later. F is the potential nonlinear dynamics however in this paper, we only consider linear dynamics and therefore $F = I$. In contrast to the existed result, we only use minimal number of successive observations of any node in the network and use a local algorithm to compute the final consensus signal.

observations: 1) there does not exist any criteria that each individual node can use to check whether the dynamic consensus is reached or not; 2) it takes arbitrarily long time to reach consensus; 3) the consensus is achieved at the expense of frequent communication [15].

Practically speaking, it is unsatisfactory to require an arbitrary long time to know the consensus value/signal, a lot of efforts have been made to increase the converge speed. For static consensus problem, Olshevsky and Tsitsiklis [18], [19] stated the fundamental limitation on the convergence speed of such consensus-type dynamics. In Zhang *et al* [20], model predictive control is used to speed up the consensus process, while a new protocol for finite-time consensus is designed in Cortes [21] and Wang and Xiao [22]. Sundaram and Hadjicostis [23] proposed an algorithm that computed the asymptotic final consensus value of the network in finite-time, Yuan *et al* [24], [25] proposed an algorithm for an arbitrarily chosen agent to compute the asymptotic final value of the network in a *minimal-time*. As shown in Fig. 2, for 6-node network, static consensus, it requires about 6^2 discrete-time steps to reach a small range of consensus value. By using the algorithm proposed in [25], it only uses 8 steps for node 1 to compute the consensus value merely using its own values.

One may then raise a similar question: how to improve the convergence time for dynamic consensus? This paper investigates the problem of computing the final consensus

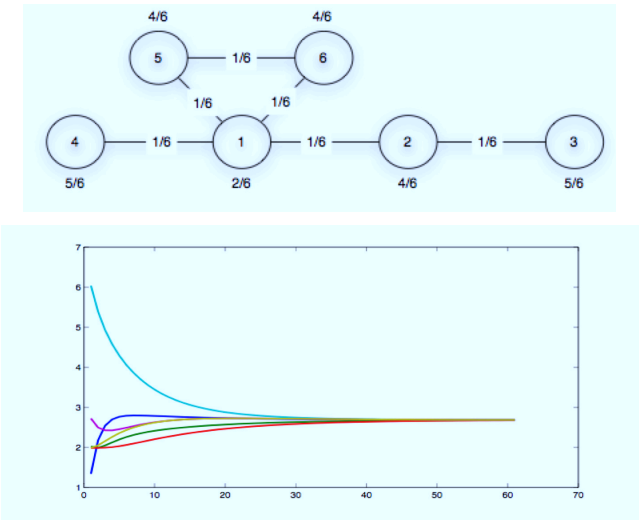


Fig. 2. Above: underlying topology with sampling time $\epsilon = 1/6$. Below: randomly stimulate the network with static input, and it requires approximately 40 steps to reach a small range (say 0.05) of consensus value.

signal of a random node (e.g., a sensor or a computer with some computational power) using only its own empirical values and in a finite and minimal amount of time.

The paper is organised as follows: after introducing background in Section II, main algorithm and results about the dynamic consensus are presented in Section III. Illustrative examples are provided in IV to verify the theoretical analysis.

II. Background

A. Notation

The notation in this paper is standard. For a matrix $A \in \mathbb{R}^{M\times N}$, $A[i,j] \in \mathbb{R}$ denotes the element in the i^{th} row and j^{th} column, $A[i,:] \in \mathbb{R}^{1\times N}$ denotes its i^{th} row, $A[:,j] \in \mathbb{R}^{M\times 1}$ denotes its j^{th} column and $A[i_1 : i_2, j_1 : j_2] \in \mathbb{R}^{(i_2-i_1+1)\times(j_2-j_1+1)}$ denotes the submatrix of A defined by the rows i_1 to i_2 and the columns j_1 to j_2. For a column vector $\alpha \in \mathbb{R}^{N\times 1}$, $\alpha[i]$ denotes its i^{th} element. Similarly for a row vector $\beta \in \mathbb{R}^{1\times N}$, $\beta[i]$ denotes its i^{th} element. We denote by $e_r^T = [0,\dots,0,1_{r^{th}},0,\dots,0] \in \mathbb{R}^{1\times N}$. I_N denotes the identity matrix of dimension N.

B. Static consensus

Consensus problem have been studied by many researchers in the literature. We consider here undirected graphs denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ (note that all results in this paper can be generalised to directed graphs), where $\mathcal{V} = \{\nu_1,\dots,\nu_n\}$ is the set of nodes, $\mathcal{E} \subset \mathcal{V}\times\mathcal{V}$ is the set of edges, and the adjacency matrix $W = W^T = \{W[i,j]\}_{i,j=1,\dots,n} \in \mathbb{N}_{\geq 0}^{n\times n}$, with $W[i,j] = 1$ when there is a link from i to j, and $W[i,j] = 0$ when there is no link from i to j. Let $x[i] \in \mathbb{R}$ denote the state of node i, which might represent a physical quantity such as attitude, position, temperature, voltage, etc. Considering the classical consensus protocol [11], the consensus dynamics of a network of continuous-time integrator agents is defined by:

$$\dot{x}(t) = -Lx(t), \tag{1}$$

where $L \in \mathbb{R}^{n\times n}$ is the Laplacian matrix induced by the topology $\mathcal{G}$ defined as $L[i,i] = \sum_{l\neq i}^{n} W[i,l]$, $\forall i = 1,\dots,n$ and $L[i,j] = -W[i,j]$, $\forall i \neq j$. Applying a time-discretisation scheme on (1) yields the following discrete-time network dynamics:

$$x_{k+1} = P_\epsilon x_k, \tag{2}$$

with $P_\epsilon = I_n - \epsilon L$, where ϵ is the sampling time. When ϵ is fixed, we consider the following update law:

$$\begin{aligned} x_{k+1} &= Ax_k, \\ y_k &= e_r^T x_k = x_k[r]. \end{aligned} \tag{3}$$

where $x_k \in \mathbb{R}^n$, $A \triangleq P_\epsilon$ and $y_k \in \mathbb{R}$ is the measurable output which corresponds to an arbitrarily chosen agent labelled r at discrete-time step k.

C. Dynamic consensus

Motivated by applications such as mobile networks and distributed Kalman filtering, we consider the scenario when the system has input signals instead of the standard static stimulus x_0. In this case, every node could receive a signal, e.g. ramp, sinusoid with possibly different amplitudes and frequencies. The goal of this consensus network is to reach a weighted average of these input signals for all nodes [14]. Hence, in contrast to the static consensus problem, the consensus is reached in a dynamic sense. More specifically, assume that each agent i has an associated signal $S[i]$ with value $s_k[i]$ at different discrete-time step k. We define the vector s_k, which contains the individual $s_k[i]$ as its components. Dynamic consensus can be viewed as a situation in which all agents asymptotically track the evolution of some aggregate network quantity. In addition to the input signals $S[i]$, each agent maintains a local variable $x_k[i]$, which is a time-varying estimate of the instantaneous average value for node i at time k.

Definition 1 (Asymptotic dynamic consensus): System (3) is said to asymptotically achieve dynamic consensus with input signal $S = [S[1],\ S[2],\ \cdots, S[n]]^T$ if for any $i,\ j$

$$\lim_{k\to\infty} \|x_k[i] - x_k[j]\| \to 0. \tag{4}$$

Under the consensus protocol that ensures tracking of the consensus signal (see next section), we focus on any chosen node r and compute the consensus signal using minimal number of its own observations. Note that additional assumptions on the input signals are needed to guarantee discrete-time consensus [15] since, for certain input signals, there may exist steady-state errors due to the poles introduced by the input signals. Fig. 1 in [14] shows a non-zero steady-state error for ramp inputs. We will then derive such conditions in the next section.

III. Decentralised minimal-time dynamic consensus computation algorithm

A. System model for dynamic consensus

Consider the discrete time LTI dynamics in eq. (3) where an arbitrarily chosen state $x[r]$ is observed. The decentralised problem is to compute the dynamic consensus signal of the network ϕ_k at time step k using only its own previously observed values $y_k = x_k[r]$. We consider the extended model of eq. (3) instead of the standard static consensus problem. The dynamic consensus will track the weighted average signal Φ asymptotically.

This section proposes the dynamic version of decentralised consensus computation algorithm, i.e. it computes the final consensus signal using minimal amount of successive discrete-time observations of any node, say node r, in the network. In particular, we assume that node r in the network does not have access to any other external information about the input signals and the network, such as, the type of signals, total number of agents n in the network (3), its local communication links or even the state values/number of its neighbours.

Similar to static consensus protocol, consider the decentralised protocol

$$\begin{aligned} x_{k+1} &= Ax_k + u_k, \\ y_k &= e_r^T x_k = x_k[r], \end{aligned} \tag{5}$$

where u_k is the unknown input/disturbance. In most cases, $u_k = s_k$; for certain applications, we let $u_k = s_k - s_{k-1}$ to guarantee the consensus of some signals with pole at 1, for example, ramp.

$$\begin{aligned} x_{k+1} &= Ax_k + s_k - s_{k-1}, \\ y_k &= e_r^T x_k = x_k[r]. \end{aligned} \tag{6}$$

Left multiplying a vector $\mathbf{1}^T$ to both sides of eq. (6) yields

$$\mathbf{1}^T(x_{k+1} - x_k) = \mathbf{1}^T(s_k - s_{k-1}),$$

since $\mathbf{1}^T A = \mathbf{1}^T$. It then follows $\mathbf{1}^T x_{k+1} = \mathbf{1}^T s_k$. Moreover, after some mathematical manipulation on eq. (6) and letting $e_{k+1} \triangleq x_{k+1} - \frac{1}{n}\mathbf{1}\mathbf{1}^T s_k$, we have

$$e_{k+1} = Ae_k + (I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)(s_k - s_{k-1}).$$

Take the Z-transform,

$$E(z) = (zI - A)^{-1}(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)(1 - z^{-1})S(z) \tag{7}$$

Let $E(z) \triangleq X(z) - \frac{\mathbf{1}\mathbf{1}^T}{n}Z(z)$ and assume the matrix A can be decomposed as

$$A = \sum_{i=1}^{n} \lambda_i(A) v_i v_i^T.$$

From the eigenvalue relation that $\lambda_i(A) = 1 - \epsilon\lambda_i(L)$ and the eigenvector corresponding to 0 is $\mathbf{1}$. Let $H(z) = (zI - A)^{-1}(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)$, and following a similar analysis as in [14], we know that the pole 1 of $(zI - A)^{-1}$ will be cancelled by multiplying $(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)$. Then we use final value theorem to find out the final error

$$e_\infty = \lim_{z \to 1}(z - 1)E(z). \tag{8}$$

$e_\infty = 0$ if $E(z)$ has at most one pole at 1.

This system is reaching consensus asymptotically under various conditions on sampling time, underlying topology which are developed in [15]. In the following part of this paper, we are assuming that these conditions for guaranteeing consensus are all satisfied. The goal of this paper is not to develop new theory or condition on consensus, but to focus on developing an algorithm that computes such dynamic consensus signal using minimal number of successive outputs.

Remark 1: The information used in the proposed algorithm was solely based on the accumulation of successive state values of the agent under consideration. No further information about the network and signal is used.

Remark 2: More importantly, so far to the authors' best knowledge there does not exist a criteria for any node in the network to check whether dynamic consensus is reached or not using merely its own observations. In other word, unlike static consensus problems, the node does not know whether the dynamic consensus is reached or not. Hence, the proposed algorithm also provides a purely decentralised way to check dynamic consensus.

For the purpose of main algorithm in Section III-B, we need to impose the following assumption for the input signals, we will later consider when the assumption does not hold in Section V.

Assumption 1: The Z-transform of such input signal at each node must have a finite number of poles.

Remark 3: In engineered systems, there is a set of signals, e.g., step, ramp, sinusoid, that are commonly used. They all satisfy the above assumption. Alternatively, one may think of the scenario that these signals are the estimations of the same linear process by different nodes. Again, the assumption is still satisfied.

Taking the Z-transform on both sides of eq. (6), $zX(z) = AX(z) + (1 - z^{-1})S(z)$ leads to

$$\begin{aligned} Y(z) = e_r^T X(z) &= e_r^T (zI - A)^{-1}(1 - z^{-1})S(z) \\ &\triangleq y_0 + y_1 z^{-1} + \cdots . \end{aligned}$$

Let $\Phi(z) \triangleq \phi_0 + \phi_1 z^{-1} + \cdots$, where $\phi_k = \frac{1}{n}\mathbf{1}^T x_k$. $Y(z)$ is usually different from the consensus signal, but it has the property for consensus in that

$$\lim_{K \to \infty} \|y_K - \phi_K\| \to 0 \tag{9}$$

B. Main algorithm for minimal-time dynamic consensus

From eq. (6) and [25], we have the following regression for the observations.

Proposition 1: Given a linear system (3) and an input signal vector Z satisfying assumption 1, there exist a $d \in \mathbb{N}$ and scalars $\alpha_0, ..., \alpha_d$ such that the following linear regression equation must be satisfied $\forall k \in \mathbb{N} \geq 0$,

$$y_{k+d+1} + \alpha_d y_{k+d} + \ldots + \alpha_1 y_{k+1} + \alpha_0 y_k = 0. \tag{10}$$

Proof: Taking the Z-transform on both sides of equation $zX(z) = AX(z) + (1 - z^{-1})S(z)$ leads to $Y(z) = e_r^T X(z) = e_r^T (zI - A)^{-1}(1 - z^{-1})S(z)$. By assuming that the number of poles of $S(z)$ is finite and noticing that $e_r^T(zI - A)^{-1}$ has finite poles [24], then the multiplication has finite poles and therefore it can be written in the form of eq. (10). ■

Remark 4: An algebraic characterisation of d for static consensus is given in [24] deriving from the Jordan block decomposition. In the dynamic case, d is a function of input signal S and linear matrix A.

Remark 5: If we can obtain the unknown coefficients in eq. (10) from data, then we can compute future outputs recursively from eq. (10) and past outputs. If we let $D_{r,s}+1$ be the length of the regression in eq. (10) then Proposition 1 also indicates that some scalars $\alpha_0, ..., \alpha_{D_{r,s}}$, the following equation always holds:

$$y_{k+D_{r,s}+1} + \alpha_{D_{r,s}} y_{k+D_{r,s}} + \ldots + \alpha_1 y_{k+1} + \alpha_0 y_k = 0. \quad (11)$$

To obtain such coefficients $\alpha_0, ..., \alpha_{D_{r,s}}$, we first introduce the definition of Hankel matrix which will be intensively used in the algorithm.

Definition 2: The Hankel matrix associated with a data set consisting of successive discrete-time values $Y_{0,1,\cdots,2k} \triangleq \{y_0 = x_0[r],\ y_1 = x_1[r], \ldots, y_{2k} = x_{2k}[r]\} (k \in \mathbb{Z})$ is defined as follows:

$$\Gamma(y_0,\ y_1,\ y_2,\ \cdots,\ y_{2k}) = \begin{bmatrix} y_0 & y_1 & y_2 & \cdots \\ y_1 & y_2 & y_3 & \cdots \\ y_2 & y_3 & \ddots & \\ \vdots & \vdots & & y_{2k} \end{bmatrix}, \quad (12)$$

with constant skew-diagonals.

A nice property of such Hankel matrix is the following Kronecker Theorem.

Theorem 1: [Kronecker Theorem][29] The Hankel matrix $\Gamma\{Y_{0,1,\ldots}\}$ has finite rank if and only if $f(z) \triangleq x_0[r] + x_1[r]/z + \cdots$ is a rational function with respect to z. What is more, the rank of the Hankel matrix $\Gamma\{Y_{0,1,\ldots}\}$ is equal to the number of poles of $f(z)$.

Remark 6: This Theorem links the the rank of a specially constructed matrix to the number of poles of a rational transfer function.

We now present the main procedure for reconstructing the coefficients. Without loss of generality, assume that the outputs start from discrete-time step 0. It is easy to remove these assumptions, e.g. see [24]. From Proposition 1 and Kronecker Theorem, when increasing the dimension of this Hankel matrix it will eventually lose rank. When it does, at discrete-time step $k = 2D_{r,s} + 2$, where $D_{r,s}$ is defined in eq. (10), compute its normalised kernel:

$$\Gamma(y_0,\ y_1,\ ,\cdots,\ y_{2D_{r,s}+2}) \begin{bmatrix} \alpha_0 & \alpha_1 & \ldots & \alpha_{D_{r,s}} & 1 \end{bmatrix}^T = 0. \quad (13)$$

It can be shown that the normalised kernel obtained from eq. (13) corresponds to the coefficients in eq. (11) [24].

Define $d(z) = z^{D_{r,s}+1} + \sum_{i=1}^{D_{r,s}} \alpha_i z^i$ and let

$$n(z) \triangleq d(z)(y_0 + y_1/z + y_2/z^2 + \cdots)$$

then by multiplication, we have $n(z) \triangleq \sum_{i=1}^{D_{r,s}} n_i z^i$.

From eq. (11), the coefficients from eq. (13) and past outputs we can predict future values of y_k, for all $k \geq 2D_{r,s} + 3$. In addition, the explicit expression of $Y(z) = e_r^T(zI - A)^{-1}(1 - z^{-1})S(z)$ can be obtained by taking the Z-transform of eq. (10).

To predict the future outputs of the observation at time K (it is actually the averaged signal under the assumption that the network reached consensus), we can use the expansion of a SISO transfer function and check the coefficient of z^{-K}.

We first define a reversion map $\mathcal{R}$ which takes z^{-m} to z^m, i.e., if $Y(z) = y_0 + y_1 z^{-1} + \cdots$, then $\mathcal{R}\{Y(z)\} = y_0 + y_1 z + \cdots$. It is not hard to show that

$$\mathcal{R}\left\{\frac{n(z)}{d(z)}\right\} = \frac{n(z^{-1})}{d(z^{-1})}$$

Then y_K can be computed by the following equality

$$y_K = \frac{1}{K!}\left\{\frac{d^K \mathcal{R}\left\{\frac{n(z)}{d(z)}\right\}}{dz^K}\right\}_{z=0}, \quad (14)$$

when $Y(z)$ has a complicated form to take the K^{th} derivative, we could first use partial expansion to decompose it to a summation of simple expressions, i.e., $Y = Y^1 + \cdots + Y^l$, where Y^is have less poles than Y. We can apply eq. (14) to Y^i to get y_K^i and then

$$y_K = y_K^1 + \cdots + y_K^l.$$

The whole algorithm can be written as follows:

Algorithm 1 Decentralised minimal-time dynamic consensus value computation with input signal constraints

Data: Successive observations of $y_i = x_i[r],\ i = 0, 1, \cdots$.
Result: Final consensus signal at time K: ϕ_K.
Step 1: Increase the dimension k of the square Hankel matrix $\Gamma\{Y_{0,1,\cdots,2k}\}$ until it loses rank and store the first defective Hankel matrix.
Step 2: The kernel $S = \begin{bmatrix} \alpha_0 & \ldots & \alpha_{D_{r,s}} & 1 \end{bmatrix}^T$ of the first defective Hankel matrix gives the coefficients of eq. (10).
Step 3: Compute $Y(z) = y_0 + y_1/z + \cdots$ and from then, we can compute $\phi_K \approx y_K$ (when K is large) using the above procedure in eq. (14).

Example 1: Let $Y_{0,1,2,3,\ldots} = (1,\ p,\ p^2,\ p^3,\ \cdots)$, then

$$\Gamma(Y_{0,1,\cdots,2k}) = \begin{bmatrix} 1 & p & \cdots \\ p & p^2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}.$$

We found that the Hankel matrix loses rank when $k = 1$. The normalised kernel of this Hankel matrix is $[-p,\ 1]^T$. Let $Y(z) = 1 + p/z + \cdots$ and $d(z) = z - p$, then $n(z) = Y(z)d(z) = z$, so $Y(z)$ writes

$$Y(z) = \frac{n(z)}{d(z)} = \frac{z}{z - p}.$$

For example, let $K = 3$, then

$$y_K = \frac{1}{K!}\left\{\frac{d^K \mathcal{R}\left\{\frac{n(z)}{d(z)}\right\}}{dz^K}\right\}_{z=0} = p^3,$$

which is consistent.

IV. Simulation

In this section, we use examples to illustrate the results stated in the previous sections.

Example 2: First, we consider a simple connected 4-node network in [14] in Fig. 3(a). We then stimulate the network with ramp inputs with different magnitude, as we can see that the consensus is reached in a dynamic and unstable sense in Fig. 3(c). Which is consistent to the consens analysis in Section III-A. Next we apply more complicated input signals and illustrate the above algorithm. Without loss of generality, we assume we can access the observations of node, i.e., $r = 1$. Let $x_0 = \begin{bmatrix}3.5784 & 2.7694 & -1.3499 & 3.0349\end{bmatrix}^T$ and $a_0 = \begin{bmatrix}1/2 & -1/2 & 1/2 & -1/2\end{bmatrix}^T$, then for any input signal i, $s_k[i] = kx_0[i]a_0[i]^k$. We stimulate the system with signals and use the local update protocol in eq. (6). The trajectory of every nodes are plotted in blue whereas the consensus signal is plotted in red in Fig. 3(d). By applying the algorithm listed above step by step, we can obtain the consensus signal in eq. (15) using 19 successive observations.

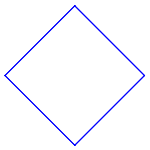
Example 3: We then consider a more complicated 10-node random directed network (small-world network) [5] with $\mathcal{G}^{sw}(n, 2d, p)$, where n is the number of nodes in the network, $2d$ is the degree of each node in the initial graph, and p is the probability of rewiring an edge. We choose here $n = 10, d = 1, p = 0.1$, see Fig. 3(b). For input signals chosen as $z_k[i] = x_0[i] * k^{\frac{1}{5}}$, with randomly chosen x_0, we can then obtain the consensus signal using 23 successive observations.

V. Assumption relaxation

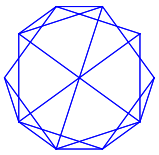
In this section, we will consider the case that when Assumption 1 does not hold. It could be the case that $u_k[r]$ in eq. (5) is noise or complicated signal. The major difference is that the following Hankel matrix will not lose rank at any finite dimension since some of the input signal does not have finite poles. Let $Y_{0,1,\cdots,2k} \triangleq \{y_0 = x_0[r],\ y_1 = x_1[r], \ldots, y_{2k} = x_{2k}[r]\}(k \in \mathbb{Z})$,

$$\Gamma(Y_{0,1,\cdots,2k}) = \begin{bmatrix} y_0 & y_1 & y_2 & \cdots \\ y_1 & y_2 & y_3 & \cdots \\ y_2 & y_3 & \ddots & \\ \vdots & \vdots & & y_{2k} \end{bmatrix}.$$

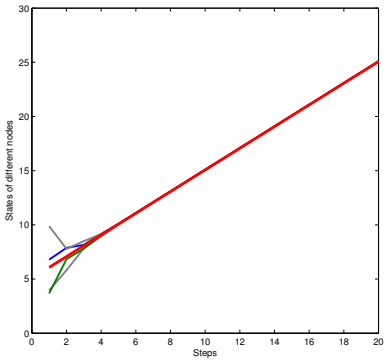
In this case, the idea is to find a Hankel matrix $\Gamma(\hat{Y}_{0,1,\cdots,2k})$ to approximate (close enough in some measure) $\Gamma(Y_{0,1,\cdots,2k})$. $\Gamma(\hat{Y}_{0,1,\cdots,2k})$ has finite rank and therefore can be used to estimate the final consensus signal.



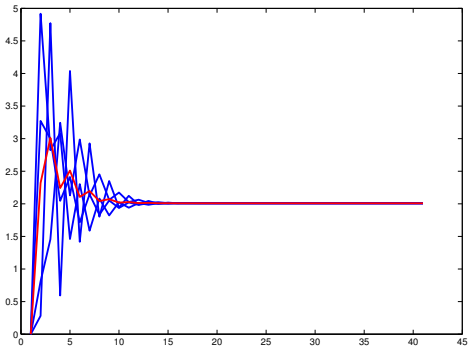
(a) 4-node ring.



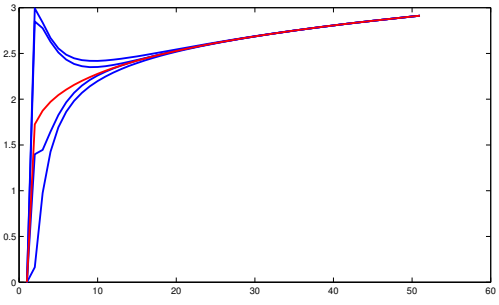
(b) 10-node small-world network.



(c) Trajectory of each node in Example 2, the red one is the consensus signal.



(d) Trajectory of each node in Example 2, the red one is the consensus signal.



(e) Trajectory of each node in Example 3, whereas the red one is the consensus signal.

Fig. 3. Networks and node trajectories in Example 2 and Example 3 with 4-node ring and 10-node small world configuration respectively.

Step 1: We start to increase the dimension k of the square Hankel matrix $\Gamma\{Y_{0,1,\cdots,2k}\}$ until it loses rank and store the first defective Hankel matrix.
Step 2: We found that when $k = 9$, the Hankel matrix loses its rank and then we can compute its kernel

$$S = \begin{bmatrix} 0 & 0.0069 & -0.0069 & -0.1181 & 0.1181 & 0.6111 & -0.6111 & -1.0000 & 1.0000 \end{bmatrix}^T.$$

Step 3: Compute $Y(z) = y_0 + y_1/z + \cdots$ and from then, we can compute y_K using eq. (14) and finally approximate the final consensus signal at large time step K, i.e., ϕ_K as y_K.

$$Y(z) = \frac{-2.952z^6 + 1.257z^5 + 1.647z^4 + 0.2067z^3 - 0.7575z^2 - 0.01911z + 0.01492}{z^7 - z^6 - 0.6111z^5 + 0.6111z^4 + 0.1181z^3 - 0.1181z^2 - 0.006944z + 0.006944}. \tag{15}$$

$$\Gamma(\hat{Y}_{0,1,\cdots,2k}) = \text{argmin}\|\Gamma(Y_{0,1,\cdots,2k}) - \Gamma(\hat{Y}_{0,1,\cdots,2k})\|, \tag{16}$$
$$\text{s.t.: } \det\Gamma(\hat{Y}_{0,1,\cdots,2k}) = 0, \Gamma(\hat{Y}_{0,1,\cdots,2k}) \text{ is Hankel}$$

here $\|\cdot\|$ can be any norm, from the fact that

$$\mathbb{E}\{(\Gamma(Y_{0,1,\cdots,2k}) - \Gamma(\hat{Y}_{0,1,\cdots,2k}))^T(\Gamma(Y_{0,1,\cdots,2k}) - \Gamma(\hat{Y}_{0,1,\cdots,2k}))\}$$
$$= \Gamma(Y_{0,1,\cdots,2k})^T\Gamma(Y_{0,1,\cdots,2k}) + \text{Noise Cov. Matrix},$$

where $\mathbb{E}\{\cdot\}$ is the expected value, this means that 2-norm can be a good candidate measure for solving the problem. To solve the above problem, we resort the following lemma.

Lemma 1: [30] Let $x \in \mathbb{R}^n$, then there exists a Hankel matrix $D \in \mathbb{R}^{n\times n}$, such that

$$Dx = x \text{ and } \|D\|_2 \le 1.$$

Proposition 2: [31] Assume that the Hankel matrix $\Gamma(Y_{0,1,\cdots,2k})$ has full rank, then

$$\min \|\Gamma(Y_{0,1,\cdots,2k}) - \boldsymbol{H}(k,k)\|_2 = \underline{\sigma}(\Gamma(Y_{0,1,\cdots,2k})) \tag{17}$$
$$\text{s.t.: } \det \boldsymbol{H}(k,k) = 0, \boldsymbol{H}(k,k) \text{ is Hankel.}$$

where $\boldsymbol{H}(k,k)$ can be obtained by the following Algorithm 2.

Proof: Before referring to the algorithm, we first define hvec operator mapping from square Hankel matrix $\mathbb{R}^{n\times n}$ to a vector $\mathbb{R}^{(2n+1)\times 1}$. For example, $\text{hvec}(\Gamma(Y_{0,1,\cdots,2k})) = \begin{bmatrix} y_0 & y_1 & \cdots & y_{2k} \end{bmatrix}^T$. We now propose the algorithm for computing the nearest defective Hankel matrix with respect to $\Gamma(Y_{0,1,\cdots,2k})$. From Algorithm 2, we can see that $\boldsymbol{H}(k,k)$ satisfies the constraints in the optimisation (16), because

1. by construction, $\boldsymbol{H}(k,k)$ is Hankel;
2. it is easy to verify that the constructed Hankel matrix D satisfying $D\underline{v}(\Gamma(Y_{0,1,\cdots,2k}) = \underline{v}(\Gamma(Y_{0,1,\cdots,2k}))$, then

$$\begin{aligned}\boldsymbol{H}(k,k)\underline{v}(\Gamma(Y_{0,1,\cdots,2k})) &= \Gamma(Y_{0,1,\cdots,2k})\underline{v}(\Gamma(Y_{0,1,\cdots,2k})) \\ &\quad - \underline{\sigma}(\Gamma(Y_{0,1,\cdots,2k}))D\underline{v}(\Gamma(Y_{0,1,\cdots,2k})) \\ &= 0.\end{aligned}$$

   As a consequence, $\boldsymbol{H}(k,k)$ does not have full rank;
3. since $\boldsymbol{H}(k,k) - \Gamma(Y_{0,1,\cdots,2k}) = -\underline{\sigma}(\Gamma(Y_{0,1,\cdots,2k}))D$ and $\|D\|_2 \le 1$, then

$$\|\boldsymbol{H}(k,k) - \Gamma(Y_{0,1,\cdots,2k})\|_2 \le \underline{\sigma}(\Gamma(Y_{0,1,\cdots,2k})).$$

Therefore, we can choose $\Gamma(\hat{Y}_{0,1,\cdots,2k}) = \boldsymbol{H}(k,k)$ as the solution of optimisation (16). ■

Algorithm 2 Computing the nearest defective Hankel matrix

Step 1: Form the observations as a square Hankel matrix, take a singular value decomposition of $\Gamma(Y_{0,1,\cdots,2k})$ and find the smallest singular value $\underline{\sigma}(\Gamma(Y_{0,1,\cdots,2k}))$ and corresponding singular vector $\underline{v}(\Gamma(Y_{0,1,\cdots,2k}))$;
Step 2: Compute the Hankel vector

$$\text{hvec}(D) = C_x^+ C_x^T e_1,$$

where C_x^+ is the Moore-Pensore pseudoinverse of C_x, $e_1 = [1,\ 0,\ \ldots,\ 0]^T$ has length of $2n-1$ and

$$C_x = \begin{bmatrix} v[1], & \ldots, & v[n] & & \\ & \ddots & & \ddots & \\ & & v[1] & \ldots & v[n] \\ v[n] & & v[1] & \ldots & v[n-1] \\ \vdots & \ddots & & \ddots & \vdots \\ v[2] & \ldots & v[n] & & v[1] \end{bmatrix}.$$

Step 3: Let $\boldsymbol{H}(k,k) = \Gamma(Y_{0,1,\cdots,2k}) - \underline{\sigma}(\Gamma(Y_{0,1,\cdots,2k}))D$.

Remark 7: $\underline{\sigma}(\Gamma(Y_{0,1,\cdots,2k}))$ quantifies how good the approximation is, more specifically, if it is large, then we probably need more observations to increase the dimension of the Hankel matrix to have a better approximation.

From above analysis, we have the following procedures (see Algorithm 3). We can apply this receding-horizon algorithm at any even discrete-time step $2k$ (for the purpose

Algorithm 3 Decentralised minimal-time dynamic consensus value computation without input signal constraints

Data: Successive observations of $y_i = x_i[r],\ i = 0, 1, \cdots$.
Result: Final consensus signal at time K: ϕ_K.
Step 1: At each time step k starting form 0, we take the singular value decomposition of $\Gamma(Y_{0,1,\cdots,2k}) = U\Sigma V^T$, where $\Sigma = \text{diag}\{\sigma_1,\ \sigma_2, \ldots, \sigma_{k+1}\}$ with $\sigma_1 \ge \sigma_2 \ldots \ge \sigma_{k+1} = \underline{\sigma}(\Gamma(Y_{0,1,\cdots,2k}))$;
Step 2: If $\underline{\sigma}(\Gamma(Y_{0,1,\cdots,2k})) \le \epsilon$, where $\epsilon \in \mathbb{R}^+$ is a prescribed value, otherwise, we increase k. $\underline{v}(\Gamma(Y_{0,1,\cdots,2k}))$ satisfies $\underline{\sigma}^2\underline{v}(\Gamma(Y_{0,1,\cdots,2k})) = (\Gamma(Y_{0,1,\cdots,2k})^T\Gamma(Y_{0,1,\cdots,2k}))\underline{v}(\Gamma(Y_{0,1,\cdots,2k}))$, we can then apply Algorithm 2 to obtain $\Gamma(\hat{Y}_{0,1,\cdots,2k})$;
Step 3: After obtaining $\Gamma(\hat{Y}_{0,1,\cdots,2k})$, we can follow Algorithm 1 to compute future estimated output $\hat{y}_K$ and approximate the consensus signal as $\phi_K \approx \hat{y}_K$.

of square Hankel matrix), and get an estimation of future outputs. Ongoing research lies in a more detailed analysis of Algorithm 3.

Remark 8: Algorithm 1 can be viewed as a special case of Algorithm 3 if we let $\epsilon = 0$.

VI. Conclusion and future work

Under the assumption that the dynamic consensus of a group of agents is reached, this paper proposes algorithms to compute the asymptotical consensus signal using the minimal number of observations of an arbitrarily chosen node in a network.

Ongoing research is extending the model (3) so as to encompass more complex situations, e.g., time-delay in the model, noise/quantisation error in the communication links, packet drop in the observations. Broadly speaking, towards improving the performance of existed results in decentralised estimation and fusion in a network, the next step is to embed the above algorithm to improve the performance in the design of distributed Kalman consensus filter [13], [32], [33].

VII. Acknowledgement

Ye Yuan acknowledges the support of Microsoft Research through the PhD Scholarship Program. Jorge Gonçalves was supported in part by EPSRC grant numbers EP/G066477/1 and EP/I029753/1. Ye wants to thank Prof. Herbert Tanner (U. Delaware), Dr. Minghui Zhu (UCSD), Prof. Alexander Megreski (MIT) and researchers from CDS, Caltech for useful discussions and comments.

References

[1] H. G. Tanner, A. Jadbabaie and G. J. Pappas, "Stable flocking of mobile agents, part I: Fixed topology," in Proceedings of IEEE Conference of Decision and Control, 2003.

[2] F. Cucker and S. Smale, "Emergent behavior in flocks," IEEE Transactions on Automatic Control, 2007.

[3] D. P. Bertsekas and J. N. Tsitsiklis, "Parallel and distributed computation: numerical methods," Englewood Cliffs, NJ: Prentice-Hall, 1989.

[4] R. Olfati-Saber, "Ultrafast consensus in small-world networks," in Proceedings of American Control Conference, 2005.

[5] D. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," Nature, 1998.

[6] L. Pecora and M. Barahona, "Synchronization of oscillators in complex networks," Chaos and Complexity Letters, 2005.

[7] M. Barahona and L. Pecora, "Synchronization in small-world systems," Physical Review Letters, 2002.

[8] G. Cybenko, "Dynamic load balancing for distributed memory multiprocessors," Journal of Parallel and Distributed Computing, 1989.

[9] Y. Wang and F. Doyle, "On influences of global and local cues on the rate of synchronization of oscillator networks," To appear, Automatica, 2011.

[10] A. Jadbabaie, J. Lin and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," IEEE Transactions on Automatic Control, 2003.

[11] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," IEEE Transactions on Automatic Control, 2004.

[12] J. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," IEEE Transactions on Automatic Control, 2004.

[13] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in Proceeding of the IEEE Conference on Decision and Control, 2007.

[14] D. Spanos, R. Olfati-Saber and R. M. Murray, "Dynamic consensus for mobile networks," IFAC World Congress, 2005.

[15] M. Zhu and S. Martinez, "On discrete-time dynamic average consensus," Automatica, 2010.

[16] W. Ren, "Consensus seeking in multi-vehicle systems with a time-varying reference state," in Proceeding of American Control Conference, 2007.

[17] R. Freeman, P. Yang and K. M. Lynch. "Stability and convergence properties of dynamic average consensus estimators," in Proceeding of the IEEE Conference on Decision and Control, 2006.

[18] A. Olshevsky and J. N. Tsitsiklis, "Convergence speed in distributed consensus and averaging." SIAM Journal on Control and Optimization, 2009.

[19] A. Olshevsky and J. N. Tsitsiklis, "A lower bound on distributed averaging," in Proceedings of IEEE Conference on Decision and Control, 2010.

[20] H. Zhang, M. Chen, G. Stan, T. Zhou and J. Maciejowski, "Collective behavior coordination with predictive mechanisms," IEEE Circuits and Systems Magazine, 2008.

[21] J. Cortes, "Finite-time convergent gradient flows with applications to network consensus," Automatica, 2006.

[22] L. Wang and F. Xiao, "Finite-time consensus problems for networks of dynamic agents," IEEE Transactions on Automatic Control, 2010.

[23] S. Sundaram and C. N. Hadjicostis, "Finite-Time distributed consensus in graphs with time-invariant topologies," in Proceedings of American Control Conference, 2007.

[24] Y. Yuan, G. Stan, L. Shi and J. Goncalves, " Decentralized final value theorem for discrete-time LTI systems with application to minimal-time distributed consensus," in Proceedings of IEEE Conference on Decision and Control, 2009.

[25] Y. Yuan, G. Stan, L. Shi, M. Barahona and J. Goncalves, " Decentralised minimal time consensus," submitted to Automatica.

[26] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," System and control letter, 2004.

[27] Vincent D. Blondel, Julien M. Hendrickx and John N. Tsitsiklis, "Continuous-time average-preserving opinion dynamics with opinion-dependent communications," SIAM Journal on Control and Optimization, 2010.

[28] K. Zhou, J. Doyle and K. Glover, "*Robust and Optimal Control*," Prentice Hall, 1996.

[29] J. Partington, "An introduction to Hankel operators," Cambridge University Press, 1988.

[30] S. Rump, "Structured perturbations part I: Normwise distances," SIAM Journal of Matrix Analysis, Application, 2003.

[31] M. Hitz, "On computing the nearest singular Hankel matrices," Proceedings of the International Symposium on Symbolic and Algebraic Computation, 2005.

[32] R. Olfati-Saber, "Distributed Kalman filter with embedded consensus filters," in Proceedings of IEEE Conference on Decision and Control, 2005.

[33] R. Olfati-Saber and J. S. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," in Proceedings of IEEE Conference on Decision and Control, 2005.