Integrating Learning and Physics based Computation for Fast Online Transient Analysis

Jiaming Li*, Yue Zhao*, and Meng Yue†
*Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA
†Interdisciplinary Science Department, Brookhaven National Laboratory, Upton, NY 11973, USA
Emails: {jiaming.li, yue.zhao.2}@stonybrook.edu, yuemeng@bnl.gov

***Abstract*—A novel method that integrates learning and physics based computation is developed for greatly accelerating the simulation of full power system transient trajectories. To solve the dynamic algebraic equations, the method replaces the time-consuming dynamic computation for generator dynamics with trained predictors, while retaining the time-efficient algebraic computation of solving AC-power flow (PF) for power systems. In particular, a predictor is trained for each generator, and the system trajectories are computed by alternating steps of calling the predictors and solving AC-PF. The proposed method also allows fully parallelizable training strategies and a flexible trade-off between training time and testing accuracy. Comprehensive evaluations of the proposed method for transient/dynamic contingency analysis of the New York/New England 16-machine 68-bus power systems demonstrate excellent performance and significant acceleration of computation.**

I. Introduction

Security and stability are of paramount importance to power systems. Power system operations often conduct contingency analysis (such as $N-k$ security analysis) to ensure the system's continued functionality and stability under potential generator and line failures. Ideally, such contingency analysis should be performed in an *online* fashion (e.g., every 5 to 15 minutes), so that the security and stability risks are analyzed for the most up-to-date power system states. Indeed, power system operators often perform such online *static* (i.e., power flow based) contingency analysis. In contrast, *dynamic* contingency analysis are only performed in *offline* settings. The reason is that the computational complexity and hence time for full-scale dynamic analysis is typically orders of magnitude greater than that for static analysis, and the sheer amount of computing time would make online dynamic contingency analysis impractical. The motivation of this work is to significantly accelerate dynamic contingency analysis so that such analysis can eventually be performed online.

The key task in performing dynamic contingency analysis is to simulate full system transient/dynamic trajectories under different contingencies which entails solving large-scale differential-algebraic equations (DAEs) many times. The common practice of solving such DAEs relies on numerical integration methods [1]. While this approach provides accurate time-domain simulations, however, it is very time consuming which renders online dynamic contingency analysis infeasible. An alternative approach is the direct method [2]. However, notable challenges and limitations remain and must be overcome for it to be practical [2]. The direct method also does not generate system trajectories which are often needed for operators' decision-making.

Recently, machine-learning-based approaches for performing transient analysis have received increasing attentions. For post-fault binary stability classification, ML technologies have seen successes (see [3] among others). [4] estimates a stability margin by taking transient trajectories as image-like data and learning from them with a convolutional neural network based hierarchical model. To further explore the inherent temporal-spatial correlations, [5] proposes a heuristic learning method towards critical sequential features. However, knowing only a binary stability indicator and/or a stability margin may not be sufficient as a tool for the system operators, who would also like to know the trajectories of various quantities in the grid during and after the faults. For generating full dynamic trajectories, physics-informed neural networks are developed to solve ordinary/partial differential equations (ODEs/PDEs, see [6] [7] among others). However, scalability challenges arise when using these approaches for dynamic contingency analysis of larger-scale systems. Moreover, when there are structural changes in the system due to, e.g., faults that change the underlying ODEs, re-training has to be performed.

In this work, we develop a learning-physics-based hybrid approach that is, to the best of our knowledge, the first work that can both a) produce full system trajectories in a fast and accurate way and b) handle system changes (e.g., due to faults) very easily. The key idea is to train a predictor for each generator (as opposed to training an overall predictor for the entire system), and connect the predictors via the algebraic equations that characterize the system's AC power flow (AC-PF). Alternating steps of calling the trained predictors and solving AC-PF will then be performed to faithfully produce the full system trajectories. We demonstrate the effectiveness of the proposed method on a comprehensive data set of $N-2$ dynamic contingency responses simulated with very sophisticated generator control and dynamics in the New York/New England 16-machine 68-bus power system.

II. Problem Description

We consider a power system that consists of N buses, on K of which there are synchronous generators. The electromechanical dynamics of the generators based on their physical

This work is supported in part by the National Science Foundation under Grant ECCS-2025152, in part by ONR under Award N00014-22-1-2001, and in part by DOE EERE SETO under Award 38456.

models and control strategies can be captured by a set of dynamic equations [1]:

$$\dot{x} = f(x, z) \tag{1}$$

where x describes a collection of generator dynamic states and z describes algebraic variables such as voltages and currents. In addition to the generator dynamic equations, the algebraic equations of the system are described by the standard AC-PF equations. As such, the dynamic algebraic equations (DAEs) that describe the entire system can be summarized as:

$$\dot{x} = f(x, z) \tag{2}$$

$$0 = g(x, z) \tag{3}$$

where (3) represents the algebraic equations. Given an initial set of values of x and z, the DAEs (2) and (3) can be solved together via various numerical integration methods to determine the subsequent trajectories of all the quantities over time. Moreover, when faults occur in the system and are subsequently cleared, *the algebraic equations* (3) *can experience multiple changes* as the underlying power system model changes. In this paper, we partition the set of all the buses into K generator buses and $N-K$ non-generator buses. We let a) the state variables of a generator at bus i and b) the algebraic variables at a non-generator bus j be

$$x_i = \left[|V_i|, \angle V_i, \delta_i\right]^T, \tag{4}$$

$$z_j = \left[|V_j|, \angle V_j\right], \tag{5}$$

where V_i (V_j) represent the generator bus' (non-generator bus') complex voltage and δ_i is the generator's rotor angle. We note that, the internal generator state variables implemented in all our simulations include many more than the above.

The goal of this work is to greatly accelerate the solution of the DAEs to generate all the dynamic trajectories in the system, and hence achieve significant speed-up of *dynamic* contingency analysis of hypothetical faults in the system. The high level idea of our approach is to a) replace the time-consuming dynamic computation of the generators with *trained predictors*, while b) retaining the time-efficient algebraic computation of solving AC-PF for the power system. As such, to generate all the trajectories, our algorithm iteratively alternates between calling trained predictors and solving AC-PFs, and does not invoke any numerical integration method at all that is traditionally required for solving DAEs.

III. Proposed Methodology

In this section, we provide the details of our proposed methodology that integrates both learning (for dynamic) and physics (for algebraic) based computation to simulate system trajectories. We first describe the overall architecture of the method in the following. First, for *each* generator, a predictor is trained for iteratively producing the dynamic states of the generator. In testing, at each time index, a) the output of all the predictors are used as inputs (similar to boundary conditions) to a AC-PF solver, and the states of the power system are solved, and b) each predictor then reads in (a corresponding small subset of) the solved system states as inputs (additional to the already computed previous system states), and produces as outputs the generator state for the next time index. This procedure iteratively alternates between these two steps to produce the full system trajectories. The general algorithm is summarized in Algorithm 1 with detailed explanations to follow. Next, we first describe a) how the learning-physics-based DAE solver is structured and used, and b) how the predictors in this solver are trained.

Algorithm 1 Integrated Learning-Physics-Based DAE Solver

Input: Initial generator state variables $\{x_{i,0}\}$ at all generator buses, initial algebraic variables $\{z_{j,0}\}$ at all non-generator buses.

Output: Generator state variables $\{x_{i,1:T}\}$ at all generator buses, algebraic variables $\{z_{j,1:T}\}$ at all non-generator buses.

1: **for** $t \leftarrow 1$ to *Trajectory Length* T **do**
2: **for** $i \leftarrow$ *Generators* **do**
3: $x_{i,t} \leftarrow$ NN_Predict$(x_{i,1:t-1}, z_{n(i),1:t-1})$
4: **end for**
5: Update $\{z_{j,t}\} \leftarrow$ AC_PF$(\{x_{i,t}\})$
6: **end for**
7: Return $\{x_{i,1:T}\}$, $\{z_{j,1:T}\}$

A. Predictor Architecture: One Predictor for Each Generator

Importantly, even though our method generates the trajectories jointly for all the generator and load buses in the system, we do *not* rely on a single "overall" predictor that jointly outputs all the trajectories. Instead, a separate predictor is trained for each generator. For a generator i, to compute its state at the current time t, its corresponding predictor takes only the following as its inputs (cf. Line 3 of Algorithm 1):

- Its own states up to the previous time step, $x_{i,1:t-1}$.
- The complex voltage(s) of its *neighboring* bus(es) $n(i)$ up to the previous time step, $z_{n(i),1:t-1}$.

Notably, each generator typically connects to just one neighboring bus. As such, the input features for a predictor include a *5-dimensional* times series (cf. (4) and (5)). There are two major advantages of the proposed approach:

1) The model complexity of the predictors is relatively *independent of the power system size*, which is a key enabler for achieving *scalability* for large-scale systems. In contrast, if a single predictor is trained to represent the entire system, (as opposed to just one generator), the model complexity of the predictor would necessarily increase significantly as the power system size increases, hindering the scalability of such learning-based approaches.
2) As a predictor only tries to capture the dynamics of one generator, and the interactions of the generators are taken care of by AC-PF, the predictor+ACPF hybrid approach can straightforwardly handle *changes in the power system*, such as topology changes due to contingencies, *without any need of re-training*. In contrast, if a single predictor is trained to represent the entire system, it will have to be re-trained when any change occurs to the power system.

As the inputs to a predictor are time series, we employ a Long Short-Term Memory (LSTM) based architecture for the predictors: based on hyperparameter tuning, two LSTM layers followed by one fully connected layer are used. The detailed

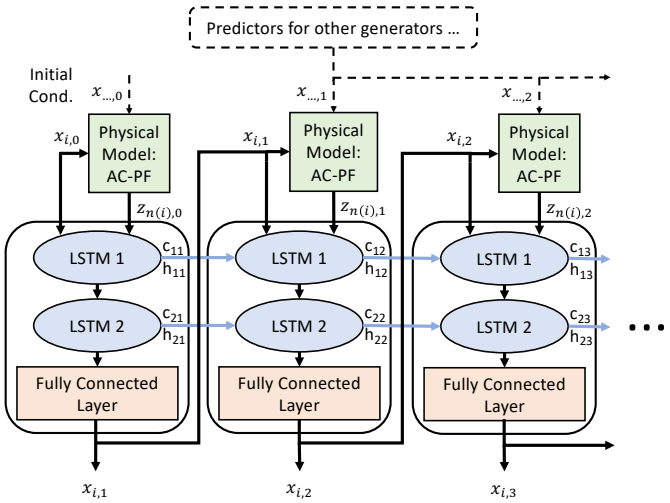

Fig. 1: Predictor Architecture for a Generator i.

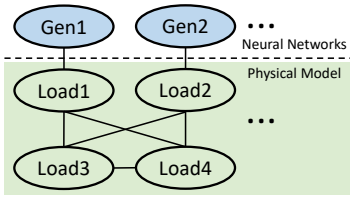

Fig. 2: A High-level system diagram of the neural-physics hybrid model. Each generator is represented by a NN predictor model. All the load buses (including those with zero load) are represented by the system-wise AC-PF model.

neural network (NN) architecture for a generator i is depicted in Fig. 1. We note that c and h are the LSTMs' hidden states and are updated at each time.

B. Physical Model: AC Power Flow

While Fig. 1 depicts the architecture of one generator's predictor, the other generators' predictors share similar architectures. As shown in Fig. 1, at every iteration, the algebraic variables z are computed based on the AC-PF solution given the latest generator state variables output by the predictors. Note that, a) this step uses the predicted generator states x for *all* the generators, b) the AC-PF is solved for the *entire* system and generates the algebraic variables z for *all* the buses, and c) each generator's predictor then takes *only the inputs it needs* — its own states and its neighboring bus' algebraic variables — to predict its states for the next time step. As such, the dynamic model of the entire power system is represented by a hybrid NN-ACPF model: each generator is represented by a NN that captures its dynamic model, and they are then connected by an AC-PF model that represents the entire power system. A high level system diagram is depicted in Fig 2.

C. Training the Predictors: Jointly, Locally, and Singularly

The predictors in the NN-ACPF model are trained based on a data set of simulated trajectories of the power system. The offline trained predictors are then utilized for online dynamic simulation of *new/unseen cases*. For a dynamic trajectory of a generator i, denoted by $x_i = [x_{i,1}, x_{i,2}, ..., x_{i,T}]$, the training labels are the state variables at the next time steps: $y_i = [y_{i,1}, y_{i,2}, ..., y_{i,T-1}]$, where $y_{i,t} = x_{i,t+1}$. We employ the Relative Mean Square Error (Relative MSE) as the training objective L_i over the NN parameters θ_i:

$$L_i(\tilde{y}; \theta_i) = \frac{1}{D} \sum_{k=1}^{D} \frac{\sum_{t=1}^{T-1} (\tilde{y}_{i,t}^{(k)} - y_{i,t}^{(k)})^2}{\sum_{t=1}^{T-1} (y_{i,t}^{(k)} - \overline{y_i^{(k)}})^2} \tag{6}$$

where $\tilde{y}$ is the predictor's output, D is the dimension of $y_{i,t}$, and $\overline{y_i^{(k)}}$ is the mean of $y_{i,t}^{(k)}$ over t for a single trajectory.

As all the generators' predictors are coupled by the AC-PF solutions at each time step during testing, one predictor's output at a time step will be fed back not only directly into the inputs of itself, but also indirectly into those of all the other predictors via AC-PF. Based on the principle that predictors' training should match how they are used in testing, all the generators' predictors should ideally be trained *jointly*. We indeed develop such a training procedure which we term *joint training*. Specifically, the losses for all the predictors are computed based on Algorithm 1. The losses thus reflect the joint performance of all the predictors interacting via AC-PF. Backpropagation is then performed for each predictor. As such, the training for all the predictors are performed in a coupled and "synchronous" fashion.

We next discuss an alternative to this joint training approach, termed *local training*. Instead of training all the predictors in sync, we decouple their training processes to be independent to each other. Specifically, when training the predictor of generator i, instead of relying on other generators' predictor outputs to solve AC-PF, we utilize certain *ground truth* data of the simulated trajectories. To clarify the ground truth data used, we introduce another key characteristic of this local training as follows: Instead of solving the AC-PF for the entire power system, for a generator i, we can limit the perspective of AC-PF to some *local system around the generator*. Specifically, the ground truth data at the *boundary* buses of this local system are used as input to solving the *local* AC-PF. As such, each generator's predictor is trained independently without dependence on each other's outputs.

One key advantage of local training is that it allows flexible control of the computational complexity and hence the raw time of training. In particular, a) the training of all the predictors do not have to be done jointly, but fully in parallel, and b) the *size* of the local system for training each predictor can be flexibly tuned, and can thus greatly reduce the complexity of AC-PF. Understandably, *there is a trade-off between the training time and the testing performance of the trained predictors*: the smaller the local system, the faster the training but also the less accurate the trained predictors during testing. The framework of local training thus provides a flexible tool for choosing an acceptable trade-off.

An extreme case of this trade-off is shrinking the local system to just a *single bus* of the generator. In this case, there is simply no AC-PF computation, and the ground truth of the neighboring bus is directly fed into the predictor during training: In Fig 1, this means that the ground truth data of the algebraic variable inputs z are used, thus completely bypassing any AC-PF. We term this extreme case *singular training*.

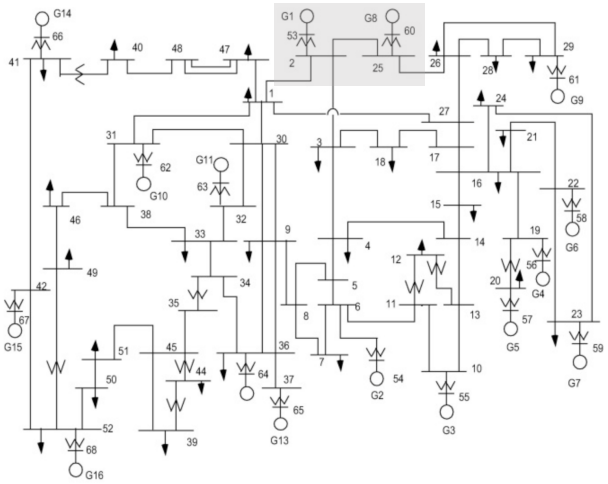

Fig. 3: Schematic diagram of the IEEE 16-machine 68-bus system. The subsystem to be investigated is in the shaded area, including 2 generator buses and 2 load buses.

We note that such singular training exactly corresponds to the method we developed in our prior work [8].

Finally, we note that the use of the trained predictors during testing is always the same (cf. Algorithm 1) regardless of how they are trained (jointly, locally, or singularly).

IV. Case Study and Discussion

A. Data Generation

We perform time-domain simulations to generate system trajectories with a software tool EPTOOL that is based on the Power System Toolbox (PST) [9]. The simulations are performed on the New York/New England 16-generator 68-bus power system (cf. Fig. 3). The generator dynamics are modeled and simulated with full control schemes, i.e. turbine governor, excitation systems, and power system stabilizers (PSSs). The load buses contain 50% of constant current load and 50% of constant impedance load. We then introduce random $N-2$ *contingencies* to the system which would cover most of the potential contingencies in practice. Each contingency consists of double permanent 3-phase transmission line faults at either of the two terminal buses of the faulted lines. We adopt the fault patterns in a practical power system [10]: The fault duration follows a normal distribution with a mean value of 100.0ms (6 cycles) and a standard deviation of 11.11ms. Each trajectory is simulated with a sampling period of 0.002s for a total length of 1.4s, and hence contains 700 time steps. The two faulted lines are then set as off-service after the faults are cleared at near and far ends of the lines. In total, we simulate $2,460$ $N-2$ contingencies and collect the simulated trajectories for learning the generators' predictors. The data set is split randomly with $1,600$ for training, 400 for validation and 460 for testing. In this paper, we focus on a 4-bus subsystem (cf. the shaded area of Fig. 3): generator 1 located at bus 53, generator 8 located at bus 60, load bus 2 and load bus 25.

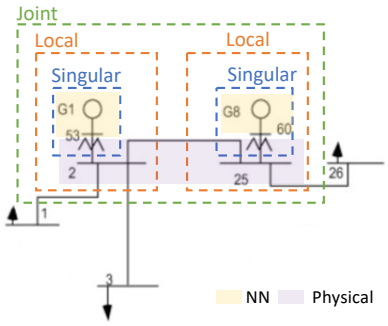

Fig. 4: Training strategies for the 4-bus subsystem.

B. Training and Implementation

We employ the three training strategies (cf. Section III-C): joint, local and singular. Specifically, for *joint training*, AC-PF is solved for the 4-bus subsystem, treating the rest of the power system as known boundary conditions. The two generators' predictor training are coupled via the 4-bus AC-PF. For *local training*, each generator's predictor training utilizes AC-PF of *only the 2-bus local system* that includes the generator bus and its neighboring load bus(es) (i.e., buses 53 and 2 for generator 1, and buses 60 and 25 for generator 8). For each generator's training, the ground truths of the physical quantities other than its 2-bus local system are treated as known boundary conditions. For *singular training*, no AC-PF is performed, and the ground truths of the physical quantities other than those of the two generator buses are treated as known boundary conditions. A schematic of these three training strategies for the 4-bus subsystem is depicted in Fig. 4.

For all the generators' predictors in all of the above training strategies, we use the *same* LSTM-based architecture (cf. Fig. 1) with the *same* set of hyperparameters for training.

C. Numerical Experiment Results

As noted at the end of Section III, during testing, the performance of the predictors are always evaluated in the same way, following Algorithm 1, regardless of the strategies (joint, local, or singular) employed for training the predictors. Specifically, a) Generator 1 and Generator 8 are replaced by 2 predictors, respectively, and b) AC-PF is iteratively solved for the 4-bus subsystem connecting the 2 generators, treating the rest of the power system as known boundary conditions.

To evaluate the performance of the trained predictors, we employ the metrics of Root Mean Squared Error (RMSE) and Relative RMSE (with the same variables as in (6)):

$$\text{Relative RMSE} = \frac{1}{D}\sum_{k=1}^{D}\frac{\sqrt{\frac{1}{T-1}\sum_{t=1}^{T-1}(\tilde{y}_{i,t}^{(k)}-y_{i,t}^{(k)})^2}}{\sqrt{\frac{1}{T-1}\sum_{t=1}^{T-1}(y_{i,t}^{(k)}-\overline{y_i^{(k)}})^2}} \quad (7)$$

We note that Relative RMSE can provide a fair evaluation for quantities of different units. We summarize the testing RMSE and Relative RMSE in Table I for different training strategies. We see that predictors from joint training performs markedly better than those from local training, which again performs significantly better than those from singular training.

A representative sample of the achieved performance is plotted in Fig. 5, in which the trajectories of the three state

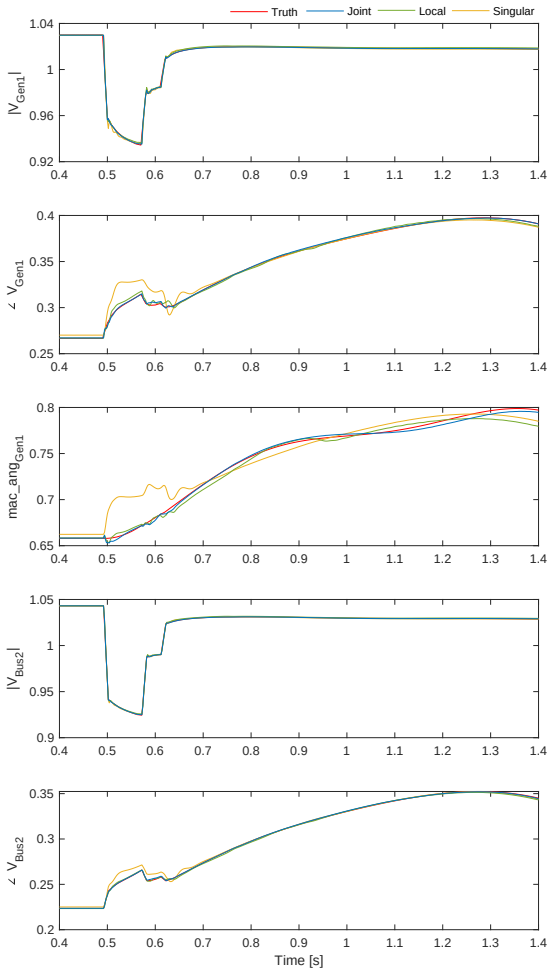

Fig. 5: A representative testing sample showing the generated trajectories after a contingency.

variables for generator 1 and those of the two algebraic variables for bus 2 are plotted. The trajectories generated with the predictors from the three training strategies are compared with the ground truths. In this example, an $N-2$ contingency occur at $0.49s$ and the two faults are cleared at $0.576s$ and $0.608s$, respectively. *We note that the trajectories include all the periods before, during, and after fault clearances, in which the power system model experiences significant changes.* The natural advantage of the proposed method in handling such changes is clearly demonstrated. As our method is the first learning-based approach that can address system changes without the need of re-training for every contingency, no available numerical comparison is available from the literature. We observe that the testing accuracy from the joint and local training are both very high. Even for singular training, while the accuracy is visibly lower, it still captures the voltage magnitudes very accurately.

We next provide the computation times and memory usages for our learning-physics-based DAE solver (in both training and testing) and those for the traditional numerical-integration based solver (in testing) in Table II. The training and testing are performed on nVidia K80 GPUs with the machine learning library PyTorch. We observe that, while joint training provides the highest accuracy, it takes a significantly longer time to train. Local training provides a reasonable trade-off between offline training time and online testing accuracy.

TABLE I: Performance Comparison of Training Strategies

Training Strategy	Avg. RMSE	Avg. Relative RMSE
Joint	$\mathbf{3.631 \cdot 10^{-3}}$	$\mathbf{4.056 \cdot 10^{-2}}$
Local	$5.372 \cdot 10^{-3}$	$5.684 \cdot 10^{-2}$
Singular	$8.720 \cdot 10^{-3}$	$8.861 \cdot 10^{-2}$

TABLE II: Computational Efficiency

Model	Offline Training time [min]	Offline Compute Memory [MB]	Online Compute Time [s]	Online Compute Memory [MB]
Singular	229	2245	2.16	1545
Local	767	2247	2.16	1545
Joint	2609	2941	2.16	1545
Numerical	-	-	19.9	269

V. Conclusion

We have developed a novel learning-physics-based method for fast and accurate computation of full power system transient trajectories. A predictor is trained for each generator, and the system trajectories are computed by alternating steps of calling the predictors and solving AC-PF. This algorithm achieves much lower computational complexity than existing numerical integration based approaches. The method also allows different training strategies — joint, local, and singular — that allow different trade-offs between offline training complexity and online testing accuracy. We evaluate the method for $N-2$ transient/dynamic contingency analysis for the New York/New England 68-bus 16-machine power system with sophisticated generator control and dynamics. Excellent performance and computation speed is demonstrated.

References

[1] B. Stott, "Power system dynamic response calculations," *Proceedings of the IEEE*, vol. 67, no. 2, pp. 219–241, 1979.

[2] H.-D. Chiang, *Direct methods for stability analysis of electric power systems: theoretical foundation, BCU methodologies, and applications*. John Wiley & Sons, 2011.

[3] J. Geeganage, U. Annakkage, T. Weekes, and B. A. Archer, "Application of energy-based power system features for dynamic security assessment," *IEEE Trans. on Power Sys.*, vol. 30, no. 4, pp. 1957–1965, 2014.

[4] L. Zhu, D. J. Hill, and C. Lu, "Hierarchical deep learning machine for power system online transient stability prediction," *IEEE Transactions on Power Systems*, vol. 35, no. 3, pp. 2399–2411, 2019.

[5] L. Zhu and D. J. Hill, "Networked time series shapelet learning for power system transient stability assessment," *IEEE Transactions on Power Systems*, vol. 37, no. 1, pp. 416–428, 2021.

[6] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[7] G. S. Misyris, A. Venzke, and S. Chatzivasileiadis, "Physics-informed neural networks for power systems," in *2020 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 2020, pp. 1–5.

[8] J. Li, M. Yue, Y. Zhao, and G. Lin, "Machine-learning-based online transient analysis via iterative computation of generator dynamics," in *IEEE SmartGridComm*, 2020, pp. 1–6.

[9] "Power system toolbox," Available at https://www.ecse.rpi.edu/ chowj/.

[10] R. Billinton and P. Kuruganty, "Probabilistic assessment of transient stability in a practical multimachine system," *IEEE Transactions on Power Apparatus and Systems*, no. 7, pp. 3634–3641, 1981.