

Comparative study of clustering techniques for real-time dynamic model reduction

Emilie Purvine¹ | Eduardo Cotilla-Sanchez² | Mahantesh Halappanavar³ |
Zhenyu Huang⁴ | Guang Lin⁵ | Shuai Lu⁶ | Shaobu Wang⁴

¹National Security Directorate, Pacific Northwest National Laboratory, Richland, Washington

²School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, Oregon

³Physical and Computational Sciences Directorate, Pacific Northwest National Laboratory, Richland, Washington

⁴Energy and Environment Directorate, Pacific Northwest National Laboratory, Richland, Washington

⁵Department of Mathematics, Purdue University, West Lafayette, Indiana

⁶EnerMod, Austin, Texas

Correspondence

Emilie Purvine, Pacific Northwest National Laboratory, 1100 Dexter Ave N., Suite 400, Seattle, WA 98125.
Email: emilie.purvine@pnnl.gov

Funding information

US Department of Energy (DOE),

Dynamic model reduction in power systems is necessary for improving computational efficiency. Traditional model reduction using linearized models or offline analysis is not adequate to capture dynamic behaviors of the power system, especially with the new mix of intermittent generation and intelligent consumption, making the power system more dynamic and nonlinear. Real-time dynamic model reduction has emerged to fill this important need. This paper explores using clustering techniques to analyze real-time phasor measurements to identify groups of generators with similar behavior, as well as a representative generator from each group for dynamic model reduction. Two clustering techniques—graph clustering and k -means—are considered. These techniques are compared with a previously developed dynamic model reduction approach using singular value decomposition. Two sample power grid datasets are used to test these different model reduction techniques. Based on the algorithms' relative performance, recommendations are provided for practical use.

KEYWORDS

graph clustering; model reduction; power system dynamics; SVD

1 | INTRODUCTION

Power engineers rely on simulation to accomplish operation and planning tasks. Dynamic simulation tools are used extensively to simulate the behavior of power systems over time subject to disturbances, such as faults, sudden loss of transmission paths, and loss of generation or load. Good models and accurate parameters are essential for credible computer simulation. In some cases, inaccurate computer models and simulations result in optimistic decisions, which can put the electric infrastructure in jeopardy. The extreme consequences of such optimistic decisions can be massive outages, such as the August 1996 western US system breakup [16]. In many other cases, inaccurate models and simulations lead to pessimistic decisions, resulting in reduced asset utilization.

Given the power grid's complexity and large footprint, a power company needs a reduced model for the region outside of its own service territory. The goal of such model reduction is to reasonably represent the external system with a simplified smaller model so that analysis can be performed more

efficiently [1, 30]. This is of particular interest for real-time power system operation, such as online dynamic security assessment [27, 28, 31].

Traditionally, model reduction is performed in the steady-state context, largely ignoring dynamics, or performed offline where the scenarios may be different from real-time conditions. This has served the power system reasonably well when the behaviors are more predictable. However, this practice is no longer adequate, as the power system is becoming increasingly dynamic and nonlinear due to the new mix of intermittent generation and intelligent consumption. Model reduction must evolve to handle these nonlinear and dynamic behaviors.

Dynamic model reduction has been studied extensively. Coherency is the most common concept adopted to identify groups of dynamic devices, e.g., generators, for model reduction purposes. Specifically, a group of generators, $G = \{g_1, \dots, g_m\}$, is coherent if their difference in voltage angles is constant over time, i.e., there exists a $c \in \mathbb{R}$ such that $\delta_{g_i}(t) - \delta_{g_j}(t) = c$ for all $t \geq 0$ and $g_i, g_j \in G$. Coherency

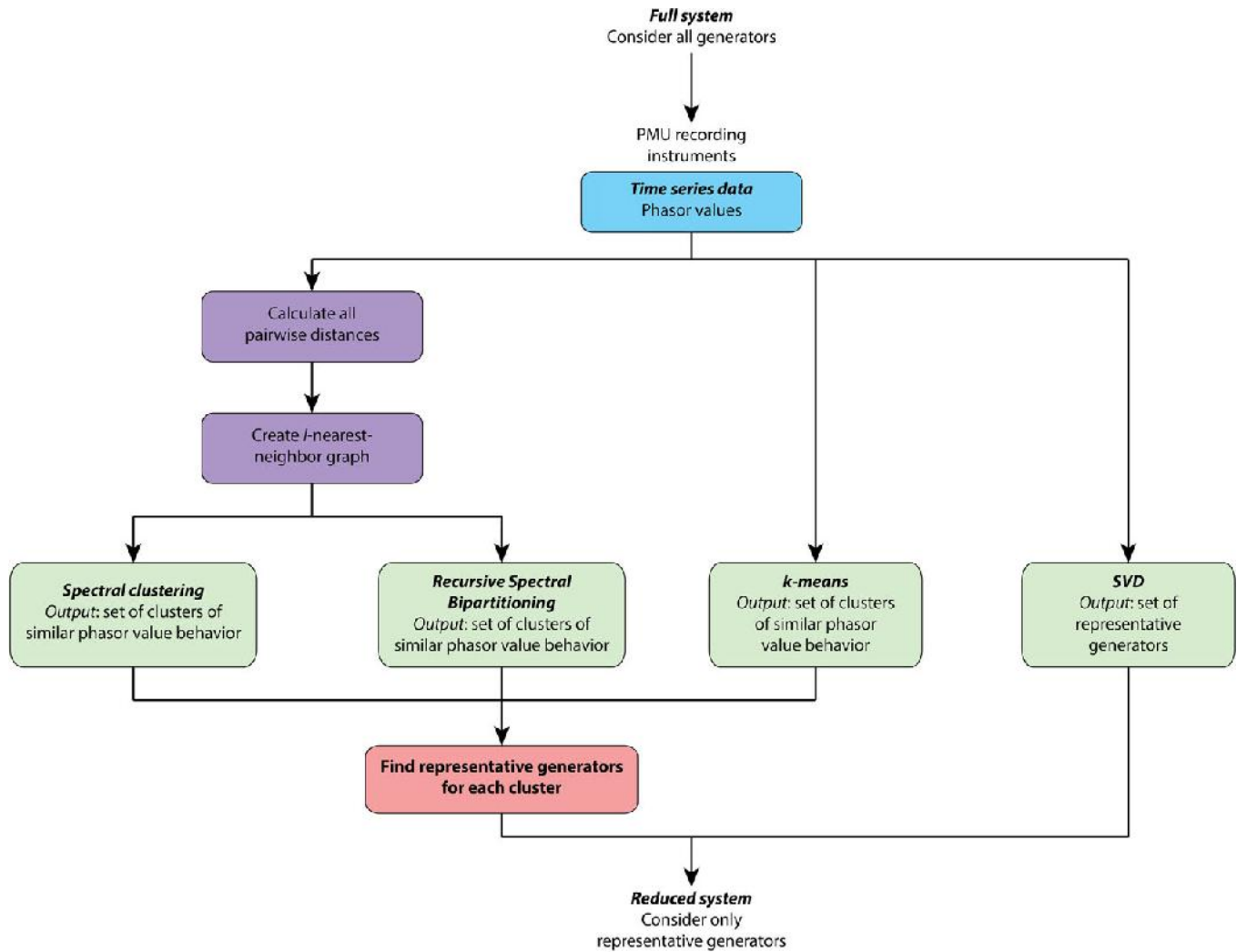


FIGURE 1 Workflow showing the four basic reduction methods compared in this paper

can be determined using a linearized model around an operating point [21, 24] or by analyzing offline simulated dynamics [22]. Either way, it is not capable of capturing real-time operating conditions, which renders the reduced model useless for timely analysis.

Coherency is inherently a clustering-based reduced-order model (ROM) method—as are the majority of the methods discussed in this paper. However, it is important to point out that there are many other types of ROM methods. Subspace projection methods, which find a simpler subspace of the full model with certain approximation guarantees, are very common. Proper orthogonal decompositions (PODs) [19] and Krylov subspace methods [2] are two examples of subspace projections. One method discussed in this paper, singular value decomposition (SVD), is also an example of a subspace method. Beyond clustering and subspace methods, there are also linearization techniques that are used to reduce the variable space of the dynamic power equations directly [10].

Recent developments and deployment of phasor technologies present an opportunity to perform dynamic model reduction in real time because high-speed phasor measure-

ments capture the majority of the power system dynamics necessary for power system operation and planning purposes. Ref. [29] proposes an SVD-based method that can be used for real-time dynamic model reduction. This method also preserves a certain level of nonlinearity in the reduced model. Following this line of research, our team continues developing real-time dynamic model reduction techniques using graph clustering methods and compares the accuracy of reduced models with the SVD-based approach, as well as traditional *k*-means clustering.

Real-time phasor measurements are used to cluster generators based on similar behaviors, and representative generators are chosen from each cluster. The final reduced model contains only the representative generators. We investigate two graph clustering methods—recursive spectral bipartitioning [25] and spectral clustering [18]. Variants of implementations of these two methods are tested alongside the SVD and *k*-means algorithms on fault scenarios within the IEEE 50-machine [4] and 16-machine systems. For reference, Figure 1 contains a workflow diagram showing each method found in the remainder of this paper. Error measurements quantify various levels of accuracy for each method,

but many possess accuracy that is adequate for power system operation and planning purposes. From a comparative perspective, this paper provides a good reference point for practical implementations.

The remainder of the paper is organized as follows: section 2 describes the system and data used in implementing and comparing the proposed model reduction methods. Sections 3 and 4 provide an overview of the clustering techniques and representative generator identification, including their computational complexity. Section 5 presents the comparative study approach and results, and Section 6 concludes the paper with suggestions for real-world use of these algorithms and future research directions.

2 | TEST SYSTEMS AND DATA

There are many types of data that can be collected from power grid systems. This paper employs data collected by phasor measurement units (PMUs). PMUs, also called synchrophasors, collect data synchronously across the power grid, providing an “online” data stream. These units are deployed to many systems throughout the grid and are time-synchronized so that measurements taken at different locations can be correlated. This work focuses on phasor data and the angle of the terminal voltage. These data are collected at millisecond resolution, namely 100 samples per second, and afford a picture of voltage angle oscillation at each generator over time. For each generator, a time series of phasor values is collected, $\vec{\delta}_i = \langle \delta_i^{(1)}, \delta_i^{(2)}, \dots, \delta_i^{(m)} \rangle$, where $\delta_i^{(j)}$ is the phasor value of generator i at the j th time step. In both test systems, data are collected for 3.8 s, $m = 380$, but, for generality, the symbolic parameter m is used throughout the paper. In future work, a study is planned to determine how many samples are needed to yield a useful reduced-order model.

2.1 | Evaluated test systems

Two small model systems are used for method validation. Specifically, the IEEE 145-bus, 50-generator system [4] (Figure 2), referred to as system S_1 , and the IEEE 68-bus, 16-generator system, system S_2 (Figure 3). In large power networks, such as the Eastern or Western Interconnects in the United States, individual power companies only control small areas, known as service territories. All these small areas are interconnected to form the entire grid. Each power company considers the generators and buses in their service territory to be their internal system, while the remaining generators and buses are external. Companies prefer to model their own internal generators fully and use model reduction to determine a simpler approximation for the external system, as it typically is much larger. In S_1 , there are $I_1 = 16$ internal system generators and $E_1 = 34$ external generators, while in S_2 , there are $I_2 = 7$ internal and $E_2 = 9$ external. In S_1 , generator 37 at bus 130 in the internal area is chosen as the reference machine,

meaning it is treated as if its phasor angle is always 0. All other machines’ phasor values are measured as deviations from the reference machine. In S_2 , generator 16 is used as the reference.

All generators are modeled using the classical model for machine dynamics with a second-order swing equation. In power systems, following a disturbance, some state variables decay fast, called fast variables, such as those from excitation systems. Meanwhile, others, known as slow variables, decay slowly, for example, rotor angle and speed. Oscillations of slow variables are determined by machine inertias and are well captured by the classical (second-order) model [5]. Moreover, slow variables dominate power oscillations in the power system. Therefore, the classical model is used in this work. Figure 4 illustrates the difference between the classical model response (solid blue line) and high-order model response (dotted red line).

For the tests on S_1 , simulated PMU datasets are created using $F_1 = 5$ different three-phase short-circuit faults within the system. The faults last for 60 ms. Then, the line is tripped to clear the fault. In S_2 , $F_2 = 3$ faults are simulated. Post-fault oscillations of the phasor values for the external system generators are recorded. The next section describes the methods used for model reduction to determine sets of representative generators.

3 | IDENTIFYING REPRESENTATIVE GENERATORS

This section describes four methods for identifying representative generators. These representative generators are meant to exhibit different types of dynamic behavior and will be used in reduced-model simulations (described in section 4). Recalling the workflow given in Figure 1, which identifies each model reduction method, this section describes their basic details. For full descriptions of these methods, refer to other detailed papers [14, 29]. The computational complexity of each method is also discussed. Table 1 summarizes the four methods in terms of their parameters and computational complexity. Runtimes for all methods on a standard desktop computer are less than 1 s for both test systems considered.

3.1 | Singular value decomposition (SVD)

SVD’s goal in the power grid model reduction setting is to find a subset of external generators, known as representative generators, whose dynamic responses, the $\vec{\delta}_i$ vectors, are as close to orthogonal as possible. The more orthogonal the representative generators are, the larger their span. Therefore, there is a better chance that the linear span of these representative vectors will contain the dynamic response vectors for the rest of the generators.

SVD goes far beyond this application and is a general method of matrix factorization into two unitary matrices and one diagonal matrix [11]. Given an initial $m \times n$ matrix A ,

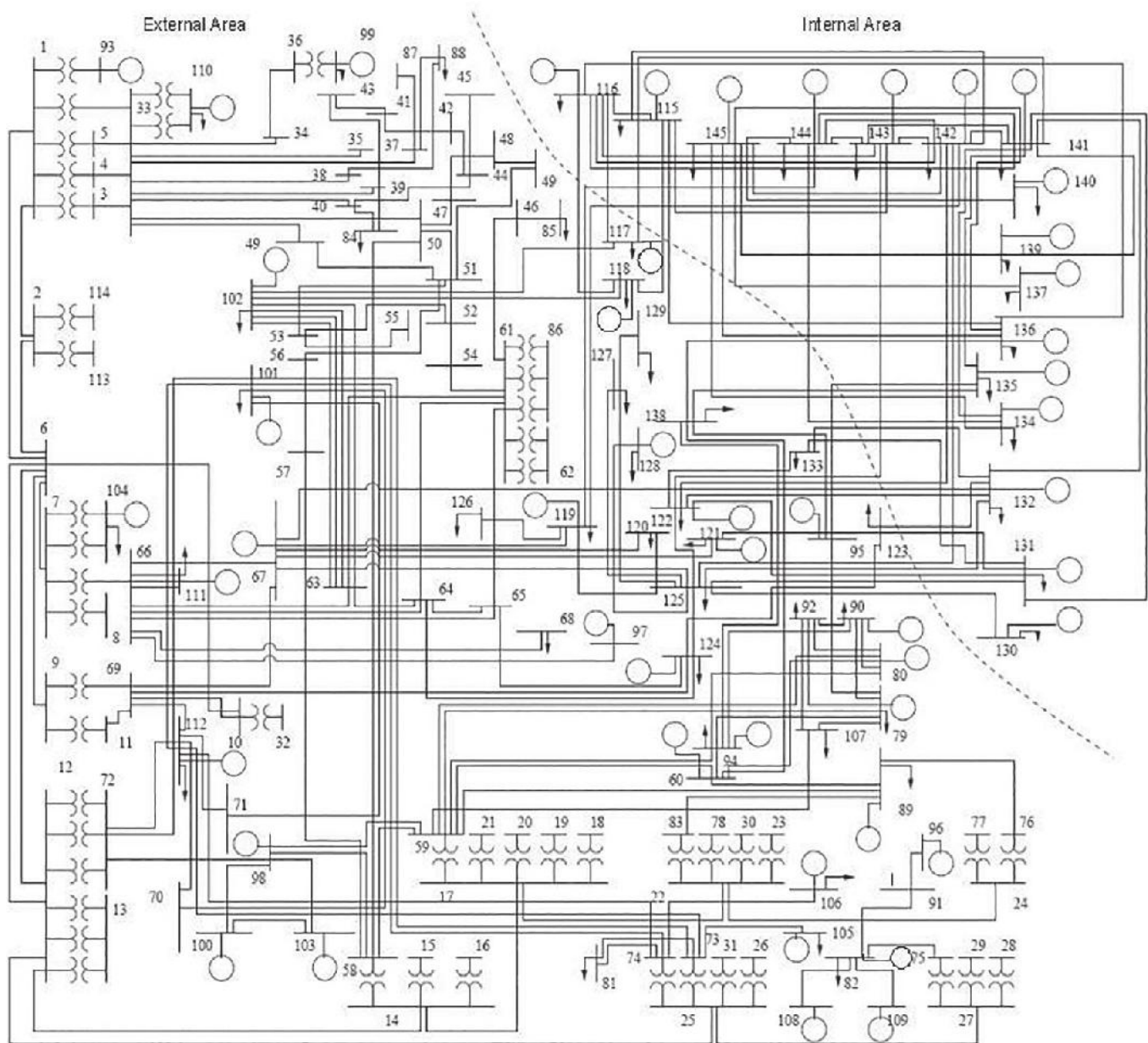


FIGURE 2 IEEE 50 generator system, S_1 . In this image, the 50 generators in the system are represented by circles, and the buses are numbered 1 through 145

the SVD method factorizes A into a product of three matrices $A = U\Sigma V^*$, where U ($m \times m$) and V ($n \times n$) are unitary and Σ is an $m \times n$ diagonal matrix. The columns of $U\Sigma$ are the principal components of A , while the diagonal values of Σ are singular values. Singular values are often used to determine how many principal components to choose because smaller singular values tend to contribute only noise to the principal components. SVD is used across many different domains, including audio verification [23] and evolutionary genomics [3]. This work focuses on the algorithm described in Ref. [29], where SVD is used for power grid model reduction.

Recall that, for each external generator i , a time series of phasor values is collected from a PMU and normalized to form the vector $\vec{\delta}_i = \langle \delta_i^{(1)}, \delta_i^{(2)}, \dots, \delta_i^{(m)} \rangle$, where $\delta_i^{(j)}$ is the normalized phasor value of generator i at the j th time step, and m is the number of time steps. This vector represents the dynam-

ics of generator i following a disturbance. The normalization is done in a standard way by subtracting the mean and dividing by the standard deviation for each $\vec{\delta}_i$ separately. Let $n = E_i$ be the number of external generators (in system S_i) and define the matrix δ to have $\vec{\delta}_i$ as column vectors.

$$\delta = \begin{bmatrix} | & | & & | \\ \vec{\delta}_1 & \vec{\delta}_2 & \dots & \vec{\delta}_n \\ | & | & & | \end{bmatrix}.$$

Given this formulation, an SVD is performed on the matrix δ , writing $\delta = U\Sigma V^*$. The first k principal components are found by taking the $k \times k$ submatrix of Σ which has the k largest singular values, denoted Σ_k , along with the corresponding k columns of U , denoted U_k . It can be written as

$$U_k \Sigma_k = X_k = \begin{bmatrix} | & | & & | \\ \vec{x}_1 & \vec{x}_2 & \dots & \vec{x}_k \\ | & | & & | \end{bmatrix}$$

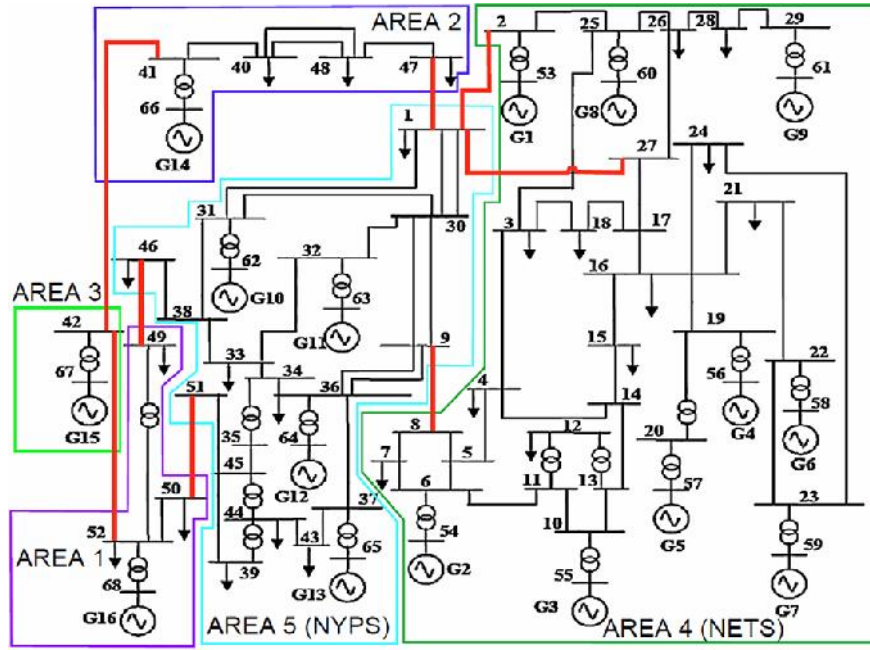


FIGURE 3 The IEEE 16 generator system, S_2 , as shown in [32]. Again, the circles are generators, labeled G1 through G16, and the buses are numbered 1 through 68

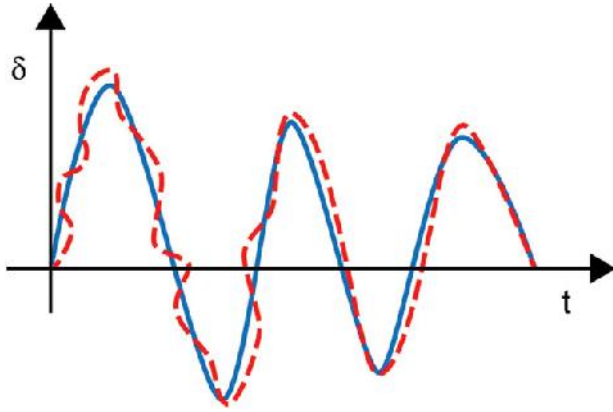


FIGURE 4 Illustration of the difference between the classical model response (solid blue line) and the high-order model response (dotted red line)

TABLE 1 Summary of model reduction methods investigated. Refer to each corresponding section for a detailed description of the parameters

Method	Parameters	Computational complexity
SVD	k	$O(n^2m + nm^2 + m^3 + nk)$
k -Means	k	$O(nk^2i)$
Recursive spectral bipartitioning	k ; ℓ ; "median" or "zero"; "size", "sum," or "avg"	$O((k-1)n^3)$
Spectral clustering	k ; ℓ	$O(n^3)$

where $\vec{x}_i$ is the i th principal component. The computational complexity of computing the principal components is $O(n^2m + nm^2 + m^3)$ [12]. For both test systems considered in this paper, the number of time steps, m , is much larger than the number of generators, n , so the m^3 term will dominate. However, this generally may not be the case, particularly in an online process where a few time steps from the recent history on a large system are used.

The two plots in Figure 5 show the singular values calculated for systems S_1 (a) and S_2 (b). Because smaller singular values can contribute noise to the principal components, these plots are used to pick the cutoffs of 4 to 15 in system S_1 and 3 to 6 in system S_2 .

Once the principal components X_k are computed, the similarity between $\vec{\delta}_i$ s and $\vec{x}_j$ s is analyzed. Namely, for each $\vec{x}_j$, find the $\vec{\delta}_i$ closest in the Euclidean distance and choose that $\vec{\delta}_i$ to be one of the representative generators. This yields the k vectors $\vec{\delta}_i$, with the highest similarity to an $\vec{x}_j$, forming the set of k representative generators. The complexity of this step is $O(nk)$ because each of the n phasor vectors must be compared to each k principal components.

3.2 | k -Means

Because the PMU data are vector data and can be thought of as points in $\mathbb{R}^m$ for some m , k -means clustering is a natural choice for comparison. k -Means is a standard recursive clustering technique for data in $\mathbb{R}^m$ [26] and is widely used [15, 17]. In each recursion step, the algorithm computes the centroids of each current cluster and then reassigns points to the cluster whose centroid is closest to it. An initialization step is needed to choose the initial centroids, which is often done by choosing k points randomly from the original dataset. The algorithm then runs as follows: assign each data point to the cluster whose randomly chosen centroid is closest, recompute centroids of the current clusters, reassign points to clusters based on distance to new centroids, recompute centroids, and so on. This is repeated for some predetermined number of iterations, or until the clusters do not change and the algorithm has converged.

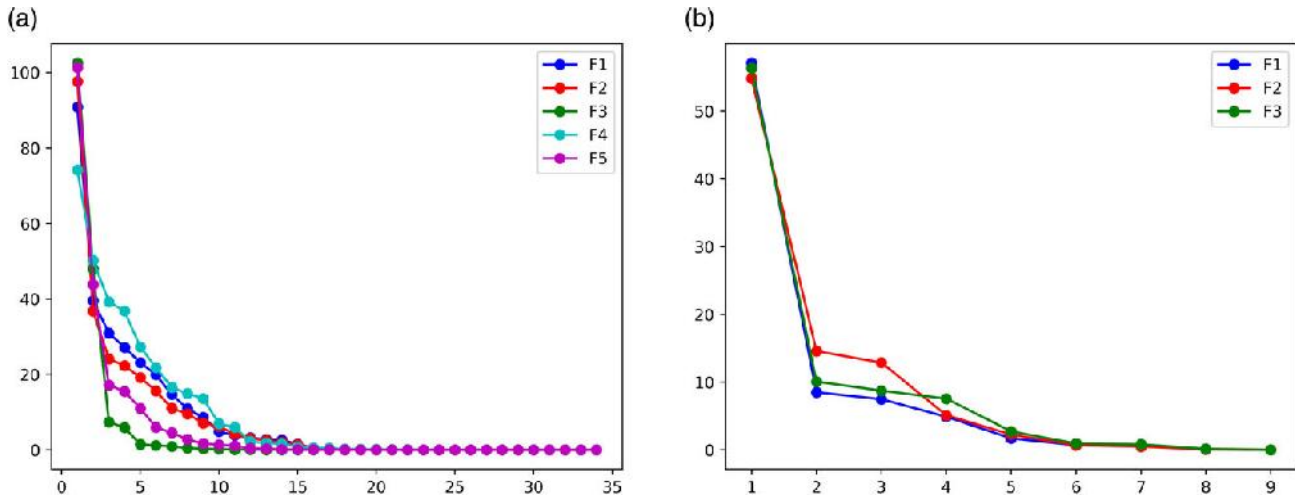


FIGURE 5 (a) Singular values for all five fault scenarios in the IEEE 50 system, system S_1 . (b) Singular values for all three fault scenarios in the IEEE 16 system, system S_2

One problem with k -means is that there is no guarantee that it will terminate in a globally optimal clustering. k -Means uses gradient descent, or a similar optimization algorithm, at each step. These are well known to have the possibility of getting stuck in a locally optimal clustering. Because the initialization step is done randomly, there can be multiple clusterings from the same input data, all being local optima. This problem can be mitigated by running k -means clustering multiple times with different random initializations and choosing the resulting clustering that minimizes an objective function. The chosen objective function is the residual sum of squares defined as

$$RSS = \sum_{i=1}^n \|\vec{\delta}_i - \mu(\vec{\delta}_i)\|^2,$$

where $\mu(\vec{\delta}_i)$ denotes the centroid for the cluster that contains $\vec{\delta}_i$ and the norm is taken to be the Euclidean norm. We use the scikit-learn python implementation of k -means, which runs a default of 10 times and chooses the clustering with the smallest RSS value.

This local optimum problem is also encountered in the spectral clustering method, which uses k -means. However, it is not an issue for SVD or recursive spectral bipartitioning. The advantage of k -means, however, is that it is the fastest of the algorithms considered in this paper. The complexity of k -means is $O(nk^2i)$, where i is the number of iterations required to converge. As k will be less than n , this is faster than the $O(n^3)$ graph clustering algorithms.

3.3 | Graph clustering

Recursive spectral bipartitioning and spectral clustering methods are not new, but they have not previously been used to perform power grid model reduction. We have made modifications that will allow for fine-tuning by an operator.

First, the graph construction method from PMU data must be described. When setting up a graph clustering problem, the graph vertices are chosen as the objects being clustered (in this case, the generators), while the edges will indicate an amount of similarity between vertices. For each generator, consider its phasor value data vector, $\vec{\delta}_i \in \mathbb{R}^m$, where m is the number of time steps recorded, and calculate a distance matrix $D = (d_{ij})_{i,j=1}^n$. The entries in D are given by the Euclidean distance between $\vec{\delta}_i$ and $\vec{\delta}_j$, $d_{ij} = \|\vec{\delta}_i - \vec{\delta}_j\|_2$. Once this matrix has been created, an ℓ -nearest-neighbor graph is formed by connecting each generator (vertex) to its ℓ closest generators. Alternate distances and graph constructions also can be used (see Ref. [14] for more details).

Both types of graph clustering will use the spectral (eigenvalue) properties of the weighted Laplacian matrix associated with the graph. The weighted Laplacian, $L = (L_{ij})_{i,j=1}^n$, is defined as follows:

$$L_{ij} = \begin{cases} \sum_{k \neq i} w_{ik} & i = j \\ -w_{ij} & i \neq j. \end{cases}$$

The entries on the diagonal, L_{ii} , are given by the sum of all edge weights on edges incident to vertex i . Off-diagonal entries, L_{ij} , are equal to the negative weight on edge e_{ij} . If an edge is absent, it is treated as an edge of weight zero. An edge's weight is defined as the similarity score between the endpoint vertices based on their distance (high distance means low similarity, and vice versa). In particular, let $w_{ij} = e^{-(d_{ij}^2/2)}$ be the Gaussian similarity between generators i and j . Other similarity functions, also known as kernels, may be used, but the Gaussian function is fairly standard [18].

Creating the graph and the Laplacian matrix has complexity $O(n^2)$. Begin by calculating all pairs of distances between the n generators. This will be $\frac{n(n-1)}{2}$ distance calculations and dominates the complexity. Then, to construct the ℓ -nearest neighbor graph, for each vertex, consider all other vertices and connect to the ℓ closest neighbors. This requires sorting the sets of neighbors for each of the n vertices. Therefore, cre-

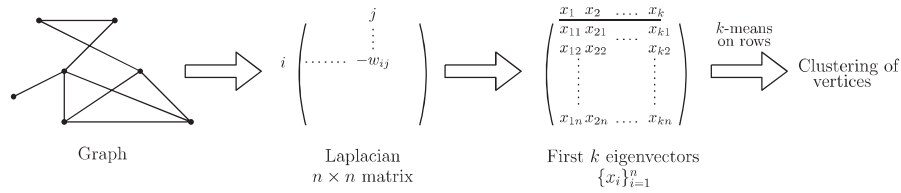


FIGURE 6 Illustration of the steps in spectral clustering

ating an ℓ -nearest neighbor graph can be done in $O(n \log n)$ time after the distances have been computed.

3.3.1 | Recursive spectral bipartitioning

The most basic type of spectral graph clustering or partitioning is recursive spectral bipartitioning [20]. This algorithm uses the eigenvector for the second smallest eigenvalue of the weighted Laplacian matrix. Clearly, as each row of the Laplacian matrix sums to zero, there is a zero eigenvalue. It is not difficult to show that L is positive semi-definite. Thus, zero is, in fact, the smallest eigenvalue, and its multiplicity is the number of connected components in the graph. The second smallest eigenvalue is the algebraic connectivity of the graph, and its associated eigenvector, commonly called the Fiedler vector after Miroslav Fiedler who first defined the theory of algebraic connectivity and its relation to graph partitioning [8, 9], has properties that define a partition of the graph vertices into two groups. The Fiedler vector contains positive, negative, and zero values. By partitioning the associated vertices into two sets—one where the value in the Fiedler vector is negative and the other in which it is positive (splitting the zero values among both or putting them all in one of the two sets)—a graph partition is obtained that minimizes the sum of the edge weights between the two partitions, and where both subgraphs are connected [8, 9]. Traditionally, the graph vertices are partitioned into those with positive values and others with negative values because of the property that both subgraphs are connected. However, this can lead to unbalanced partitions because there is no guarantee that half of the vertices will have positive values while the other half will be negative. To construct more balanced partitions, the split can be made based on the median of the Fiedler vector. This choice only guarantees that one of the two induced subgraphs is connected. However, since the partition will be recursively continued, having disconnected induced subgraphs should not create problems because they will have the opportunity to split eventually. Both these splitting methods have been tested and are reported as part of the comparison in section 5. In particular, splitting at zero often gives degenerate partitions into fewer than k sets.

Using the Fiedler vector to partition the graph vertices into two disjoint sets is only the first step. A second degree of freedom in this method is how to continue the recursive partitioning. In traditional recursive spectral bipartitioning, both the sets are further partitioned using the Laplacian of the subgraph induced by each set of vertices. If this process of

repeatedly splitting each set is continued for N steps, it will yield a partition with 2^N sets. A more targeted approach to the recursive splitting is explored to achieve any number of sets rather than just powers of 2. Instead of arbitrarily splitting each set of the partition into two at each step, a search is performed among all current sets for one that is the least tight. We define "tight" as the sum of all pairwise distances in that set, the average of all pairwise distances, or simply the size of the set. These three possible tightness schemes are considered, in addition to the two methods for splitting the Fiedler vector (zero or median). All the results are summarized in Ref. 5. In the remainder of this paper, recursive spectral bipartitioning may be referred to as Fiedler partitioning because of the prominent use of the Fiedler vector.

The computational complexity of recursive spectral bipartitioning is dominated by the Fiedler vector calculation, which has complexity $O(n^3)$. This must be done many times, and each time a set is split into two. Therefore, the complexity is $O((k-1)n^3)$.

3.3.2 | Spectral clustering

For general spectral clustering, more than just one eigenvector of the weighted Laplacian is used. Instead of using only the Fiedler vector, the first k eigenvectors are considered. Each entry of an eigenvector corresponds to a vertex in the graph from which the weighted Laplacian was formed. An $n \times k$ matrix can be formed where the columns are the first k eigenvectors. Each row can be thought of as a new vector representation for each of the vertices. k -Means is then used to cluster these new vector representations, thereby clustering the vertices themselves. The pipeline illustration in Figure 6 shows this sequence. In this case, eigenvectors are only calculated once, at $O(n^3)$ complexity, followed by a single k -means calculation. As the k -means complexity is less than n^3 , the total complexity of spectral clustering is dominated by the n^3 term. For a more in-depth discussion of spectral clustering, including other variants not investigated here, see [18].

3.4 | Choosing cluster representatives

Both k -means and the graph methods produce a partition of the generators into clusters of similar dynamic behavior. However, in the context of model reduction, we need a set of representative generators, as produced by the SVD method. This is achieved by choosing one representative generator from each cluster: the medoid.

For each cluster, $C_i = \{\delta_{ij}\}_{j=1}^{|C_i|}$, so it is preferable to choose the average time series, the true centroid:

$$\delta_{C_i} = \left\langle \frac{\sum_{j=1}^{|C_i|} \delta_{ij}^{(k)}}{|C_i|} \right\rangle_{k=1}^m.$$

However, this true centroid is unlikely to correspond exactly to the time series of phasor values for a generator in the given cluster. Instead, the generator whose time series is closest to this centroid in the Euclidean distance is chosen. This element closest to the true centroid is known as the medoid of the cluster.

4 | FROM REPRESENTATIVE GENERATORS TO A REDUCED MODEL SIMULATION

Once a set C of representative generators has been produced from any one of the methods described in section 3, a reduced model simulation can be performed. For each non-representative generator, $\vec{\delta}_i \notin C$, coefficients $\alpha_j \in \mathbb{R}$ are found via regression such that $\|\vec{\delta}_i - \sum_{\vec{\delta}_j \in C} \alpha_j \vec{\delta}_j\|_2$ is minimized. In other words, the phasor values for each nonrepresentative generator are approximated as a linear combination of the phasor values for the representative generators. Then, only the set of representative generators are simulated, by solving power flow equations, and responses for the remaining generators are approximated using the same linear combinations of the responses for the representative generators.

5 | COMPARATIVE STUDY APPROACH AND RESULTS

In this section, a measure that quantifies the amount of error between a reduced model and the full system is defined. In addition, an overview of performance profiles, the method for comparing the error in reduction methods across multiple scenarios, is provided. Finally, performance profiles are used to compare the reduction methods described in the previous section.

5.1 | Measures for comparison

To judge the error of a particular model reduction method, M , for test system S_x ($x = 1, 2$) under fault scenario $1 \leq u \leq F_x$, first the full system of external and internal generators (E_x external + I_x internal) is simulated and then the reduced system (r representative external + I_x internal, depending on the reduction ratio r). Recall that the internal system represents the set of generators and buses owned by a particular power company of interest and are not reduced, while the external machines are interconnected but owned by others. Responses,

or phasor values, of the I_x internal generators are recorded during both full and reduced simulations. Let $\delta_{u,i}^f(t)$ be the phasor value response of the i th internal generator in the full simulation at time t following fault scenario u , and let $\delta_{u,i}^{M,r}(t)$ be the same for the reduced model. Notice that $\delta_{u,i}^f$ does not have the M superscript because it is independent of any reduced model. Then, for each internal system generator, define the following metric to measure the mismatch of response curves of the full and reduced systems:

$$J_{u,r}^M(i) = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} \left| \delta_{u,i}^{M,r}(t) - \delta_{u,i}^f(t) \right| dt, \quad (1)$$

which is the L_1 norm between vectors $\delta_{u,i}^{M,r}$ and $\delta_{u,i}^f$. For example, $J_{3,7}^{\text{SVD}}(38)$ for system S_1 is the error on internal generator $i = 38$ for fault scenario $u = 3$ using the $M = \text{SVD}$ model reduction method to $r = 7$ generators. While using the L_2 norm or some other L_p norm is an option, the results are nearly identical in our test cases. Hence, only one is presented.

To simplify the comparisons, define

$$J_{u,r}^M = \frac{1}{I_x} \sum_{i \in I_x} J_{u,r}^M(i),$$

to be the average $J_{u,r}^M(i)$ values over all internal generators. Of note, there is a slight abuse of notation in the sum over $i \in I_x$. Here, we use I_x to mean both the set of internal generators and the number of internal generators, as was originally defined.

5.2 | Performance profiles

The comparison method we use, perfprof (for “performance profile”) [6, 7, 13], comparatively plots the performance of different algorithms against each other. This type of analysis is typically used when comparing the runtimes of multiple algorithms against each other, preferring low runtime over high. This strategy is adopted here because the premise is the same: to choose the reduction method that most often across multiple comparable tests (fault scenarios) has smallest error value. Comparisons will only be made within the same reduction ratio r and within the same test system. Once a system ($x = 1$ or 2) and an r have been chosen, there are nine methods to compare across F_x fault scenarios, which are called tests. Table 2 summarizes these values for the example where $x = 2$ and $r = 5$.

Note that some of the methods did not return a reduced model for all r values in a given fault scenario, so some of these data may be missing. This is because the clustering may have degenerated into more than r clusters. For example, if the graph clustering resulted in r clusters where one of the clusters consisted of only v isolated vertices, the algorithm treats it as $r + (v - 1)$ clusters. If this is the case, we let the corresponding $J_{u,r}^M$ measure be a value larger than any of the other values returned in the table. This indicates that it did a “bad job” at that particular reduction ratio in that test or scenario. Too many of these degenerate reductions reflects poorly on

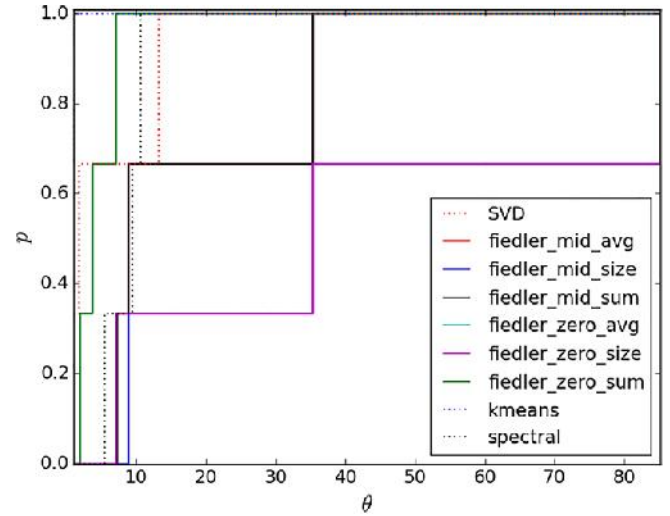
TABLE 2 Summary of the values compared for system S_2 and reduction ratio $r = 5$

M	Test 1	Test 2	Test 3
SVD	$J_{1,5}^{\text{SVD}}$	$J_{2,5}^{\text{SVD}}$	$J_{3,5}^{\text{SVD}}$
k -Means	$J_{1,5}^{k\text{-means}}$	$J_{2,5}^{k\text{-means}}$	$J_{3,5}^{k\text{-means}}$
Fiedler, zero, sum	$J_{1,5}^{\text{Fzsu}}$	$J_{2,5}^{\text{Fzsu}}$	$J_{3,5}^{\text{Fzsu}}$
Fiedler, zero, avg	$J_{1,5}^{\text{Fza}}$	$J_{2,5}^{\text{Fza}}$	$J_{3,5}^{\text{Fza}}$
Fiedler, zero, size	$J_{1,5}^{\text{Fzsi}}$	$J_{2,5}^{\text{Fzsi}}$	$J_{3,5}^{\text{Fzsi}}$
Fiedler, mid, sum	$J_{1,5}^{\text{Fmsu}}$	$J_{2,5}^{\text{Fmsu}}$	$J_{3,5}^{\text{Fmsu}}$
Fiedler, mid, avg	$J_{1,5}^{\text{Fma}}$	$J_{2,5}^{\text{Fma}}$	$J_{3,5}^{\text{Fma}}$
Fiedler, mid, size	$J_{1,5}^{\text{Fmsi}}$	$J_{2,5}^{\text{Fmsi}}$	$J_{3,5}^{\text{Fmsi}}$
spectral	$J_{1,5}^{\text{spec}}$	$J_{2,5}^{\text{spec}}$	$J_{3,5}^{\text{spec}}$

the method and will lower the perfprof score, described next, as its error will always be much higher than the lowest error.

The performance profile method produces a plot with a tolerance factor θ on the x -axis and a proportion of the tests $0 \leq p \leq 1$ on the y -axis. Each of the model reduction methods corresponds to a staircase-shaped curve in the plot. Continuing to use Table 2 as an example, a point (θ, p) for method M in this example means that $J_{u,5}^M$ is within a factor of θ of the best $J_{u,5}$ in proportion p of the three tests. It is likely that the best method is different for each u , but the $J_{u,5}^M$ measurement produced by this model reduction method is within some factor of whatever is the best for each test. In particular, a point $(1, p)$ means that $J_{u,5}^M$ is the optimal value (i.e., within a factor of 1 of the best) in proportion p of the tests, and a point $(\theta, 1)$ denotes that $J_{u,5}^M$ is always (proportion $p = 1$) within a factor of θ of the optimal value. Figure 7 contains the perfprof plot for this specific case. The method Fiedler-zero-size (purple line) passes through the point $\sim(35, 0.66)$, meaning that for roughly 66% of the tests (u values), the Fiedler-zero-size $J_{u,5}^{\text{Fzsi}}$ value is within a factor of 35 of the best $J_{u,5}^M$. This factor of 35 may sound high, but these measures range between ~ 0.002 and 1, making the maximum possible factor around 500.

Given the perfprof plot for an S_x and r , observe that a line which stays close to the p -axis the longest and reaches $p = 1$ first is optimal. However, there may not be a single method that achieves both these objectives. Instead, we posit that generally higher and further left is better. Therefore, much like

**FIGURE 7** Performance profile for system S_2 and reduction ratio $r = 5$. A point (θ, p) for method M means that $J_{u,5}^M$ is within a factor of θ of the best J_u in proportion p of the three fault scenarios

judging the accuracy of a binary classification algorithm by the area under an ROC curve, we compare the accuracy of model reduction methods using area under its perfprof curve. Unlike ROC curves, the x -axis (θ) is not bound between 0 and 1. Instead, the maximum θ value is the maximum ratio between any two accuracy measures for the given reduction ratio. That is to say, for a given r , the maximum θ will be

$$\max \theta = \max_{M_1, M_2, u} \frac{J_{u,r}^{M_1}}{J_{u,r}^{M_2}},$$

where the maximum is taken over all pairs of methods M_1 and M_2 and all fault scenarios u . To normalize the area, divide by the total possible area, which is $\max \theta$, as the y -axis maximum is 1. In the next section, this perfprof comparison method is used to determine optimal model reduction methods for each system and reduction ratio.

5.3 | Results of comparison

For system S_1 , reduction ratios $4 \leq r \leq 15$ are considered, and for system S_2 we let $3 \leq r \leq 6$. This totals $12 + 4 = 16$ different perfprof comparisons. Therefore, rather than includ-

TABLE 3 Relative areas under the perfprof curves for system S_1 , rows ordered by average value

Method	r												Avg
	4	5	6	7	8	9	10	11	12	13	14	15	
SVD	0.989	0.999	0.978	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.997
Fiedler-mid-sum	0.699	1.000	1.000	1.000	1.000	0.977	0.993	0.998	0.800	1.000	0.990	0.996	0.954
Fiedler-mid-size	0.699	1.000	0.982	0.994	0.999	1.000	0.800	1.000	0.994	0.994	0.990	0.996	0.954
Fiedler-mid-avg	0.699	0.975	0.975	1.000	0.775	0.991	1.000	1.000	0.988	1.000	0.991	0.756	0.929
Spectral	0.979	0.745	0.974	0.985	1.000	0.787	0.763	1.000	0.990	0.792	0.772	0.710	0.875
k -Means	0.779	0.745	0.965	0.985	1.000	0.787	0.960	0.992	0.996	0.792	0.772	0.710	0.874
Fiedler-zero-avg	0.755	0.880	0.911	0.733	0.631	0.765	0.753	0.598	0.777	0.787	0.763	0.710	0.755
Fiedler-zero-sum	0.755	0.680	0.711	0.733	0.631	0.765	0.753	0.598	0.777	0.787	0.763	0.710	0.722
Fiedler-zero-size	0.755	0.680	0.711	0.733	0.631	0.765	0.753	0.598	0.777	0.787	0.763	0.710	0.722

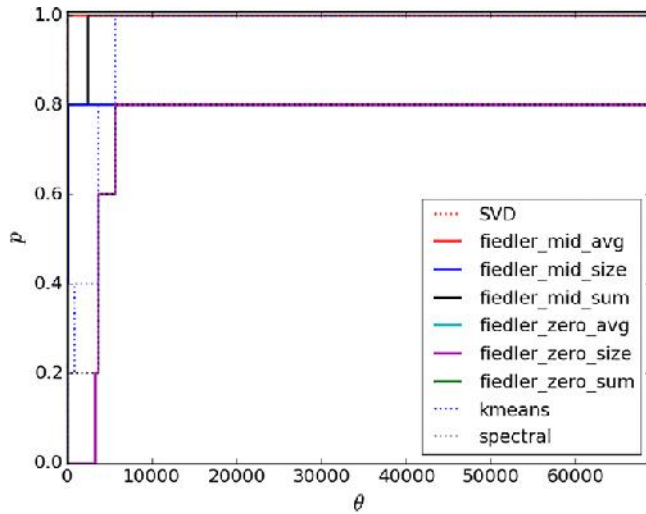


FIGURE 8 Perfprof plot for system S_1 and $r = 10$. The curves for Fiedler-zero-avg and Fiedler-zero-sum are hidden under the purple Fiedler-zero-size curve, as all three did not return any reductions for $r = 10$. A point (θ, p) for method M means that $J_{u,10}^M$ is within a factor of θ of the best $J_{u,10}$ in proportion p of the five fault scenarios

ing all perfprof plots, two tables of normalized areas under the perfprof curves will be provided, one for each test system, showing all r values. In each table, values are rounded to three significant digits. In particular, all 1.000 values are rounded to that value and are not equal to it. To reinforce the tables, some perfprof plots will also be included.

5.3.1 | System S_1

In system S_1 , the methods Fiedler-zero-sum and Fiedler-zero-size only return reductions for $r = 4$ and Fiedler-zero-avg only for $r = 4, 5$, and 6. Therefore, those three methods have the worst performance in our comparisons. In this system, splitting the Fiedler vector at zero sometimes yields a degenerate partition, where one part is empty. Therefore, the splitting process reaches a stable point (nothing else can be split using the Fiedler vector) prior to obtaining the desired number of clusters.

Table 3 shows the normalized area measurements for system S_1 . The methods are in decreasing order by their average area under the curve over all r values. It is clear that SVD dominates in this case. However, notice that for $r = 5$ and 6, SVD does not yield the maximum area under the perfprof curve. In this case, it is beaten by Fiedler-mid-sum and Fiedler-mid-size, which are methods with the second and third largest average areas. Figure 8 shows an example perfprof plot for $r = 10$. The curves for Fiedler-mid-size (solid blue) and k -means (dotted blue) cross around (3000, 0.8), but the area under k -means is higher than that for Fiedler-mid-size, indicating that k -means would be preferred between the two for $r = 10$.

These perfprof area comparisons may seem far removed from the actual PMU phasor values. To further illustrate the comparisons, some PMU traces for full and reduced systems under four different methods are produced. Figure 9 contains

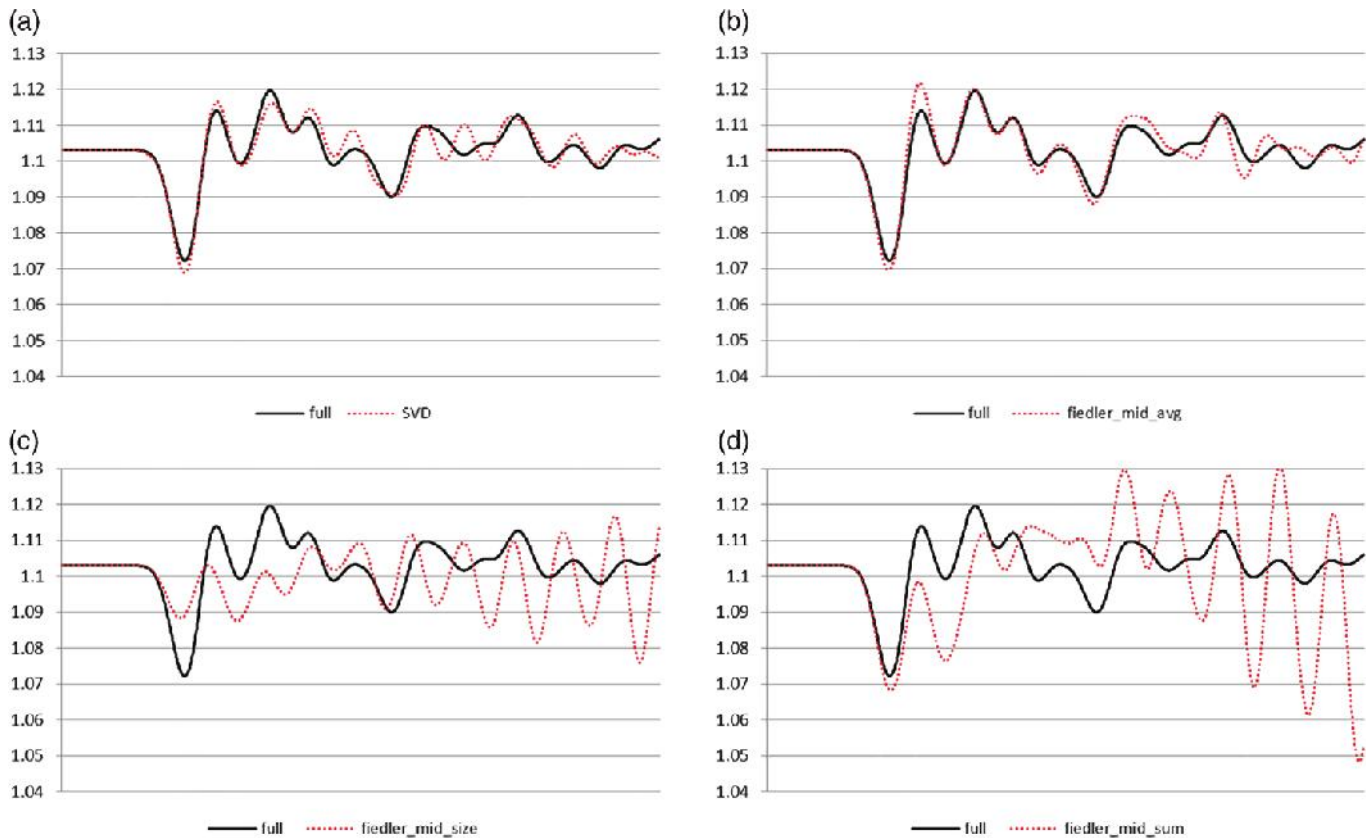


FIGURE 9 PMU traces for generator 45 in system S_1 with the first fault scenario under the full model (no reduction) and three separate reduced models for $r = 10$. (a) Full model vs SVD. (b) Full model vs Fiedler-mid-avg. (c) Full model vs Fiedler-mid-size. (d) Full model vs Fiedler-mid-sum

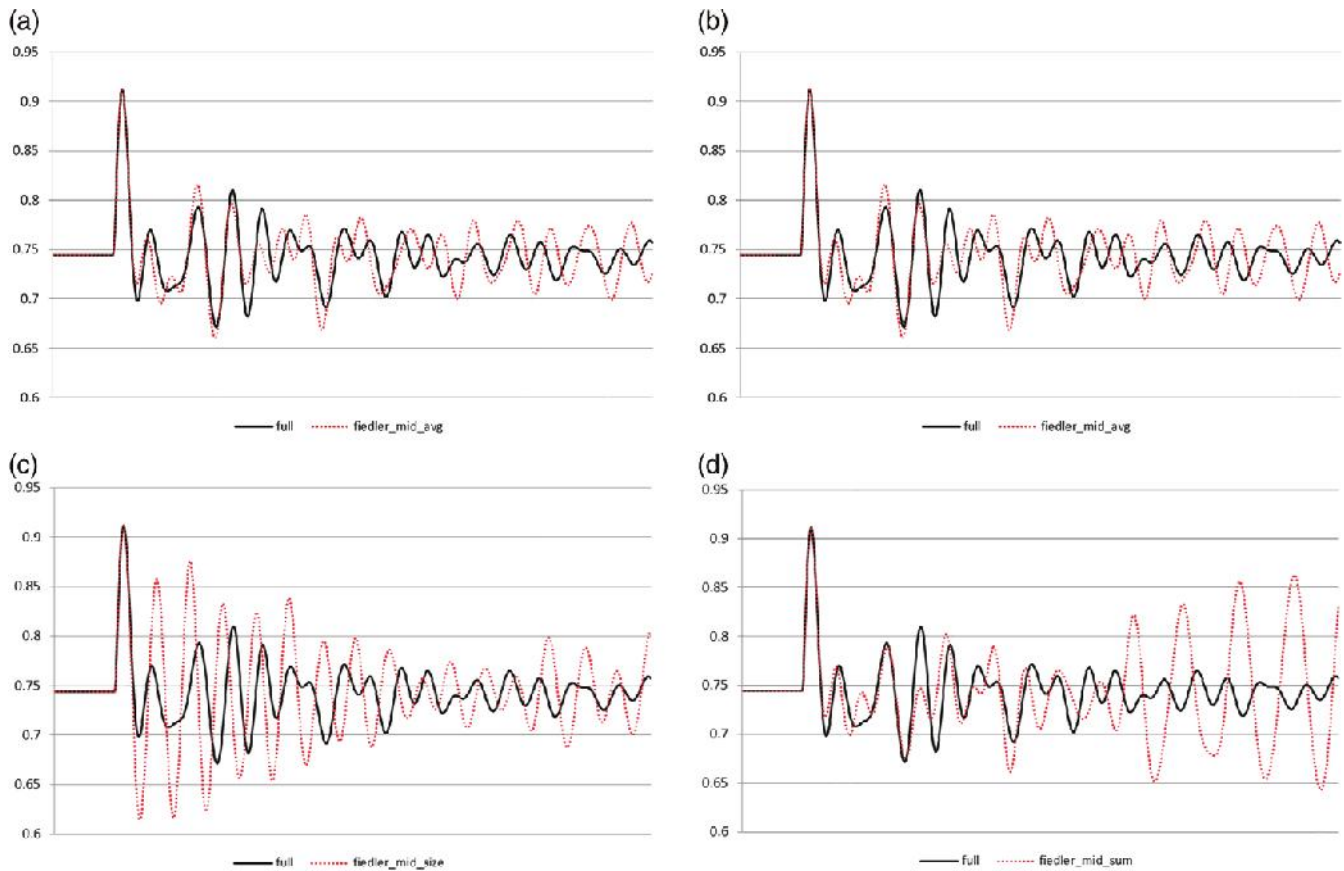


FIGURE 10 PMU traces for generator 29 in system S_1 with the first fault scenario under the full model (no reduction) and three separate reduced models for $r = 10$. (a) Full model vs SVD. (b) Full model vs Fiedler-mid-avg. (c) Full model vs Fiedler-mid-size. (d) Full model vs Fiedler-mid-sum

four PMU response comparisons for internal system generator #45 following fault scenario 1. In all four subfigures, the solid black line represents the phasor values over time for the full system without any model reduction, and the dotted red line represents the phasor values simulated using the indicated model reduction for $r = 10$. The $r = 10$ column in Table 3 indicates that SVD and Fiedler-mid-avg have the best reductions fairly consistently across all five fault scenarios. Fiedler-mid-sum scores very high, and Fiedler-mid-size does not perform as well. One might draw the same conclusions from looking at the PMU traces, though it may not be as clear-cut as it is when looking at the perfprof areas. In particular, Fiedler-mid-sum appears to perform very poorly, but its perfprof area remains quite high. Of course, this is only one of 16 internal generators and only a single fault scenario, so the Fiedler-mid-sum method must have performed much better for other generators.

Generator 45 is chosen because most methods seem to perform fairly well in this fault scenario on it. In contrast, generator 29 is one where most reductions perform fairly poorly. In Figure 10, similar PMU traces are shown for generator 29 in S_1 still under the first fault scenario. It is much more difficult to judge a well-performing vs a poorly performing reduction just by looking at these particular PMU trace plots, which is one of the reasons for choosing the more global perfprof areas as the comparison method.

TABLE 4 Relative areas under the perfprof curves for system S_2 , ordered by the average value

Method	r				Avg
	3	4	5	6	
k -Means	0.816	0.984	0.988	0.933	0.931
SVD	0.714	0.967	0.938	0.822	0.860
Fiedler-mid-avg	0.718	0.952	0.798	0.779	0.812
Fiedler-mid-sum	0.718	0.952	0.798	0.779	0.812
Spectral	0.646	0.962	0.900	0.479	0.747
Fiedler-zero-sum	0.350	0.943	0.950	0.684	0.732
Fiedler-zero-avg	0.350	0.943	0.950	0.684	0.732
Fiedler-zero-size	0.636	0.934	0.501	0.688	0.689
Fiedler-mid-size	0.618	0.640	0.493	0.656	0.602

5.3.2 | System S_2

System S_2 features a much smaller set with only nine generators in the external area. There still are a few degenerate cases where a method did not return a reduction for some r values, but it is not as widespread as in system S_1 . In addition, methods Fiedler-mid-avg and Fiedler-mid-sum always return identical reductions. For completeness, both methods are shown in the tables although they have exactly the same values.

As in the previous section, Table 4 contains the areas under perfprof plots summaries. In this system, the three Fiedler-zero methods perform poorly just as in S_1 . Yet, there

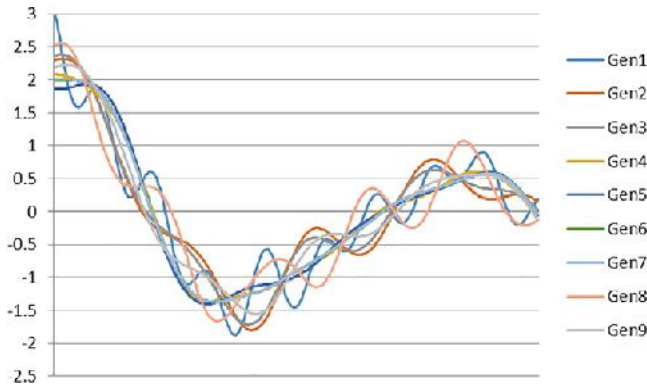


FIGURE 11 PMU phasor values for all nine generators in the internal system for S_2 over 380 readings, totaling 3.8 s, following fault scenario 1

are two major differences between S_1 and S_2 . Instead of SVD on top, k -means leads by a wide margin, whereas in S_1 k -means is not a top performer. Additionally, Fiedler-mid-size does very poorly in S_2 compared with S_1 .

Rather than providing PMU traces, an example partitioning of the input data is shown. In Figure 11, the full set of PMU phasor values recorded at all nine internal generators are displayed on the same axes. Then, Figures 12 and 13 show the clustering into four sets of generators for Fiedler-mid-sum and k -means, respectively. Although k -means did perform better overall in S_2 , looking at the clusters themselves, one could argue that the representative generators found using the Fiedler clustering seem more representative of the system as

a whole. In particular, k -means split up generators 4 and 5 from 6 and 7, even though they appear to be very similar as shown in Figure 12(b). Thus, having generators 4, 5, and 6 as representative generators in the k -means reduction might be redundant.

6 | CONCLUSION

In this paper, we provided a survey of three clustering techniques and an SVD algorithm for dynamic model reduction along with a comparison of these methods against two test systems: the IEEE-50 and IEEE-16 generator systems. We compared two graph methods—recursive spectral bipartitioning and spectral clustering—as well as an SVD method and the standard k -means clustering algorithm. Our analysis (detailed in Section 5.3) leads us to the following conclusions.

First, we remark that the k -means algorithm does not appear to be very reliable. Although it does perform well on the smaller system but did not rank highly in the larger system, we must conclude that either k -means is not as well suited for larger systems, or it is not expected to consistently work well. The latter statement is consistent with known problems using the k -means algorithm: the data must be sufficiently separated and distributed to yield useful conclusions. Thus, it seems that k -means, though sometimes quite suited to this problem, may not always be dependable.

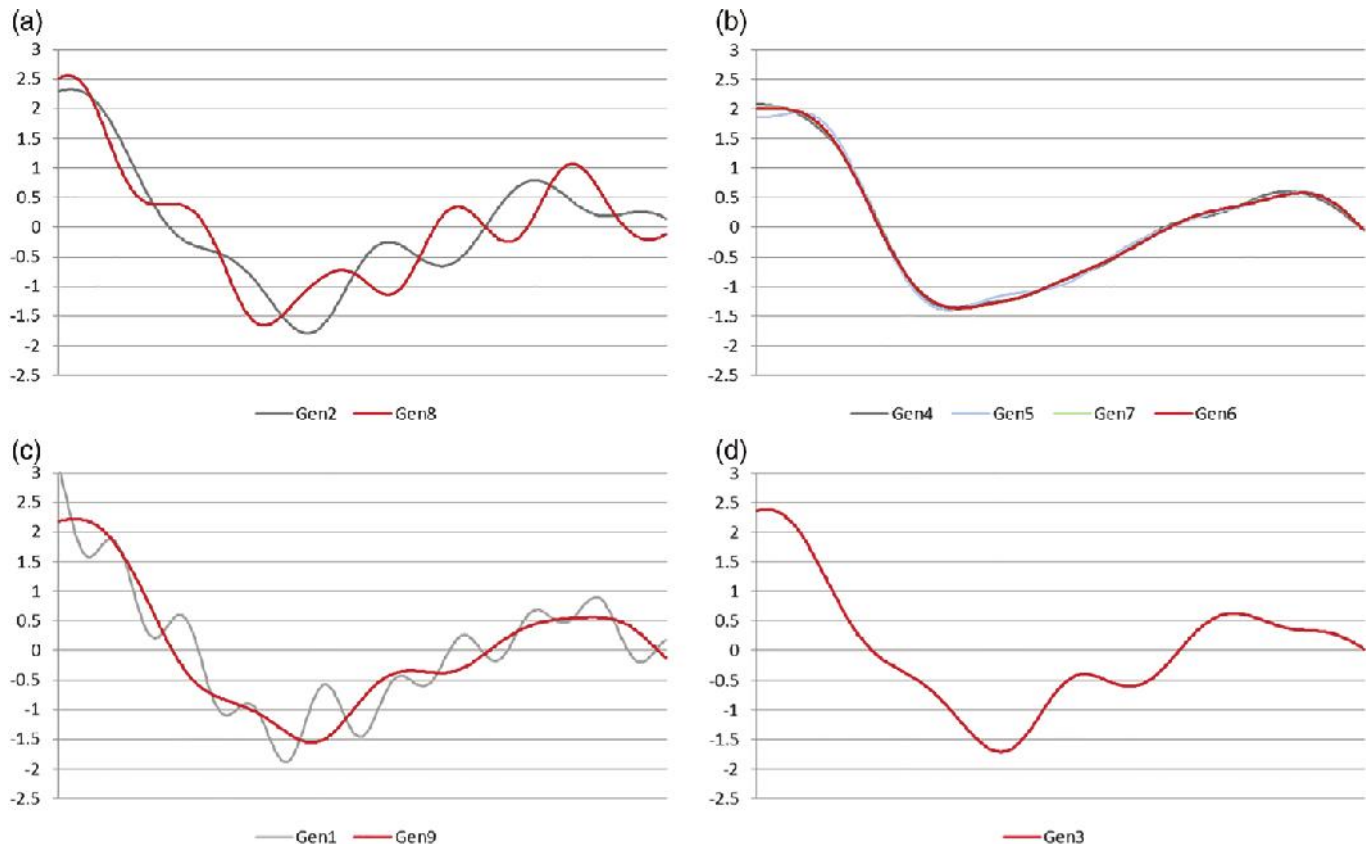


FIGURE 12 Clustering via Fiedler-mid-sum for system S_2 following fault scenario 1 with $r = 4$. Representative generators are in red in each plot

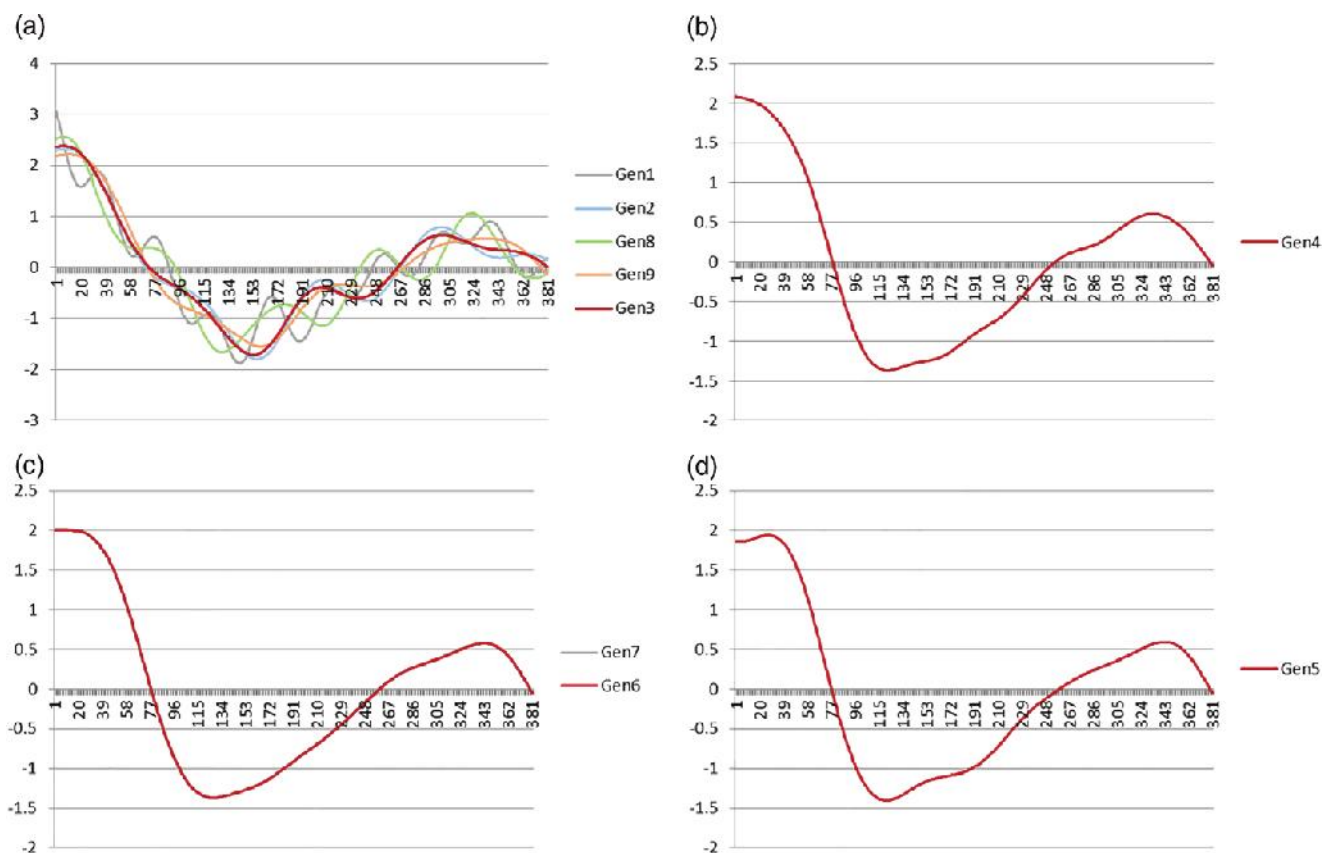


FIGURE 13 Clustering via k -means for system S_2 following fault scenario 1 with $r = 4$. Representative generators are in red in each plot

In contrast, we observe that the SVD method does seem to be a consistently high performer. In S_1 , SVD dominates almost all of the r values and has similarly high area under the perfprof curve in S_2 . This consistency across multiple systems will be necessary in a broadly applicable dynamic model reduction algorithm. Similarly, the Fiedler-mid-avg and Fiedler-mid-sum methods also appear to be repeatedly well-performing reduction techniques.

There also are three perpetually poorly performing algorithms: the Fiedler-zero methods. Recall from Section 3.3.1 that the standard method of Fiedler partitioning is to split the set of vertices according to positive or negative Fiedler value. However, in this application, it is clear that splitting at the midpoint of the Fiedler values is much more advantageous. Especially in the larger S_1 system, splitting at the zero point can yield a degenerate partition. Even in the smaller S_2 system, despite not creating degenerate partitions, the Fiedler-zero methods also perform poorly.

Lastly, the spectral method ranks in the middle of our list of average area under the curve for both test systems. However, looking at its performance on specific r values, there is high fluctuation. Recall that the final step of the spectral clustering method involves using k -means. Given our earlier conclusions that k -means may not be consistently reliable, it is not unexpected to draw the same conclusion about the spectral clustering method.

Overall, given the analysis of these specific methods on these two relatively small systems, we recommend that (1)

dynamic model reduction techniques should not rely on any kind of k -means clustering tools; (2) SVD is useful if a quick solution is needed, there are many widely available software packages for calculating SVDs, and it tended to be repeatedly well-performing if not always the top method; (3) if more time is available for a test study on a specific system of interest, we suggest that SVD be compared against Fiedler-mid-sum and Fiedler-mid-avg methods to see which performs best on that system for a desired r value. This paper provided a useful method for comparing performance using performance profiles given full and reduced model simulation data.

ACKNOWLEDGMENTS

The authors wish to thank the reviewers of this paper for their helpful comments and suggestions, which substantially improved it. This work was supported in part by the Applied Mathematics Program of the Office of Advanced Scientific Computing Research within the Office of Science of the U.S. Department of Energy (DOE) through the Multifaceted Mathematics for Complex Energy Systems (M2ACS) project.

REFERENCES

1. A. C. Antoulas, D. C. Sorensen, and S. Gugercin, *A survey of model reduction methods for large-scale systems*, Contemp. Math. **280** (2001), 193–219.
2. Z. Bai, *Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems*, Appl. Numer. Math. **43** (2002), 9–44.

3. N. M. Bertagnolli et al., *SVD identifies transcript length distribution functions from DNA microarray data and reveals evolutionary forces globally affecting GBM metabolism*, PLoS ONE **8** (2013), no. 11, 1–18.
4. C. A. Canizares et al., *Linear performance indices to predict oscillatory stability problems in power systems*, IEEE Trans. Power Syst. **19** (2004), no. 2, 1104–1114.
5. J. Chow, P. Accari, and W. Price, *Inertial and slow coherency aggregation algorithms for power system dynamic model reduction*, IEEE Trans. Power Syst. **10** (1995), no. 2, 680–685.
6. N. J. Dingle and N. J. Higham, *Reducing the influence of tiny normwise relative errors on performance profiles*, ACM Trans. Math. Softw. **39** (2013).
7. E. D. Dolan and J. J. Moore, *Benchmarking optimization software with performance profiles*, Math. Program. **91** (2002), 201–213.
8. M. Fiedler, *Algebraic connectivity of graphs*, Czech Math. J. **23** (1973).
9. M. Fiedler, *A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory*, Czech Math. J. **25** (1975), 619–633.
10. R. W. Freund, *Reduced-order modeling techniques based on Krylov subspaces and their use in circuit simulation* Birkhäuser Boston, Boston, MA, 1999.
11. J. E. Gentle, *Numerical linear algebra for applications in statistics*, Chapter: Singular value factorization, § 3.2.7, Springer-Verlag, 1998, pp. 102–103.
12. G. H. Golub and C. F. Van Loan, *Matrix computations*, 3rd ed. The Johns Hopkins University Press, Baltimore, MD, 1996.
13. D. J. Higham and N. J. Higham, *MATLAB guide*, 2nd ed. SIAM, 2005.
14. E. Hogan et al., *Towards effective clustering techniques for the analysis of electric power grids*, Proc. 3rd Workshop on HiPCNA-PG, 2013.
15. M. Honarkhah and J. Caers, *Stochastic simulation of patterns using distance-based pattern modeling*, Math. Geosci. **42** (2010), no. 5, 487–517.
16. D. N. Kosterev, C. W. Taylor, and W. A. Mittelstadt, *Model validation for the August 10, 1996 WSCC system outage*, IEEE Trans. Power Syst. **14** (1999), 967–979.
17. D. Lin and X. Wu, *Phrase clustering for discriminative learning*, Annual meeting of the ACL and IJCNLP, 2009, pp. 1030–1038.
18. U. von Luxburg, *A tutorial on spectral clustering*, Stat. Comput. **17** (2007), no. 4, 395–416.
19. R. Pinnau, *Model reduction via proper orthogonal decomposition* Springer, Berlin, Heidelberg, 2008.
20. A. Pothén, H. D. Simon, and K.-P. Liou, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Matrix Anal. Appl. **11** (1990), no. 3, 430–452.
21. G. Rogers, *Power system oscillations*, Power electronics and power systems, Springer, 2000.
22. H. Rudnick, R. I. Patino, and A. Brameller, *Power-system dynamic equivalents: coherency recognition via the rate of change of kinetic energy*, IEEE Proc. C **128** (1981), no. 6, 325–333.
23. M. Sahidullah and T. Kinnunen, *Local spectral variability features for speaker verification*, Digit. Signal Process. **50** (2016), 1–11.
24. S. Sastry and P. Varaiya, *Coherency for interconnected power systems*, IEEE Trans. Autom. Control **26** (1981), no. 1, 218–226.
25. S. E. Schaeffer, *Graph clustering*, Comput. Sci. Rev. **1** (2007), no. 1, 27–64.
26. S. Z. Selim and M. A. Ismail, *K-means-type algorithms: a generalized convergence theorem and characterization of local optimality*, IEEE Trans. Pattern Anal. Mach. Intell. **6** (1984), no. 1, 81–87.
27. K. Sun et al., *An online dynamic security assessment scheme using phasor measurements and decision trees*, IEEE Trans. Power Syst. **22** (2007), no. 4, 1935–1943.
28. R. Tiako, D. Jayaweera, and S. Islam, *Real-time dynamic security assessment of power systems with large amount of wind power using case-based reasoning methodology*, 2012 IEEE Power and Energy Society General Meeting, July 2012.
29. S. Wang et al., *Measurement-based coherency identification and aggregation for power systems*, 2012 IEEE Power and Energy Society General Meeting, 2012.
30. S. Wang et al., *Dynamic-feature extraction, attribution, and reconstruction (DEAR) method for power system model reduction*, IEEE Trans. Power Syst. **29** (2014), no. 5, 2049–2059.
31. Y. Xue et al., *A new tool for dynamic security assessment of power systems*, Control Eng. Pract. **6** (1998), no. 12, 1511–1516.
32. L. Marinovici et al., *Distributed hierarchical control architecture for transient dynamics improvement in power systems*, 2014 IEEE PES General Meeting | Conference & Exposition, National Harbor, MD, 2014.

How to cite this article: Purvine E, Cotilla-Sanchez E, Halappanavar M, Huang Z, Lin G, Lu S, and Wang S. Comparative Study of Clustering Techniques for Real-Time Dynamic Model Reduction, *Stat Anal Data Min.* 2017;10:263–276. <https://doi.org/10.1002/sam.11352>