ADMM Optimization Strategies for Wide-Area Oscillation Monitoring in Power Systems Under Asynchronous Communication Delays

Jianhua Zhang, *Student Member, IEEE*, Seyedbehzad Nabavi, *Member, IEEE*, Aranya Chakrabortty, *Senior Member, IEEE*, and Yufeng Xin

***Abstract*—In this paper, we present a suite of asynchronous distributed optimization algorithms for wide-area oscillation estimation in power systems using alternating direction method of multipliers (ADMMs). We first pose the estimation problem as a real-time, iterative, and distributed consensus problem. Thereafter, we consider a probabilistic traffic model for modeling delays in any typical wide-area communication network, and study how the delays enter the process of information exchange between distributed phasor data concentrators that are employed to execute this consensus algorithm in a coordinated fashion. Finally, we propose four different strategies by which the convergence rate and accuracy of this consensus algorithm can be made immune to the asynchrony resulting from the network traffic. We carry out extensive simulations to show possible numerical instabilities and sensitivities of the ADMM convergence on our proposed strategies. Our results exhibit a broad view of how the convergence of any distributed estimation algorithm in a generic cyber-physical system depends strongly on the uncertainties of the underlying communication models.**

***Index Terms*—Distributed optimization, wide-area monitoring, mode estimation, alternating direction method of multipliers (ADMM), synchrophasors.**

I. Introduction

FOLLOWING the Northeast blackout of 2003, Wide-Area Measurement System (WAMS) technology using Phasor Measurement Units (PMUs) has largely matured for the North American grid [1]. However, as the number of PMUs scales up to the thousands in the next few years under the U.S. Department of Energy's smart grid demonstration initiative, Independent System Operators (ISO) and utility companies are struggling to understand how the resulting gigantic volumes of real-time data can be efficiently harvested, processed, and utilized to solve wide-area monitoring and control problems for any realistic power system interconnection. It is intuitive that the current state-of-the-art centralized communication and information processing architecture of WAMS will no longer be sustainable under such a data explosion, and a completely distributed cyber-physical architecture will need to be developed [2]. Research is currently being carried out by the Data and Network Management Task Team (DNMTT) of North American Synchrophasor Initiative (NASPI) on the implementation of this distributed architecture with the prime research focus being protocols, Quality-of-Service, latency, bandwidth and security [3].

However, very little attention has been paid to perhaps two of the most critical consequences of this envisioned distributed architecture - namely a) formulation of traditional centralized strategies for wide-area monitoring and control as distributed algorithms, and b) investigation of convergence and accuracy properties of these algorithms when they are implemented in a completely asynchronous environment that is bound to arise in any practical wide-area communication network. We recently proposed a distributed cyber-physical architecture to address the first problem in [2]. In this paper we address the second problem, focusing on the fundamental question on how asynchrony in a communication network can influence the performance of one of the most critical wide-area monitoring applications, namely, modal estimation of electro-mechanical oscillations using Synchrophasors. Several centralized algorithms for solving this problem have been proposed over the past decade including the Eigenvalue Realization Algorithm (ERA) and Prony analysis [4], mode metering [5], matrix pencil [6], and Hilbert-Huang transform [7]. However, all of these algorithms are based on offline techniques, and that too using only a handful (but observable) set of PMUs. In contrast, we formulate the mode estimation problem as a global consensus problem for the coefficients of the characteristic polynomial of the system, and then solve it using Alternating Direction Method of Multipliers (ADMM) [8]. The physical grid is assumed to be divided into multiple balancing regions or *areas*, which may or may not be coherent, but belong to different utility companies. PMUs in each area communicate their data in real-time to estimator(s) or Phasor Data Concentrators (PDC) located at the local control center via a Virtual Private Network (VPN). These local PDCs can then share information between each other and also with a

Manuscript received September 1, 2015; revised January 20, 2016; accepted March 17, 2016. Date of publication March 29, 2016; date of current version June 17, 2016. This work was supported in part by the U.S Department of Energy, and in part by the U.S National Science Foundation under Grant ECCS 1054394, Grant ECCS 1329780, and Grant CNS 1329745. Paper no. TSG-01069-2015.

J. Zhang and A. Chakrabortty are with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695 USA (e-mail: jzhang25@ncsu.edu; achakra2@ncsu.edu).

S. Nabavi is with New York Power Authority, White Plains, NY 10601 USA (e-mail: seyedbehzad.nabavi@nypa.gov).

Y. Xin is with the Renaissance Computing Institute, University of North Carolina, Chapel Hill, NC 27517 USA (e-mail: yxin@renci.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSG.2016.2547939

central PDC located at the Independent System Operator (ISO) through a wide-area communication network, examples of which can range from standard Internet to any controllable network such as Software Defined Network (SDN). We consider a probabilistic traffic model for modeling delays in such controllable wide-area networks, and study how these delays enter the process of information exchange between the PDCs. The definition of *information* for this specific application pertains to the vector of local estimates of the coefficients of the characteristic polynomial of the system transfer function, estimated by each PDC using local PMU measurements. The reason for exchanging these small-size local estimates instead of large volumes of PMU data between the various PDCs follows from transmission costs as well as real-time computation costs. We propose four different update rules by which the convergence rate of ADMM and the accuracy of estimation of these coefficients can be made immune to the asynchrony resulting from the network traffic as much as possible. We carry out extensive simulations using an IEEE prototype power system model to show possible numerical instabilities and sensitivities of the ADMM convergence on our proposed strategies. Preliminary results on two of the proposed algorithms, considering only uni-directional delays, have recently been reported in our conference paper [9]. This paper extends those initial findings to two new strategies, each with multiple sub-scenarios, and considers the combined impact of both uplink and downlink delays in asynchronous ADMM.

Several recent papers on distributed optimization such as [10]–[14], and references therein, have presented seminal results on how ADMM can be used for distributed estimation under asynchronous communication. The optimization architecture involving a master-slave strategy proposed in [13], for example, is especially relevant to our problem. Applications of these algorithms to various practical examples have also been illustrated [15]–[18]. However, majority of these papers do not consider any explicit model of the network delays from where the asynchrony arises. The novelty of our work in contrast to these existing papers is to bridge this gap, and investigate how various parameters of the network traffic models for wide-area communication networks, for example, can impact convergence and accuracy of distributed estimation. The underlying motivation for this approach, again, is to emphasize on the controllability properties of future SDNs by which the delay model parameters can be appropriately controlled in order to guarantee desired rates of convergence for our proposed algorithms.

The remainder of the paper is organized as follows. Section II formulates the mode estimation problem using Synchrophasors, and casts it as a distributed consensus problem using ADMM. Section III provides a stochastic delay model for wide-area communication networks. Section IV proposes multiple asynchronous ADMM (A-ADMM) strategies, and also derives their convergence conditions. Section V presents simulation results for the different variants of A-ADMM, and compares them with a distributed subgradient method. Section VI concludes the paper.

II. Problem Statement and Recap of Mode Estimation Using Synchronous ADMM

A. Problem Statement

Consider a power system with n synchronous generators and n_l loads connected by a given topology. Each synchronous generator is modeled by a second-order swing equation, and each bus is modeled by two algebraic equations for active and reactive power balance. We convert this differential-algebraic model to a completely differential model using standard techniques of Kron reduction, and arrive at a linearized state variable model for the n-machine system as:

$$\begin{bmatrix} \Delta\dot{\delta}(t) \\ \Delta\dot{\omega}(t) \end{bmatrix} = \underbrace{\left[\begin{array}{c|c} \mathbf{0}_{n\times n} & \omega_s I_n \\ \hline \mathcal{M}^{-1}L & -\mathcal{M}^{-1}\mathcal{D} \end{array}\right]}_{A} \begin{bmatrix} \Delta\delta(t) \\ \Delta\omega(t) \end{bmatrix},$$

$$\mathbf{y}(t) = \mathrm{col}(y_p(t)), \quad \text{for } \; p = 1, \dots, P, \tag{1}$$

where $\Delta\delta = [\Delta\delta_1 \;\cdots\; \Delta\delta_n]^T$, $\Delta\omega = [\Delta\omega_1 \;\cdots\; \Delta\omega_n]^T$, $\mathcal{M} = \mathrm{diag}(M_1, \dots, M_n)$, and $\mathcal{D} = \mathrm{diag}(D_1, \dots, D_n)$, with $\Delta\delta_i$, $\Delta\omega_i$, M_i, and D_i being the small-signal angle deviation, the small-signal frequency deviation, inertia, and mechanical damping of the i^{th} generator, respectively. I_n is the $(n \times n)$ identity matrix, and ω_s is the synchronous speed of the system. The $(n \times n)$ matrix L is the Laplacian matrix whose elements, neglecting the line resistances, are defined as

$$[L]_{ij} = \frac{E_i E_j}{x_{ij}} \cos\bigl(\delta_{i0} - \delta_{j0}\bigr), \quad [L]_{ii} = -\sum_j [L]_{ij} \tag{2}$$

where x_{ij} is the equivalent reactance between generators i and j in the Kron-reduced form, E_i is the internal voltage and δ_{i0} is the equilibrium angle of the i^{th} generator. We consider the output vector $\mathbf{y}(t)$ to be a set of phase angle and frequency measurements $y_p(t)$, $p = 1, \dots, P$, measured by PMUs at p designated buses. Other outputs such as bus voltages and currents may also be considered but we restrict our analysis to phase angles and frequencies only. The eigenvalues of A are denoted by $(-\sigma_k \pm j\Omega_k)$, $(j = \sqrt{-1})$, where Ω_k and $\sigma_k > 0$ are the frequency and damping factor of the k^{th} mode for $k = 1, \dots, 2n$, respectively. Our objective is to estimate these $2n$ eigenvalues of A from $\mathbf{y}(t)$ in a distributed fashion using multiple computational resources. For this purpose, we next describe how the commonly used Prony algorithm for modal estimation can be cast as a distributed optimization problem. We first recall the centralized Prony algorithm, and thereafter reformulate it as a distributed algorithm.

B. Distributed Prony With Synchronous ADMM [2]

A generic expression for the solution of $y_p(t)$ in (1) can be written as

$$y_p(t) = \sum_{k=1}^{2n} r_{p,k} e^{(-\sigma_k + j\Omega_k)t} + r_{p,k}^{*} e^{(-\sigma_k - j\Omega_k)t}. \tag{3}$$

Each component in the RHS of (3) is referred to as a *mode*. σ_k is the damping, Ω_k is the frequency of the k^{th} mode, and r_{pk} is the residue of the k^{th} mode reflected in the p^{th} output. Sampling $y_p(t)$ with a uniform sampling period of T, a generic

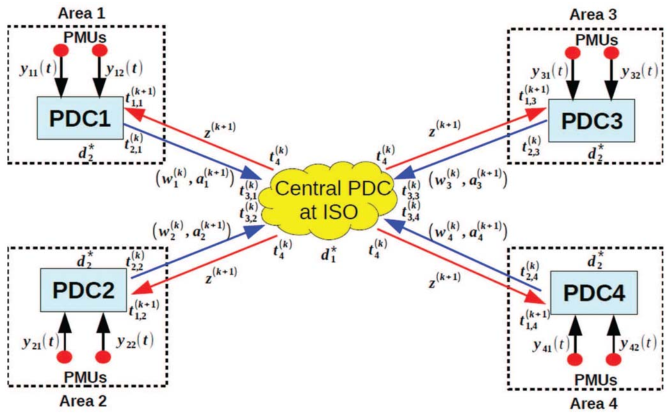


Fig. 1. Distributed architecture for a 4-area power system network.

expression for the z-transform of $y_p(m) \triangleq y_p(t)|_{t=mT}$, ($m = 0, 1, \ldots, M$), can be written as

$$Y_p(z) = \frac{b_{p,0} + b_{p,1}z^{-1} + b_{p,2}z^{-2} + \cdots + b_{p,2n}z^{-2n}}{1 + a_1z^{-1} + a_2z^{-2} + \cdots + a_{2n}z^{-2n}}, \quad (4)$$

where a's and b's are constant coefficients of the characteristic polynomial and the zero polynomial, respectively, assuming that the incoming small-signal disturbance input can be modeled as a unit impulse. The roots of the characteristic polynomial will give us the discrete-time poles of the system. One may, therefore, first estimate the coefficient vector $\mathbf{a} : \{a_1, \ldots, a_{2n}\}$, compute the discrete-time poles, and finally convert them to the continuous-time poles to obtain σ_k and Ω_k, for $k = 1, \ldots, 2n$, using the following algorithm, often referred to as the Prony algorithm [6]:

Step 1. Taking the inverse z-transform of (4) one may write

$$\underbrace{\begin{bmatrix} y_p(2n) \\ y_p(2n+1) \\ \vdots \\ y_p(2n+\ell) \end{bmatrix}}_{\mathbf{c}_p} = \underbrace{\begin{bmatrix} y_p(2n-1) & \cdots & y_p(0) \\ y_p(2n) & \cdots & y_p(1) \\ \vdots & & \vdots \\ y_p(2n+\ell-1) & \cdots & y_p(\ell) \end{bmatrix}}_{\mathbf{H}_p} \underbrace{\begin{bmatrix} -a_1 \\ -a_2 \\ \vdots \\ -a_{2n} \end{bmatrix}}_{\mathbf{a}} \quad (5)$$

where ℓ is an integer satisfying $2n+\ell \leq M-1$. Concatenating $\mathbf{c}_p$ and $\mathbf{H}_p$ in (5) for $p = 1, \ldots, P$, one can find $\mathbf{a}$ by solving a least-squares (LS) problem

$$\min_{\mathbf{a}} \frac{1}{2} \left\| \begin{bmatrix} \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_P \end{bmatrix} \mathbf{a} - \begin{bmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_P \end{bmatrix} \right\|^2, \quad (6)$$

where $\|\cdot\|$ denotes the 2-norm of a vector.

Step 2. Find the roots of the discrete-time characteristic polynomial, say denoted by z_k, $k = 1, \ldots, 2n$. Then, the desired eigenvalues λ_k are equal to $\ln(z_k)/T$.

The LS problem (6) can be reformulated as a global consensus problem over a network of N computational areas spanning the power grid. An example with four areas is shown in Fig. 1. We assume each area to be equipped with one local PDC (located at its control center) as shown in the figure. These PDCs receive local PMU measurements, run a local LS using these measurements, and then share the estimated parameters with a supervisory PDC at the ISO. For convenience, we will refer to the PDCs inside the areas as 'local PDC' and the PDC at the ISO as 'central PDC' as indicated in Fig. 1. The problem (6) then can be rewritten as

$$\min_{\mathbf{a}_1,\ldots,\mathbf{a}_N,\mathbf{z}} \sum_{i=1}^{N} \frac{1}{2} \left\| \hat{\mathbf{H}}_i \mathbf{a_i} - \hat{\mathbf{c}}_\mathbf{i} \right\|^2, \quad \text{s.t} \quad \mathbf{a_i} - \mathbf{z} = \mathbf{0}, \quad (7)$$

for $i = 1, \ldots, N$, where, $\hat{\mathbf{H}}_i = \text{col}(\mathbf{H}_p)$, $\hat{\mathbf{c}}_i = \text{col}(c_p)$, $p = 1, \ldots, P_i$ (where P_i is the number of sensors in Area i), $\mathbf{a}_i$ is the vector of the *primal variables*, and $\mathbf{z}$ is the *global consensus variable*. The value of P_i may be different in each Area i. The global consensus solution of (7) is achieved when the local estimates of the N regional PDCs, $\mathbf{a}_i$, $\forall\ i = 1, \ldots, N$, reach the same value. To solve (7) in a distributed way, we use ADMM [8], which reduces to the following set of recursive updates by using an *augmented Lagrangian* (for details, please refer to [2]):

$$\mathbf{w}_i^{(k)} = \mathbf{w}_i^{(k-1)} + \rho\left(\mathbf{a}_i^{(k)} - \mathbf{z}^{(k)}\right), \quad (8a)$$

$$\mathbf{a}_i^{(k+1)} = \left(\left(\hat{\mathbf{H}}_i^{(k)}\right)^T \hat{\mathbf{H}}_i^{(k)} + \rho\mathbf{I}\right)^{-1} \times \left(\left(\hat{\mathbf{H}}_i^{(k)}\right)^T \hat{\mathbf{c}}_i^{(k)} - \mathbf{w}_i^{(k)} + \rho\mathbf{z}^{(k)}\right), \quad (8b)$$

$$\mathbf{z}^{(k+1)} = \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{a}_i^{(k+1)} + (1/\rho)\mathbf{w}_i^{(k)}\right), \quad (8c)$$

where $\mathbf{w}_i^{(k)}$ is the vector of the *dual variables* or the Lagrange multipliers associated with (7) at iteration k, and $\rho > 0$ denotes the *penalty factor*. $\hat{\mathbf{H}}_i^{(0)} = \hat{\mathbf{H}}_i$ as defined in (7), and new PMU measurements are added to this matrix with every subsequent iteration. Alternatively, one may also construct $\hat{\mathbf{H}}_i^{(k)}$ over a sliding window of measurements to maintain a fixed matrix size. In [2] we developed the cyber-physical architecture by which local PDCs and the central PDC can exchange information between each other for executing (8). We summarize that architecture as follows. Consider the k^{th} iteration. Referring to Fig. 1: **Step 1)** at time $t_{1,i}^{(k)}$, any local PDC i runs the dual-primal update to gain ($\mathbf{w}_i^{(k)}$, $\mathbf{a}_i^{(k+1)}$) using (8a) and (8b) after receiving the consensus variable $\mathbf{z}^{(k)}$ from the central PDC; **Step 2)** at time $t_{2,i}^{(k)}$, the local PDC i transmits updated dual-primal variable set ($\mathbf{w}_i^{(k)}$, $\mathbf{a}_i^{(k+1)}$) to central PDC; **Step 3)** at time $t_{3,i}^{(k)}$, the central PDC calculates the consensus variable $\mathbf{z}^{(k+1)}$ using (8c); **Step 4)** at time $t_{4,i}^{(k)}$, the central PDC broadcasts $\mathbf{z}^{(k+1)}$ to the local PDCs in each area for their next update. The underlying assumption here is that every update in (8) can be performed by the local PDCs in parallel, i.e., they are all synchronized with each other. The algorithm (8) is, therefore, referred to as synchronous ADMM (S-ADMM).

In practice, however, the communication between the local PDCs and the central PDC will always involve communication delays, thereby leading to asynchrony in message arrivals at all PDCs. The timing diagram under that condition, as shown in Fig. 2, will consist of three main delay-sensitive components: (1) local PMU measurements stream in real-time to the

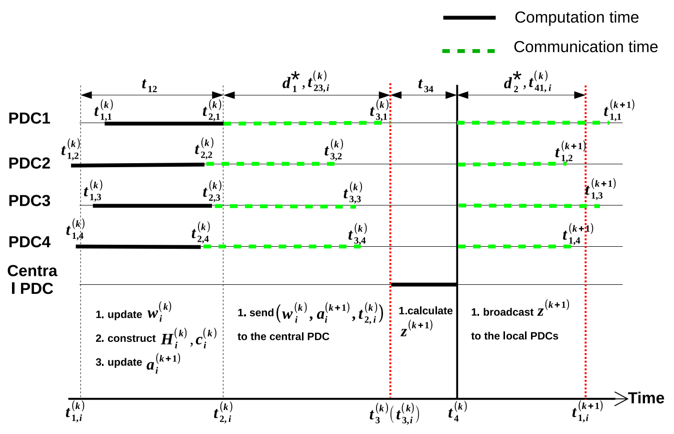

Computation time
Communication time
t_{12}
$d_1^*, t_{23,i}^{(k)}$
t_{34}
$d_2^*, t_{41,i}^{(k)}$
PDC1
PDC2
PDC3
PDC4
Central PDC
$t_{1,1}^{(k)}$ $t_{2,1}^{(k)}$ $t_{3,1}^{(k)}$ $t_{1,1}^{(k+1)}$
$t_{1,2}^{(k)}$ $t_{2,2}^{(k)}$ $t_{3,2}^{(k)}$ $t_{1,2}^{(k+1)}$
$t_{1,3}^{(k)}$ $t_{2,3}^{(k)}$ $t_{3,3}^{(k)}$ $t_{1,3}^{(k+1)}$
$t_{1,4}^{(k)}$ $t_{2,4}^{(k)}$ $t_{3,4}^{(k)}$ $t_{1,4}^{(k+1)}$
1. update $w_i^{(k)}$
2. construct $H_i^{(k)}, c_i^{(k)}$
3. update $a_i^{(k+1)}$
1. send $\left(w_i^{(k)}, a_i^{(k+1)}, t_{2,i}^{(k)}\right)$ to the central PDC
1.calculate $z^{(k+1)}$
1. broadcast $z^{(k+1)}$ to the local PDCs
Time
$t_{1,i}^{(k)}$ $t_{2,i}^{(k)}$ $t_3^{(k)}(t_{3,i}^{(k)})$ $t_4^{(k)}$ $t_{1,i}^{(k+1)}$

Fig. 2. Timing diagram for Asynchronous ADMM.

local PDCs. Since this communication happens over a private network, we ignore this communication delay throughout this paper; (2) the green dash lines between time $t_{2,i}^{(k)}$ and time $t_{3,i}^{(k)}$ at any iteration k show that the local estimates $(\mathbf{w}_i^{(k)}, \mathbf{a}_i^{(k+1)})$ arrive at the central PDC at different instants $t_{3,i}^{(k)}, i = 1, \ldots, N$ due to variable uplink delays; (3) the green dash lines between time $t_4^{(k)}$ and time $t_{1,i}^{(k+1)}$ show that the consensus variable $\mathbf{z}^{(k+1)}$ arrives at different instants $t_{1,i}^{(k+1)}, i = 1, \ldots, N$ at different local PDCs due to variable downlink delays. The resulting algorithm is referred to as asynchronous-ADMM or A-ADMM [10]. One trivial way to counteract the asynchrony would be to force all PDCs to wait till they receive every scheduled message at every iteration. This, however, can lead to unacceptably slow convergence times, and, depending on network congestion, can even turn out to be very risky in case any message gets lost or delayed for an uncertain period of time. Instead, we wish to counteract asynchrony by defining a set of *flexible deadlines* for message arrival in every PDC, and accordingly by modifying the update rules in (8) based on only those messages that respect these deadlines. In order to understand how these deadlines should be constructed in accordance to the network traffic, we first develop a probability distribution model for the network delays.

III. Delay Model for Wide-Area Communication

Following [20], we model the stochastic end-to-end delay experienced by a message between the central PDC and local PDCs in terms of three components: the minimum deterministic delay, denoted by m; the Internet traffic delay with Probability Density Function (PDF), denoted by ϕ_1; and the router processing delay with PDF, denoted by ϕ_2. Then, the PDF of the total delay at any time t is given as

$$\phi(t) = p\phi_2(t) + (1-p)\phi_1(t) * \phi_2(t), \qquad t \geq 0, \tag{9}$$

with $\phi_1(t) * \phi_2(t) = \int_0^t \phi_2(u)\phi_1(t-u)du$. Here p is the probability of open period of the path with no Internet traffic, and the router processing delay can be well approximated by a Gaussian density function $\phi_2(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}$, where $\mu > m$. The Internet traffic delay is modeled by an alternating renewal process with exponential closure period when the Internet traffic is on, with the PDF $\phi_1(t) = \lambda e^{-\lambda t}$, where λ^{-1} models the mean length of the closure period. The benchmark value of all parameters of this model are set as: $p = 0.58, \lambda = 1.39, \mu = 5.3, \sigma = 0.078$, following [20].

We next derive the Cumulative Distribution Function (CDF) of the delay model. First, (9) can be rewritten as

$$\phi(t) = \frac{p}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} + \frac{\lambda(1-p)}{\sigma\sqrt{2\pi}} e^{-\lambda t} \int_0^t e^{\lambda s - \frac{(s-\mu)^2}{2\sigma^2}} ds. \tag{10}$$

We rewrite the integral part of (10) by using the error function $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$, to get

$$\begin{aligned}\phi(t) = {} & \frac{p}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} \\ & + \frac{\lambda(1-p)}{2} e^{\left(\frac{1}{2}\lambda^2\sigma^2 + \mu\lambda\right)} e^{-\lambda t} \mathrm{erf}\left(\frac{t-\lambda\sigma^2-\mu}{\sqrt{2}\sigma}\right) \\ & + \frac{\lambda(1-p)}{2} e^{\left(\frac{1}{2}\lambda^2\sigma^2 + \mu\lambda\right)} \mathrm{erf}\left(\frac{\lambda\sigma^2+\mu}{\sqrt{2}\sigma}\right) e^{-\lambda t}. \end{aligned} \tag{11}$$

By using the partial integral method and the first derivative of the error function, $\frac{d}{ds}\mathrm{erf}(s) = \frac{2}{\sqrt{\pi}} e^{-s^2}$, we derive the CDF of the delay model as

$$\begin{aligned} P(t) = {} & \int_{-\infty}^t \phi(s) ds = \frac{1}{2}\left[\mathrm{erf}\left(\frac{\mu}{\sqrt{2}\sigma}\right) + \mathrm{erf}\left(\frac{t-\mu}{\sqrt{2}\sigma}\right)\right] \\ & + \frac{(p-1)}{2} e^{\left(\frac{1}{2}\lambda^2\sigma^2 + \mu\lambda\right)} e^{-\lambda t} \\ & \times \left[\mathrm{erf}\left(\frac{\lambda\sigma^2+\mu}{\sqrt{2}\sigma}\right) + \mathrm{erf}\left(\frac{t-\lambda\sigma^2-\mu}{\sqrt{2}\sigma}\right)\right]. \end{aligned} \tag{12}$$

Random delays from this CDF will next be imposed on the communication links to emulate A-ADMM.

IV. Proposed A-ADMM Strategies

A. Strategy I

In this strategy PDCs *skip* messages that do not arrive within a chosen deadline. We define two deadlines or *delay thresholds*, namely $d_1^* > 0$ and $d_2^* > 0$ in milliseconds, for the uplink and downlink delays, respectively. Without loss of generality, we assume that the counting of these deadlines start from the instant at which any PDC *sends out* any message at any iteration. For simplicity of notations, we also assume that every local PDC is assigned the same threshold d_2^* although same exact analyses will hold for different threshold values at different PDCs. Following the timing diagram in Fig. 2, if any local update $(\mathbf{w}_i^{(k)}, \mathbf{a}_i^{(k+1)})$ does not arrive at the central PDC within time d_1^*, the central PDC skips them, and computes the consensus variable $\mathbf{z}^{(k+1)}$ in (8c) simply as,

$$\mathbf{z}^{(k+1)} = \frac{1}{\left|S_1^{(k)}\right|} \sum_{i \in S_1^{(k)}} \left(\mathbf{a}_i^{(k+1)} + (1/\rho)\mathbf{w}_i^{(k)}\right). \tag{13}$$

where $S_1^{(k)}$ is the index set of local PDCs whose messages arrive on time, and $S_2^{(k)}$ is the index set for PDCs that can receive $\mathbf{z}^{(k+1)}$ within time, at the k^{th} iteration.

Algorithm 1 A-ADMM With Strategy I

```
procedure CentralPDC($\epsilon$)
    initialize: $k = 1$, $d_1^*$
    repeat
        repeat
            wait
            receive updates ($\mathbf{w}_i^{(k)}$, $\mathbf{a}_i^{(k+1)}$)
        until timer $\geq d_1^*$ or all updates received
        update $\mathbf{z}^{(k+1)}$ by (13)
        broadcast $\mathbf{z}^{(k+1)}$ to all local PDCs
        $k \leftarrow k + 1$
    until $\Delta\mathbf{z} = ||\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}|| \leq \epsilon$
end procedure
procedure LocalPDC $i$
    initialize: $k = 1$, $\mathbf{w}_i^{(0)} = 0$, $\mathbf{a}_i^{(0)} = 1$, $d_2^*$;
    repeat
        repeat
            wait
        until timer $\geq {d_2}^*$, or receive $\mathbf{z}^{(k)}$
        update $y_i(k)$, $\hat{\mathbf{H}}_i^{(k)}$, $\hat{\mathbf{c}}_i^{(k)}$ by (5)
        if global update received then
            update ($\mathbf{w}_i^{(k)}$, $\mathbf{a}_i^{(k+1)}$) by (8a), (8b)
        else
            $\mathbf{w}_i^{(k)} = \mathbf{w}_i^{(k-1)}$; $\mathbf{a}_i^{(k+1)} = \mathbf{a}_i^{(k)}$
        end if
        send ($\mathbf{w}_i^{(k)}$, $\mathbf{a}_i^{(k+1)}$) to the central PDC
        $k \leftarrow k + 1$
    until termination
end procedure
```

On the other hand, if any local PDC i does not receive $\mathbf{z}^{(k)}$ from the central PDC within the delay threshold d_2^*, it skips the update (8a)-(8b) altogether, and simply returns the local estimate ($\mathbf{w}_i^{(k-1)}$, $\mathbf{a}_i^{(k)}$) (i.e., the estimate from the previous iteration) to the central PDC. For a practical Internet model, the skipping strategy is most suitable for scenarios that involve exceptionally long delays or possible packet loss. The algorithm for this strategy is listed in Algorithm 1.

B. Strategy II

In this strategy PDCs use internal memory to replace messages that do not arrive within the respective deadlines with their values from previous iteration. Memorized data from PDCs, for example, can be stored and retrieved via a distributed storage service [21]. Following the timing diagram in Fig. 2, if any local update ($\mathbf{w}_i^{(k)}$, $\mathbf{a}_i^{(k+1)}$) does not arrive at the central PDC within time d_1^*, the central PDC computes the consensus variable $\mathbf{z}^{(k+1)}$ in (8c) as

$$\mathbf{z}^{(k+1)} = \frac{1}{N}\left(\sum_{i\in S_1^{(k)}}\left(\mathbf{a}_i^{(k+1)} + \frac{\mathbf{w}_i^{(k)}}{\rho}\right) + \sum_{i\notin S_1^{(k)}}\left(\mathbf{a}_i^{(l_i+1)} + \frac{\mathbf{w}_i^{(l_i)}}{\rho}\right)\right), \tag{14}$$

where $l_i \in \{k-1, k-2, k-3, \ldots\}$ denotes the index of the *latest* message that arrived successfully at the central PDC from the i^{th} local PDC. Similarly, if any local PDC $i, i \notin S_2^{(k)}$ does not receive $\mathbf{z}^{(k)}$ within its deadline, then it updates (8a)-(8b) as

$$\mathbf{w}_i^{(k)} = \mathbf{w}_i^{(k-1)} + \rho\left(\mathbf{a}_i^{(k)} - \mathbf{z}_i^{(l_i)}\right), \tag{15a}$$

$$\mathbf{a}_i^{(k+1)} = \left(\left(\hat{\mathbf{H}}_i^{(k)}\right)^T\hat{\mathbf{H}}_i^{(k)} + \rho\mathbf{I}\right)^{-1} \times \left(\left(\hat{\mathbf{H}}_i^{(k)}\right)^T\hat{\mathbf{c}}_i^{(k)} - \mathbf{w}_i^{(k)} + \rho\mathbf{z}_i^{(l_i)}\right). \tag{15b}$$

where $l_i \in \{k-1, k-2, k-3, \ldots\}$ denotes the index of the *latest* message that arrived successfully at the i^{th} local PDC. Note that $l_i \neq l_j$, in general. However, since every PMU data is time-stamped by GPS, the PDCs will have the ability to stamp or decipher the iteration number corresponding to any message they send or receive.

1) Adjustment in Stopping Criterion: The usual practice in S-ADMM is to keep track of $\Delta\mathbf{z} = ||\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}||$, and terminate the algorithm once $\Delta\mathbf{z}$ falls below a chosen tolerance. This step needs to be modified for A-ADMM with Strategy II. The reason is as follows. In case every local update ($\mathbf{w}_i^{(k)}$, $\mathbf{a}_i^{(k+1)}$), $i = 1, \ldots, N$ arrive after time d_1^* at the central PDC, then the newest update of $\mathbf{z}^{(k+1)}$ will be computed from its stored latest update, which is exactly same as $\mathbf{z}^{(k)}$ causing $\Delta\mathbf{z}$ to be zero. This may force the algorithm to terminate prematurely. To alleviate impact of delayed updates, we first modify the stopping criteria to

$$\Delta\mathbf{z} = \frac{1}{\left|S_1^{(k)}\right|}\sum_{i\in S_1^{(k)}}\left|\left|\left(\mathbf{a}_i^{(k+1)} - \mathbf{a}_i^{(l_i)}\right) + (1/\rho)\left(\mathbf{w}_i^{(k)} - \mathbf{w}_i^{(l_i-1)}\right)\right|\right|. \tag{16}$$

where ($\mathbf{w}_i^{(l_i-1)}$, $\mathbf{a}_i^{(l_i)}$) is the latest local update received from the i^{th} local PDC at the $(k-1)^{th}$ iteration at the central PDC. The tolerance factor, denoted by ϵ, is set to be a constant value for every iteration (10^{-6} in our simulations). Another way would be to adjust ϵ adaptively according to the number of non-delayed estimates. It chooses smaller values of ϵ for smaller values of $|S_1^{(k)}|$. That is, we define a vector $\epsilon = [\epsilon_1, \ldots, \epsilon_N]$ such that for $|S_1^{(k)}| = 1$ we use ϵ_1, for $|S_1^{(k)}| = 2$ we use ϵ_2, and so on, where $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_N$, [9]. The different steps of A-ADMM with Strategy II are shown in Algorithm 2.

C. Conditions for Convergence

A valid question that may arise at this point is under what conditions will Strategy 1 and Strategy 2 guarantee convergence of A-ADMM. In this section we derive these conditions, first by expressing the consensus model (8) in a structured discrete-time state-space form, and thereafter applying small-gain theorem [22]. For simplicity of notations, and without any loss of generality, we assume the matrices $\hat{\mathbf{H}}_i$ and $\hat{\mathbf{c}}_i$ to be time-invariant. Denoting $A_i = ((\hat{\mathbf{H}}_i)^{\mathrm{T}}\hat{\mathbf{H}}_i + \rho\mathbf{I}_{2n})^{-1}$, after a few calculations one can easily derive that (8)

Algorithm 2 A-ADMM With Strategy II

```
1: procedure CentralPDC($\epsilon$)
2:     initialize: $k = 1$, $d_1^*$
3:     repeat
4:         repeat
5:             wait
6:             receive updates $(\mathbf{w}_i^{(k)}, \mathbf{a}_i^{(k+1)})$
7:         until timer $\geq d_1^*$ or all updates received
8:         update $\mathbf{z}^{(k+1)}$ by (14), update $\Delta\mathbf{z}$ by (16)
9:         broadcast $\mathbf{z}^{(k+1)}$ to all local PDCs
10:        $k \leftarrow k+1$
11:    until $\Delta\mathbf{z} \leq \epsilon$
12: end procedure
13: procedure LocalPDC $i$
14:    initialize: $k = 1$, $\mathbf{w}_i^{(0)} = 0$, $\mathbf{a}_i^{(0)} = 1$, $d_2^*$;
15:    repeat
16:        repeat
17:            wait
18:        until timer $\geq d_2^*$, or receive $\mathbf{z}^{(k)}$
19:        update $y_i(k)$, $\hat{\mathbf{H}}_i^{(k)}$, $\hat{\mathbf{c}}_i^{(k)}$ by (5)
20:        if global update received then
21:            update $(\mathbf{w}_i^{(k)}, \mathbf{a}_i^{(k+1)})$ by (8a), (8b)
22:        else
23:            update $(\mathbf{w}_i^{(k)}, \mathbf{a}_i^{(k+1)})$ by (15a), (15b)
24:        end if
25:        send $(\mathbf{w}_i^{(k)}, \mathbf{a}_i^{(k+1)})$ to the central PDC
26:        $k \leftarrow k+1$
27:    until termination
28: end procedure
```

can be written as

$$\mathbf{a}^{(k+1)} = \underbrace{\begin{bmatrix} \mathbf{L}_{11} & \mathbf{L}_{12} \\ \mathbf{I} & 0 \end{bmatrix}}_{L_0} \mathbf{a}^{(k)}, \tag{17}$$

where, $\mathbf{a}^{(k)} = [\mathbf{a}_1^{(k)^T} \ldots \mathbf{a}_N^{(k)^T}\, \mathbf{a}_1^{(k-1)^T} \ldots \mathbf{a}_N^{(k-1)^T}]$, and the submatrices in the RHS are given by

$$\mathbf{L}_{11} = \begin{bmatrix} I - \rho A_1 + \frac{2}{N}\rho A_1 & \ldots & \frac{2\rho A_1}{N} \\ \frac{2\rho A_2}{N} & \ldots & \frac{2\rho A_2}{N} \\ \vdots & \ddots & \ddots \\ \frac{2\rho A_N}{N} & \frac{2\rho A_N}{N} & I - \rho A_N + \frac{2}{N}\rho A_N \end{bmatrix} \tag{18}$$

$$\mathbf{L}_{12} = \begin{bmatrix} -\frac{\rho A_1}{N} & \ldots & -\frac{\rho A_1}{N} \\ \vdots & \ddots & \vdots \\ -\frac{\rho A_N}{N} & \ldots & -\frac{\rho A_N}{N} \end{bmatrix}. \tag{19}$$

It can be verified that every row of the matrix L_0 sums to one, and, hence, the system has a consensus manifold defined at $\mathbf{a}_1^{(k)} = \mathbf{a}_2^{(k)} = \cdots = \mathbf{a}_N^{(k)}$ as $k \to \infty$ (for details please see [23]).

Next, to model the delays in A-ADMM, we define N binary numbers $\{d_1^k, d_2^k, \ldots, d_N^k\}$ for any iteration k where d_i^k expresses whether the estimate of the local PDC i at iteration k arrives at the central PDC within the threshold d_1^* or not:

$$d_i{}^k = \begin{cases} 1, & arrive \\ 0, & delay. \end{cases}$$

Using Strategy 1 for the averaging step, it can then be easily derived that the state-space representation of A-ADMM takes the form

$$\mathbf{a}^{(k+1)} = \Big(L_0 + P\,Q^k\,R\Big)\mathbf{a}^{(k)} \tag{20}$$

where, L_0 is the stable state matrix for the delay-free model (17), and the matrices P, Q^k and R are given as

$$P = \begin{bmatrix} C_1 & C_2 \\ 0 & 0 \end{bmatrix},\; Q^k = \begin{bmatrix} F^k & \frac{1}{2N}I \\ \frac{2}{N}I & F^{k-1} \end{bmatrix},\; R = I \tag{21}$$

where,

$$C_1 = \begin{bmatrix} 2\rho A_1 & 2\rho A_1 & \ldots & 2\rho A_1 \\ 2\rho A_2 & 2\rho A_2 & \ldots & 2\rho A_2 \\ \vdots & \vdots & \ddots & \vdots \\ 2\rho A_N & 2\rho A_N & \ldots & 2\rho A_N \end{bmatrix}, \tag{22}$$

$$C_2 = \begin{bmatrix} -\rho A_1 & -\rho A_1 & \ldots & -\rho A_1, \\ -\rho A_2 & -\rho A_2 & \ldots & -\rho A_2 \\ \vdots & \vdots & \ddots & \vdots \\ -\rho A_N & -\rho A_N & \ldots & -\rho A_N, \end{bmatrix}, \tag{23}$$

$$F^k = \text{diag}\left\{ \frac{d_1^k}{\sum_{i=1}^N d_i^k},\; \frac{d_2^k}{\sum_{i=1}^N d_i^k},\; \ldots,\; \frac{d_N^k}{\sum_{i=1}^N d_i^k} \right\}, \tag{24}$$

$$F^{k-1} = \text{diag}\left\{ \frac{d_1^{k-1}}{\sum_{i=1}^N d_i^{k-1}},\; \frac{d_2^{k-1}}{\sum_{i=1}^N d_i^{k-1}},\; \ldots,\; \frac{d_N^{k-1}}{\sum_{i=1}^N d_i^{k-1}} \right\}. \tag{25}$$

From small-gain theorem [22] it, therefore, follows that A-ADMM with Strategy 1, given by the model (20), is convergent if

$$\left|\left|P\,(zI - L_0)^{-1}\,R\right|\right|_\infty < \frac{1}{\gamma} \tag{26}$$

where $||Q^k|| < \gamma$, i.e., γ is the maximum upper bound on the norm of the delay-matrix Q^k over all iterations k. If $d_i^k = 1$ for every i, then $PQ^kR = 0$ for every k, and (20) retains its stability and convergence. Equation (26), therefore, models the trade-off in stability for A-ADMM when d_i^k deviates from unity for a certain set of PDCs i. A similar state-space form and small-gain condition can be derived for A-ADMM with Strategy 2 using the exact same approach as above, but we skip its derivation here for space limitations.

D. Strategy II With Gradient Update

It is well-known that ADMM algorithms typically show steep initial convergence to a ball around the optimal point, and slow convergence towards the optimal thereafter [24], [25]. This fact motivates us to add a gradient update term to (15)

TABLE I
COMPARISON OF A-ADMM STRATEGIES W.R.T SENSITIVITY TO NETWORK CHARACTERISTICS

Strategy Type	Sensitivity to Packet Loss	Sensitivity to Packet Delay	Sensitivity to Delay Threshold
Strategy I - Skipping	high	high	high
Strategy II - Previous Update	high	low	medium
Strategy II with Gradient Method	medium	low	low

to improve convergence as:

$$\mathbf{w}_i^{(k)} = \mathbf{w}_i^{(k-1)} + \rho\Big(\mathbf{a}_i^{(k)} - \Big(\mathbf{z}^{(l_i)} + \gamma_i\Big(\mathbf{z}^{(l_i)} - \mathbf{z}^{(l_i-1)}\Big)\Big)\Big), \tag{27a}$$

$$\mathbf{a}_i^{(k+1)} = \Big(\Big(\hat{\mathbf{H}}_i^{(k)}\Big)^T \hat{\mathbf{H}}_i^{(k)} + \rho \mathbf{I}\Big)^{-1} \Big(\Big(\hat{\mathbf{H}}_i^{(k)}\Big)^T \hat{\mathbf{c}}_i^{(k)} - \mathbf{w}_i^{(k)} + \rho\Big(\mathbf{z}^{(l_i)} + \gamma_i\Big(\mathbf{z}^{(l_i)} - \mathbf{z}^{(l_i-1)}\Big)\Big)\Big), \tag{27b}$$

where, $l_i \in \{k-1, k-2, \dots\}$, and γ_i is the step size whose value can be flexibly adjusted to expedite convergence. That is, now the local PDC $i, i \notin S_2^{(k)}$ uses $\mathbf{z}^{(l_i)} + \gamma_i(\mathbf{z}^{(l_i)} - \mathbf{z}^{(l_i-1)})$ instead of $\mathbf{z}^{(l_i)}$ in its update equations. Since (8) is solving consensus, this gradient information can improve the closeness of $\mathbf{z}^{(k)}$ to $\mathbf{z}^{(k+1)}$ compared to $\mathbf{z}^{(l_i)}$. The update at the central PDC remains same as in (14).

Table I compares the three A-ADMM strategies in terms of their sensitivity to different communication bottlenecks.

E. Other Strategies

Other heuristic strategies can also be used to mitigate asynchrony by taking advantage of fact that (8) is a consensus problem. One such strategy can be to use spatial correlation between the estimates generated by the local PDCs at every iteration. Local estimates $\mathbf{a}_i^{(k)}$ and $\mathbf{a}_j^{(k)}$ for any two PDCs i and j, located at two different spatial locations in the grid, are likely to be close in their magnitudes over iteration k, and exactly equal as $k \to \infty$. The closeness at any given k, especially for smaller values of k, of course depends on the difference between the initial conditions $\mathbf{a}_i^{(0)}$ and $\mathbf{a}_j^{(0)}$, and individual convergence rates of the two PDCs depending on their network traffic. The idea, therefore, is to keep track of the correlation factors between every pair of local estimates at the central PDC. The correlation coefficient $\rho_{(\mathbf{a}_i^{(k)},\mathbf{a}_j^{(k)})}$ between $\mathbf{a}_i^{(k)}$ and $\mathbf{a}_j^{(k)}$ with expected values $\mu_{\mathbf{a}_i^{(k)}}$ and $\mu_{\mathbf{a}_j^{(k)}}$ and standard deviations $\sigma_{\mathbf{a}_i^{(k)}}$ and $\sigma_{\mathbf{a}_j^{(k)}}$ is defined as

$$\rho_{\left(\mathbf{a}_i^{(k)},\mathbf{a}_j^{(k)}\right)} = \frac{cov\Big(\mathbf{a}_i^{(k)}, \mathbf{a}_j^{(k)}\Big)}{\sigma_{\mathbf{a}_i^{(k)}}\sigma_{\mathbf{a}_j^{(k)}}} = \frac{E\Big[\Big(\mathbf{a}_i^{(k)} - \mu_{\mathbf{a}_i^{(k)}}\Big)\Big]E\Big[\Big(\mathbf{a}_j^{(k)} - \mu_{\mathbf{a}_j^{(k)}}\Big)\Big]}{\sigma_{\mathbf{a}_i^{(k)}}\sigma_{\mathbf{a}_j^{(k)}}}. \tag{28}$$

The central PDC computes the correlation matrix $\mathcal{C}^k$ as

$$\mathcal{C}^k = \begin{bmatrix} \rho_{\left(\mathbf{a}_1^{(k)},\mathbf{a}_1^{(k)}\right)} & \rho_{\left(\mathbf{a}_1^{(k)},\mathbf{a}_2^{(k)}\right)} & \cdots & \rho_{\left(\mathbf{a}_1^{(k)},\mathbf{a}_N^{(k)}\right)} \\ \rho_{\left(\mathbf{a}_2^{(k)},\mathbf{a}_1^{(k)}\right)} & \rho_{\left(\mathbf{a}_2^{(k)},\mathbf{a}_2^{(k)}\right)} & \cdots & \rho_{\left(\mathbf{a}_2^{(k)},\mathbf{a}_N^{(k)}\right)} \\ \vdots & \vdots & \cdots & \vdots \\ \rho_{\left(\mathbf{a}_N^{(k)},\mathbf{a}_1^{(k)}\right)} & \rho_{\left(\mathbf{a}_N^{(k)},\mathbf{a}_2^{(k)}\right)} & \cdots & \rho_{\left(\mathbf{a}_N^{(k)},\mathbf{a}_N^{(k)}\right)} \end{bmatrix}. \tag{29}$$

If at iteration $(k+1)$ any estimate $\mathbf{a}_i^{(k+1)}$ does not arrive on time, then the central PDC scans $\mathcal{C}^k$, and locates the index j such that $\rho_{(\mathbf{a}_i^{(k)},\mathbf{a}_j^{(k)})}$ has the highest magnitude among all entries in the i^{th} row of $\mathcal{C}^k$. If $\mathbf{a}_j^{(k+1)}$ has arrived on time, then it substitutes the missing value $\mathbf{a}_i^{(k+1)}$ by $\mathbf{a}_j^{(k+1)}$ instead of $\mathbf{a}_i^{(l_i+1)}$ as in (14). In other words, (14) now takes the form

$$\mathbf{z}^{(k+1)} = \frac{1}{N}\left(\sum_{i\in S_1^{(k)}}\left(\mathbf{a}_i^{(k+1)} + \frac{\mathbf{w}_i^{(k)}}{\rho}\right) + \sum_{j\notin S_1^{(k)}}\left(\mathbf{a}_{J_j}^{(k+1)} + \frac{\mathbf{w}_{J_j}^{(k)}}{\rho}\right)\right). \tag{30}$$

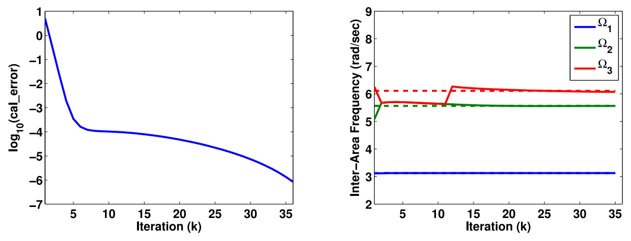
where $J_j = \arg\max_{i\in S_1^{(k)}} \mathcal{C}^k(j,i)$.

V. SIMULATION RESULTS

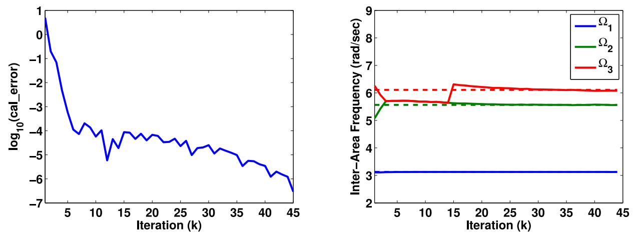
To verify the A-ADMM algorithms described in Section IV, we consider the IEEE 68-bus system. The system is divided into 4 areas, each with one local PDC and 3 PMUs. The simulated measurements of bus frequency are obtained using the Power Systems Toolbox (PST) nonlinear dynamics simulation routine. A three-phase fault is considered occurring at the line connecting buses 1 and 2. The fault starts at $t = 0.1$ sec, clears at bus 1 at $t = 0.15$ sec and at bus 2 at $t = 0.20$ sec. The measurements are downsampled and the sampling period T is increased up to 0.2 seconds. Our objective is to estimate the post-fault inter-area oscillation modes of the system using real-time PMU measurements of bus frequency. Since there are 16 generators, each with six states, our proposed algorithms should ideally solve a 96^{th}-order polynomial. However, offline analysis of the model revealed that choosing $2n = 40$ yields a satisfactory estimate of the inter-area modes as most of the residues for the high frequency modes are practically zero. The initial 10 samples (2 seconds) of the measurements are gathered before starting the optimization iterations. We set $\rho = 10^{-9}$, and $\epsilon = 10^{-6}$.

A. S-ADMM v.s. A-ADMM

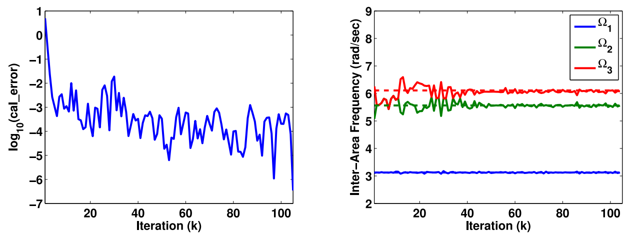
We first consider a fundamental comparison of S-ADMM and A-ADMM with Strategy I. From our delay model, we note that $P(X \le 5.67) = 0.8$, meaning that $d_1^* = d_2^* = 5.67$ ms



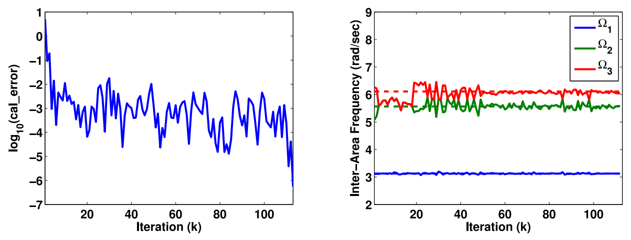
(a) S-ADMM



(b) Downlink Case of A-ADMM with Strategy I



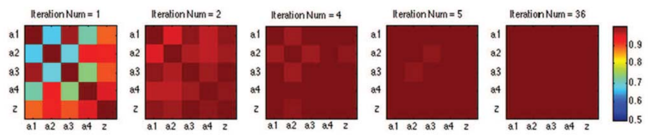
(c) Uplink Case of A-ADMM with Strategy I



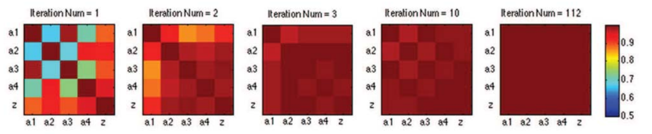
(d) Bi-link Case of A-ADMM with Strategy I

Fig. 3. Comparison of convergence and accuracy of Strategy I and S-ADMM.

will lead to 20% of the messages be delayed. Using these values, we simulate 1000 runs of the A-ADMM experiments with Strategy I, and plot the expected values of the convergence error in Fig. 3. The strategy is subdivided into three different cases, namely: 1) Downlink case, where we only consider those runs where downlink deadlines are missed, but uplink deadlines are always met; 2) Uplink case, where the reverse happens; and 3) Bi-link case, which is the usual A-ADMM with Strategy I. The left panel of Fig. 3 shows that compared to the smooth convergence of S-ADMM, the convergence of A-ADMM becomes jittery as more asynchrony is added. Setting the accuracy of estimation to be fixed at 99.5% or more, convergence rate clearly slows down from 36 iterations in S-ADMM to 45 for the Downlink case, 105 for the Uplink case, and 113 for general A-ADMM with bi-directional delay. The right panel of this figure shows the convergence and accuracy of the estimated values of the frequency of the

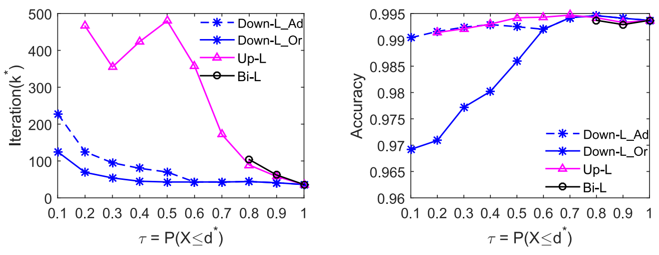


(a) S-ADMM

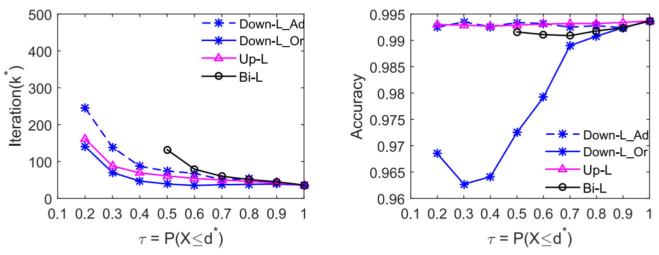


(b) Bi-link Case of A-ADMM with Strategy I

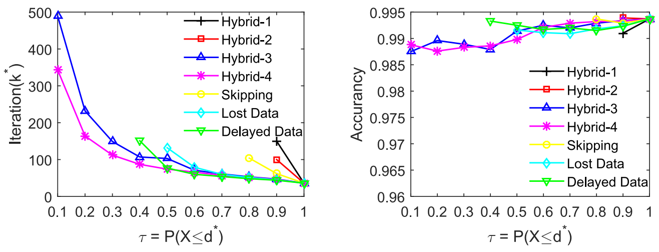
Fig. 4. Correlation analyses for S-ADMM v.s A-ADMM with Strategy I.



(a) Strategy I - Skipping



(b) Strategy II - Lost Data



(c) Strategy II - Delayed Data

Fig. 5. Comparison of convergence and accuracy of Strategy I and II w.r.t delay threshold.

three dominant modes. Fig. 4 shows the correlation plots of every local estimate and the consensus variable over different iterations, illustrating the relative convergence rates of the estimates at the five different PDCs.

B. Sensitivity of A-ADMM to Delay Thresholds

We next test the impact of d_1^* and d_2^* on the convergence and accuracy of the A-ADMM strategies. For each strategy, we run 200 runs for each different choice of the delay thresholds by using 200 pre-generated sets of delays using model (12). Fig. 5 and 6 show the relationships between different delay thresholds and the average convergence rate and accuracy.

TABLE II
SENSITIVITY OF GRADIENT UPDATE TO DELAY THRESHOLD

$\tau = P(X \leq d^*)$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.9
Lost Data - Downlink (k^*/α)	-	355/0.9914	218/0.9914	160/0.9916	132/0.9917	118/0.9916	108/0.9916	83/0.9913
Gradient Method (k^*/α)	1117/0.9897	317/0.9913	218/0.9914	162/0.9916	130/0.9916	113/0.9917	102/0.9918	77/0.9914
Step Size γ	-0.4	-0.2	0.001	0.015	0.01	0.9	0.8	1.9
Effective γ Range	-	[-0.7 0.1]	[-0.9 0.5]	[-1.2 0.9]	[-1.6 1]	[-2.2 1.5]	[-2.5 2]	[-6 5]
Lost Data - Bi-link (k^*/α)	-	-	-	-	239/0.9917	170/0.9912	148/0.9918	100/0.9917
Gradient Method (k^*/α)	-	-	-	628/0.9911	230/0.9918	168/0.9912	133/0.9918	94/0.9917
Step Size γ	-	-	-	-0.3	-0.3	-0.3	0.5	1
Effective γ Range	-	-	-	-	[-1 0.3]	[-1.7 0.6]	[-1.8 1]	[-4.6 3.5]
Delayed Data - Downlink (k^*/α)	113/0.9915	116/0.9914	114/0.9914	118/0.9915	118/0.9914	111/0.9915	102/0.9912	85/0.9913
Gradient Method (k^*/α)	112/0.9916	107/0.9914	100/0.9915	89/0.9915	84/0.9914	77/0.9915	75/0.9913	78/0.9917
Step Size γ	0.01	0.2	0.35	0.6	0.8	0.9	1	1.8
Effective γ Range	[-0.3 0.12]	[-0.6 0.25]	[-1 0.45]	[-1.4 0.7]	[-1.8 0.8]	[-2 1.2]	[-2.5 1.7]	[-5 5.2]
Delayed Data - Bi-link (k^*/α)	-	-	-	-	130/0.9913	128/0.9913	119/0.9914	93/0.9913
Gradient Method (k^*/α)	-	-	-	196/0.9913	123/0.9914	105/0.9913	101/0.9914	85/0.9914
Step Size γ	-	-	-	-0.02	0.3	0.6	0.8	2.3
Effective γ Range	-	-	-	[-0.3 0.2]	[-1.2 0.5]	[-2 1]	[-2.5 1.5]	[-5.5 3.5]

1) Strategy I: From the left subfigure in Fig. 5(a), we see that in the Downlink case, the number of iterations increases slowly when the CDF τ of the delay threshold d_2^* decreases from 1 to 0.2. Even when $\tau = 0.1$, meaning that $\mathbf{z}^{(k)}$ only has 10% possibility to arrive by the deadline, the algorithm still converges in 100 iterations. However, in Uplink case, once $\tau = 0.1$, the algorithm faces numerical instability, and therefore diverges. Also, as τ decreases from 1 to 0.5, k^* increase dramatically up to 500, while it remains around the value of 400 for $\tau \in [0.2, 0.4]$. As expected, the bi-link case has the shortest stable range of $\tau \in [0.8, 1]$, and largest value of k^*, compared to the other two cases. If $\tau \leq 0.7$, the algorithm diverges irrespective of the choice of the deadlines, indicating that Strategy I is very sensitive to bi-directional delays. Estimation accuracy of 99.1% or more is maintained by adjusting ϵ of the stop criterion from 10^{-6} to 10^{-8} or 10^{-7}, as shown in the blue dash lines in Fig. 5(a). After the adjustment, the Downlink case still has much better convergence than the Uplink case, indicating that message-skipping impacts the averaging step far more than the local least-square update step.

2) Strategy II - Lost Data: Strategy II is further divided into two subcases - namely, lost data and delayed data. Lost data essentially implies that if any message does not arrive on time at any PDC it is considered to be unusable for future iterations. Fig. 5(b) shows that similar to Strategy I, the accuracy of this strategy drops dramatically from 99.08% to 96.27% when τ decreases from 0.8 to 0.2 in Downlink case. Even after adjustment of ϵ, it has worse convergence than Strategy I. This observation implies that if the local PDC i does not receive any message on time it is more advisable to skip the update rather than using out-of-date information about $\mathbf{z}^{(k)}$. For the Uplink case, however, this strategy has significantly better convergence and accuracy guarantees than Strategy I, especially for $\tau \in [0.2, 1]$, implying that when the probability of data loss is high, it is more advisable for the central PDC to use stored values of local updates rather than skipping them.

3) Strategy II - Delayed Data: This is the usual A-ADMM with Strategy II, where delayed data are used in future iterations. It has the best convergence and accuracy guarantees among the three representative cases described so far, partly because of zero packet loss rate, and partly due to limited maximum communication delay. In Downlink case, the blue line in the left subfigure of Fig. 5(c), indicates that the iteration number k^* has the lowest value for each given τ, compared to the blue lines of both Strategy I and Strategy II - Lost data before adjustment. In Uplink case, the magenta line shows smallest k^* in the same effective range of $\tau \in [0.2, 1]$. For the Bi-link case, this not only converges in the smallest number of iterations k^* for a given τ, but also has the longest effective range of $\tau \in [0.4, 1]$ guaranteeing numerical stability.

TABLE III
HYBRID CONTROL STRATEGIES

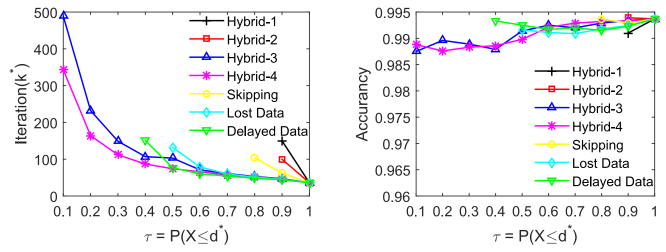
Strategy Type	Uplink	Downlink
Hybrid-1	Skipping	Lost Data
Hybrid-2	Skipping	Delayed Data
Hybrid-3	Lost Data	Skipping
Hybrid-4	Delayed Data	Skipping

4) Hybrid Strategies: So far we have considered cases where Strategy I and Strategy II are employed independently for the estimation. We next consider the cases when these two strategies are used in combinations. Four possible such hybrid strategies, shown in Table III, are considered. Fig. 6 shows the relationship between the delay threshold and the convergence and accuracy guarantees of these hybrid cases. Three main conclusions can be drawn from Fig. 6: (1) Hybrid-1 and Hybrid-2 have worse convergence because Strategy I is employed in the uplink communication, while Strategy II is applied to the downlink communication. When $\tau \leq 0.8$, they force the algorithm to start to diverge, as shown with black and red lines in the left subfigure of Fig. 6(a), respectively; (2) A magnified view of Fig. 6(b) shows that when the delay thresholds (d_1^*, d_2^*) are larger than 5.35 ms, namely $\tau \geq 0.5$, the standard Strategy II-Delayed Data has best convergence with accuracy above 99.13%; (3) Finally, when $\tau \in [0.1, 0.4]$, Hybrid-3 and Hybrid-4 still guarantee convergence, while both Strategy II-Lost data and Strategy-II Delayed data start to diverge. Therefore, if we need to set the delay thresholds to small values, Hybrid-3 is the best choice.

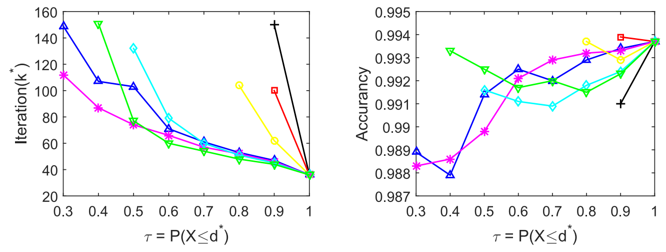
5) Strategy II With Gradient Update: This strategy is most effective when the initial conditions for the local updates

TABLE IV
COMPARISON OF CONVERGENCE OF A-ADMM AND A-DSM

Delay Type	A-ADMM			A-DSM		
	Skipping	Lost Data	Delayed Data	Skipping	Lost Data	Delayed Data
Bi-link	121	54	51	91009	107190	99635
Downlink	41	39	39	91445	91440	87217
Uplink	82	45	44	73844	91556	87274



(a) Hybrid Strategies



(b) Hybrid v.s. Standard

Fig. 6. Comparison of convergence and accuracy of hybrid strategies w.r.t delay threshold.

are noticeably different from each other. To demonstrate this method we set $\mathbf{a}_1^{(0)} = 0.01\mathbf{1}$, $\mathbf{a}_2^{(0)} = 0.1\mathbf{1}$, $\mathbf{a}_3^{(0)} = \mathbf{1}$, $\mathbf{a}_4^{(0)} = 10\mathbf{1}$, where $\mathbf{1}$ is a vector of ones with 40 rows (the number of unknown coefficients of the characteristic polynomial is considered to be 40). The convergence guaranteed by S-ADMM in this case is $k^* = 117$, and accuracy (on a scale of 1) is $\alpha = 0.9938$. Table II shows that the gradient method improves convergence k^* for relatively the same accuracy factor α by tuning the step size γ, considering $\gamma_i = \gamma$, $\forall i = 1, .., N$. The symbol '$-$' in the table means that the algorithm diverges. The stable range of $\tau \in [0.2, 0.9]$ for Downlink and $\tau \in [0.4, 0.9]$ for Bi-link is increased to $\tau \in [0.1, 0.9]$, and $\tau \in [0.5, 0.9]$, respectively, after the gradient update. Table II also shows the optimal value of γ for a given τ. It should be noted that when τ is large, i.e., when the delay threshold is large, then γ must be chosen to be large as well for optimal convergence. The range of stable γ, however, is increased in that condition.

The heuristic strategy involving spatial correlation analysis, as in (29), for this example shows comparable results to the above, especially when the initial conditions are scattered. However, we defer the details of those results to future for the sake of brevity and space limitation. Also, we note that since the basic S-ADMM problem involves only two simple steps of local least-squares and averaging, all of these proposed A-ADMM strategies are highly scalable with network size and the number of PMUs.

C. A-ADMM v.s. A-DSM

We next compare our A-ADMM algorithm with Distributed Subgradient Method (DSM), another commonly used distributed optimization algorithm [26], [27], in terms of the convergence and the sensitivity of convergence to Strategy I and II. For this purpose, we employ both strategies to DSM by setting $d_1^* = d_2^* = 5.67$ ms, and the accuracy factor to be 0.9914. The resulting algorithm is referred to as asynchronous-DSM or A-DSM. The simulation results are tabulated in Table IV, showing notably slower convergence for A-DSM compared to the A-ADMM for both strategies, each with three communication delay types. It should be noted, however, that ADMM needs more computation per iteration due to the computation of the matrix inverse. Regarding how these two algorithms are sensitive to Strategy I and II, we have three interesting observations: (1) In contrast to A-ADMM, Strategy I of A-DSM has better convergence rate in Uplink case than Downlink case, indicating that message-skipping impacts the local sub-gradient update step more than the central averaging step. (2) For both Uplink and Bi-link cases, Strategy I of A-DSM has the best convergence while the performance of Strategy II - Delayed Data is best for A-ADMM, as noted before. (3) Regarding Strategy II - Lost Data and Delayed Data, A-DSM shows similar convergence between Uplink and Downlink cases, implying that the use of stale messages has almost the same impact on both central averaging and local sub-gradient update steps.

VI. CONCLUSION

In this paper, we presented four cyber-physical estimation algorithms for power system oscillation monitoring using Synchrophasors. Our algorithms demonstrate how multitudes of geographically dispersed PMUs and PDCs can communicate with each other, and how the various binding factors in the network protocols can pose bottlenecks for their communication. We construct a suite of methods that one may choose to eliminate asynchrony in wide-area estimation problems. The results, thereby provide valuable insights and guidance in deploying future PMU and PDC infrastructures, not only for power systems but for any generic cyber-physical sensor network. Our future work will be to evaluate the sensitivity of the proposed algorithms to models of bandwidth allocation as well to data corruption and denial-of-service cyber-attacks.

REFERENCES

[1] A. G. Phadke and J. S. Thorp, *Synchronized Phasor Measurements and Their Applications*. New York, NY, USA: Springer, 2008.

[2] S. Nabavi, J. Zhang, and A. Chakrabortty, "Distributed optimization algorithms for wide-area oscillation monitoring in power systems using an inter-regional PMU-PDC architecture," *IEEE Trans. Smart Grid*, vol. 6, no. 5, pp. 2529–2538, Sep. 2015.

[3] R. Hasan, R. Bobba, and H. Khurana, "Analyzing NASPInet data flows," in *Proc. IEEE Power Syst. Conf. Expo.*, Seattle, WA, USA, 2009, pp. 1–6.

[4] J. J. Sanchez-Gasca and J. H. Chow, "Performance comparison of three identification methods for the analysis of electromechanical oscillations," *IEEE Trans. Power Syst.*, vol. 14, no. 3, pp. 995–1002, Aug. 1999.

[5] N. Zhou, D. J. Trudnowski, J. W. Pierre, and W. A. Mittelstadt, "Electromechanical mode online estimation using regularized robust RLS methods," *IEEE Trans. Power Syst.*, vol. 23, no. 4, pp. 1670–1680, Nov. 2008.

[6] G. Liu, J. Quintero, and V. M. Venkatasubramanian, "Oscillation monitoring system based on wide area synchrophasors in power systems," in *Proc. Bulk Power Syst. Dyn. Control iREP Symp.*, Charleston, SC, USA, 2007, pp. 1–13.

[7] A. R. Messina and V. Vittal, "Nonlinear, non-stationary analysis of inter-area oscillations via Hilbert spectral analysis," *IEEE Trans. Power Syst.*, vol. 21, no. 3, pp. 1234–1241, Aug. 2006.

[8] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[9] J. Zhang, S. Nabavi, A. Chakrabortty, and Y. Xin, "Convergence analysis of ADMM-based power system mode estimation under asynchronous wide-area communication delays," in *Proc. IEEE PES Gen. Meeting*, Denver, CO, USA, Jul. 2015.

[10] T. H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Trans. Signal Process.*, vol. 63, no. 2, pp. 482–497, Jan. 2015.

[11] M. Hong, "A distributed, asynchronous and incremental algorithm for nonconvex optimization: An ADMM based approach," *arXiv.org*, 2014.

[12] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, "Asynchronous distributed optimization using a randomized alternating direction method of multipliers," in *Proc. IEEE 52nd Annu. Conf. Decis. Control*, Florence, Italy, 2013, pp. 3671–3676.

[13] R. Zhang and J. T. Kwok, "Asynchronous distributed ADMM for consensus optimization," in *Proc. 31st Int. Conf. Mach. Learn.*, Beijing, China, 2014, pp. 1701–1709.

[14] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. I. Jordan, "A general analysis of the convergence of ADMM," *arXiv.org*, 2015.

[15] Y. Zhang, E. Dall'Anese, and G. B. Giannakis, "Distributed optimal beamformers for cognitive radios robust to channel uncertainties," *IEEE Trans. Signal Process.*, vol. 60, no. 12, pp. 6495–6508, Dec. 2012.

[16] H. Zhu and G. B. Giannakis, "Power system nonlinear state estimation using distributed semidefinite programming," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 6, pp. 1039–1050, Dec. 2014.

[17] Y. Xin and A. Chakrabortty, "A study on group communication in distributed wide-area measurement system networks in large power systems," in *Proc. IEEE Glob. Conf. Signal Inf. Process. (GlobalSIP)*, Austin, TX, USA, Dec. 2013, pp. 543–546.

[18] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Trans. Signal Process.*, vol. 62, no. 7, pp. 1750–1761, Apr. 2014.

[19] A. A. Fouad and P. M. Anderson, *Power System Control and Stability*. Piscataway, NJ, USA: IEEE Press, 2003.

[20] G. Hooghiemstra and P. Van Mieghem, "Delay distributions on fixed Internet paths," Dept. Inf. Technol. Syst., Delft Univ., Delft, The Netherlands, Tech. Rep. 20-011-020, 2001.

[21] T. Qian, A. Chakrabortty, F. Mueller, and Y. Xin, "A distributed storage system for multi-resolution virtual synchrophasors," in *Proc. IEEE PES Gen. Meeting*, Washington, DC, USA, Jul. 2014.

[22] P. P. Khargonekar, I. R. Petersen, and K. Zhou, "Robust stabilization of uncertain linear systems: Quadratic stabilizability and H_∞ control theory," *IEEE Trans. Autom. Control*, vol. 35, no. 3, pp. 356–361, Mar. 1990.

[23] T. Erseghe, D. Zennaro, E. Dall'Anese, and L. Vangelista, "Fast consensus by the alternating direction multipliers method," *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5523–5537, Nov. 2011.

[24] E. Wei and A. Ozdaglar, "On the $\mathcal{O}(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers," in *Proc. IEEE Conf. Global Conf. Signal Inf. Process. (GlobalSIP)*, Austin, TX, USA, 2013, pp. 551–554.

[25] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc WSNs with noisy links—Part I: Distributed estimation of deterministic signals," *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 350–364, Jan. 2008.

[26] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[27] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Asynchronous gossip algorithm for stochastic optimization: Constant step size analysis," in *Recent Advances in Optimization and its Applications in Engineering*. Heidelberg, Germany: Springer-Verlag, 2010, pp. 51–60.

Jianhua Zhang (S'12) received the M.S. degree in electrical engineering from the New Mexico Institute of Mining and Technology, Socorro, NM, USA, in 2010. She is currently pursuing the Ph.D. degree in electrical and computer engineering with North Carolina State University, Raleigh, NC, USA. Her research interests are in wide-area monitoring of power systems, especially in the cyber-physical implementation of the synchrophasor technology.

Seyedbehzad Nabavi (S'12–M'16) received the B.S. degree from the Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2009, and the M.S. and Ph.D. degrees from North Carolina State University, Raleigh, NC, USA, in 2011 and 2015, respectively, all in electrical engineering. He is currently a Research and Technology Development Engineer with New York Power Authority, White Plains, NY, USA.

Aranya Chakrabortty (S'04–M'08–SM'15) received the Ph.D. degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 2008. He is currently an Associate Professor of Electrical and Computer Engineering with North Carolina State University, Raleigh, NC, USA. His research interests are in power system dynamics, identification, and control using wide-area measurement systems technology.

Yufeng Xin received the Ph.D. degree in operations research and computer science from North Carolina State University, Raleigh, NC, USA, in 2002. He is currently a Senior Researcher with the Renaissance Computing Institute, University of North Carolina at Chapel Hill. His research interests are in networking, cloud computing, and cyber-physical systems.