(Invited Paper)  

Jianhua Zhang\textsuperscript{1}, Aranya Chakrabortty\textsuperscript{1}, and Yufeng Xin\textsuperscript{2}  
\textsuperscript{1}North Carolina State University, \textsuperscript{2}Renaissance Computing Institute, UNC Chapel Hill  
Emails: jzhang25@ncsu.edu, aranya.chakrabortty@ncsu.edu, yxin@renci.org

Abstract—In this paper we address the problem of implementing wide-area oscillation monitoring algorithms for large power system networks using distributed processing of Synchrophasor measurements. We consider two computational approaches, namely decentralized least squares (DLS) and its recursive implementation (RLS). Both algorithms are executed using multiple phasor data concentrators (PDC), deployed as virtual computing machines communicating over a fiber-optic communication network. Results are demonstrated using the US-Wide ExoGENI communication network connected to a PMU testbed at NC State University, and analyze the end-to-end computational and communication delays for both algorithms.

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 into the thousands in the next few years under the US 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 rather 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], [3]. In the Eastern Interconnection (EI) of the US grid, for example, about 60 PMUs are currently streaming data via the Internet to a super phasor data concentrator (SPDC) which is handling about 100,000 data points per second. This architecture will no doubt become untenable as the EI scales up to 300-400 PMUs by 2015.

The North American Synchrophasor Initiative (NASPI) is currently addressing this architectural problem by developing new communication and computing protocols for WAMS through NASPInet and Phasor Gateway [4] to facilitate PMU data communication between multiple utilities and control centers. Data from PMUs owned by any specific utility is first passed through the Phasor Gateway, and the output is either sent to ‘Applications’ or to a ‘Historian’ for local use, or sent directly to a NASPInet data bus for sharing information with other utilities, which themselves may be exchanging data through the same bus from PMUs under their own local territories with the neighboring control regions. Research is currently being carried out by the Data and Network Management Task Team (DNM TT) of NASPI on the implementation of this distributed architecture with the prime research focus being - protocols, Quality-of-Service, latency, bandwidth and security [5].

However, almost no attention has yet been paid to perhaps the most critical consequence of this envisioned architecture - namely distributed algorithms. Partly due to a lack of a cyber-physical research infrastructure and partly due to the priorities set forth by PMU installations, the NASPI community has not yet delved into investigating how the currently used centralized algorithms for wide-area monitoring and control [6] can be translated into a distributed computing framework once the aforementioned decentralized WAMS architecture is realized in the next three to four years. Development of such algorithms will be imperative not only for increasing reliability by eliminating single-point failures, but also for minimizing network transit. As shown in [7], transmitting data across a wide-area communication network (WAN) is expensive, the links can be relatively slow, and the bandwidth per dollar will indeed grow slower than other computing resources leading to distributed PMU data processing followed by transmission of full or partially processed outputs as a natural choice [8], [9].

Motivated by this challenge, in this paper we address the problem of implementing wide-area monitoring algorithms over a distributed communication infrastructure using massive volumes of real-time PMU data, and analyze the typical cyber-physical challenges that one may encounter in the process. The specific monitoring algorithm for our interest is the so-called estimation of electro-mechanical oscillation modes of a large power grid arising from the swing dynamic equations of its generators. If the system size is small, and only a handful set of PMUs is installed, then one may estimate these oscillation modes, or equivalently the eigenvalues of the state variable model of the system, in a centralized way using well-known centralized algorithms such as Eigenvalue Realization Algorithm (ERA), Prony analysis, and mode metering [10], [11], all of which have been widely used by the WAMS community over the past decade. In comparison, in this work we consider two com-
computational approaches, namely decentralized least squares (DLS) and its recursive implementation, i.e., recursive least squares (RLS) [12]. Both are implemented in a decentralized fashion using a cloud-computing mechanism overlaid on top of fiber-optic communications. The network chosen for our purpose is the US-wide ExoGENI communication testbed [13], which has recently been integrated with a hardware-in-loop PMU testbed at NC State University using Real-time Digital Simulators (RTDS) and hardware PMUs from different vendors. Multiple phasor data concentrators (PDC), deployed as virtual computing machines communicating over the communication network, receive Synchrophasor measurements from multiple Phasor Measurement Units (PMU), and run an independent least squares algorithm to compute the coefficients of the characteristic polynomial of the transfer function of the power system. These estimates are then transmitted to a central server where the estimated coefficients are averaged, and the global estimates of the dominant oscillation modes are evaluated. In the second case, the same algorithm is executed recursively in real-time. Using the ExoGENI-WAMS testbed we show how the end-to-end computational and communication delays as well as the corresponding accuracy of estimation compare for both of these decentralized algorithms with respect to their centralized implementation.

The remainder of the paper is organized as follows. Section II presents how the oscillation modes of this model can be estimated using DLS and RLS. Section III describes the ExoGENI network testbed on which these algorithms are implemented. Section IV presents numerical results on delay evaluation. Section V concludes the paper.

II. LEAST-SQUARES-BASED DECENTRALIZED ALGORITHMS

We consider a standard small-signal model of a \( n \)-machine power system of the form

\[
\begin{bmatrix}
\Delta \delta \\
\dot{\Delta} \omega
\end{bmatrix} = \begin{bmatrix}
A & B \\
0_{n \times n} & D
\end{bmatrix}
\begin{bmatrix}
\Delta \delta \\
\Delta \omega
\end{bmatrix} + \begin{bmatrix} 0 \\ e_p \end{bmatrix} u. \quad (1)
\]

The details of this model can be found in [6]. Let us denote the \( i^{th} \) eigenvalue of the matrix \( M^{-1}L \) by \( \lambda_i \). The largest eigenvalue of this matrix is equal to 0, and all other eigenvalues are negative, i.e., \( \lambda_m \leq \cdots \leq \lambda_2 < \lambda_1 = 0 \). The eigenvalues of \( A \) are denoted by \( \{-\sigma_1, \cdots, \sigma_{\nu} \} \), \( \sigma = \sqrt{-1} \). Our objective is to estimate these eigenvalues using measurements of voltage, phase angle, and frequency from PMUs in a decentralized fashion. For this we next state the DLS and RLS algorithms, and explain their cyber-physical implementation architectures.

We open the problem by considering a fixed input bus, i.e., a node through which a disturbance input \( u(t) \) enters the system, and two distinct output nodes, say bus \( p \) and bus \( q \), (which may or may not be the same as the input bus) where PMUs are installed, i.e., nodes whose outputs \( y_p(t) \) and \( y_q(t) \) are known. As stated above, in practice \( y \) may refer to either voltage magnitude, or phase angle of frequency recorded by the PMU at that specific bus. Also, in reality, there may be many more than just two outputs. But for simplicity of discussion, we just restrict our discussion to two outputs measured respectively by two PMUs. Since there are \( m \) generators, each modeled by a second-order dynamic model, the total system order is \( n = 2m \). The corresponding discrete-time transfer functions defined for nodes \( p \) and \( q \) from the small-signal disturbance input \( u(t) \) can be expressed as, respectively,

\[
G_p(z) = \frac{Y_p(z)}{U(z)} = \frac{a_0 + a_1z^{-1} + \cdots + a_{m_p}z^{-m_p}}{1 + b_1z^{-1} + \cdots + b_nz^{-n}} \quad (2)
\]

\[
G_q(z) = \frac{Y_q(z)}{U(z)} = \frac{c_0 + c_1z^{-1} + \cdots + c_{m_q}z^{-m_q}}{1 + b_1z^{-1} + \cdots + b_nz^{-n}} \quad (3)
\]

where \( m_p \leq n \) and \( m_q \leq n \) are the orders of the respective polynomial. Taking inverse \( z \)-transform, these \( z \)-domain transfer functions (2-3) can be converted into the sequence-domain equations represented by the block-matrix at the sample index \( k \in \{0, 1, \ldots, \infty\} \), as

\[
y_p(k) = \begin{bmatrix} \phi_p(k) & U_p(k) \end{bmatrix} \begin{bmatrix} \gamma_3 \\ \gamma_1 \\ \gamma_2 \end{bmatrix} ;
\]

\[
y_q(k) = \begin{bmatrix} \phi_q(k) & U_q(k) \end{bmatrix} \begin{bmatrix} \gamma_3 \\ \gamma_1 \\ \gamma_2 \end{bmatrix} \quad (4)
\]

where,

\[
\phi_p(k) = \begin{bmatrix} y_p(k-1) & y_p(k-2) & \cdots & y_p(k-n) \end{bmatrix} \\
\phi_q(k) = \begin{bmatrix} y_q(k-1) & y_q(k-2) & \cdots & y_q(k-n) \end{bmatrix} \\
U_p(k) = \begin{bmatrix} u(k) & u(k-1) & \cdots & u(k-m_p) \end{bmatrix} \\
U_q(k) = \begin{bmatrix} u(k) & u(k-1) & \cdots & u(k-m_q) \end{bmatrix} \\
\gamma_1 = \begin{bmatrix} a_0 & a_1 & \cdots & a_{m_p} \end{bmatrix} \\
\gamma_2 = \begin{bmatrix} c_0 & c_1 & \cdots & c_{m_q} \end{bmatrix} \\
\gamma_3 = \begin{bmatrix} -b_1 & -b_2 & \cdots & -b_n \end{bmatrix}
\]

Our objective is to simply estimate the common characteristic polynomial of the two transfer functions captured by the parameter vector \( \gamma_3 \) from the known input sequence \( u(k) \) and the output sequences \( y_p(k) \) and \( y_q(k) \). Once estimated, these coefficients can be used to compute the system poles, and to get an indication of the dominant frequencies of oscillation and their corresponding damping factors. To achieve this purpose, without any loss of generality, we assume the incoming disturbance \( u(t) \) to be an impulsive input, and apply a batch as well as recursive decentralized least squares approach to compute a global estimate for \( \gamma_3 \). Thereafter, we implement the two algorithms as well as their centralized version in the distributed wide-area EcoGENI communication testbed, run them in real-time, and draw conclusions regarding the accuracy, communication and computation times of each.

A. CENTRALIZED VS DECENTRALIZED LEAST-SQUARES ALGORITHMS

We are interested in investigating two problems: 1) how much accuracy we may have to lose, and 2) how much total time including communication and computation time we can save, if instead of considering the centralized implementation of the least-squares estimation of \( \beta = -\gamma_3 \), using both \( y_p(k) \) and \( y_q(k) \) taken together, we consider each of them individually leading to decentralized, independent local estimates \( \beta_p \) and \( \beta_q \), and thereafter merge these local estimates to form the global estimate \( \beta \). We first consider the centralized
estimation of $\beta$ with both $y_p(k)$ and $y_q(k)$ stacked together in the measurement vector. From (4), we can write

$$\begin{bmatrix} y_p(k) \\ y_q(k) \end{bmatrix} = \begin{bmatrix} \phi_p(k) & U_p(k) \\ \phi_q(k) & U_q(k) \end{bmatrix} \begin{bmatrix} \gamma_p \\ \gamma_q \end{bmatrix}$$

(5)

By assuming that any variable with a negative sample index is zero by default, we construct matrices $A$ and $B$ for $k \in \{0, 1, ..., K\}$ with $K > n$ being a sufficiently large integer, as below,

$$A = \text{col}(y_p(1), y_q(1), y_p(2), y_q(2), ..., y_p(K), y_q(K))$$

$$B = \begin{bmatrix} \phi_p(1) & U_p(1) & 0 \\ \phi_q(1) & 0 & U_q(1) \\ \phi_p(2) & U_p(2) & 0 \\ \phi_q(2) & 0 & U_q(2) \\ \vdots & \vdots & \vdots \\ \phi_p(K) & U_p(K) & 0 \\ \phi_q(K) & 0 & U_q(K) \end{bmatrix}_{2K \times (n+m_p+m_q+2)}$$

(6)

The problem in the centralized case, therefore, is to generate the parameter vector $\Lambda$ that solves $\Lambda = B^{-1}A$ (the matrix inverse operator here means pseudoinverse), then extract the first $n$ entries of $\Lambda$, flip their sign to get the common parameter vector $\beta$, which can be written as $\beta = -[B^{-1}A]^{+n}$.

On the other hand, the decentralized estimation is carried out using the two node outputs independently. The derivation process is exactly same as above, except $y_p(k)$ and $y_q(k)$ are handled as in equation (4) instead of equation (5). Taking the output $y_p(k)$ as the example, we construct the matrices $A_p$ and $B_p$ for $k \in \{0, 1, ..., K\}$ as

$$A_p = \text{col}(y_p(1), y_p(2), ..., y_p(K)) \in \mathbb{R}^{K \times 1}$$

(7)

$$B_p = \begin{bmatrix} \phi_p(1) \\ \phi_p(2) \\ \vdots \\ \phi_p(K) \end{bmatrix} U_p(K)_{K \times (n+m_p+1)}$$

(8)

Similarly, we can get the matrices $A_q$, $B_q$ for the other output $y_q(k)$. Then, the respective estimates will be

$$\beta_p = -[B_p^{-1}A_p]^{+n}, \beta_q = -[B_q^{-1}A_q]^{+n}$$

(9)

To construct a global estimate of $\beta$, we simply average these two decentralized (or local) estimates $\beta_p$ and $\beta_q$, namely $\bar{\beta} = \frac{\beta_p + \beta_q}{2}$ as an arithmetic mean, or $\bar{\beta} = \sqrt{\beta_p \beta_q}$ as the geometric mean. If there are $l$ PMU measurements, then the average will simply result to $\bar{\beta} = \frac{1}{l} \sum_{i=1}^{l} \beta_i$ for the arithmetic mean, or $\bar{\beta} = \left(\prod_{i=1}^{l} \beta_i\right)^{1/l}$ for the geometric mean.

B. Recursive Least-Squares (RLS) Algorithm

The standard least-squares algorithms described in the above subsection, however, is more suited to be carried out in an offline mode, meaning that the algorithm is applicable only when a large chunk of PMU data from various locations is available to the operator in a batch. The recursive least-square (RLS) algorithm, on the other hand, is a real-time parameter estimation method, where PMU data from various locations may stream in online to the operator central server (in the centralized case) or to the local PDCs (in the decentralized case). To formulate the RLS problem, equation (4) can be rewritten as $y_k = \phi_k^T \theta$ for all $k$, where $\theta = \begin{bmatrix} \gamma_3 \\ \gamma_1 \end{bmatrix}$ is the unknown parameter vector and $\phi_k^T = [\phi_p(k) U_p(k)]$ is the regressor vector at the sample index $k$ containing the past input and output information. Following the standard least-squares algorithm, the problem in the case, therefore, is to generate $\theta_K$ that solves

$$\min_{\theta_K} \sum_{k=0}^{K-1} (y_k - \phi_k^T \theta_K)^2 + (\theta_K - \theta_0)^T R_0 (\theta_K - \theta_0)$$

(10)

where, $R_0$ is a positive-definite matrix, and $\theta_0$ is a best initial guess. With this new cost function, the problem reduces to

$$\theta_K = \left( R_0 + \sum_{k=0}^{K-1} \phi_k \phi_k^T \right)^{-1} \left( R_0 \theta_0 + \sum_{k=0}^{K-1} y_k \phi_k \right)$$

(11)

To save computational time, the RLS step can next be applied as follows. Let $P_K = \left( R_0 + \sum_{k=0}^{K-1} \phi_k \phi_k^T \right)^{-1}$, we can get

$$P_{K+1} = \left( P_K^{+1} + \phi_K \phi_K^T \right)^{-1}$$

(12)

By the matrix inversion lemma of $(A + BC)^{-1} = A^{-1} - A^{-1}B(I + CA^{-1}B)^{-1}CA^{-1}$, we may rewrite (12) as,

$$P_{K+1} = P_K - \frac{P_K \phi_K \phi_K^T P_K}{1 + \phi_K P_K \phi_K}$$

(13)

where $P_0 = R_0^{-1}$. By substituting (13) into (11), the iteration for the unknown parameter vector $\theta_K$ can be written as

$$\theta_{K+1} = \theta_K + P_{K+1} \phi_K (y_K - \phi_K^T \theta_K)$$

(14)

The stop condition of this recursive algorithm is when $||\theta_{K+1} - \theta_K|| < \epsilon$ where $\epsilon$ is a chosen tolerance. The estimated common parameter vector then is $\beta = -[\theta_K]^{+n}$. The decentralized version of this algorithm follows similarly as for the batch case. Each PDC, receiving $y_p(k)$ and $y_q(k)$ respectively in real-time, may estimate decentralized estimates $\beta_p$ and $\beta_q$, transmit these two estimates to another PDC, often referred to as a central server in our simulations, where the coefficients are averaged to form the global estimate $\bar{\beta}$. Using this global estimate the characteristic polynomial can be easily constructed and the poles can be solved for. For example, considering $\beta = \{b_1, b_2, ..., b_n\}$, estimates of the actual system poles can then be calculated by solving for the characteristic polynomial

$$1 + b_1 z^{-1} + b_2 z^{-2} + ... + b_n z^{-n} = 0$$

(15)

It is worth noting that depending on need it may not be necessary to estimate all poles of the system but only a specific set of poles, for example those corresponding to inter-area oscillations. The primary goal of wide-area monitoring, in face, often involves selective estimation of oscillation modes. If that is the case, least squares may not
be the optimal approach as it tends to estimate the entire characteristic polynomial as well as the zeros of the transfer function. Instead input-independent methods such as Prony may be more suitable and time-saving.

III. ExoGENI Testbed and Experimental Topology

Compared to the conventional approach of running distributed algorithms using offline software programs, the main contribution of this paper is to implement the aforesaid least-squares-based algorithms for power system mode estimation using C/C++ codes in the real-world networking experimental testbed named ExoGENI [13], which is designed to support research and innovation in networking, operating systems, future Internet architectures, and networked data-intensive cloud computing. We use the ExoGENI network to serve as a platform for wide-area monitoring using PMU data streaming from multiple PMUs to multiple PDCs, realized using virtual computers. We first describe the ExoGENI testbed, and then design three experimental network topologies to investigate the end-to-end delay of the centralized (CLS), decentralized (DLS), and recursive (RLS) least-squares estimation.

A. ExoGENI-WAMS Testbed

Over the past one year the authors have developed a hardware-in-loop simulation framework where high fidelity detailed models of large power systems can be simulated in Real-time Digital Simulators (RTDS), and the dynamic responses can be captured via real hardware Phasor Measurement Units that are synchronized via a common GPS reference. This PMU testbed has recently been connected to a state-funded, metro-scale, multi-layered advanced dynamic optical network testbed called Breakable Experimental Network (BEN) owned by Renaissance Computing Institute (RENCI). BEN connects distributed cloud resources in local universities, with four PoPs located at RENCI, UNC Chapel Hill, NC State Centennial campus, and Duke University, using CISCO and Juniper routers on top of WDM/TDM bandwidth virtualization technology from Infinera. These allow us to set up dynamic multi-layer connections of up to 10 Gbps between the sites. The RTDS output ports at NC State have been connected to this network. The necessary software codes have been developed to extract streaming PMU data from PMUs and PDCs, software images have been installed in all the virtual computers on the BEN PoPs, and finally, tests are being run to check if the data can be successfully accessed by all the nodes in sync with each other.

BEN also has 10 Gbps linkages to the Internet2 network which allows us to set up a Software Defined Network (SDN), referred to as ExoGENI, with dynamic high-speed connections to other universities in the nation, as shown in Figure 1 [13]. We are currently using ExoGENI to access computing resources via Layer 2 networks. We next show how we can simulate different types of disturbance events (short-circuit faults, loss of generation, loss of load, line loss, etc.) in power system models in RTDS, collect the emulated responses via hardware PMUs, and communicate these data via ExoGENI for running our distributed estimation algorithms DLS and RLS. The resulting network is referred to as the ExoGENI-WAMS network. This network is an ideal resource to simulate, demonstrate and validate the fundamental challenges of distributed computation for any wide-area monitoring and control problem.

ExoGENI allows users to create custom topologies using resources from multiple federated providers via a control and management software called the Open Resource Control Architecture (ORCA) to orchestrate the networked cloud resource provisioning. We argue that the current design practice based on the centralized servers and IP-based Internet architecture is not an economical and efficient solution to satisfy the real-time requirement of processing large volume of Synchronized data. We envision a IaaS based solution would be more suitable. In fact, ExoGENI service allows dynamic provisioning of virtual machines of different CPU and memory capacities with customized software images. With this capability, the WAMS communication network can automatically request for the right virtual machine to run the best real-time algorithm, for example, for the least-squares based oscillation mode estimation problem, among other monitoring, state estimation and control problems. With this capability, one can experiment various WAMS applications under real-time constraints, and answer critical questions such as: where to deploy the computing facilities, how to design better communication topologies, and what data transport protocols to use for more efficient control.

B. Experiment Topology Design

To test the estimation algorithms described in Section III, and compare their accuracy and end-to-end delay including communication time and computation time, we design three experimental network topologies. The first topology is for the Centralized Least-Squares (CLS) algorithm, shown in Figure 2(a). It consists of one client Virtual Machine (VM), being charge of generating the simulated PMU measured data $y_1(k)$ and $y_2(k)$, executing the centralized least-squares
algorithm, and sending out the estimated parameter vector \( \beta \), and one server VM, which is responsible for receiving the estimates and solving for the roots of the continuous-time transfer function for the power system. This can, for example, be done by first estimating the roots of the discrete-time polynomial and converting them to their continuous-time counterparts by the relationship \( s = \ln(\alpha)/T \) where \( T \) is the sampling time. Table I shows each component of the end-to-end delay. As obvious from the formulation, the averaging step for \( \beta \) is not included in the case of centralized LS algorithm. The second topology for the Decentralized Least-Squares (DLS) algorithm is shown in Figure 2(b). Compared with first one, the decentralized topology employs two client VMs to generate simulated PMU data \( y_1(k) \) or \( y_2(k) \), run the standard least-squares algorithm and send the estimate \( \beta_1 \) or \( \beta_2 \) to server VM, respectively. The server VM averages the respective coefficients contained in \( \beta_1 \) or \( \beta_2 \), constructs the global characteristic equation, solves the root finding problem, and converts the discrete-time poles to continuous-time poles. For both first and second topologies, the end-to-end delay consists of: (a) \( T_2 \), which is the computation time of least-squares algorithm; (b) \( T_3 \) which is the communication time between the client VM and the central server; and (c) \( T_4 \), which is the computation time for root-finding with or without averaging the elements of \( \beta \).

To further verify the performance of end-to-end day between DLS and RLS algorithms, the third topology shown in Figure 2(c), is designed by adding two PMU VMs, which generate simulated PMU source data \( y_1(k) \) and \( y_2(k) \), respectively. Thus, each client VM needs to stream the PMU source data from the corresponding PMU VM, except for the execution of DLS or RLS algorithm and communication of the estimated sets. In this case, the end-to-end delay contains all four components shown in Table I. The step-by-step architectural diagram for executing the different steps of computing these delays is shown in Figure 3.

### IV. Experiment and Results

#### A. Experiment Setup

We carry out two experiments to verify the performance of CLS, DLS and RLS. Experiment I is to investigate the performance of standard least-squares algorithm in centralized and decentralized fashions. Experiment II is to compare the performance of DLS and RLS algorithms. In both experiments, we choose a 4th-order power system model, consisting of two pairs of complex conjugate poles. The system is a two-machine power system with a single tie-line connecting the two, excited by an impulsive input disturbance at machine 1, with two PMUs measuring their terminal bus frequencies. These measured outputs are sampled once every 0.01 second.
In this case study, the parameter averaging and root-finding are also implemented in the client VM. Thus, $T_3$ component in this section means total computation time of algorithm, plus averaging (if needed) and root finding.

B. Analysis of Results

Accuracy: The final estimates of the coefficients of the characteristic polynomial for the three algorithms are shown in Table II. It can be seen that DLS with averaging preserves the accuracy of the CLS solution, which may be treated as the ideal solution. RLS, however, loses accuracy from the ideal solution to certain extent, mainly due to wrong guesses for the initial guess. More educated guesses for the initial estimate will improve the accuracy of RLS.

End-to-End Delay: Five different scenarios are implemented to investigate the performance of the end-to-end delay. Table III shows the result of Experiment I for comparing the end-to-end delay between CLS and DLS. Note that the RENCI rack is located in the triangle area of North Carolina while the UvA rack is at Netherlands. Also, each PMU source data file has 1001 samples. Two conclusions can be drawn from Experiment I: 1) Scenario 1 shows that the end-to-end delay of DLS decreases to about 54 ms from 147 ms of CLS, clearly pointing that DLS saves computational time $T_2$. 2) When the communication time is dominated with the long distance between the VMs in Scenario 2&3, the advantage of DLS is not as much impressive, but still clear. The result of Experiment II is shown in Table IV. In this case, each PMU source data file only has 200 samples. From Scenario 1&2, the end-to-end delay of RLS is only slightly less than that of CLS, because the iteration number of VM1 is 144, while of VM2 2 is 76, which means that depending on the initial guess the RLS algorithm may take significant number of iterations to converge. Another point worth noticing is that the end-to-end delay is also determined by the type of PMU source data, meaning that if phase angle is used instead of frequency then the algorithm may converge faster, which clearly follows from the participation factor of the modes in the outputs.

V. CONCLUSION

In this paper we presented a comparative study of three well-known least-squares-based algorithms for power system mode estimation using Synchronphasors implemented via an advanced networking and distributed application testbed called ExoGENI. Our results testify that DLS can significantly improve the end-to-end delay without losing accuracy compared to the centralized solution. RLS shows similar time-saving characteristics but loses accuracy if the initial guess is far from the actual solution. Our future work will embrace delay testing of more advanced algorithms such as Prony and Matrix Pencil, and validating them with real PMU data. We are also currently working towards extending these distributed algorithms from monitoring to closed-loop control, and testing their sensitivity with respect of different types of malicious cyber-physical attacks.

REFERENCES


<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS</td>
<td>-0.2172</td>
<td>0.7047</td>
<td>1.2757</td>
<td>-0.7553</td>
</tr>
<tr>
<td>DLS</td>
<td>-0.2273</td>
<td>0.7049</td>
<td>1.2756</td>
<td>-0.7552</td>
</tr>
<tr>
<td>RLS</td>
<td>-0.2471</td>
<td>0.7870</td>
<td>1.4393</td>
<td>-0.8136</td>
</tr>
</tbody>
</table>

**TABLE II**

**ACCURACY OF CLS, DLS, RLS ESTIMATES**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$T_2$(us)</th>
<th>$T_3$(us)</th>
<th>Total (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS</td>
<td>3,178,939</td>
<td>3,348,240</td>
<td></td>
</tr>
<tr>
<td>DLS</td>
<td>3,187,137</td>
<td>3,299,170</td>
<td></td>
</tr>
<tr>
<td>RLS</td>
<td>3,267,583</td>
<td>3,447,497</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE III**

**END-TO-END DELAY OF EXPERIMENT I: CLS vs DLS**

**TABLE IV**

**END-TO-END DELAY OF EXPERIMENT II: DLS vs RLS**