This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSG.2016.2608965, IEEE Transactions on Smart Grid

Fast Parallel Stochastic Subspace Algorithms for Large-Scale Ambient Oscillation Monitoring

Tianying Wu, *Student Member*, *IEEE*, Vaithianathan "Mani" Venkatasubramanian, *Fellow*, *IEEE,* and Alex Pothen

***Abstract*—With the installation of synchrophasors widely across the power grid, measurement-based oscillation monitoring algorithms are becoming increasingly useful in identifying the real-time oscillatory modal properties in power systems. When the number of PMU channels grows, the computational time of many PMU data based algorithms is dominated by the computational burden in processing large-scale dense matrices. In order to overcome this limitation, this paper presents new formulations and computational strategies for speeding up an ambient oscillation monitoring algorithm, namely, Stochastic Subspace Identification (SSI). Based on previous work, two fast Singular Value Decomposition (SVD) approaches are first applied to the SVD evaluation within the SSI algorithm. Next, block structures are exploited so that the large-scale dense matrix computations can be processed in parallel. This helps in memory savings as well as in overall computational time. Experimental results from three sets of archived data of the Western Interconnection demonstrate that the new approaches can provide significant speedups while retaining modal estimation accuracy. With proposed fast parallel algorithms, the real-time oscillation monitoring of the large-scale system using hundreds of PMU measurements becomes feasible.**

***Index Terms*—Power system oscillations, Stochastic Subspace Identification, large-scale dense matrix computations, parallel computing, Synchrophasors**

I. Introduction

DEVELOPMENT of wide-area measurement system (WAMS) which is composed primarily of phasor measurement units (PMUs) [1] has enabled a variety of measurement-based algorithms for system-wide stability analysis [2], [3] and control [4], [5] in recent years. Among them, measurement-based oscillation monitoring algorithms have matured considerably and played an important role in the small signal stability analysis of power systems [6]-[16].

The computational complexity of the above PMU-based oscillation monitoring algorithms largely depends on the number of PMU measurements being processed in the analysis. Currently, only a handful of PMU signals can be handled in real-time implementations. On the other hand, there are several hundreds of PMUs installed in the Western and the Eastern systems of North America already, and it is important to develop fast algorithms that can process all available PMU data together to detect and to locate the likely cause of any oscillatory problems in the observed PMU measurements. More generally, there is an urgent need to develop new computational strategies that can overcome the limitations of PMU processing algorithms in handling hundreds of PMU measurements simultaneously.

Among various methods, Stochastic Subspace Identification (SSI) methods [4], [6], [12]-[16] are recognized as effective algorithms for the modal estimation of systems with unknown inputs like the power system. SSI was first introduced to power systems in [12] to identify the critical modes from the ambient noise measurements. In [14], a recursive adaptive SSI method was proposed in order to reduce the computational burden. Recursive methods generally are more sensitive to data quality issues compared to data block approaches, and our focus in this paper is on fast implementation of the block SSI approach SSI-Covariance in [14]. A recent work [15] tackled the challenge of model-order determination and extended the application of SSI to ring-down analysis.

SSI methods in general are known for accurate estimation of electromechanical modes and their mode shapes from ambient data [3]-[16]. Specifically SSI-Covariance method (denoted simply as SSI in the rest of this paper) has a special advantage over other ambient oscillation identification methods in its ability to simultaneously estimate forced oscillations and inter-area modes even when their frequencies are close to each other [17]-[18].

The main challenge in applying SSI method for real-time ambient modal analysis in power systems is the computational burden in SSI from SVD processing of large dense matrices [16]. An improved parallel SVD algorithm has been introduced in [19]. In [16], we took advantage of the properties of the SVD problem in power systems and introduced two fast SVD approaches for speeding up the SVD: Randomized SVD method [20], and the augmented Lanczos bidiagonalization method [21]. They were tested on SSI algorithm of [14].

This paper shows that the faster SVD methods from [16] by themselves are not sufficient. The construction of the matrix that requires SVD is in itself an extremely time consuming task because of the enormity of the matrix size. We first note that the matrix whose SVD needs to be computed in PMU is large and dense, not sparse [16]. While there exists a rich history of

T. Wu, and V. Venkatasubramanian are with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99163, USA.

A. Pothen is with Department of Computer Science, Purdue University, West Lafayette, Indiana 47907, USA.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSG.2016.2608965, IEEE Transactions on Smart Grid

power engineering literature on the solution of large-scale sparse matrix problems [22], matrix computations involving large dense matrices have remained a challenge. In this paper, we propose new formulations and computational strategies for how such large-scale dense matrices can be handled using parallel implementations.

This paper proposes strategies to get around the problem by solving the SVD problem involving large-scale dense matrix without even constructing the matrix. By deeper analysis of the structure of the dense matrix, it is shown that *all the computational steps in the fast SVD methods of [16] require only the evaluation of product terms of the large dense matrix with a small matrix or a vector.* Moreover, the product can be evaluated in fast parallel fashion by exploiting certain block structures in the matrices.

Using the new strategies introduced in the paper, it is shown that SSI can be implemented to run faster than real time in our test servers even for oscillation monitoring using 400 measurement signals. The contribution of this paper is in demonstrating dramatic speedup from the proposed high performance implementation strategies thus enabling SSI algorithm to be useful for online ambient oscillation monitoring of large power systems.

The rest of the paper is organized as follows: Section II is a brief introduction of the SSI-Covariance method. The block SSI algorithms based on the two fast SVD approaches are then proposed in Section III. In Section IV, test results using three sets of recorded PMU data of the Western Interconnection (of North America) are provided. Section V concludes the paper.

II. Stochastic Subspace Identification - Covariance

The state space model of a stochastic system is defined as

$$\begin{cases} x_{k+1} = Ax_k + w_k \\ y_k = Cx_k + v_k \end{cases}, \tag{1}$$

where $x_k \in \Re^{n\times 1}$ is the state of system; $y_k \in \Re^{l\times 1}$ is the output or measurements; w_k is the process noise and v_k is the measurement noise; n is the system order and l is the number of input channels. All the modal properties of the given system can be obtained from state and output matrices A and C. The first step in SSI is to construct the past and future output matrices Y_p and Y_f from $2I+J$ data elements. That is,

$$Y_p = \begin{bmatrix} y_0 & y_1 & \cdots & y_{J-1} \\ y_1 & y_2 & \cdots & y_J \\ \vdots & \vdots & \ddots & \vdots \\ y_{I-1} & y_I & \cdots & y_{I+J-2} \end{bmatrix} \in \Re^{Il\times J},$$
$$Y_f = \begin{bmatrix} y_I & y_{I+1} & \cdots & y_{I+J-1} \\ y_{I+1} & y_{I+2} & \cdots & y_{I+J} \\ \vdots & \vdots & \ddots & \vdots \\ y_{2I-1} & y_{2I} & \cdots & y_{2I+J-2} \end{bmatrix} \in \Re^{Il\times J}. \tag{2}$$

where J is known as the initial window and I is a user defined parameter that should be larger than n.

The covariance matrix H can be calculated as

$$H = Y_f Y_P^T = O_I G, \tag{3}$$

where O_I the extended observability and G, are defined as

$$O_I = [C \quad CA \quad CA^2 \quad \cdots \quad CA^{I-1}]^T, \tag{4}$$

$$G \equiv E\{x_k y_k^T\}. \tag{5}$$

In (5), the operator $E\{\cdot\}$ denotes the expected value. For estimating the matrices A and C from H, we need to obtain the matrix O_I. It is seen from [23] that:

$$\begin{aligned} H &= O_I G = USV^T, \\ O_I &= U_1 \times S_1^{1/2}, \end{aligned} \tag{6}$$

where U_1 contains the first n left singular vectors corresponding to the n dominant singular values in S_1. The state and output matrices then can be estimated as

$$A = \underline{O}_I^{\dagger} \overline{O}_I, \quad C = O_I(1:l,:), \tag{7}$$

where $\cdot^{\dagger}$ denotes the Moore-Penrose pseudo-inverse, $\underline{O}_I$ is the matrix O_I without the last l rows and $\overline{O}_I$ is the matrix O_I without the first l rows. The continuous-time equivalents of these matrices will be

$$A_c = \frac{1}{T_S}\log A, \quad C_c = C. \tag{8}$$

where T_S is the sampling rate. Modal properties including mode frequency and damping ratio for the oscillation modes of the system can be obtained from an eigenvalue analysis of the matrix A_c and the mode shapes can be calculated using the right eigenvectors of A_c and the output matrix C_c.

As stated before, the new computational burden comes from the construction of the covariance matrix H. It can be seen from (3) that the matrix H is square with size of $I\cdot l \times I\cdot l$. Based on the experiments in [14], a value for $2I$ between 5 and 20 seconds multiplying the sampling rate achieves good estimation accuracy. With a six-second window and sampling rate of 30 Hz which is usually the case in North American power systems, the value for $2I$ will be 180. The size of matrix H then is highly dependent on the number of channels l. Table I shows different sizes of the matrix H that need to be constructed. It is clear from Table I that the size grows with the number of measurements to be handled in the SSI-Covariance algorithm.

TABLE I
The Size of Matrix H

$2I$	l	Size of H
180	10	900×900
	100	9000×9000
	1000	90000×90000

For instance, when 200 PMU signals are to be processed, the covariance matrix H becomes a square matrix with size 18,000 × 18,000. A typical value of the initial window is four minutes,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSG.2016.2608965, IEEE Transactions on Smart Grid

so J equals to 7,200 with a sampling rate of 30 Hz. The construction of the matrix H requires a product of two large-scale matrices Y_f and Y_p in (2) which costs approximately 4.666×10^{12} floating-point operations (flops). The matrix H requires 2.41 GB of memory for analyzing 200 PMU signals and the size of the matrix grows quadratically with increase in the number of PMU signals. Clearly the computation and storage of the covariance matrix H are not scalable for analyzing a large number of PMU signals. In this paper, we show that *the growth in computational time can be made linear with respect to the number of PMU signals* using the strategies introduced in Section III-C.

III. Block SSI Algorithms

The construction of the covariance matrix H is simply the multiplication of two large-scale dense data matrices Y_f and Y_p^T in (2). It costs $O((I \cdot l)^2 \cdot J)$ flops. Instead of applying a full SVD to the H matrix, the two fast SVD approaches multiply the H matrix by a much smaller matrix or a vector, so that construction of the H matrix is no longer necessary. A better procedure is to multiply the Y_p^T matrix by the smaller matrix or the vector first, and then multiply the Y_f matrix by the result from the former step. Next, PMU data matrices Y_p and Y_f are decomposed into I blocks in rows as follows:

$$Y_p = \begin{bmatrix} Y_{p,0} \\ Y_{p,1} \\ \vdots \\ Y_{p,I-1} \end{bmatrix}, Y_{p,i} = \begin{bmatrix} y_i & y_{i+1} & \cdots & y_{i+J-1} \end{bmatrix} \in \Re^{l \times J},$$

$$Y_f = \begin{bmatrix} Y_{f,0} \\ Y_{f,1} \\ \vdots \\ Y_{f,I-1} \end{bmatrix}, Y_{f,i} = \begin{bmatrix} y_{I+i} & y_{I+i+1} & \cdots & y_{I+i+J-1} \end{bmatrix} \in \Re^{l \times J}, \quad (9)$$

$$i = 0, 1, \ldots, I-1.$$

Here, parallel computing can be applied in order to get more time-savings. Moreover, *each block of Y_p or Y_f can be assembled directly from the PMU data matrices without constructing the Y_p and Y_f matrices.* Therefore, the proposed block SSI algorithms demonstrate better performances both in computational time and in memory storage.

A. SSI based on Randomized SVD method

The main steps of the block SSI algorithm based on the Randomized SVD method [20] are as follows:

1) Generate Gaussian random matrices Ω_i, where $\Omega_i \in \Re^{l \times h}$, $i = 0, 1, \ldots, I-1$. h is the rank of approximation matrix (denoted as k in [16]).

2) Form the blocks $Y_{p,i}$ and $Y_{f,i}$ in (9) directly from the PMU output measurements to construct the $I \cdot l \times h$ sample matrix Y.

$$Y = \begin{bmatrix} Y_{f,0} \cdot \sum_{i=0}^{I-1} \left(Y_{p,i}^T \cdot \Omega_i \right) \\ Y_{f,1} \cdot \sum_{i=0}^{I-1} \left(Y_{p,i}^T \cdot \Omega_i \right) \\ \vdots \\ Y_{f,I-1} \cdot \sum_{i=0}^{I-1} \left(Y_{p,i}^T \cdot \Omega_i \right) \end{bmatrix}; \quad (10)$$

for $j = 1 : q$

$$Y_{temp} = \begin{bmatrix} Y_{p,0} \cdot \sum_{i=0}^{I-1} \left(Y_{f,i}^T \cdot Y_i \right) \\ Y_{p,1} \cdot \sum_{i=0}^{I-1} \left(Y_{f,i}^T \cdot Y_i \right) \\ \vdots \\ Y_{p,I-1} \cdot \sum_{i=0}^{I-1} \left(Y_{f,i}^T \cdot Y_i \right) \end{bmatrix}; Y = \begin{bmatrix} Y_{f,0} \cdot \sum_{i=0}^{I-1} \left(Y_{p,i}^T \cdot Y_{temp,i} \right) \\ Y_{f,1} \cdot \sum_{i=0}^{I-1} \left(Y_{p,i}^T \cdot Y_{temp,i} \right) \\ \vdots \\ Y_{f,I-1} \cdot \sum_{i=0}^{I-1} \left(Y_{p,i}^T \cdot Y_{temp,i} \right) \end{bmatrix};$$

endfor

where q is the number of the power iterations [20]. Y_{temp} is an intermediate matrix with no physical meaning. Note that $Y_i \in \Re^{l \times h}$ and $Y_{temp,i} \in \Re^{l \times h}$ can be obtained by decomposing matrix Y and Y_{temp} into I blocks, respectively.

Based on the fact that each block $Y_{p,i}$ (or $Y_{f,i}$) is independent from others, parallel computing can be applied in this step. Take the equation (10) as the example. First, compute I independent tasks $\left(Y_{p,i}^T \cdot \Omega_i \right)$ in parallel and sum the results. Second, multiply each block $Y_{f,i}$ to the summation result $\sum_{i=0}^{I-1} \left(Y_{p,i}^T \cdot \Omega_i \right)$ in I parallel tasks and put the multiplication results into the corresponding places in matrix Y. Repeat this procedure until we obtain the final matrix Y.

3) Form an $I \cdot l \times h$ orthogonal matrix Q through QR decomposition such that $Y = QR$, where R is an upper triangular matrix.

4) Construct the $h \times I \cdot l$ approximation matrix B.

$$B = \begin{bmatrix} \sum_{i=0}^{I-1} \left(Q_i^T \cdot Y_{f,i} \right) \cdot Y_{p,0}^T \\ \sum_{i=0}^{I-1} \left(Q_i^T \cdot Y_{f,i} \right) \cdot Y_{p,1}^T \\ \vdots \\ \sum_{i=0}^{I-1} \left(Q_i^T \cdot Y_{f,i} \right) \cdot Y_{p,I-1}^T \end{bmatrix};$$

Similar to step 2, parallel computing is applied in the construction of matrix B.

5) Calculate the SVD of the $h \times h$ matrix $B = \hat{U} \tilde{\Sigma} \tilde{V}^T$.

6) Form the approximate left-singular vectors $\tilde{U} = Q\hat{U}$.

7) Obtain state and output matrices A_c and C_c using (6) to (8).

8) Calculate modal properties and mode shapes.

The computational complexity is dominated by matrix

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSG.2016.2608965, IEEE Transactions on Smart Grid

multiplications in steps 2 and 4, which costs $O(I{\cdot}l{\cdot}J{\cdot}h)$ flops in serial. Assuming that this is equally distributed among T threads, the computational cost becomes $O(I{\cdot}l{\cdot}J{\cdot}h/T)$ flops in parallel implementation.

B. SSI using Augmented Lanczos Bidiagonalization method

The augmented Lanczos bidiagonalization method [21] only requires the matrix H in the partial Lanczos bidiagonalization. With a unit initial vector $p_1 \in \Re^{I\cdot l}$, we get the decomposition,

$$\begin{aligned} HP_k &= Q_k D_k, \\ H^T Q_k &= P_k D_k^T + r_k e_k^T, \end{aligned} \tag{11}$$

Where $k \ll I{\cdot}l$ is the number of bidiagonalization steps, $p_1 = P_k e_1$, $P_k \in \Re^{I\cdot l\times k}$ and $Q_k \in \Re^{I\cdot l\times k}$ are orthogonal, $D_k \in \Re^{k\times k}$ (denoted as B_k in [16]) is upper bidiagonal

$$D_k = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \alpha_3 & \ddots & \\ & & & \ddots & \beta_{k-1} \\ & & & & \alpha_k \end{bmatrix},$$

and $r_k \in \Re^{I\cdot l}$ is the residual. The pseudo code of the block algorithm for partial Lanczos bidiagonalization is listed below:

1) $P_1 = p_1;\ q_1 = \begin{bmatrix} Y_{f,0}\cdot\sum_{i=0}^{I-1}\left(Y_{p,i}^T\cdot p_{1,i}\right) \\ Y_{f,1}\cdot\sum_{i=0}^{I-1}\left(Y_{p,i}^T\cdot p_{1,i}\right) \\ \vdots \\ Y_{f,I-1}\cdot\sum_{i=0}^{I-1}\left(Y_{p,i}^T\cdot p_{1,i}\right) \end{bmatrix};$
2) $\alpha_1 = \|q_1\|;\ q_1 = q_1/\alpha_1;\ Q_1 = q_1;$
3) *for* $j = 1:k$
4) $r_j = \begin{bmatrix} Y_{p,0}\cdot\sum_{i=0}^{I-1}\left(Y_{f,i}^T\cdot q_{j,i}\right) \\ Y_{p,1}\cdot\sum_{i=0}^{I-1}\left(Y_{f,i}^T\cdot q_{j,i}\right) \\ \vdots \\ Y_{p,I-1}\cdot\sum_{i=0}^{I-1}\left(Y_{fi}^T\cdot q_{j,i}\right) \end{bmatrix} - \alpha_j p_j;$
5) $r_j = r_j - P_j(P_j^T r_j);$
6) *if* $j < k$, *then*
7) $\beta_j = \|r_j\|;\ p_{j+1} = r_j/\beta_j;\ P_{j+1} = [P_j \quad p_{j+1}];$
8) $q_{j+1} = \begin{bmatrix} Y_{f,0}\cdot\sum_{i=0}^{I-1}\left(Y_{p,i}^T\cdot p_{(j+1),i}\right) \\ Y_{f,1}\cdot\sum_{i=0}^{I-1}\left(Y_{p,i}^T\cdot p_{(j+1),i}\right) \\ \vdots \\ Y_{f,I-1}\cdot\sum_{i=0}^{I-1}\left(Y_{p,i}^T\cdot p_{(j+1),i}\right) \end{bmatrix} - \beta_j q_j;$
9) $q_{j+1} = q_{j+1} - Q_j(Q_j^T q_{j+1});$
10) $\alpha_{j+1} = \|q_{j+1}\|;\ q_{j+1} = q_{j+1}/\alpha_{j+1};\ Q_{j+1} = [Q_j \quad q_{j+1}];$
11) *endif*
12) *endfor*

Note that $p_{1,i}, q_{j,i}, p_{(j+1),i} \in \Re^{l}$ can be obtained by decomposing vectors p_1, q_j and p_{j+1} into I blocks, respectively. Similar to the Randomized SVD-based block SSI algorithm in Section A, parallel computing can be applied in pseudo code lines 1, 4, and 8.

Next, follow the remaining steps of the Augmented Lanczos bidiagonalization method in [16] and SSI-Covariance method in Section II to obtain the modal properties and mode shapes.

The computational complexity is dominated by the partial Lanczos bidiagonalization, which is $O(I{\cdot}l{\cdot}J{\cdot}k)$ flops in serial. Assuming that we have T threads, the computational cost becomes $O(I{\cdot}l{\cdot}J{\cdot}k/T)$ flops in parallel.

C. Implementation Strategy

Four strategies are considered for both fast SVD methods in the computational time comparison, in order to test the performance of the block algorithms and parallel computing. Please note that the four strategies provide the same estimation results because the block algorithms will only affect the computational time but not the accuracy of the results.

All four strategies are implemented using a commercial, highly-optimized and extensively-threaded C# math library, NMath.NET [24], which is in turn based on Intel Math Kernel Library (MKL) [25]. The four strategies are listed as follows:

Strategy 1: Construct the covariance matrix H. This is the algorithm introduced in [16], the traditional SSI with the fast SVD methods. The computational complexity is dominated by the construction of the matrix H in (3). It costs $O((I{\cdot}l)^2{\cdot}J)$ flops.

Strategy 2: Compute matrix-vector products involving the covariance matrix H without forming H explicitly. The two fast SVD approaches multiply the H matrix by a much smaller matrix or a vector. In Strategy 2, we multiply the $Y_p{}^T$ matrix by the smaller matrix or the vector first, and then multiply the Y_f matrix by the result from the former step, to avoid the construction of matrix H. The computational complexity is then dominated by the SVD decomposition, which is $O(I{\cdot}l{\cdot}J{\cdot}h)$ and $O(I{\cdot}l{\cdot}J{\cdot}k)$ flops for the two fast SVD methods as discussed in Section III, respectively. There is additional speedup from

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSG.2016.2608965, IEEE Transactions on Smart Grid

native parallel computing features of NMath.NET matrix routines.

Strategy 3: Apply a block algorithm based on NMath.NET native parallel computing. The block structure is introduced in this strategy. We first decompose the PMU data matrices Y_p and Y_f into I blocks as in (9). Then, we assemble each block of Y_p *or* Y_f directly from the PMU data matrices without constructing the Y_p and Y_f matrices. The computational complexity is the same with Strategy 2, which is $O(I{\cdot}l{\cdot}J{\cdot}h)$ or $O(I{\cdot}l{\cdot}J{\cdot}k)$ flops.

Strategy 4: Apply a block algorithm based on explicit parallel computing. In Strategy 4, parallel computing is applied to the SVD decomposition as discussed in Section III (steps 2 and 4 in the Randomized SVD, and pseudo code lines 1, 4, and 8 in the partial Lanczos bidiagonalization). We parallelize the matrix computations in the above steps into I independent tasks, since I is 90, which is higher than the number of threads for our testing machines. The computational cost of Strategy 4 is $O(I{\cdot}l{\cdot}J{\cdot}h/T)$ or $O(I{\cdot}l{\cdot}J{\cdot}k/T)$ flops.

IV. Case Studies

The proposed block SSI methods are applied to analyze the measurement data from three recent recordings in the Western Interconnection with a total of 102, 271 and 400 available PMU channels, respectively. The results in this section are tested with the implementations on the C# platform of Visual Studio 2012. The computational time is reported from the tests on four machines with different numbers of threads. The processor and memory information of the four test machines are listed in Table II.

TABLE II
Testing Machine Information

Mach. No.	CPU	No. of Threads	RAM (GB)	Level 1 Cache (KB)
1	Intel Core i7-4930MX @ 3.0 GHz (Turbo Boost up to 3.9 GHz)	8	32	128
2	Intel Xeon E5-2643 v2 @ 3.5 GHz (Turbo Boost up to 3.8 GHz)	24	64	384
3	Intel Xeon E5-2650 v2 @ 2.6 GHz (Turbo Boost up to 3.4 GHz)	32	32	512
4	Intel Xeon E5-2697 v2 @ 2.7 GHz (Turbo Boost up to 3.5 GHz)	48	128	768

A. *Case 1 with 102 Voltage Phase Angle Channels*

PMU measurements provide a choice of different signal types for ambient oscillation monitoring. The different signal choices include bus voltage phase angles, bus voltage magnitudes, bus frequencies, line current magnitudes and line current phase angles. Bus voltage phase angles have been widely used in the literature and in industry installations because they have excellent observability of electromechanical modes [6]-[15]. Phase angle signals are relatively easy to detrend by subtracting a suitable phase angle [6]-[15]. In later sections of this paper such as in Section IV-B, we show that line current magnitudes also serve as good signals for ambient oscillation monitoring. Typically the number of available PMU line current measurements is much higher than that of bus voltage phase angle measurements. In this context, line current magnitudes provide more observability especially of local oscillatory properties when compared with bus voltage phase angle measurements. In general, the ambient modal estimation results should be consistent across different signal types.

The first case analyzes 102 PMU channels of voltage phase angle measurements. Fig. 1 shows three randomly chosen examples of bus voltage phase angle channels for this case after subtracting a reference phase angle from all of them. The angle reference here is arbitrarily chosen to be a pre-specified PMU bus voltage phase angle. The size of the covariance matrix H is 9,180 × 9,180. The system order for SSI is chosen to be 10. According to the discussion in [16], h = 20 and q = 1 are selected for the Randomized SVD method, and k = 16 is chosen for the augmented Lanczos bidiagonalization method.

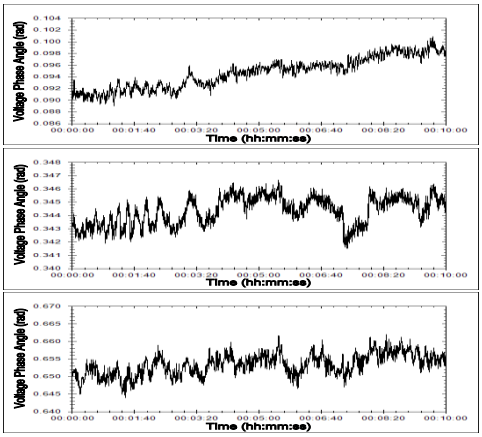

Fig. 1. PMU bus voltage phase measurements of Case 1.

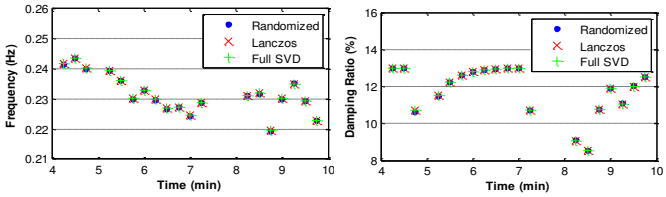

Fig. 2. Plots of the estimation results for different SSI algorithms for Case 1.

TABLE III
Summary Results of Different SSI Algorithms for Case 1

Mode Results		Full SVD	Randomized SVD h = 20, q = 1	Lanczos SVD k = 16
f (Hz)	*Mean*	0.2317	0.2316	0.2317
	SD	0.0063	0.0063	0.0063
ζ (%)	*Mean*	11.8046	11.8088	11.8046
	SD	1.3161	1.3104	1.3161

Ten minutes of PMU data are analyzed and the estimation results of the traditional SSI with full SVD are compared with the two fast SVD methods in Fig. 2. The analysis uses an initial window of four minutes, and is repeated every 15 seconds in a moving window formulation. So J is 7200, and there are 24 moving windows, which implies 24 SSI computations in total. The mean value and the standard deviation (SD) of the frequency and the damping ratio for the mode are calculated and are listed in Table III.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSG.2016.2608965, IEEE Transactions on Smart Grid

TABLE IV
COMPUTATIONAL TIME COMPARISON OF DIFFERENT SSI ALGORITHMS FOR CASE 1

Mac. No.	Full SVD (sec)	Randomized SVD $h = 20$, $q = 1$				Lanczos SVD $k = 16$			
		Strategy 1 sec (speedup)	Strategy 2 sec (speedup)	Strategy 3 sec (speedup)	Strategy 4 sec (speedup)	Strategy 1 sec (speedup)	Strategy 2 sec (speedup)	Strategy 3 sec (speedup)	Strategy 4 sec (speedup)
1	112,075	105,826 (1.06x)	206.8 (542x)	135.4 (828x)	**105.7 (1060x)**	106,234 (1.05x)	**81.2 (1380x)**	100.2 (1119x)	96.0 (1167x)
2	125,807	121,007 (1.04x)	237.9 (529x)	174.0 (723x)	**141.8 (887x)**	115,458 (1.09x)	**87.4 (1439x)**	148.0 (850x)	122.0 (1031x)
3	138,236	135,304 (1.02x)	298.4 (463x)	230.3 (600x)	**134.2 (1030x)**	133,961 (1.03x)	**101.9 (1357x)**	177.0 (781x)	115.0 (1202x)
4	170,518	168,617 (1.01x)	304.4 (560x)	226.2 (754x)	**140.0 (1218x)**	163,028 (1.05x)	**104.8 (1627x)**	158.0 (1079x)	138.3 (1233x)

According to Fig. 2 and Table III, the estimation results of the block SSI algorithms based on the two fast SVD methods are comparable to the estimations of the full SVD-based traditional SSI algorithm, which demonstrates the accuracy of the proposed block SSI algorithms.

The four implementation strategies discussed in Section III as considered in this case for testing the computational performance of the block algorithm. First let us summarize the values for all the variables in the flop computation formulas for this test case: $I = 90$, $l = 102$, $J = 7200$, $h = 20$, $q = 1$, and $k = 16$.

In the Randomized SVD-based method, the computational complexity is dominated by the construction of the covariance matrix H (only for Strategy 1), the sample matrix Y, and the small matrix B. The construction of matrix H requires the multiplication of matrices Y_f and Y_p^T, which costs $2\times(I\cdot l)^2\times J$ flops, i.e. 1.214×10^{12} flops in this test case. And it costs $2\times(I\cdot l)^2\times h\times(2q+1)$ flops and $2\times(I\cdot l)^2\times h$ flops in order to build the matrices Y and B, respectively. So there are 1.348×10^{10} more flops in Strategy 1. Strategy 2 requires $4\times I\cdot l\times J\times h\times(2q+1)$ flops and $4\times I\cdot l\times J\times h$ flops to obtain matrices Y and B, respectively. So it costs only 2.115×10^{10} flops in total, which is a 58-fold speedup compared to Strategy 1.

In the Lanczos SVD-based method, the flop counts of Strategy 1 include $2\times(I\cdot l)^2\times J$ flops for building matrix H and $2\times(I\cdot l)^2\times(2k+1)$ flops for the partial Lanczos bidiagonalization, whereas Strategy 2 requires only $4\times I\cdot l\times J\times(2k+1)$ flops for the partial Lanczos bidiagonalization. So the speedup of Strategy 2 over Strategy 1 is 140. (The flop counts are 1.220×10^{12} and 8.725×10^{9} for Strategy 1 and Strategy 2, respectively.)

For both Randomized and Lanczos SVD-based methods, Strategy 3 costs same number of flops as Strategy 2 in theory. Strategy 4 applies parallel computing, so the time savings compared to Strategy 3 is based on the number of the threads available in the test server.

The computational time of the entire analysis is tested on four different machines and the results are listed in Table IV. The comparison is made among all the four strategies for both fast SVD methods as well as the traditional SSI method with full SVD calculation. The fastest times for the two SVD methods are each marked in bold.

After subtracting the initial window, a total of 360 seconds of PMU data have been analyzed. The traditional SSI method which calculates the full SVD costs almost two days to finish the analysis of six-minute data. Moreover, it costs 642.9 MB of memory to store matrix H, which can be saved in Strategy 2.

The number of flops is the same for Strategy 2 and Strategy 3 for both fast SVD methods. Strategy 3 will cost additional time in order to assemble the blocks. However, since the construction of the Y_p and Y_f matrices is avoided in Strategy 3, it can save 0.985 GB of memory storage in this test case. By introducing the two fast SVD methods (Strategy 1), we have saved around one hour in the computation time, which is insignificant compared to the total computational time.

However, by simply switching the order of the matrix computations (Strategy 2), the calculation times drop to minutes instead of days. The speedup for the Randomized SVD-based algorithm is around 500 times, and is over 1000 times for the Lanczos SVD-based method. The actual speedup from Strategy 2 to Strategy 1 is even higher than the theoretical one. This is because the matrix H is too big to fit inside the cache memory, which costs extra time when processing it.

The block structure is utilized in Strategy 3. The computations are faster than Strategy 2 on all the four machines for the Randomized SVD-based method. But they are slower for the Lanczos SVD-based method. This is because, the matrix H multiplies a vector x in the Lanczos SVD method, whereas it multiplies a small matrix in the Randomized SVD method. The vector x can be stored in 71.7 KB of memory, so it fits inside the Level 1 cache for all the machines. However, in the Randomized SVD method, the small matrix requires 1.40 MB of memory, which is larger than the Level 1 cache size. The block algorithms decompose the matrix into smaller blocks to make them fit inside the Level 1 cache, so that the computations become faster with blocking in this case.

Parallel computing is applied in Strategy 4. The computational time is faster than Strategy 3 on all the machines. Since NMath.Net library has the inherent multi-threading feature, even in Strategies 2 and 3, almost half of the threads have been used in the computation. Therefore, the speedup for Strategy 4 is not significant compared to Strategies 2 and 3, although the manual parallel computing in Strategy 4 uses all the available threads for calculation.

Comparing the four machines in our tests, Machine 1 takes the shortest time to complete the entire analysis for all the strategies, because of its higher CPU clock rate. However, if we consider the speedup of the block SSI methods compared to the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSG.2016.2608965, IEEE Transactions on Smart Grid

traditional SSI, Machine 4 is the best. It has the largest number of threads among four machines, so more speedup can be obtained on Machine 4 with parallel computing.

For the Lanczos SVD-based algorithm, Strategy 4 is still slightly slower than Strategy 2. This suggests that changing the order of the matrix multiplication is good enough for this test case. However, the block structure and parallel computing are necessary for the Randomized SVD-based algorithm.

Based on the results in Table IV, Strategy 4 is the fastest for the Randomized SVD-based algorithm. And, Strategy 2 is the fastest for the Lanczos SVD-based algorithm. Both of them take less than three minutes on all four machines which are faster than real time. This is remarkable given that the traditional SSI-Covariance algorithm implementation would take more than one day to analyze the same set of PMU data on the same machines.

B. Case 2 with 271 Current Magnitude Channels

In order to test the proposed block SSI algorithms for a large-scale system, we analyze the second case with 271 PMU channels of current magnitude measurements. Fig. 3 shows three examples of PMU line current magnitude measurements for this case.

The size of the covariance matrix H in this case is 24,390 × 24,390. Same parameters with Case 1 are chosen for different SSI algorithms. Ten minutes of PMU data are analyzed.

In this test case, the traditional SSI algorithm is no longer feasible. The program will generate an error says "Array dimensions exceeded supported range" in the process of constructing the covariance matrix H. This is because of the memory limitations in the C# platform that "No single object can be larger than 2GB". Whereas the matrix H in this test case costs 4.43 GB of memory to store it.

The mean and standard deviation (SD) of the estimation results of the two fast SVD methods are shown in Fig. 4 and Table V. The analysis uses an initial window of four minutes, and is repeated every 15 seconds in a moving window formulation.

Based on Fig. 4 and Table V, the estimation results of the two block SSI algorithms agree with each other. The proposed block SSI algorithms can guarantee the accuracy when the number of PMU channels being processed has increased. Same strategies are considered to test the computational time for the two block SSI algorithms. Because the construction of the matrix H is infeasible, Strategy 1 is only analyzed in theory, but not implemented for this test case.

The construction of the covariance matrix H costs 8.566 × 10^{12} flops in this test case. The flop counts speedups of Strategy 2 from Strategy 1 are 154 times and 371 times for the Randomized SVD-based and Lanczos SVD-based algorithms, respectively. Strategy 3 requires extra time for addressing the blocks, but can save 2.62 GB of memory for not constructing the Y_p and Y_f matrices in this test case. With parallel computing applied in Strategy 4, more time savings can be achieved compared to Strategy 3.

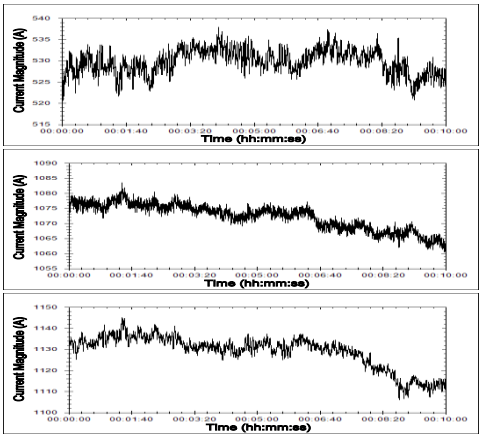

Fig. 3. PMU line current magnitude measurements of Case 2.

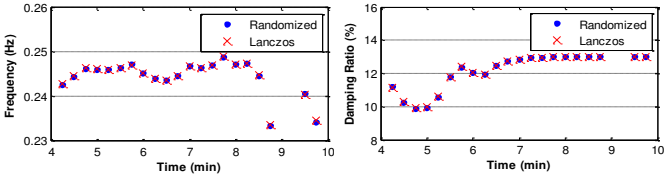

Fig. 4. Plots of the estimation results for different SSI algorithms for Case 2.

TABLE V
SUMMARY RESULTS OF DIFFERENT SSI ALGORITHMS FOR CASE 2

Mode Results		Randomized SVD $h = 20, q = 1$	Lanczos SVD $k = 16$
f (Hz)	*Mean*	0.2445	0.2446
	SD	0.0039	0.0038
ζ (%)	*Mean*	12.1434	12.1522
	SD	1.0809	1.0646

TABLE VI
COMPUTATIONAL TIME (SEC) COMPARISON FOR CASE 2

Mac. No.	Randomized SVD $h = 20, q = 1$			Lanczos SVD $k = 16$		
	Strat.2	Strat.3	Strat.4	Strat.2	Strat.3	Strat.4
1	539.7	306.7	**289.0**	200.2	199.0	**171.0**
2	573.0	373.4	**269.8**	212.0	220.0	**197.0**
3	674.0	425.2	**269.4**	226.6	249.7	**208.7**
4	785.8	491.6	**262.5**	209.6	261.0	**203.0**

Table VI summarized the comparison results for the computational time of different strategies tested on four different machines.

Similar to Case 1, Strategy 4 is the fastest for the Randomized SVD-based block algorithm. For the block SSI algorithm based on the Lanczos SVD method, Strategy 3 is still slower than Strategy 2, except on Machine 1 where the times for the two strategies are almost the same. But with the parallel computing technique, Strategy 4 becomes faster than Strategy 2 in this test case. Therefore, when the number of PMU channels increases, exploiting the block structure and employing the parallel computing are useful for the Lanczos SVD-based algorithm.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSG.2016.2608965, IEEE Transactions on Smart Grid

The fastest times for both block SSI algorithms are in bold in Table VI. They are faster than real time on all the machines, whereas the traditional SSI algorithm is not even feasible.

C. Case 3 with 400 Current Magnitude Channels

In the third case, there are 400 PMU channels of current magnitude measurements.

The size of the covariance matrix H in this case is 36,000 × 36,000. Same parameters with the first two cases are chosen for comparing different SSI algorithms. Ten minutes of PMU data are analyzed.

A memory space of 9.65 GB is needed to store the covariance matrix H, which is higher than the single object memory limit of the C# platform. Therefore, it is impossible to construct the matrix H, which means the traditional SSI algorithm is infeasible in this case. The mean and standard deviation (SD) of the modal estimation results of the two fast SVD methods are shown in Fig. 5 and Table VII. The analysis uses an initial window of four minutes, and is repeated every 15 seconds in a moving window formulation.

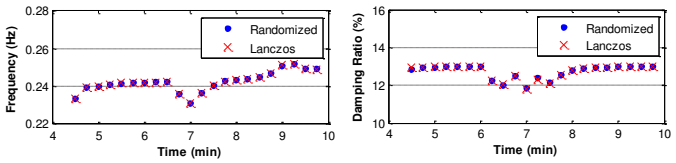

Fig. 5. Plots of the estimation results for different SSI algorithms for Case 3.

TABLE VII
SUMMARY RESULTS OF DIFFERENT SSI ALGORITHMS FOR CASE 3

Mode Results		Randomized SVD $h = 20, q = 1$	Lanczos SVD $k = 16$
f (Hz)	*Mean*	0.2425	0.2425
	SD	0.0052	0.0052
ζ (%)	*Mean*	12.7344	12.7331
	SD	0.3521	0.3677

TABLE VIII
COMPUTATIONAL TIME (SEC) COMPARISON FOR CASE 3

Mac. No.	Randomized SVD $h = 20, q = 1$			Lanczos SVD $k = 16$		
	Strat.2	Strat.3	Strat.4	Strat.2	Strat.3	Strat.4
1	787.0	553.0	**482.5**	285.8	301.2	**253.0**
2	804.5	553.7	**426.0**	282.8	266.0	**225.0**
3	928.5	641.3	**371.0**	303.0	301.4	**231.8**
4	1122.5	768.3	**386.0**	301.0	378.0	**270.4**

According to Fig. 5 and Table VII, the estimations of the two block SSI algorithms match well with each other. Again, this indicates good performance of the proposed algorithms in terms of estimation accuracy. We use the same implementation strategies to test the computational time. Similar to Case 2, Strategy 1 is only analyzed in theory, but not implemented for this test case, owing to memory limitations.

It takes 1.866×10^{13} flops to construct the matrix H in this 400-channel case. Strategy 2 provides a 227-fold speedup in flops compared to Strategy 1 for the Randomized SVD-based algorithm, and the speedup for the Lanczos SVD-based method is 548 times. The matrices Y_p and Y_f in this case take up 3.862 GB of memory space, and this can be reduced in Strategy 3. Strategy 4 applies parallel processing to save more time. The comparison among different strategies on all four testing machines is summarized in Table VIII.

According to Table VIII, Strategy 4 (in bold) is the fastest for both block SSI algorithms. Especially for the Lanczos SVD-based algorithm, the block structure with parallel processing shows much better performance in terms of speed, when the number of PMU signals becomes larger.

The computational time is longer than six minutes for the Randomized SVD-based algorithm, especially on Machine 1 where it takes more than eight minutes to complete the entire analysis. In fact, Machine 1 is the fastest one for the first three strategies. But it becomes the slowest one after parallel processing is employed. This is because it has only eight threads which is the smallest among all the testing machines.

Since the overall computational time is less than six minutes (360 seconds), the Lanczos SVD-based algorithm is still faster than real time even with 400 signals on all four test machines.

D. Discussion

In Case 1, the block structure is not advantageous for the Lanczos SVD-based algorithm when there are only around one hundred PMU signals. However, the benefits of the parallel computing gradually emerge when the number of PMU channels keeps growing.

In order to identify the break-even point of number of PMU channels that would result in nearly equal computational performance between Strategies 2 and 4 for the Lanczos SVD-based algorithm, we start with Case 2 in which Strategy 4 is faster than Strategy 2. Then, we keep decreasing the number of PMU signals by 20 until the computational time of Strategy 2 becomes smaller than that of Strategy 4.

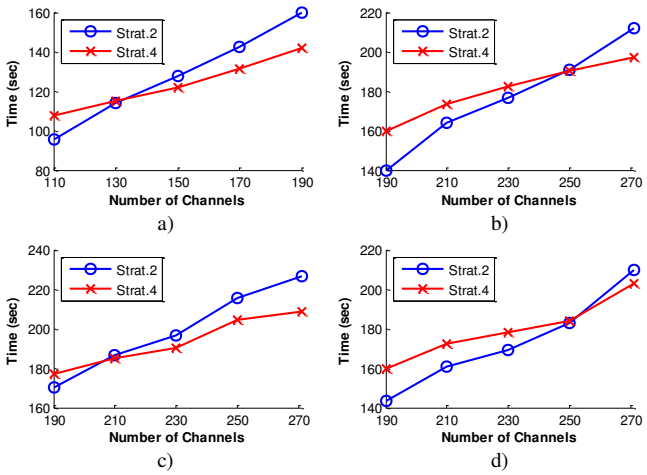

Fig. 6. Speed comparison of Strategies 2 versus 4: a) Machine 1, b) Machine 2, c) Machine 3, and d) Machine 4.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSG.2016.2608965, IEEE Transactions on Smart Grid

TABLE IX
COMPUTATIONAL TIME COMPARISON OF DIFFERENT TEST CASES

Case No.	No. of PMU Signals	Randomized SVD				Lanczos SVD			
		Machine 1 sec (GF)	Machine 2 sec (GF)	Machine 3 sec (GF)	Machine 4 sec (GF)	Machine 1 sec (GF)	Machine 2 sec (GF)	Machine 3 sec (GF)	Machine 4 sec (GF)
1	102	105.7 (-)	141.8 (-)	134.2 (-)	140.0 (-)	96.0 (-)	122.0 (-)	115.0 (-)	138.3 (-)
2	271	289.0 (2.73x)	269.8 (1.90x)	269.4 (2.01x)	262.5 (1.88x)	171.0 (1.78x)	197.0 (1.61x)	208.7 (1.81x)	203.0 (1.47x)
3	400	482.5 (4.56x)	426.0 (3.00x)	371.0 (2.76x)	386.0 (2.78x)	253.0 (2.64x)	225.0 (1.84x)	231.8 (2.02x)	270.4 (1.95x)
4	800	-	-	-	-	471.3 (4.90x)	536.7 (4.40x)	480.3 (4.18x)	463.4 (3.35x)
5	1200	-	-	-	-	673.3 (7.01x)	662.5 (5.43x)	649.7 (5.65x)	652.7 (4.72x)

Note: GF stands for growth factor.

Fig. 6 shows the computational time taken by Strategies 2 and 4 on the four testing machines with different number of signals. According to Fig. 6, Machines 1, 2, 3, and 4 reach their break-even points in terms of the same computational time between Strategies 2 and 4 at around 130, 250, 210, and 250 PMU channels respectively. For all four machines, the proposed block strategy is the fastest when the number of PMU signals becomes high enough as shown in Fig. 6.

In order to test the performance of speed for the proposed block SSI algorithms with more PMU channels, we duplicate the measurement signals from Case 3 twice and three times to make two larger test cases of 800 and 1200 channels, respectively. The underlying assumption in deriving the Randomized SVD method that the data is random does not hold by repeating the channels, and the Randomized SVD-based estimation results suffer for the cases of 800 and 1200 channels. Therefore, only the Lanczos SVD-based methods are tested with Case 4 and Case 5.

A comparison of the computational time for all the five test cases is made in Table IX. The times for Strategy 4 are used here, because they are the shortest among all the strategies for most cases excepting Lanczos SVD-based method for Case 1. From Case 1 to Case 2, the number of PMU measurements has increased 2.66 times, but the computational time for Case 2 is roughly twice of Case 1 for both methods, except for the Randomized SVD-based method on Machine 1 which increases 2.73 times.

When the number of PMU channels increases by a factor of 3.92 in Case 3, the computational time grows by less than three times compared to Case 1, except for the Randomized SVD-based method on Machine 1 which grows 4.56 times. In Case 4, the computational time increases by less than five times on all the machines, compared to the fact that the number of PMU signals is 7.84 times of Case 1. When there are more than one thousand channels in Case 5, the computational time only increases around seven times on Machine 1, and around five times on Machine 2, 3, and 4.

To conclude, *the computational time of the proposed fast parallel methods rises in a linear fashion with the number of PMU measurements.*

The growth factor of Machine 1 for the Randomized SVD-based method is much higher than the ones of other machines, which suggests that eight threads are not enough for processing large number of PMU signals. Parallel computing is important for future power systems with a larger number of PMU signals.

V. CONCLUSION

In this paper, two fast parallel SSI algorithms are proposed based on the two fast SVD approaches introduced in [16]. The algorithms have been tested using the measurement data from three PMU recordings of the Western Interconnection. They provide accurate estimation results and at the same time, speed up the computation significantly.

The block structure and parallel processing are not necessary for the Lanczos SVD-based algorithms in the first test case. But they become useful when the number of PMU measurements increases. The computational complexity of the traditional SSI algorithm grows quadratically with the number of PMU channels being processed in the analysis. However, the strategies introduced in this paper show that proposed block algorithms can make the growth linear.

In summary, this paper shows that large-scale dense matrices in PMU problems can be handled effectively in parallel implementations by employing strategies such as the ones illustrated here for the SSI-Covariance algorithm.

VI. ACKNOWLEDGEMENT

We thank US Department of Energy for supporting this research through funding from the Consortium for Electric Reliability Technology Solutions, and also through grant DE-SC0010205. We acknowledge National Science Foundation grant 1552323. We also thank Dr. Ananth Kalyanaraman, Washington State University for helpful discussions on different test machines.

REFERENCES

[1] J. F. Hauer, et al., "Use of the WECC WAMS in wide-area probing tests for validation of system performance and modeling," *IEEE Trans. Power Syst*., vol.24, no.1, pp.250-257, Feb. 2009.

[2] H. Yuan and F. Li, "Hybrid voltage stability assessment (VSA) for N-1 contingency," *Electr. Pow. Syst. Res.*, vol. 122, pp. 65–75, May 2015.

[3] R. Eriksson, and L. Söder, "Wide-area measurement system-based subspace identification for obtaining linear models to centrally coordinate controllable devices," *IEEE Trans. Power Del.*, pp. 988- 997, Apr. 2011.

[4] P. Zhang, D. Y. Yang, K. W. Chan, and G. W. Cai, "Adaptive wide-area damping control scheme with stochastic subspace identification and

signal time delay compensation," *IET Gen., Transm,. Distrib.*, vol. 6, no. 9, pp. 844–852, Sep. 2012.
[5] H. Yuan, T. Jiang, H. Jia, F. Li, et al., "Real-time wide-area loading margin sensitivity (WALMS) in Power Systems," *Proc. IEEE PES General Meeting*, Denver, CO, USA, Jul. 2015.
[6] N. Zhou, J. W. Pierre, and J. F. Hauer, "Initial results in power system identification from injected probing signals using a subspace method, " *IEEE Trans. Power Syst.*, vol.21, no.3, pp. 1296–1302, Aug. 2006.
[7] I. Kamwa, A. K. Pradhan, and G. Joós, "Robust detection and analysis of power system oscillations using the Teager-Kaiser energy operator," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 323–333, Feb. 2011.
[8] J. C. Peng, and N. C. Nair, "Enhancing Kalman filter for tracking ringdown electromechanical oscillations," *IEEE Trans. Power Syst*., vol. 27, no.2, pp. 1042-1050, May 2012.
[9] N. Zhou, J.W. Pierre, D. J. Trudnowski, and R. T. Guttromson, "Robust RLS methods for online estimation of power system electromechanical modes," *IEEE Trans. Power Syst*., pp. 1240–1249, Aug. 2007.
[10] J. Ning, X. Pan, V. Venkatasubramanian, "Oscillation modal analysis from ambient synchrophasor data using distributed frequency domain optimization," *IEEE Trans. Power Syst.*, pp. 1960-1968, May 2013.
[11] H, Khalilinia, V. Venkatasubramanian, "Modal analysis of ambient PMU measurements using orthogonal wavelet bases," *IEEE Trans. Smart Grid*, vol. 6, no. 6, pp. 2954-2963, Nov. 2015.
[12] H. Ghasemi, C. Canizares and A. Moshref, "Oscillatory stability limit prediction using stochastic subspace identification, " *IEEE Trans. Power Syst*., vol.21, no.2, pp. 736-745, May 2006.
[13] T. Jiang, *et al.*, "A Novel Dominant Mode Estimation Method for Analyzing Inter-Area Oscillation in China Southern Power Grid," *IEEE Trans. Smart Grid*, vol. 7, no. 5, pp. 2549-2560, Sept. 2016.
[14] S. A. Nezam Sarmadi, and V. Venkatasubramanian, "Electromechanical mode estimation using recursive adaptive Stochastic Subspace Identification", *IEEE Trans. Power Syst.*, pp. 349–358, Jan. 2014.
[15] T. Jiang, H. Yuan, H. Jia, N. Zhou, and F. Li, "Stochastic subspace identification-based approach for tracking inter-area oscillatory modes in bulk power system utilising synchrophasor measurements," *IET Gen., Transm., Distrib.,* vol. 9, no. 15, pp. 2409-2418, Nov. 2015.
[16] T. Wu, S. A. Nezam Sarmadi, V. Venkatasubramanian, A. Pothen, and A. Kalyanaraman, "Fast SVD computations for synchrophasor algorithms," *IEEE Trans. Power Syst.*, vol. 31, no. 2, pp. 1651-1652, Mar. 2016.
[17] S. A. N. Sarmadi and V. Venkatasubramanian, "Inter-area resonance in power systems from forced oscillations," *IEEE Trans. Power Syst*., vol. 31, no. 1, pp. 378–386, Jan. 2016.
[18] S. A. N. Sarmadi, V. Venkatasubramanian, and A. Salazar, "Analysis of November 29, 2005 Western American Oscillation Event," *IEEE Trans. Power Syst.*, DOI: 10.1109/TPWRS.2016.2521319, to appear.
[19] A. Haidar, J. Kurzak, and P. Luszczek, "An improved parallel singular value algorithm and its implementation for multicore hardware," *Proc. of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis*, Denver, CO, USA, Nov. 2013.
[20] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions", *SIAM Rev*., vol. 53, no. 2, pp. 217–288, 2011.
[21] J. Baglama and L. Reichel, "Augmented implicitly restarted Lanczos bidiagonalization methods," *SIAM J. Sci. Comput.*, pp. 19–42, 2005.
[22] P. Kundur, Power *System Stability and Control*, McGraw-Hill, 1992.
[23] P.V. Overchee and B.D. Moore, *Subspace Identification for Linear Systems: Theory, Implementation and Applications,* Dordrecht: Kluwer Academic Publishers, 1996.
[24] http://www.centerspace.net/products/nmath
[25] https://software.intel.com/en-us/intel-mkl

Biographies

Tianying Wu (S'13) received the B.Sc. degree in electrical engineering from Southeast University, Nanjing, China, in 2011. She is currently pursuing the Ph.D. degree at School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, USA. Her research interests include power system stability and power system dynamics.

Vaithianathan "Mani" Venkatasubramanian is currently a Professor at Washington State University, Pullman, WA. His research interests include nonlinear system theory, power system stability and control.

Alex Pothen is a professor of computer science at Purdue University. His research interests are in combinatorial scientific computing (CSC), high-performance computing, and bioinformatics. Alex received an undergraduate degree from the Indian Institute of Technology, Delhi, and a PhD from Cornell University. He helped found the CSC community, which organizes a Society for Industrial and Applied Mathematics (SIAM) Workshop on CSC every two years, since 2004. He was the Director of the Combinatorial Scientific Computing and Petascale Simulations (CSCAPES) Institute, a pioneering research project funded by the U.S. Department of Energy during 2006-2012 to design combinatorial algorithms for extreme-scale computers. He serves as an editor of the Journal of the Association for Computing Machinery (ACM) and the SIAM Review, and has served on the editorial board of SIAM Books, SIAM Spotlights, SIAM Journal on Scientific Computing, and other publications.