An Asynchronous Distributed Algorithm for Solving a Linear Algebraic Equation*

Ji Liu Shaoshuai Mou A. Stephen Morse

Abstract—A distributed algorithm is described for solving a linear algebraic equation of the form $Ax = b$ where $A$ is a matrix for which the equation has at least one solution. The equation is simultaneously and asynchronously solved by $m$ agents assuming each agent knows only a subset of the rows of the partitioned matrix $[A \ b]$, the estimates of the equation’s solution generated by its neighbors, and nothing more. Each agent recursively updates its estimate of a solution at its own event times by utilizing estimates generated by each of its neighbors which are transmitted with delays. Each agent has its own event time sequence and the event time sequences of different agents are not assumed to be synchronized. Neighbor relations are characterized by a time-dependent directed graph whose vertices correspond to agents and whose arcs depict neighbor relations. It is shown that for any matrix $A$ for which the equation has a solution and any repeatedly jointly strongly connected sequence of neighbor graphs defined on the merged sequence of all agents’ event times, the algorithm causes all agents’ estimates to converge exponentially fast to the same solution to $Ax = b$.

I. INTRODUCTION

Certainly the most well known and probably the most important of all numerical computations involving real numbers is solving a system of linear algebraic equations. Efforts to develop distributed algorithms to solve such systems have been under way for a long time especially in the parallel processing community where the main objective is to achieve efficiency by somehow decomposing a large system of linear equations into smaller ones which can be solved on parallel processors more accurately or faster than direct solution of the original equations would allow [1]–[5]. In some cases, notably in sensor networking [6] and some filtering applications [7], the need for distributed processing arises naturally because processors onboard sensors or robots are physically separated from each other. In addition, there are typically communication constraints which limit the flow of information across a robotic or sensor network and consequently preclude centralized processing, even if efficiency is not the central issue. It is with these thoughts in mind that we are led to consider the following problem.

We are interested in a network of $m > 1$ autonomous agents which are able to receive information from their “neighbors”. Neighbor relations can be conveniently characterized by a time-dependent directed graph with $m$ vertices and a set of arcs defined so that there is an arc in the graph from vertex $j$ to vertex $i$ just in case agent $j$ is a neighbor of agent $i$. Thus the directions of arcs represent the directions of information flow. Each agent $i$ has a real-time dependent state vector $x_i(t)$ taking values in $\mathbb{R}^n$, and we assume that the only information agent $i$ receives from neighbor $j$ is the state vector of neighbor $j$. We also assume that agent $i$ knows a pair of real-valued matrices $(A_i^{(n\times n)}, B_i^{(n\times 1)})$. The problem of interest is to devise local algorithms, one for each agent, which will enable all $m$ agents to iteratively and asynchronously compute solutions to the linear equation $Ax = b$ where $A = \text{column} \{ A_1, A_2, \ldots, A_m \}_{k\times n}$, $b = \text{column} \{ b_1, b_2, \ldots, b_m \}_{k\times n}$ and $n = \sum_{i=1}^m n_i$. We will assume that $Ax = b$ has a solution although we will not require it to be unique.

The most widely studied version of the problem just formulated is when $m = n$, the $A_i$ are linearly independent row vectors $a_i$, the $b_i$ are scalars, and the “neighbor graph” is a constant, symmetric and strongly connected graph. For this version of the problem, $A$ is therefore an $n \times n$ nonsingular matrix, $b$ is an $n$ vector and agent $i$ knows the state $x_i(t)$ of each of its neighbors as well as its own state. The problem in this case is thus for each agent $i$ to compute $A_i^{-1}b_i$ given $a_i$, $b_i$ and $x_i(t)$, $j \in N_i$, where $N_i$ is the set of labels of agent $i$’s neighbors. In this form, there are several classical parallel algorithms which address closely related problems. Among these are Jacobi iterations [5], so-called “successive over-relaxations” [1] and the classical Kaczmarz method [2]. Although these are parallel algorithms, all rely on “relaxation factors” which cannot be determined in a distributed way unless one makes special assumptions about $A$. Additionally, the implicitly defined neighbor graphs for these algorithms are generally strongly complete: i.e., all processors can communicate with each other.

Perhaps most closely related to the problem under discussion, is the problem posed in [8], [9]. The authors develop several algorithms and give sufficient conditions for them to work correctly.

An obvious approach to the problem we have posed is to reformulate it as a distributed optimization problem [10]–[12]. There are some especially nice ideas along these lines which one might bring to bear on the problem we have formulated. One of the most interesting stems from the approach taken in [10] which makes quite clever use of the idea of consensus. Here is a key idea motivated by the work in [10]: Observe that if $A_i x_i = b_i$ for $i \in \{1,2,\ldots,m\}$, and in addition, if a consensus is reached in that all $x_i$ are equal, then automatically all $x_i$ satisfy $A_i x_i = b$. So one can tackle the problem by trying to get each agent $i$ to solve its own equation and at the same time making sure that a consensus is reached. The algorithm proposed in [10] almost accomplishes this. However it is based on gradient descent and thus has the disadvantages normally associated with that methodology: e.g., slow convergence as the optimal solution is approached.

Another approach to the problem is to view it as a consensus problem in which the goal is for all $m$ agents to reach a consensus [by ultimately causing all of the $x_i$ to be equal], subject to the requirement that each $x_i$ satisfies the convex constraint $A_i x_i = b_i$. An algorithm for solving a large class of constrained consensus problems of this type in a synchronous manner, appears in [13]. Specialization of that algorithm to the problem of interest here, yields an algorithm similar to synchronous version of the algorithm which we will consider. The principle difference between the two - apart from correctness proofs - is that the algorithm stemming from [13] relies on convergence properties of doubly stochastic matrices whereas the synchronous version of the algorithm developed in this paper does not. As a consequence, the former cannot be implemented without assuming that each agent knows at least an
upper bound on the number of neighbors of each of its current neighbors, whereas the latter can. An extension of the constrained consensus algorithm developed in [13], which gets around this assumption, has appeared recently in [14].

Rather than go through the intermediate step of reformulating the distributed computation problem posed as an optimization problem or a constrained consensus problem, we approach the problem directly as was done in [15] for the case when neighbors do not change. For the case when neighbors change over time, the synchronous version and a restricted asynchronous version of the problem in which transmission delays are ignored, are addressed in [16]. In this paper we consider a more general asynchronous version of the problem in which transmission delays are explicitly taken into account. Each agent independently updates its state vector at times determined by its own clock. What makes the problem asynchronous is that it is not assumed that the agents’ clocks are synchronized or that the “event times” between which any one agent updates its state vector are evenly spaced. The aim of this paper is to prove that the conditions under which all \( x_i(t) \) converge to the same solution to \( Ax = b \) are essentially the same as those in the \{delay-free\} synchronous case derived in [16] provided the notion of an agent’s neighbor between its event times is appropriately defined.

A. Preliminaries

If \( M \) is a matrix, \( M \) denotes its column span. If \( n \) is a positive integer, \( n = \{1, 2, \ldots, n\} \). The graph of an \( n \times n \) matrix \( M \) with nonnegative entries is an \( n \)-vertex directed graph \( \gamma(M) \) defined so that \((i, j)\) is an arc from \( i \) to \( j \) in the graph just in case the \( j \)th entry of \( M \) is nonzero. Throughout this paper we write \( G \) for the set of all directed graphs with \( m \) vertices and \( G_{oa} \) for the set of all those graphs in \( G \) which have self-arcs at all vertices.

II. The Algorithm

We associate with each agent \( i \), a strictly increasing, infinite sequence of \textit{event times} \( t_1, t_2, \ldots \) with the understanding that \( t_1 \) is the time agent \( i \) initializes its state and the remaining \( t_{ik}, k > 1 \) are the times at which agent \( i \) updates its state. Between any two successive event times \( t_{ik} \) and \( t_{i(k+1)} \), \( x_i(t) \) is held constant. We assume that for any \( k \geq 1, x_i(t) \) equals its limit from above as \( t \) approaches \( t_{ik} \); thus \( x_i(t) \) is constant on each open half interval \( [t_{ik}, t_{i(k+1)}), k \geq 1 \).

We assume that for \( i \in \text{mm} \), agent \( i \)'s event times satisfy

\[
\bar{T}_i \geq t_{i(k+1)} - t_{ik} \geq T_i, \quad k \geq 1
\]

(1)

where \( \bar{T}_i \) and \( T_i \) are positive numbers such that \( \bar{T}_i > T_i \). Thus the event times of agent \( i \) are distinct and the difference between any two successive event times cannot be too large. We make no assumptions at all about the relationships between the event times of different agents. In particular, two agents may have completely different unsynchronized event time sequences.

Agent \( i \) communicates with the network at each of its event times by transmitting its current state to other agents in the network. For \( k > 1 \), we say that agent \( j \) is a \textit{neighbor} of agent \( i \) at time \( t_{ik} \) if agent \( i \) receives a state estimate from agent \( j \) at some time \( \tau \) in the interval \( [t_{i(k-1)}, t_{ik}) \). If agent \( i \) receives more than one state estimate from neighbor \( j \) during this interval, the last of these, called the \textit{final state}, is the one used by agent \( i \) in the computation of \( x_i(t_{i(k+1)}) \). We take agent \( i \) to be a neighbor of itself at every event time \( t_{ik} \). In the sequel \( N_i(t_1) = \{i\} \) while for \( k > 1 \), \( N_i(t_{ik}) \) is the set of labels of agent \( i \)'s neighbors at time \( t_{ik} \). Since agent \( i \) is always taken to be a neighbor of itself, \( N_i(t_{ik}) \) is never empty.

Prompted by the work in [15], the update rule for agent \( i \) we want to consider is

\[
x_i(t_{i(k+1)}) = x_i(t_{ik}) - \frac{1}{m_i(t_{ik})} P_i \left( m_i(t_{ik}) x_i(t_{ik}) - \sum_{j \in N_i(t_{ik})} x_j(t_{jk}) \right), \quad k \geq 1
\]

(2)

where for \( j \in N_i(t_{ik}), x_j(t_{jk}) \) is the final state agent \( i \) has received from agent \( j \) during the interval \( [t_{i(k-1)}, t_{ik}), m_i(t_{ik}) \) is the number of labels in \( N_i(t_{ik}) \), and \( P_i \) is the orthogonal projection on the kernel of \( A_i \).

It is possible to rewrite (2) in a form which is more convenient for analysis. For this, fix \( k > 1 \) and \( j \in N_i(t_{ik}) \). Suppose agent \( j \) transmits its state \( x_j(t_{jk}) \) to agent \( i \) at event time \( t_{jk} \) and agent \( i \) receives \( x_j(t_{jk}) \) at time \( \tau \in [t_{i(k-1)}, t_{ik}] \). Agent \( i \) then holds this state until time \( t_{ik} \) at which point it is used in the computation of \( x_i(t_{i(k+1)}) \) via (2). The \textit{transmission time} for this event is \( \tau - t_{jk} \), whereas the \textit{hold time} is \( t_{ik} - \tau \). By the \textit{delay time} for this event, written \( \delta_{ij}(k) \), is meant the sum of the transmission time and the hold time. Thus \( \delta_{ij}(k) = t_{ik} - t_{jk} \). Note that the hold time \( t_{ik} - \tau \) is bounded above by \( T_i \) because of (1). We assume that the transmission time \( \tau - t_{jk} \) is bounded above as well. Thus there is a positive bound \( \delta_{ij} \) such that \( \delta_{ij}(k) \leq \delta_{ij}, k \geq 2 \).

The preceding implies that \( x_j(t_{jk}) = x_j(t_{ik} - \delta_{ij}(k)) \). Thus it is possible to rewrite (2) as

\[
x_i(t_{i(k+1)}) = x_i(t_{ik}) - \frac{1}{m_i(t_{ik})} P_i \left( m_i(t_{ik}) x_i(t_{ik}) - \sum_{j \in N_i(t_{ik})} x_j(t_{ik} - \delta_{ij}(k)) \right), \quad k \geq 1
\]

(3)

with the understanding that \( \delta_{ij}(k) = 0, k \geq 1 \).

To proceed we need a common time scale on which all \( m \) agent update rules can be defined. For this, let \( t_1 = \max \{t_{ik} \} \) and write \( T_i \) for the event times of agent \( i \) which are greater than or equal to \( t_1 \). Let \( T \) denote the set of all event times of \( m \) agents which are greater than or equal to \( t_1 \). Thus \( T \) is the union of the \( T_i \). Relabel the times in \( T \) so that \( t_s < t_{s+1} \) for \( k \geq 1 \). For each \( i \in \text{mm} \) and \( t_r \in T \) define the \textit{extended neighbor set}

\[
N_i(t_r) = \{N_i(t_s) : t_s \in T_i, t_r \notin T_i\}
\]

Thus \( N_i(t_r) \) coincides with \( N_i(t_s) \) whenever \( t_s \) is an event time of agent \( i \) and the simple index \( i \) otherwise. We describe all defined neighbor relationships at time \( \tau \in \{1, 2, \ldots\} \) to be the directed graph \( \bar{\Gamma}(\tau) \) with vertex set \( \mathbb{V} = \text{mm} \) and arc set \( \bar{\Sigma}(\tau) \subset \mathbb{V} \times \mathbb{V} \) which is defined so that \( (i, j) \) is an arc from \( i \) to \( j \) just in case label \( i \) is in the neighbor set \( N_i(t_r) \). Thus each \( \bar{\Gamma}(\tau) \) is a directed graph on \( m \) vertices with at most one arc between each ordered pair of vertices and with exactly one self-arc at each vertex. We call \( \bar{\Gamma}(\tau) \) the \textit{extended neighbor graph} of the asynchronous system (2) at time \( \tau \).

III. Main Results

To state our main result, we need a few concepts. We say that a directed graph is \textit{strongly connected} if there is a directed path between each pair of distinct vertices. By the \textit{composition} of two directed graphs \( G_p, G_q \) with the same vertex set, written \( G_q \circ G_p \), is meant that directed graph with the same vertex set and arc set...
defined so that \((i, j)\) is an arc in the composition just in case there is a vertex \(k\) such that \((i, k)\) is an arc in \(G_p\) and \((k, j)\) is an arc in \(G_q\). The definition extends unambiguously to any finite sequence of directed graphs with the same vertex set. Let us agree to say that an infinite sequence of graphs \(G_1, G_2, \ldots\) with the same vertex set is \textit{repeatedly jointly strongly connected}, if for some positive integer \(l\) and each integer \(k > 0\), the composed graph \(G_k = G_{k-1} \circ G_{k-2} \circ \cdots \circ G_{l+1}\) is strongly connected.

In the \{delay-free\} synchronous version of the problem treated in [16], for each \(k \geq 1\), the \(k\)th event times \(t_{1k}, t_{2k}, \ldots, t_{mk}\) of all \(m\) agents are the same. Agent \(i\) receives the states from its neighbors only at its event times and no delays occur in the values of the states which agent \(i\) received \((i.e., \delta_{ij}(k) = 0 \text{ for all } i, j \in m)\). Thus in this case equations (3) can be written as

\[
x_i(t_{k+1}) = x_i(t_k) - \frac{1}{m_k(t_k)} P_i \left( m_k(t_k) x_i(t_k) \right)
\]

where \(t_k = t_{ik}\). Define \(N(k)\) to be the directed graph with vertex set \(V = m\) and arc set \(A(k) \subseteq V \times V\) which is defined so that \((i, j)\) is an arc from \(i\) to \(j\) just in case label \(i\) is in the neighbor set \(N_j(t_k)\). We call \(N(k)\) the \textit{neighbor graph} of the synchronous system (4) at time \(t_k\). The result for this version of the problem can be found in [16] and is as follows.

\textbf{Theorem 1:} For any trajectory of the synchronous system defined by (4) whose associated sequence of neighbor graphs \(N(1), N(2), \ldots\) is repeatedly jointly strongly connected, then there exists a real nonnegative number \(\mu < 1\) such that all \(x_i(t)\) converge to the same solution to \(Ax = b\) as \(t \to \infty\), as fast as \(\mu^n\) converges to zero.

The aim of this paper is to prove that essentially the same result holds in the face of asynchronous updating and transmission delays.

\textbf{Theorem 2:} For any trajectory of the asynchronous system defined by (2) whose associated sequence of extended neighbor graphs \(\tilde{N}(1), \tilde{N}(2), \ldots\) is repeatedly jointly strongly connected, then there exists a real nonnegative number \(\mu < 1\) such that all \(x_i(t)\) converge to the same solution to \(Ax = b\) as \(t \to \infty\), as fast as \(\mu^n\) converges to zero.

\section{IV. Analysis}

In this section we explain why Theorem 2 is true. First we will use the concept of \textit{analytic synchronization} [17] to develop a synchronous system \(S\) for the asynchronous system under consideration.

\textbf{A. Analytical Synchronization}

1) \textbf{Definition of \(S\)}: We begin with the final state \(x_j(t_{ik} - \delta_{ij}(k))\) in (3). It is clear from the preceding that \(x_j(t_{ik} - \delta_{ij}(k)) = x_j(t_{ijk})\) for some \(t_{ijk} \in T_j\). Note that \(t_{ik}\) and \(t_{ijk}\) are two different event times in \(T\). Set \(t_{ik} = t_r\) and \(t_{ijk} = t_s\) where \(\sigma, \tau \in \{1, 2, \ldots\}\) and \(\sigma < \tau\). We write \(d_{ij}(\tau) = \tau - \sigma\) for the number of distinct event times in \(T\) during the time interval \((t_\sigma, t_\tau)\). As a consequence of the assumption that inequality (1) holds, there must exist a bounded integer \(d\) such that \(d_{ij}(\tau) < d\) for all \(i, j \in m\) and \(t_\tau \in T\). Then \(x_j(t_k - \delta_{ij}(k)) = x_j(t_{r,d_{ij}(\tau)})\) and \(d_{ij}(\tau) \in \{0, 1, \ldots, d - 1\}\). Because each agent can always access the latest value of its own state, \(d_{ij}(\tau) = 0\) for all \(i\) \in \(m\) and \(t_\tau \in T\).

For each \(i \in m\) and \(t_q \in T\), define

\[
x_i(t_r) = x_i(t_{r,q}), \quad \sigma < \tau \leq q'
\]

where \(t_q\) is the first event time of agent \(i\) after \(t_r\). Note that for any \(t_q \in T\), there is always such a \(q'\) because we have assumed that inequality (1) holds. Then each agent’s state is well defined at any other agent’s event times. Let \(\bar{x}_i(\tau) = x_i(t_r)\). Doing this enables us to extend the domain of applicability of update rule (3) from \(T_r\) to all of \(T\). In particular

\[
\bar{x}_i(\tau + 1) = \bar{x}_i(\tau) - \frac{1}{\bar{m}_i(\tau)} P_i \left( \bar{m}_i(\tau) \bar{x}_i(\tau) \right) - \sum_{j \in N_i(\tau)} \bar{x}_j(\tau) - d_{ij}(\tau) = 0 \quad j = i
\]

where \(\bar{m}_i(\tau)\) is the number of labels in \(\bar{N}_i(\tau)\), \(d_{ij}(\tau)\) \(\in\) \\{0, 1, ..., \(d - 1\)\} if \(j \neq i\) and \(d_{ij}(t) = 0\) if \(j = i\).

2) \textbf{State Space Model:} It is possible to represent the system defined by (5) using a state space model similar to the model for the delay-free case, much as was done in [18]. Towards this end, let \(G\) denote the set of all directed graphs with vertex set \(V = V_1 \cup V_2 \cup \cdots \cup V_m\) where \(V_1 = \{v_1, v_2, \ldots, v_{m1}\}\). Here each \(v_{ij}\) \(j > 1\) labels the \((j - 1)\)th possible delay value of \(\bar{x}_i(\tau)\), namely \(\bar{x}_i(\tau - j + 1)\). Sometimes we write \(i\) for \(v_{1i}, i \in m\), \(V\) for the subset of vertices \(\{v_{11}, v_{21}, \ldots, v_{m1}\}\), and think of \(v_{1i}\) as an alternative label of agent \(i\).

To take account of the fact that each agent can use its own current state in its update formula (5), we will utilize those graphs in \(G\) which have self-arcs at each vertex in \(V\). We will also require the arc set of each such graph to have, for \(i \in m\), an arc from each vertex \(v_{ij}\) in \(V_i\) except the last, to its successor \(v_{ij+1}\) in \(V_i\). Finally we stipulate that for each \(i \in m\), each vertex \(v_{ij}\) with \(j > 1\) has in-degree of exactly \(1\). In the sequel we call any such graph a \textit{delay graph} and write \(D\) for the subset of all such graphs. Note that unlike the class of graphs \(G_{\infty}\) considered before, there are graphs in \(D\) possessing vertices without self-arcs.

There is a simple relationship between \(D(\tau)\) and the neighbor graph \(\bar{N}(\tau)\) defined earlier. In particular,

\[
\bar{N}(\tau) = Q(D(\tau))
\]

where \(Q(D(\tau))\) is the “quotient graph” of \(D(\tau)\). By the \textit{quotient graph} of any \(G \in \bar{G}\), written \(Q(G)\), is meant that directed graph in \(\bar{G}\) with vertex set \(V\) whose arc set consists of those arcs \((i, j)\) for which \(G\) has an arc from some vertex in \(V_i\) to some vertex in \(V_j\). The quotient graph of \(D(\tau)\) thus models which states are being used by each agent in updates at time \(\tau\) without describing the specific delayed states actually being used.

To proceed, let \(x^*\) be any solution to \(Ax = b\). Then \(x^*\) must satisfy \(A_i x^* = b_i\) for \(i \in m\). Thus if we define

\[
\bar{y}_i(\tau) = \bar{x}_i(\tau) - x^*, \quad i \in m, \quad \tau \geq 1
\]

then \(\bar{y}_i(\tau) \in P_i \hat{A} \text{ image } P_i, \quad \tau \geq 1\), because \(P_i = \ker A_i\). Therefore \(P_i \bar{y}_i(\tau) = \bar{y}_i(i), \quad i \in m, \quad \tau \geq 1\). From (5),

\[
\bar{y}_i(\tau + 1) = \frac{1}{\bar{m}_i(\tau)} P_i \sum_{j \in N_i(\tau)} P_j \bar{y}_j(\tau - d_{ij}(\tau)), \quad i \in m
\]

It is possible to combine these \(m\) update equations into one linear recursion equation. To accomplish this, define \(\bar{y}(t)\) to be that \(dn\) vector whose first \(m\) elements are \(y_1(t)\) to \(y_m(t)\), whose next \(m\) elements are \(y_1(t - 1)\) to \(y_m(t - 1)\), and so on. Let \(\hat{A}\) be a diagonal block matrix whose first \(m\) diagonal blocks are \(P_i\), whose next \(m\) diagonal blocks are \(P_2\), and so on.

We also write \(A_{\hat{N}(\tau)}\) for the adjacency matrix of \(\bar{N}(\tau)\), \(D_{\hat{N}(\tau)}\) for the diagonal matrix whose
The $ij$th diagonal entry is the in-degree of vertex $v_{ij}$ in $\bar{N}(\tau)$, and let 
$F(\tau) = D^{-1} \bar{N}(\tau) A' \bar{N}(\tau)$. It is straightforward to verify that
\[ \bar{g}(\tau + 1) = \bar{P}(F(\tau) \otimes I) \bar{P} \bar{y}(\tau), \quad \tau \geq 1 \] 
(8)

Note that $F(\tau)$ is a stochastic matrix; in the literature it is sometimes referred to as a flocking matrix. But unlike the situation in the delay-free case, not all the diagonal entries of $F(\tau)$ are positive. We write $\bar{F}$ for the set of all such matrices. $F(\tau) \otimes I$ is the $d\times d\times n^3$ matrix which results when each entry $f_{ij}(\tau)$ of $F(\tau)$ is replaced by $f_{ij}(\tau)$ times the $n \times n$ identity. Note that $P^2 = P$ because each $P_i$ is idempotent.

**B. Uniqueness**

First we study the special case when $Ax = b$ has a unique solution. This is exactly when $\cap_{i=1}^m \ker A_i = 0$. Since $\ker A_i = P_i$, $i \in m$, the uniqueness assumption is equivalent to the condition
\[ \bigcap_{i=1}^m P_i = 0 \] 
(9)

Our goal is to derive conditions under which $\bar{y} \to 0$ since this will imply that all $x_i$ approach the desired solution $z^*$ in the limit at $t \to \infty$. To state the key technical condition to accomplish this, we need to define a special “mixed-matrix” norm. Let $\| \cdot \|_\infty$ denote the induced infinity norm and write $\mathbb{R}^{dn \times dn}$ for the vector space of all $dm \times dm$ block matrices $Q = [Q_{ij}]$ whose $ij$th entry is an $n \times n$ matrix $Q_{ij} \in \mathbb{R}^{n \times n}$. We define the mixed matrix norm of $Q \in \mathbb{R}^{dm \times dm}$, written $\| Q \|$, to be
\[ \| Q \| = \| (Q) \|_\infty \]
where $(Q)$ is the $dm \times dm$ matrix in $\mathbb{R}^{dm \times dm}$ whose $ij$th entry is $[Q_{ij}]$. It is very easy to verify that $\| \cdot \|_\infty$ is in fact a norm. It is even sub-multiplicative [cf. Lemma 3].

To state our main technical result, we also need the following idea. Let $l$ be a positive integer. A compact subset $\mathcal{C}$ of $dm \times dm$ stochastic matrices with graphs in $\bar{F}$ is $l$-compact if the set $\mathcal{C}_l$ consisting of all sequences $S_1, S_2, \ldots, S_l,$ $S_i \in \mathcal{C}$, for which $Q(\gamma(S_1)) \circ \cdots \circ Q(\gamma(S_2)) \circ Q(\gamma(S_1))$ is strongly connected, is nonempty and compact.

The key technical result we will need is as follows.

**Theorem 3:** Suppose that (9) holds. Let $l$ be a positive integer. Let $\mathcal{C}$ be an $l$-compact subset of $dm \times dm$ stochastic matrices in $\bar{F}$ and define
\[ \lambda = \left( \sup_{H_{\omega \in \mathcal{C}_L}} \sup_{H_{\omega - 1 \in \mathcal{C}_{L - 1}}} \cdots \sup_{H_1 \in \mathcal{C}_1} \| \bar{P}(Q_{\omega \cdots 1} \otimes I) \bar{P}(Q_{\omega - 1 \cdots 1} \otimes I) \cdots \bar{P}(Q_{1} \otimes I) \bar{P} \| \right)^{\frac{1}{l}} \]
where $\omega = m(dm - 1)$ and for $i \in \{1, 2, \ldots, \omega\}$, $H_i$ is the subsequence $Q(i-1)_{i+1} Q(i-1)_{i+2} \ldots, Q(i)_{i+1}$. Then $\lambda < 1$, and for any infinite sequence of stochastic matrices $S_1, S_2, \ldots$ in $\mathcal{C}$ whose graphs form a sequence $Q(\gamma(S_1)), Q(\gamma(S_2)), \ldots$ which is repeatedly jointly strongly connected by contiguous subsequences of length $l$,
\[ \| \bar{P}(S_{\omega \cdots 1} \otimes I) \bar{P}(S_{\omega - 1 \cdots 1} \otimes I) \cdots \bar{P}(S_{1} \otimes I) \bar{P} \| \leq \lambda^{(\tau - lw)} \] 
(10)

This theorem will be proved in IV-D.3.

**Proof of Theorem 2 (assuming (9) holds):** Since directed graphs in $D$ are bijectively related to flocking matrices in $\bar{F}$, the set $\bar{F}$ of distinct subsequences $F((k - 1)l + 1), F((k - 1)l + 2), \ldots, F(kl)$, $k \geq 1$, encountered along any trajectory of (8) must be a finite and thus compact set. Moreover the composed graphs $Q(\gamma(F(kl))) \circ \cdots \circ Q(\gamma(F(l)))$ is strongly connected. For $l$ and $Q(\gamma(F(\tau))) = \bar{N}(\tau)$, $\tau \geq 1$. Hence Theorem 3 is applicable to (8) $\bar{y}(\tau)$ converges to zero exponentially fast. From this and (6) it follows that Theorem 2 is true for the case when $Ax = b$ has a unique solution.

**C. Nonuniqueness**

We now consider the general case in which (9) is not presumed to hold. This is the case when $Ax = b$ does not have a unique solution. To deal with this case we will first [in effect] “quotient out” the subspace $\cap_{i=1}^m P_i$ thereby obtaining a subsystem to which Theorem 3 can be applied. The steps involved in doing this are as follows.

Let $Q'$ be any matrix whose columns form an orthonormal basis for the subspace $(\cap_{i=1}^m P_i)^\perp$ and define $P_i = Q_i P_i Q'$. $i \in m$. Then the following statements are true.

**Lemma 1:** (1) Each $P_i$, $i \in m$, is an orthogonal projection matrix. (2) Each $P_i$, $i \in m$, satisfies $QP_i = P_i Q$. (3) $\cap_{i=1}^m P_i = 0$.

Note that property 2 of Lemma 1 implies that $QP_i P_j = P_j P_i Q$ for all $i, j \in m$. Thus if we define $z_i = \bar{Q}_{ij} y_i$, $i \in m$, from (7)
\[ z_i(\tau + 1) = \frac{1}{m_i(\tau)} \bar{P}_i \sum_{j \in N_i(\tau)} P_j z_j(\tau - d_{ij}(\tau)), \quad i \in m \] 
(11)

Observe that (11) has exactly the same form as (7) except for the $\bar{P}_i$ which replaces the $P_i$. But in view of Lemma 1, the $\bar{P}_i$ are also orthogonal projection matrices and $\cap_{i=1}^m \bar{P}_i = 0$. Thus Theorem 3 is also applicable to the system of iterations (11). Therefore $z_i \to 0$ exponentially fast as $\tau \to \infty$.

Define $\bar{z}_i = \bar{y}_i - Q' z_i$, $i \in m$. Note that $Q' \bar{z}_i = Q' \bar{y}_i - z_i$ so $Q' \bar{z}_i = 0$, $i \in m$. Thus $\bar{z}_i(\tau) \in \cap_{i=1}^m P_i$, $i \in m$. Clearly $P_j \bar{z}_i(\tau) = \bar{z}_i(\tau), i, j \in m$. Moreover from property 2 of Lemma 1, $P_i Q' = Q' P_i$. These facts and (11) imply that
\[ z_i(\tau + 1) = \frac{1}{m_i(\tau)} \sum_{j \in N_i(\tau)} \bar{z}_j(\tau - d_{ij}(\tau)), \quad i \in m \] 
(12)

These equations are the update equations for the standard consensus problem with measurement delays treated in [18] and elsewhere for case when the $\bar{z}_i$ are scalars. It is well known that for the scalar case, a sufficient condition for all $\bar{z}_i$ to converge exponentially fast to the same value is that the neighbor graph sequence $\bar{N}(\tau)$, $\tau \geq 1$ be repeatedly jointly strongly connected for some $l$ [18]. But since the vector update (12) decouples into $n$ independent scalar update equations, the convergence conditions for the scalar equations apply without change to the vector case as well. Thus all $\bar{z}_i$ converge exponentially fast to the same limit $z^* \in \cap_{i=1}^m P_i$. Therefore all $\bar{z}_i$ defined by (5) converge to the same limit $z^* + z^* = z^*$ which solves $Ax = b$. This concludes the proof of Theorem 2 for the case when $Ax = b$ does not have a unique solution.

**D. Justification for Theorem 3**

In this section we develop the ideas needed to prove Theorem 3. We begin with the following lemma about projection matrices.

**Lemma 2:** For any nonempty set of $n \times n$ real orthogonal projection matrices $\{P_1, P_2, \ldots, P_k\}$,
\[ |P_k \cdots P_1| \leq 1 \] 
(13)
Moreover $|P_kP_{k-1} \cdots P_1| < 1$ if and only if \( \bigcap_{i=1}^k P_i = 0 \).

1) Projection Matrix Polynomials: Let \( \{P_1, P_2, \ldots, P_m\} \) be a set of \( n \times n \) orthogonal projection matrices for which (9) holds. We will be interested in matrices of the form

\[
\mu(P_1, P_2, P_3, \ldots, P_m) = \sum_{i=1}^c \lambda_i P_{h(i)}(1)P_{h(i)}(2) \cdots P_{h(i)}(q_i)
\]

(14)

where \( q_i \) and \( c \) are positive integers, \( \lambda_i \) is a real positive number, and for each \( j \in \{1, 2, \ldots, q_i\} \), \( h(j) \) is an integer in \( m \). We call such matrices together with the \( n \times n \) zero matrix, projection matrix polynomials. The set of projection matrix polynomials, written \( \mathbb{P} \), is clearly closed under matrix addition and multiplication. Let us note from the triangle inequality, that

\[
|\mu(P_1, P_2, P_3, \ldots, P_m)| \leq \sum_{i=1}^c |\lambda_i| P_{h(i)}(1)P_{h(i)}(2) \cdots P_{h(i)}(q_i)
\]

From this and (13) it follows that

\[
|\mu(P_1, P_2, P_3, \ldots, P_m)| \leq |\mu(P_1, P_2, P_3, \ldots, P_k)|
\]

(15)

where \( |\mu(P_1, P_2, P_3, \ldots, P_k)| = \sum_{i=1}^k \lambda_i \). We call \( |\mu| \) the nominal bound of \( \mu \). Notice that the actual 2-norm of \( \mu \) will be strictly less than its nominal bound provided at least one “component” of \( \mu \) has a 2-norm less than one where by a component of \( \mu \) we mean any matrix product \( P_{h_i(1)}P_{h_i(2)} \cdots P_{h_i(q_i)} \) appearing in the sum in (14) which defines \( \mu \). In view of Lemma 2, a sufficient condition for \( P_{h(i)}(1)P_{h(i)}(2) \cdots P_{h(i)}(q_i) \) to have a 2-norm less than 1 is that

\[
\bigcap_{j=1}^{q_i} \text{Im}(P_{h(i)(j)}) = 0
\]

As a consequence of (9), this in turn will always be true if each of the projections matrices in the set \( \{P_1, P_2, \ldots, P_m\} \) appears in the component \( P_{h_i(1)}P_{h_i(2)} \cdots P_{h_i(q_i)} \) at least once. Prompted by this we say that a nonzero projection matrix polynomial \( \mu(P_1, P_2, P_3, \ldots, P_m) \) is complete if it has a component \( P_{h_i(1)}P_{h_i(2)} \cdots P_{h_i(q_i)} \) within which each of the projections matrices \( Pi \), \( j \in \mathbb{M} \) appears at least once. Complete projection matrix polynomials are thus a class of projection matrix polynomials with 2-norms strictly less than their nominal bounding values. The converse of course is not necessarily so.

The ideas just discussed extend in an external way to “projection block matrices.” By an \( dm \times dm \) projection block matrix is meant a block partitioned matrix of the form

\[
M = [\mu_{ij}(P_1, P_2, P_3, \ldots, P_m)]_{dm \times dm}
\]

An \( dm \times dm \) projection block matrix is thus an \( dm n \times dm n \) matrix of real numbers partitioned into \( n \times n \) sub-matrices which are projection matrix polynomials. The set of all \( dm \times dm \) projection block matrices, written \( \mathbb{P}^{dm \times dm} \), is clearly closed under multiplication. By the nominal bound of \( M = [\mu_{ij}(P_1, P_2, P_3, \ldots, P_m)]_{dm \times dm} \in \mathbb{P}^{dm \times dm} \), written \( |M| \), we mean the \( dm \times dm \) matrix whose \( ij \)th entry is the nominal bound of \( \mu_{ij}(P_1, P_2, P_3, \ldots, P_m) \). Using (15) it is quite easy to verify that

\[
[|\mu_{ij}(P_1, P_2, P_3, \ldots, P_m)|]_{dm \times dm} \leq |M|
\]

(16)

where for any real matrices \( X \) and \( Y \) of the same size, \( X \leq Y \) means that \( X - Y \) is a matrix of nonnegative numbers. The definition of nominal bound of a projection matrix polynomial implies that for all \( \mu_1, \mu_2 \in \mathbb{P}, [\mu_1] [\mu_2] = [\mu_1 + \mu_2] \) and \( |\mu_1 + \mu_2| = |\mu_1| + |\mu_2| \). From this it follows that

\[
|M_1 M_2| = |M_1| |M_2|, \quad M_1, M_2 \in \mathbb{P}^{dm \times dm}
\]

(17)

In order to measure the sizes of matrices in \( \mathbb{P}^{dm \times dm} \) we shall make use of the mixed matrix norm \( ||\cdot|| \) defined earlier. A critical property of this norm is that it is sub-multiplicative.

Lemma 3:

\[
||AB|| \leq ||A|| ||B||, \quad \forall A, B \in \mathbb{R}^{dm \times dm}
\]

It is worth noting that the preceding properties of \( ||\cdot|| \) remain true for any pair of standard matrix norms provided both are sub-multiplicative. It is conceivable that the mixed matrix norm which results when the 1-norm is used in place of the 2-norm, will find application in the study of distributed compressed sensing algorithms [19].

Let \( M = [\mu_{ij}]_{dm \times dm} \) be a matrix in \( \mathbb{P}^{dm \times dm} \). Since \( \langle M \rangle = ||\mu_{ij}||_{dm \times dm} \), it is possible to rewrite (16) as

\[
\langle M \rangle \leq |M|, \quad M \in \mathbb{P}^{dm \times dm}
\]

Therefore

\[
||M|| \leq ||M||_{\infty}, \quad M \in \mathbb{P}^{dm \times dm}
\]

Thus in the case when \( |M| \) turns out to be a stochastic matrix, which is exactly the case we are interested in, \( ||M|| \leq 1 \). In other words, when \( |M| \) is a stochastic matrix, \( M \) is non-expansive.

What we are especially interested in are conditions under which \( M \) is a contraction in the mixed matrix norm we have been discussing.

Proposition 1: Any matrix \( M \in \mathbb{P}^{dm \times dm} \) whose nominal bound is stochastic, is non-expansive in the mixed matrix norm. If, in addition, at least one entry in each block row of \( M \) is complete and (9) holds, then \( M \) is a contraction in the mixed matrix norm.

2) The Closure of \( D \): In the sequel we will use some concepts and ideas which have been introduced in [18]. We call a vertex \( i \) of a directed graph \( G \), a root of \( G \) if for each other vertex \( j \) of \( G \), there is a directed path from \( i \) to \( j \). Thus \( i \) is a root of \( G \), if it is the root of a directed spanning tree of \( G \). We will say that \( G \) is rooted at \( i \) if \( i \) is in fact a root.

We say that a rooted graph \( G \in \mathbb{G} \) is a hierarchical graph with hierarchy \( \{v_1, v_2, \ldots, v_m\} \) if it is possible to re-label the vertices in \( \mathbb{V} \) as \( v_1, v_2, \ldots, v_m \) in such a way so that \( v_1 \) is a root of \( G \) with a self-arc and for \( i > 1 \), \( v_i \) has a neighbor \( v_{ij} \) “lower” in the hierarchy where lower we mean \( j < i \). It is clear that any graph \( G \) in \( \mathbb{G} \) possessing a self-arc is hierarchical. Note that a graph may have more than one hierarchy and two graphs with the same hierarchy need not be equal.

It can be shown by example that \( D \) is not closed under composition. We deal with this problem as follows. First, let us agree to say that a vertex \( v \) in a graph \( G \in \mathbb{G} \) is a neighbor of a subset of \( G \)'s vertices \( U \), if \( v \) is a neighbor of at least one vertex in \( U \).

Next, we say that a graph \( G \in \mathbb{G} \) is an extended delay graph if for each \( i \in \mathbb{M} \) (i) every neighbor of \( v_i \) which is not in \( V_i \) is a neighbor of \( v_{1i} \) and (ii) the subgraph of \( G \) induced by \( V_i \) has \( \{v_{1i}, v_{2i}, \ldots, v_{ia}\} \) as a hierarchy. We write \( D \) for the set of all extended delay graphs in \( G \). It is easy to see that every delay graph is an extended delay graph. The converse however is not true. \( D \) is closed under composition (cf. Proposition 3 of [18]).

By the agent subgraph of \( G \in \mathbb{G} \) is meant the subgraph of \( G \) induced by \( V \). Note that while the quotient graph of \( G \) describes relations between distinct agent hierarchies, the agent subgraph of \( G \) only captures the relationships between the roots of the hierarchies. Note in addition that both the agent subgraph of \( G \) and the quotient graph of \( G \) are graphs in \( \mathbb{G} \) because all \( m \) vertices of \( G \) in \( V \) have self-arcs.
Proposition 2: The composition of any set of at least \( dm - 1 \) extended delay graphs will have a complete agent subgraph and the sequences which are repeatedly divided by \( \omega \); thus \( \lambda q_r \) is the unique integer quotient of \( \tau \) divided by \( \omega \).

Moreover \( \tau = \omega q_r + \rho_r \) where \( \rho_r \) is the unique integer remainder of \( \tau \) divided by \( \omega \). Thus \( \lambda^{\omega q_r} = \lambda^\tau \rho_r \). But \( \rho_r < \omega \) and \( \lambda < 1 \) so \( \lambda^{(\tau - \rho_r)} \leq \lambda^{(\tau - \omega)} \). It follows from this and (18) that (10) is true.

V. CONCLUDING REMARKS

A shortcoming of the algorithm discussed in this paper is that it requires each agent to communicate its full state \( x_i(t) \) to each of its neighbors. Another is that the algorithm is not suitable for over specified linear equations \( Ax = b \) where a least squares solution might still be valuable [11]. These are issues for future research.

REFERENCES