Machine-Learning-Based Online Transient Analysis via Iterative Computation of Generator Dynamics

Jiaming Li*, Meng Yue†, Yue Zhao*, and Guang Lin‡
*Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA
†Sustainable Energy Technologies Department, Brookhaven National Laboratory, Upton, NY 11973, USA
‡Departments of Mathematics, Department of Statistics, School of Mechanical Engineering, and
Department of Earth, Atmospheric, and Planetary Sciences, Purdue University, West Lafayette, IN 47907, USA
Emails: {jiaming.li, yue.zhao.2}@stonybrook.edu, yuemeng@bnl.gov, guanglin@purdue.edu

***Abstract*—Transient analysis is vital to the planning and operation of electric power systems. Traditional transient analysis utilizes numerical methods to solve the differential-algebraic equations (DAEs) to compute the trajectories of quantities in the grid. For this, various numerical integration methods have been developed and used for decades. On the other hand, solving the DAEs for a relatively large system such as power grids is computationally intensive and is particularly challenging to perform online. In this paper, a novel machine learning (ML) based approach is proposed and developed to predict post-contingency trajectories of a generator in the time domain. The training data are generated by using an off-line simulation platform considering random disturbance occurrences and clearing times. As a proof-of-concept study, the proposed ML-based approach is applied to a single generator. A Long Short Term Memory (LSTM) network representation of the selected generator is successfully trained to capture the dependencies of its dynamics across a sufficiently long time span. In the online assessment stage, the LSTM network predicts the entire post-contingency transient trajectories given initial conditions of the power system triggered by system changes due to fault scenarios. Numerical experiments in the New York/New England 16-machine 86-bus power system show that the trained LSTM network accurately predicts the generator's transient trajectories. Compared to existing numerical integration methods, the post-disturbance trajectories of generator's dynamic states are computed much faster using the trained predictor, offering great promises for significantly accelerating both offline and online transient studies.**

I. Introduction

A key objective of transient analysis is to determine whether all the generators in a power system can maintain synchronism after a major disturbance such as transmission line fault or generator trip, which is of significant interests for power system planners and operators. Traditionally, there are two categories of methods for transient analysis. One is performed by numerically solving differential algebraic equations (DAEs) that characterize the power grid [1], and another one is the direct method [2]. The numerical solutions provide the evolution of the system dynamic behaviors. This process has been common practice and feasible. However, it is very time-consuming, and often has numerical issues especially for large-scale power networks. This can be exacerbated by the need for evaluating many more scenarios to capture the uncertainties of increasing renewable generation in the power grid. While parallelization of the numerical integration methods can be explored to speed up the computation, it is highly resource consuming. The direct method, on the other hand, is based on the energy function or Lyapunov function defined for a dynamic system, which is system specific and usually very difficult for a relatively large system [2]. In addition, it produces the stability margin but not the system state trajectories that are often needed as well. As such, there has not been functioning online transient contingency analysis for realistic sized power grids, where full numerical solutions of large-scale DAEs need to be computed for thousands of contingency scenarios within a few minutes.

We identify that the major challenge for transient analysis, especially in an online manner, is the extremely heavy computational efforts and time. To address this challenge, high hopes have been given to machine learning (ML) based approaches for this type of applications, as researched in a large body of work (see, e.g., [3] [4]). However, the existing ML-based transient studies primarily focus on the development of Transient Stability Assessment (TSA) system for determining system stability given the input disturbances, i.e., a binary indicator of whether the system is stable or not under a specific disturbance. The ML-based TSA tool can be fast, but is however insufficient as a tool for the system planners or operators, who would also like to know the trajectories of various quantities in the grid during and after the disturbances. Such trajectories are crucial for examining if there will be any voltage or frequency violation that may trigger load shedding. On generating full trajectories, recent ML-based ordinary/partial differential equation solvers [5] have seen successes in power systems of small sizes [6]. However, major challenges arise when using these approaches for large-scale dynamic contingency analysis, a) as the dimensions of the ODEs increase (e.g., as the size of the power system increases), and b) when there are structural changes in the system due to, e.g., faults that change the underlying ODEs.

In this study, an innovative ML-based approach is proposed

This work is supported in part by the National Science Foundation under Grant ECCS-2025152. G. Lin gratefully acknowledges the support from the National Science Foundation (DMS-1555072, DMS-1736364, CMMI-1634832, and CMMI-1560834), and Brookhaven National Laboratory Subcontract 382247.

to learn from and then predict the time-domain responses of the system states, where we leverage the capability of dynamic neural networks, in particular, long-short-term-memory (LSTM) networks. Recognizing that the grid dynamics are dominated by the generators and their associated control, the focus of this study is to obtain an LSTM network for a single generator to reproduce the generator dynamics, trained using time-domain simulation data of post-disturbance generator dynamics. In particular, starting from any given initial condition, the trained predictor is applied iteratively to generate the entire trajectories of generator states, reproducing the ground-truth dynamics by solving the DAEs. The capability of the trained LSTM network is demonstrated by accurately predicted trajectories under many different disturbances. As such, the proposed approach is different from the existing ML applications in TSA analyses and can be considered an important step to develop a platform for the *full ML-based transient analysis* for power systems.

The remainder of the paper is organized as follows. Section II formulates the problem. The overall methodology and detailed discussions are presented in Section III. Section IV provides numerical case studies. Section V presents conclusions and future research on a roadmap to develop a full transient analysis platform based on the proposed ML approach.

II. Problem Description

We consider a power system whose dynamics are determined by a set of DAEs,

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

$$0 = g(x, z) \tag{2}$$

where x is a collection of generator state variables and z the algebraic variables representing, e.g., voltages and currents. f mainly represents the generator dynamics such as inertial dynamics and control. g indicates the power balance in the transmission network that connect generators and load. Given an initial set of values of x and z (e.g., as a result of a disturbance occurred in the system), (1) and (2) can be solved together via various numerical integration methods to determine the subsequent trajectories of all the quantities over time. Since the grid dynamics are dominated by generators and the associated controls, in this paper, we limit our scope to computing the trajectories of the state variables of a single generator. In the context of this paper, we let the state variables of a generator be

$$x = [|V|, \angle V, \delta, \omega, P_{mech}, P_{gen}, Q_{gen}]^T, \tag{3}$$

where V represents the bus complex voltage, δ is the rotor angle, ω is the rotor speed, P_{mech} is the mechanical power input to the geneartor, P_{gen} is the real power generation, and Q_{gen} is the reactive power generation. We note that, the internal generator state variables that we consider in our simulations include more than the above. In other words, the state variables (3) are the "agent's state", rather than the state of the world. Nonetheless, we will show that (3) as the agent's state is sufficient for training predictors of full trajectories.

Instead of using conventional methods of numerically solving the DAEs (1) and (2), we propose to use a machine-learning-based method that computes, using the outputs of a trained predictor, the trajectories of the state variables given any initial condition. Specifically, a) every generator is connected with the rest of the system via line(s) to a neighboring bus (see. e.g., Fig. 6 and 7, where Generator 1, located at bus 53, is connected to the system via a single line between bus 53 and 2), b) for the generator of interest, we assume that the current dynamics on its connecting line(s) and the voltage dynamics on its neighboring bus(es) are given as external inputs, and c) based on the initial condition of the generator and these external inputs, we seek to compute the subsequent trajectories of all or a subset of the state variables of this generator.

In essence, we replace the numerical solver of the DAE with a trained predictor which is *iteratively* applied to the trajectories over time, starting from an initial condition, to generate the full trajectories. Such a trained predictor can be viewed as a neural network (NN) representation of the dynamics of the generator. With the ML-based method, we aim to greatly accelerate the computation of the trajectories compared to the traditional numerical solvers of DAEs.

III. Proposed Methodology

We now provide the details of the proposed methodology of ML-based fast computation of generator dynamics.

A. Machine-Learning-Based Computation of Trajectories

A high level diagram for the training and use of the predictor is depicted in Fig. 1. The overall methodology consists of three elements for ML applications: data generation, offline training, and online assessment. The data for transient trajectories of a power system are generated considering a large number of faulted scenarios with 60% of the data for training, 20% for validation, and 20% for testing.

A depiction of the input and output for an NN representation of a generator is shown in Fig. 2. For a generator at bus k that connects to the system via line jk at a neighboring bus j, the state variables $x_k(t)$ are as specified in (3), and the external input variables are the current injection $I_{jk}(t)$ on line jk and the voltage $V_j(t)$ at bus j. we seek to train a predictor that, at time t, a) takes in the current state variables $x_k(t)$ and the external inputs $z(t)$ that consists of $I_{jk}(t)$ and $V_j(t)$, and b) outputs the state variables at the next time step $x_k(t+1)$. Note that $I_{jk}(t)$ can be calculated by using $V_j(t)$, $V_k(t)$, and the line impedance $Z_{jk}(t)$. Iteratively applying such a predictor will then produce the full trajectories of all the state variables for this generator. Importantly, the predicted state variables are fed back as (part of) the input in the next time step. This implementation process is better illustrated by using the sequential data flow unrolled over time in Fig. 3.

To obtain such a predictor, we utilize extensive *offline* data generation and training. The trained predictor is then applied in an *online* fashion to new initial conditions unseen in the training data. While the data generation and training may take a reasonably long time, they are performed offline with

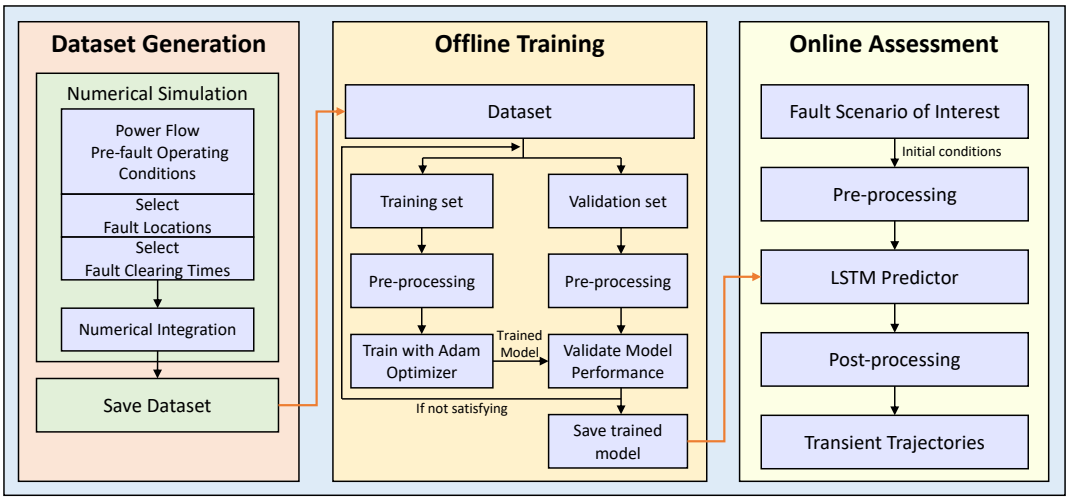

Fig. 1: Overview of the proposed approach for ML-based transient analysis.

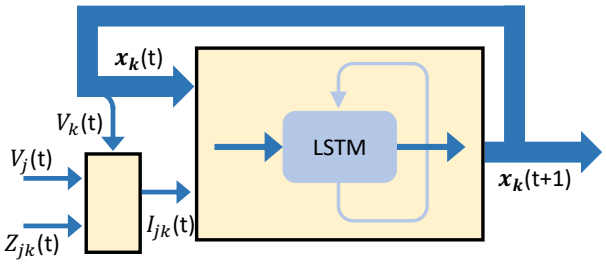

Fig. 2: Input and output for a generator.

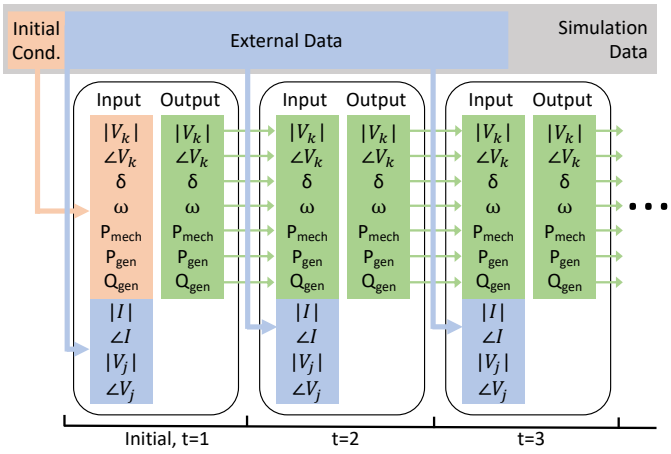

Fig. 3: The flow of data in the implementation of the ML-based trajectory computation.

abundant computation resources. Moreover, The use of the trained predictor in the testing stage is very fast, and thus fulfilling our objective of accelerating the online computation of full trajectories of generator dynamics. This shares a spirit similar to the Learning-to-Infer method developed in [7].

a) Data Generation and Off-line Training: We simulate power system dynamics under varying initial conditions (cf. Section IV-A later for more details). The training, validation and testing data are obtained by collecting the trajectories of the generator dynamics as well as that of the external inputs. For a trajectory of generator dynamics, we denote it by $x = [x_1, x_2, ..., x_T]$, where x_t consists of all the state variables at time step t. Note that subscript k is dropped since generator k is the only one considered here and we denote $x(t)$ as x_t and $z(t)$ as z_t for simplicity.

As our prediction is for the next time step's state variables, the *training labels* are freely built-in: $y = [y_1, y_2, ..., y_{T-1}]$, where $y_t = x_{t+1}$. Given a collection of N pairs of training samples $\{x^{(n)}, y^{(n)}\}_{n=1}^{N}$, we train the predictor model to minimize a Mean Square Error (MSE) objective function L over the NN parameters θ:

$$L(\theta) = \frac{1}{D}\frac{1}{N}\frac{1}{T}\sum_{i=1}^{D}\sum_{n=1}^{N}\sum_{t=1}^{T}(\tilde{y}_{i,t}^{(n)} - y_{i,t}^{(n)})^2, \qquad (4)$$

where $\tilde{y}^{(n)}$ is the predictor's output and D is the dimension of $y_t^{(n)}$, i.e., the number of state variables which equals 7 in our context (cf. (3)).

b) Online Assessment: In the testing stage, we compute, via iterative prediction, the entire trajectories $[x_1, x_2, ..., x_T]$ based on the initial condition x_1 as well as the external inputs $[z_1, z_2, ..., z_T]$. As such, all that is performed are $T-1$ queries of the trained predictor, which is a lot faster than numerically solving the DAEs to compute the generator dynamics.

B. Neural Network Architecture

To efficiently represent the generator dynamics, the neural network architecture employed in the predictor plays an important role. Due to the time-series nature of the trajectories of interest, we consider incorporating recurrent neural networks (RNNs) to model the predictor as opposed to using only fully connected (FC) multilayer perceptron (MLP).

A traditional RNN unit maps the current (at time t) input $\mathbf{x}_t$ to a hidden state $\mathbf{h}_t$ according to the previous hidden state $\mathbf{h}_{t-1}$. To further improve the memory encoding capability,

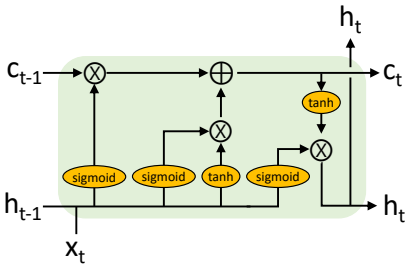

Fig. 4: LSTM Cell.

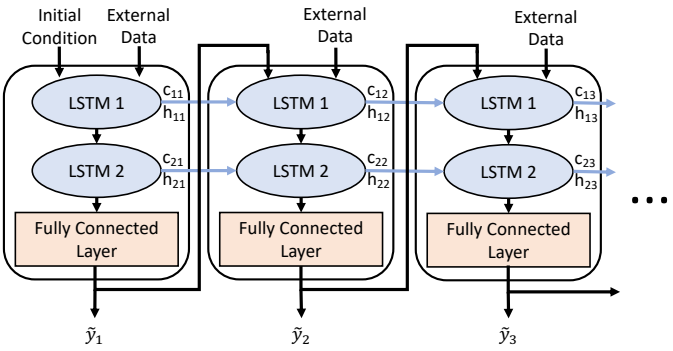

Fig. 5: NN architecture and implementation.

an RNN layer can consist of multiple RNN units, and an RNN model can consist of multiple RNN layers. Specifically, we employ a well-established type of RNN unit — Long Short Term Memory (LSTM) unit [8] — in our RNN. The architecture of an LSTM unit is depicted in Fig. 4 [9]. Notably, LSTM provides the capability to not only capture inter-temporal dynamics but also resolve the vanishing and exploding gradient problems commonly observed in simple RNNs. These favorable features make LSTM-based architectures able to achieve outstanding performance in a wide range of sequence learning tasks. Specifically, we employ two LSTM layers followed by one FC layer. The detailed NN architecture in our implementation is depicted in Fig. 5, which instantiates the structure depicted in Fig. 3.

IV. Case Study and Discussion

A. Data Generation

We perform time-domain simulations to generate system trajectories using a software tool EPTOOL that was developed based on the Power System Toolbox (PST) [10]. In particular, we generate a library of system dynamic responses to a variety of disturbances.

The simulations are performed on the New York/New England 16-generator 68-bus power system (cf. Fig. 6). The full generator dynamics are modeled and simulated with turbine governor, excitation systems, and power system stabilizers (PSSs), even though only a subset of generator states are selected to be reproduced using the NN (3). The load buses contain 50% of constant current load and 50% of constant impedance load. We then introduce $N-2$ *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 as presented in [11]: The fault duration (i.e., the period between the start and the clearance of a fault) follows a normal distribution with a mean value of 100.0ms (6 cycles) and a standard deviation of 11.11ms. We assume that the system is in steady-state operating condition before each contingency. Two transmission lines are picked at random from the entire system, and both faults are applied at 0.01s. Each contingency is simulated for a length of 9.0s with a sampling rate of 100Hz, and hence each trajectory contains 900 samples. The two transmission lines are then set as off-service after the faults are cleared at near ends and far ends of the lines. In total we simulate $3,000$ $N-2$ contingencies and collect the corresponding data of trajectories.

TABLE I: Hyperparameters of the Network

Parameter	Value
# nodes in LSTM1	400
# nodes in LSTM2	400
# nodes in Fully Connected Layer	7
Learning Rate	0.001
Minibatch size	200
Gradient clip	0.0001

We focus on Generator 1 located at bus 53. We collect the post-fault-clearance trajectories of Generator 1's $[|V|, \angle V, \delta, \omega, P_{mech}, P_{gen}, Q_{gen}]^T$. Since Bus 53 is connected to the system via Bus 2 (cf. Fig. 7), we further collect a) trajectories of $[|V|, \angle V]^T$ at Bus 2, and b) trajectories of current injection on line 2-53, denoted by I. To avoid discontinuities, $\pm 2\pi$ are added appropriately to the angle data, i.e. $\angle V, \angle I$ and δ. In sum, the training data set is denoted by $x = [x_1, x_2, ..., x_T]$, where

$$x_t = [|V_{53,t}|, \angle V_{53,t}, \delta_t, \omega_t, P_{mech,t}, P_{gen,t}, Q_{gen,t}, |I_t|, \angle I_t, |V_{2,t}|, \angle V_{2,t}] \quad (5)$$

is an 11-*dimensional* input vector to the LSTM network. We note that there's a slight abuse of notation here, as we have included external inputs z into x (5) as well. Meanwhile, we construct labels $y = [y_1, y_2, ..., y_T]$, where

$$y_t = [|V_{53,t+1}|, \angle V_{53,t+1}, \delta_{t+1}, \omega_{t+1}, P_{mech,t+1}, P_{gen,t+1}, Q_{gen,t+1}] \quad (6)$$

is a 7-*dimensional* vector.

B. Training Details

We provide the hyperparameters of the LSTM-based neural network (cf. Fig. 5) and the training algorithm in Table I. Here, the number of LSTM units are set relatively large as over-parameterization tend to speed up training progress and improve generalization capability [12]. We employ the Adam optimizer [13] for training.

We perform the training and testing on a desktop computer with a 2nd generation 3.60-Ghz Intel Xeon CPU with 32GB RAM. An nVidia GTX 1060 GPU is configured to support efficient machine learning library PyTorch [14].

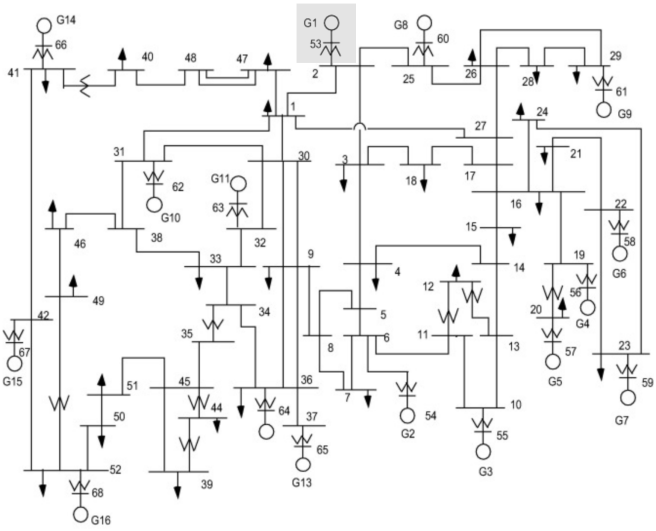

Fig. 6: Schematic diagram of IEEE 16-machine 68-bus system. The generator to be investigated is in the shaded area.

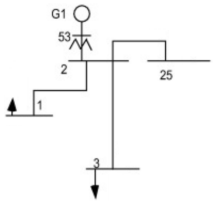

Fig. 7: Local connections of generator 1

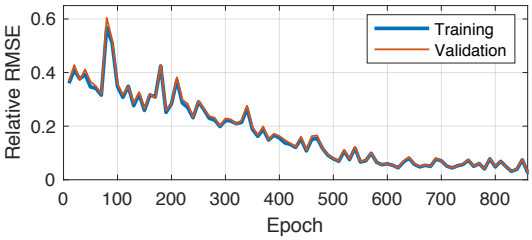

Fig. 8: Relative RMSE as training progresses.

C. Numerical Experiment Results

To evaluate the performance of the trained predictor, we introduce the following metric of Relative RMSE to fairly account for quantities of different units:

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

where $\overline{y_i^{(n)}}$ is the mean of $y_{i,t}^{(n)}$ over t and M is the size of validation data set. The progression of the Relative RMSE over the training and validation data set as the training progresses is depicted in Fig. 8. We observe almost no overfitting as the gap between training and validation performance is almost zero. We summarize the achieved testing MSE and

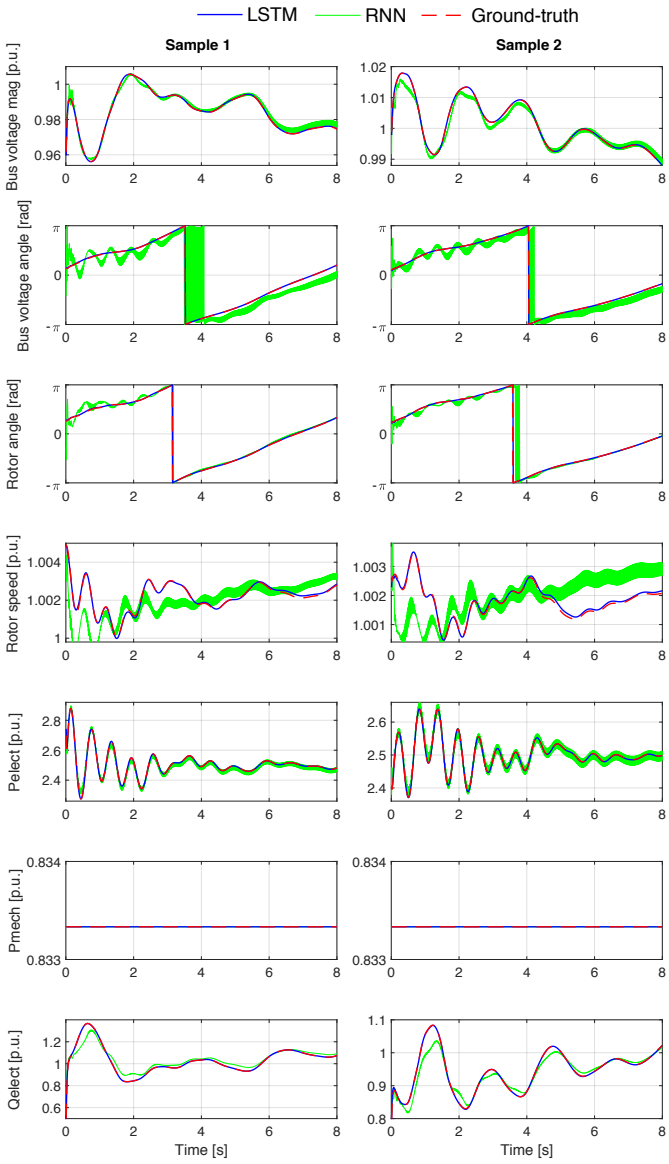

Fig. 9: Two testing samples. The voltage and rotor angles are re-wrapped in the plot (whereas in our computation they are unwrapped to be continuous). LSTM has a relative RMSE of 0.0434 for sample 1 and 0.0261 for sample 2. RNN captures the temporal dynamics less accurately with undesirable oscillations at various places. MLP's predicted trajectories are nonsensical and not plotted.

Relative MSE in Table II, compared with those achieved with a "vanilla" RNN. We see that the LSTM-based NN performs significantly better than the vanilla RNN. We further note that, we also evaluated the performance with MLP as opposed to LSTM/RNN, and the performance of MLP is much worse and not at all acceptable. These demonstrate that the use of LSTM is crucial for performance improvement.

To provide an intuitive observation of the achieved performance, we plot in Fig. 9 the ground-truth and predicted generator state trajectories for two representative sample cases, each

TABLE II: Performance Comparison of Different Predictor Models

Model	MSE	Relative RMSE
LSTM	$9.6 \cdot 10^{-5}$	0.0269
Vanilla RNN	0.0208	0.3850

TABLE III: Comparison of Computational Efficiency

Model	Offline Processing time [min]	Offline Compute Memory [MB]	Online Compute Time [s]	Online Compute Memory [MB]
LSTM	165.9	4014	0.4445	60
Vanilla RNN	28.1	3642	0.4052	18
Numerical	-	-	71.5*	182

corresponding to an $N-2$ contingency. The trajectories of *all the predictor outputs — the* 7 *state variables of Generator 1* (3) — are plotted. We observe that the ground-truth and LSTM-predicted trajectories overlap very closely. As such, the trained LSTM-based NN is capable to precisely predict the 7-dimensional generator dynamics given just the generator's initial condition as well as the external inputs. We also observe that RNN is less desirable than LSTM with evident deviations from the ground-truth as well as significant oscillations at various places (corresponding to the thick stripes in the plots).

Next, we provide the computation times and memories in both the offline training and online testing stages, comparing the LSTM-based NN, the vanilla RNN, and the conventional numerical solver of DAE. All the results are based on the desktop machine as described in Section IV-B, and are summarized in Table III. We observe that, *while LSTM achieves almost the same accuracy as the conventional solver, it is more than two orders of magnitude faster.* This demonstrates that our objective of greatly accelerating the transient analysis/computation of generator dynamics is successfully achieved by the proposed method. Here, we would like to clarify that, while the testing time in Table III is for predicting the dynamics of a single generator, in comparison, the conventional numerical solver is used to compute on the entire power system with 16 generators. Nonetheless, even by multiplying the testing time of the LSTM-based predictor by 16, which assumes a naive sequential computation for all the generators, the total testing time would be $0.4445s/\text{gen} * 16$ generators $= 7.112s$. This is still 10x faster than the conventional numerical solver. We note that the predictor based trajectory computation for all the generators can be trivially parallelized and hence more gain in computation speed can be easily achieved.

V. Conclusion

This paper proposes a novel ML-based method for post-contingency generator transient simulation for the purpose of comprehensive transient analysis. The generation of training, validation and testing data was performed by an automated contingency generation and simulation in a PST-based simulation platform. An LSTM network is utilized to learn the intrinsic characteristics of the generator dynamics. In the New York/New England 68-bus 16-machine power system, we have trained an LSTM network for a single generator's dynamic trajectories, and have shown it to reproduce the ground-truth trajectories very accurately. The proposed approach in this paper is fundamentally different from the existing ML-based transient analysis, and is considered a promising technique that can potentially replace the numerical integration methods in the area of online/real-time transient analyses with much faster computation.

The effectiveness and speed of the proposed approach make it an important first step and lay the foundation toward the development of complete transient analysis platform for entire power systems, which will be the focus of our future work. Leveraging the physics of the grid together with the proposed ML approach is currently being explored to achieve the goal of large-scale online transient analysis. In addition to line faults, other events such as line switching and load changes can also be easily taken into account.

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] R. Zhang, Y. Xu, Z. Y. Dong, and K. P. Wong, "Post-disturbance transient stability assessment of power systems by a self-adaptive intelligent system," *IET Generation, Transmission & Distribution*, vol. 9, no. 3, pp. 296–305, 2015.

[4] J. James, D. J. Hill, A. Y. Lam, J. Gu, and V. O. Li, "Intelligent time-adaptive transient stability assessment system," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 1049–1058, 2017.

[5] 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.

[6] G. S. Misyris, A. Venzke, and S. Chatzivasileiadis, "Physics-informed neural networks for power systems," *arXiv preprint arXiv:1911.03737*, 2019.

[7] Y. Zhao, J. Chen, and H. V. Poor, "A Learning-to-infer method for real-time power grid multi-line outage identification," *IEEE Transactions on Smart Grid*, vol. 11, no. 1, pp. 555–564, 2020.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] "Understanding LSTM networks, 2015." Available at https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

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

[11] 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.

[12] B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro, "Towards understanding the role of over-parametrization in generalization of neural networks," *arXiv preprint arXiv:1805.12076*, 2018.

[13] D. Kinga and J. B. Adam, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. IEEE, 2015, pp. 1–15.

[14] A. Paszke, S. Gross, F. Massa, A. Lerer *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8026–8037.