Learning Power System Dynamics with Noisy Data Using Neural Ordinary Differential Equations

Shaorong Zhang, Koji Yamashita, and Nanpeng Yu
Department of Electrical and Computer Engineering
University of California, Riverside
California, USA
nyu@ece.ucr.edu

***Abstract*—The ability to learn complex system dynamics is crucial to enhancing the reliability and stability of power systems. In this paper, we develop a novel neural ordinary differential equation (ODE) based algorithm to predict the transient trajectories of power systems. To handle noisy measurements, we propose a noise-removal module, which is implemented before the neural ODE module. The proposed algorithm is validated using the IEEE 118-bus system. The numerical study results demonstrated the superior accuracy of the proposed model over the baseline neural network (NN) and its robustness against measurement noise. Furthermore, the analytics results verified the generalization performance across different fault durations and locations.**

***Index Terms*—Neural ordinary differential equation, power system dynamics, transient trajectory.**

I. Introduction

Learning the dynamic behaviors of power systems is essential to performing stability analysis and to designing adaptive real-time control mechanisms. However, with the integration of renewable energy resources and the emergence of distributed generation, the behavior of power system dynamics becomes increasingly fast-evolving and complex. Under this changing environment, the ability to learn complex power system dynamics becomes indispensable for the reliable operation of power systems.

Traditionally, power system dynamic simulations have been widely used for transient stability assessment. This method relies on accurate physical dynamic parameters and the solving of a large set of differential algebraic equations. However, as electrical networks grow in complex and unpredictable ways, obtaining precise dynamic parameters for the various components within the system becomes a daunting task. Furthermore, solving a large number of differential equations is computationally intensive, making the real-time power system analysis and control extremely challenging.

In recent years, many machine learning (ML) algorithms have been proposed to learn the dynamic behavior of real physical systems with limited observation [1]. ML-based methods have the potential to address the challenges of computation burden and inaccurate dynamic parameters. In the domain of power system dynamic simulation, researchers often use binary classification models such as gated recurrent units (GRU) [2], recurrent neural network (RNN) [3], convolutional neural network (CNN) [4]–[6] to identify system stability issues. However, binary classification models do not provide detailed system dynamic trajectories to facilitate decision-making. To address the issue of inaccurate dynamic parameters, researchers have proposed physics-based neural ODEs to learn the parameters of the generator dynamic model using data from phasor measurement units (PMU) [7]. However, it is important to note that this approach does not accelerate the dynamic simulation process.

Therefore, data-driven machine learning approaches have emerged to predict power system dynamic trajectories. To fully utilize physical information, physics-informed approaches have been developed to learn the dynamic model by embedding the exact physical equations [8] or energy conservation law [9]. However, this group of models is still in the early stage of its development and has difficulties generalizing to a larger power system. Deep learning methods, such as long short-term memory (LSTM) network [10] and extreme learning [11], have been developed to predict future transient trajectories based on given initial conditions or past sensor measurements. But learning the power system dynamics directly does not guarantee good performance due to the high level of network complexity, the limited number of sensors and low sampling rate.

Neural ODEs present a strong and elegant method for learning the differential equations governing a dynamic system through trajectory data [1]. This technique efficiently utilizes state information from both nearby and distant time points, showing particular strength in scenarios with sparse or irregularly spaced sampling. In this paper, we propose a novel framework for learning the power system dynamic model using neural ODE with noisy power system measurements. The developed neural ODE-based algorithm is validated using the IEEE 118-bus system, which can reproduce complex power system dynamic behavior. The software modules and data for this research are available at [12].

The main contributions of this paper are listed below:

- We developed a neural ODE-based algorithm for learning complex power system dynamics and transient trajectories with limited training data.
- To mitigate noisy measurements in the system, we incorporated a noise-removal module in the proposed solution.
- Numerical study results show that the proposed algorithm achieves better performance in transient trajectory predic-

tion than the baseline neural network.
- Furthermore, the proposed algorithm demonstrates not only strong robustness against measurement noise but also great generalization capability across different system fault durations and locations.

The remainder of this paper is organized as follows. Section II formulates the power system dynamic model. Section III introduces the proposed technical methods. Section IV presents the numerical studies. Section V gives the conclusion.

II. Power System Dynamic Model

This study employs the classical power system dynamic model as the foundation and adds the governor model to make the transient trajectories more realistic. This provides a simplified yet insightful representation of the generator and system dynamics to evaluate the performance of the baseline and the proposed neural ODE-based algorithms.

In the dynamic simulations, each generator k ($k = 1, 2, \cdots, g$) is modeled with a set of differential equations representing the dynamics of the generators and governors. The classical generator model used in this study for simulating the dynamics is essentially the swing equation as shown in (1):

$$\frac{M_k}{\omega_0}\frac{d^2\delta_k}{dt^2} + \frac{D_k}{\omega_0}\frac{d\delta_k}{dt} = T_{m_k} + \Delta T_{m_k} - \frac{P_{e_k}}{\omega_k}, \tag{1}$$

where M_k denotes the inertia constant, ω_0 is the synchronous speed of the generator rotor, D denotes damping coefficient of generators, T_{m_k} denotes mechanical torque, ΔT_{m_k} denotes deviation in mechanical torque, P_{e_k} denotes active power output, δ_k denotes rotor angle, and ω_k denotes rotor speed.

To make the system dynamics more realistic in the simulations, a governor model is integrated with the classical model:

$$\begin{cases} \dfrac{d\mathrm{RY}_k}{dt} = -5\mathrm{RY}_k - 125\Delta\omega_k, \\ \dfrac{d\mathrm{CV}_k}{dt} = -5\mathrm{CV}_k + 5\mathrm{RY}_k, \\ \dfrac{d\mathrm{HP}_k}{dt} = -4\mathrm{HP}_k + 4\mathrm{CV}_k, \\ \dfrac{d\Delta T_{m_k}}{dt} = \dfrac{1}{9}\Delta T_{m_k} + \dfrac{7}{10}\dfrac{d\mathrm{HP}_k}{dt} + \dfrac{1}{9}\mathrm{HP}_k, \end{cases} \tag{2}$$

where RY_k denotes speed relay output, CV_k denotes control valve opening degree, HP_k denotes high-pressure turbine output. $\Delta\omega_k = \omega_k - \omega_0$ denotes deviation in rotor speed.

Finally, the electrical power of generator k, P_{e_k}, is calculated using the Kron reduced admittance matrix $\boldsymbol{Y}_g \in \mathbb{R}^{g\times g}$ and the internal voltages of the generator E_k,

$$P_{e_k} = \sum_{i=1}^{g} |Y_{ki}||E_k||E_i|\sin(\delta_{ki} - \theta_{ki}), \tag{3}$$

where E_k denotes internal voltage of synchronous generators and $\theta_{ij} = \frac{\pi}{2} + \arg((\boldsymbol{Y}_g)_{ij})$. Note that arg denotes the argument of a complex number.

III. Technical Methods

A. Overview of Neural ODE

Neural ODE parameterizes the derivative of the hidden state using a neural network, which can capture the intricate evolution of dynamic system states.

Given initial states $\boldsymbol{z}(t_0)$, neural ODE learns the function f parameterized by a neural network with weights θ [1]:

$$\frac{d\boldsymbol{h}(t)}{dt} = f(\boldsymbol{h}(t), t, \theta). \tag{4}$$

The corresponding loss function is defined as:

$$L(\boldsymbol{z}(t_1)) = L(\boldsymbol{z}(t_0) + \int_{t_0}^{t_1} f(\boldsymbol{z}(t), t, \theta)dt). \tag{5}$$

To update the model parameters during training, we use the adjoint method proposed in [1] to calculate the derivative of the loss function with respect to model parameters: $\frac{dL}{d\theta}$. Specifically, the dynamics of adjoint $\boldsymbol{a}(t) = \frac{\partial L}{\partial \boldsymbol{z}(t)}$ are given by another ODE:

$$\frac{d\boldsymbol{a}(t)}{dt} = -\boldsymbol{a}(t)^T\frac{\partial f(\boldsymbol{z}(t), t, \theta)}{\partial \boldsymbol{z}}. \tag{6}$$

The gradient of the loss function with respect to the parameters θ depends on both $\boldsymbol{a}(t)$ and $\boldsymbol{z}(t)$:

$$\frac{dL}{d\theta} = -\int_{t_1}^{t_0} \boldsymbol{a}(t)^T\frac{\partial f(\boldsymbol{z}(t), t, \theta)}{\partial\theta}dt. \tag{7}$$

In practice, Ricky et al. provides the “torchdiffeq“ library, which is equipped with ODE solvers implemented in PyTorch [13]. This ODE solvers have the option to output the state at multiple time steps $\mathbf{z}(t_1), \cdots, \mathbf{z}(t_T)$ and supports backpropagation using the adjoint method, where T denotes batch time.

B. Learning System Transient Trajectories with Neural ODE

The power system data utilized in this study is generated by solving Differential-Algebraic Equations (DAEs), which accurately encapsulate the intricate dynamics of the power system. In general, DAE have a semi-explicit form [14]:

$$\begin{aligned} \dot{\boldsymbol{y}} &= \bar{f}(\boldsymbol{y}, \boldsymbol{z}), \quad \boldsymbol{y}(t_0) = \boldsymbol{y}_0, \\ \mathbf{0} &= \bar{g}(\boldsymbol{y}, \boldsymbol{z}), \quad \boldsymbol{z}(t_0) = \boldsymbol{z}_0, \end{aligned} \tag{8}$$

where $\boldsymbol{y} = \boldsymbol{y}(t) \in \mathbb{R}^n$ are the dynamic states, $\boldsymbol{z} = \boldsymbol{z}(t) \in \mathbb{R}^m$ are the algebraic variables, $\bar{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ describes differential equations, $\bar{g} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$ describes algebraic equations. Under the assumption that the algebraic equation have a unique solution for $\boldsymbol{z}$: $\boldsymbol{z} = \boldsymbol{G}(\boldsymbol{y})$, the DAE is equivalent to the following system of ODEs:

$$\dot{\boldsymbol{y}} = \bar{f}(\boldsymbol{y}, \boldsymbol{G}(\boldsymbol{y})), \quad \boldsymbol{y}(t_0) = \boldsymbol{y}_0. \tag{9}$$

In the classical power system dynamic model, the algebraic variables $\boldsymbol{P}_{e_k}$ have a unique analytical solution (see equation (3)). Thus, we assume that the power system dynamic model can be approximated as a neural network parameterized by θ: $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \theta)$. Theoretically, $\boldsymbol{x}$ should include all the state variables. However, not all of them are observable. To simplify the model, we let $\boldsymbol{x} = [\boldsymbol{\delta}, \boldsymbol{\omega}]^T$, $\boldsymbol{\delta} = [\delta_1, \delta_2, \cdots, \delta_g]^T$,

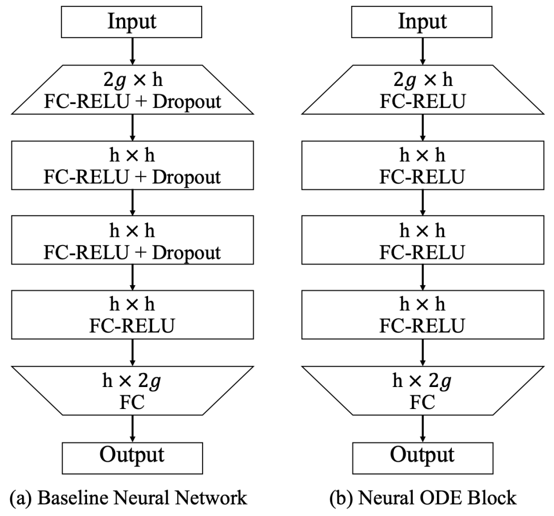

Fig. 1. Baseline neural network and neural ODE model

$\boldsymbol{\omega} = [\omega_1, \omega_2, \cdots, \omega_g]^T$. It has been observed from numercial study that the performance difference between the model with complete state vector and the simplified model is negligible. Next we present two learning schemes.

We first present a baseline NN to learn the dynamic model. The baseline NN is parameterized as a multi-layer perception (MLP) f_1. The corresponding loss function is defined as:

$$\mathcal{L}_1 = \|f_1(\boldsymbol{x}, \theta_1) - \dot{\boldsymbol{x}}\|_2^2. \tag{10}$$

In contrast, neural ODE parameterizes the dynamic model by neural ODE block $f_2(\boldsymbol{x}, \theta_2)$. Given a pair of states $\boldsymbol{x}(t_0)$ and a set of future states $\{\boldsymbol{x}(t_k)\}_{k=1}^{L}$, where $t_0 < t_k$ for all k, the corresponding loss function is:

$$\mathcal{L}_2 = \sum_{k=1}^{L} \|\boldsymbol{x}(t_0) + \int_{t_0}^{t_k} f_2(\boldsymbol{x}(t), \theta_2) dt - \boldsymbol{x}(t_k)\|_2^2. \tag{11}$$

The architecture of the baseline NN and neural ODE block are shown in Fig. 1. The dimension of the input and output layers are both $2g$, where g is the number of generators. The hidden layer dimension is h. To make it a fair comparison, the baseline NN and neural ODE models have similar network structures. The only difference is that three dropout layers were added in the baseline NN to mitigate overfitting.

C. Noise Removal Block

Suppose the measurements of rotor angle $\tilde{\boldsymbol{\delta}}$ and rotor speed $\tilde{\boldsymbol{\omega}}$ are noisy observations: $\tilde{\boldsymbol{\delta}} = \boldsymbol{\delta} + \boldsymbol{n}_\delta$, $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} + \boldsymbol{n}_\omega$, where $\boldsymbol{n}_\delta$ and $\boldsymbol{n}_\omega$ are Gaussian noise. The presence of measurement noise can significantly increase the transient trajectory prediction error. To mitigate this problem, we introduce a noise removal block before feeding the data to the neural networks for model training. Specifically, the function of smoothing is estimating $\bar{\boldsymbol{x}}$ at point k given a set of data points $\{\{t_k, \tilde{\boldsymbol{x}}(t_k)\}\}$. $\bar{\boldsymbol{x}}$ can be calculated as:

$$\bar{\boldsymbol{x}}(k) = \frac{\sum_{i=k-w}^{k+w} K_u(t_i - t_k)\tilde{\boldsymbol{x}}(t_k)}{\sum_{i=k-w}^{k+w} K_u(t_i - t_k)}, \tag{12}$$

where $K_u(t) = \frac{1}{u}K(\frac{t}{u})$ is a kernel function with bandwidth u. The window size is $2u+1$. Different kernels can be selected. Depending on the need for simplicity or precision, either the boxcar kernel or the Gaussian kernel can be adopted. This noise removal block will degenerate to moving average when selecting Boxcar as kernel for equidistant sampling time series.

The noise removal block includes padding and convolution, we add padding to handle boundary conditions and use convolution for smoothing data. The bandwidth and window size critically influences the regression's extent of smoothing. A smaller u and w yield a regression that closely follows the data, while a larger u provides a smoother estimate.

IV. Numerical Studies

A. Simulation Setup of IEEE 118-bus System

The transient trajectories of rotor angle and rotor speed are generated using time-domain simulations of the IEEE 118-bus system [15]. The simulation covers $11s$ of dynamic behavior of the system. During this period, two events occurred: at $t_1 = 1s$, a balanced ground fault causes the breakers of the transmission line l to open; at $t_2 (> 1)$, the fault is cleared. The simulation time step is set as 10ms for all events.

Three datasets were generated using a commercially available software[16]. They have the same initial states, highlighting the variation across fault durations and diverse fault points. The specific configuration is shown in Table I. For example, the disturbance location of the first dataset is in the middle of the 345 kV transmission line between buses 38 and 65. 471 trajectories are generated for different fault durations ranging from 30ms to 500ms with an interval of 1ms.

TABLE I
Simulation Dataset

Dataset	Fault location Bus I	Fault location Bus J	Fault duration range	Increment
1	38	65	[30ms, 500ms]	1ms
2	69	70	[50ms, 500ms]	50ms
3	110	112	[50ms, 500ms]	50ms

B. Generalization Across Fault Durations

To evaluate the generalization capability of the neural ODE and baseline NN models across different fault durations while fixing the fault location, dataset 1 is selected for training and testing. Specifically, the training data comprises dynamic trajectories corresponding to fault durations of $\{30, 80, 130, 180, 230, 280, 330, 380, 430, 480\}ms$. The test data consists of trajectories with fault duration of $100ms$; To ensure the same differential equations of the trajectories, we only use the post-fault state (after $2s$) for training and testing.

The hyperparameters of the neural network models are shown in Table II. Both neural networks use mean square error (MSE) as the loss function and Adam as the optimizer. During the training of neural ODE, the selection of $\{\boldsymbol{x}(t_k)\}_{k=1}^{L}$ will significantly influence the performance and training time of neural ODE. When $\Delta t = t_L - t_0$ is excessively large, the training time increases significantly. Conversely, an overly small Δt compromises the accuracy of trajectory prediction. We make a trade-off by setting $t_k = t_0 + k \times 10ms$ and $L = 10$, with L referring to the batch time as shown in Table II. For each training iteration, we randomly select an initial timestamp with replacement from the range $[t_2, t_3 - 100ms]$. Then starting from the selected initial state, we step forward-in-time through the ODE solver for 10 contiguous timestamps and back-propagate to update the neural network parameters, using the ODE solver Runge-Kutta of order 5 of Dormand-Prince-Shampine with absolute tolerance equal to 10^{-6}.

TABLE II
HYPERPARAMETERS OF NEURAL NETWORK MODELS

	Baseline NN	Neural ODE
Hidden dimension	1000	1000
Learning rate	0.001	0.001
Iterations	10000	5000
Batch time	-	10
Batch size	1024	256
Dropout rate	0.5	-

We evaluate the performance of the learned neural networks with the trajectory prediction task: given the initial states $[\boldsymbol{\delta}_0, \boldsymbol{\omega}_0]$, we use neural ODE and baseline NN to solve this initial value problem and then compare the trajectory prediction results. We calculate the predicted rotor angle and rotor speed as $\hat{\boldsymbol{\delta}}$ and $\hat{\boldsymbol{\omega}}$. The MSEs for predicted and actual trajectories are calculated by: $\text{MSE}(\hat{\boldsymbol{\delta}}, t) = \frac{1}{g}\|\hat{\boldsymbol{\delta}}(t) - \boldsymbol{\delta}(t)\|^2$ and $\text{MSE}(\hat{\boldsymbol{\omega}}, t) = \frac{1}{g}\|\hat{\boldsymbol{\omega}}(t) - \boldsymbol{\omega}(t)\|^2$.

The trajectories with fault duration of $100ms$ are used for testing and the predicted trajectories are shown in Fig. 2. The MSEs of the predicted and actual trajectories are shown in Fig. 3. The experimental results show that the baseline NN achieves acceptable results in the task of trajectory prediction. However, neural ODE demonstrates significantly better performance, especially in the long-term predictions. In addition, the average simulation time of the proposed method is only 1.22 s, which is much shorter than the 14.49 s required by traditional dynamic simulation software on the same computing device.

C. Robustness Against Measurement Noise

The robustness of our proposed model against measurement noise was evaluated in this subsection. First, measurement noises are superimposed on the generated trajectories in dataset 1. In practice, rotor speed usually cannot be directly measured and must be computed based on the measurements from PMUs. For simplicity, it is assumed that the measurement noises on the rotor angle and rotor speed follow a Gaussian distribution: $e \sim N(0, 0.02)$. Next, we evaluate the effectiveness of the noise removal block by conducting two groups of experiments: in the first group, the neural ODE model is learned from noisy data, in the second group, we add a noise removal block before the neural ODE model.

In the noise removal block, the bandwidth is set to 0.07 and window size to 15. The kernel is selected as Gaussian for convolution. The result in Fig. 4 shows that the noise will significantly affect the performance of neural ODE model, and

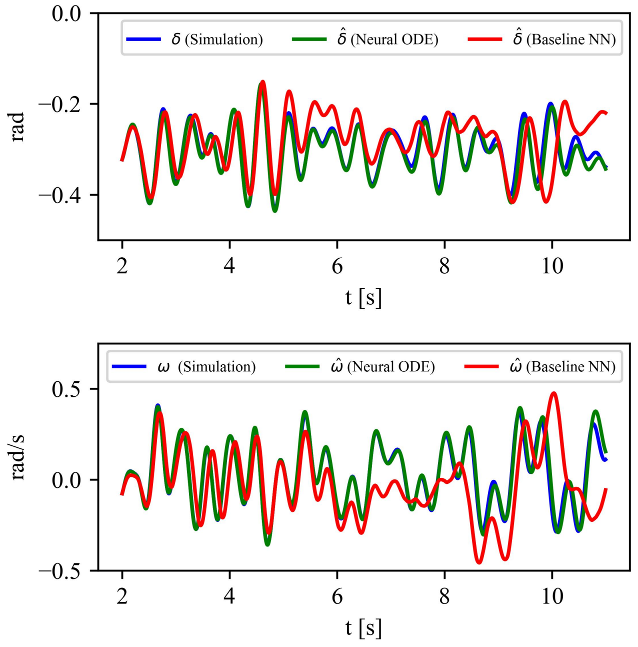

Fig. 2. Predicted and actual trajectories of the generator at bus 49 with fault between bus 38 and bus 65 and fault duration of $100ms$

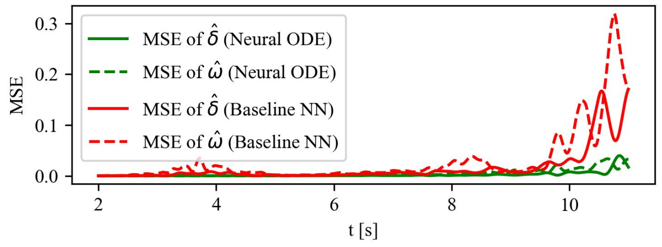

Fig. 3. MSEs of predicted trajectories of all 19 generators with fault between bus 38 and bus 65 and fault duration of $100ms$

adding the noise removal block can significantly reduce the MSE of the predicted trajectories.

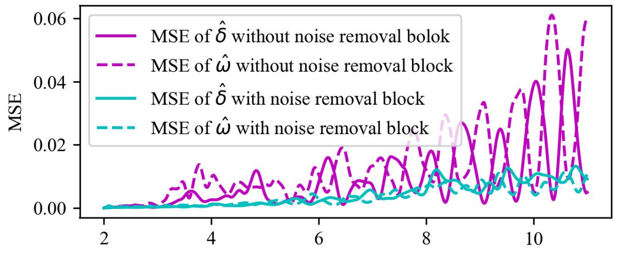

Fig. 4. MSE of predicted trajectories of all 19 generators using neural ODE with fault between bus 38 and bus 65 and fault duration of $100ms$

D. Generalization Across Fault Locations

The generalization capability of the neural ODE model is evaluated through datasets 2 and 3. The training dataset contains all the trajectories in the dataset 2 and the test dataset include rotor angle trajectories in dataset 3 with fault duration of 100ms. The result in Fig.5 shows that the neural ODE performed well predicting trajectories related to fault locations not included in the training dataset, but it doesn't guarantee perfect generalization across all fault locations.

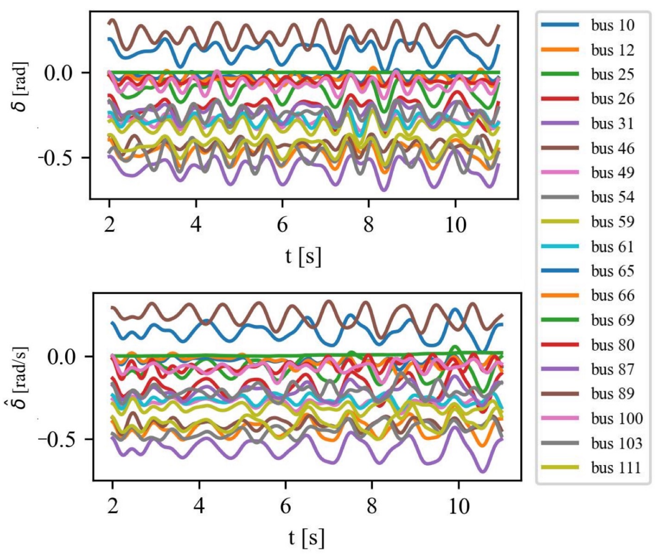

Fig. 5. Predicted and the actual rotor angles of all 19 generators, the training data related to fault between bus 69 and bus 70, the test data related to fault between bus 110 and bus 112 with a fault duration of $100ms$

V. Conclusion

In this paper, we propose a novel framework that employs neural ODEs to learn complex power system dynamics from noisy measurements. The performance of the algorithms is validated using the IEEE 118-bus system. The results of numerical studies highlight the superior performance of the proposed neural ODE-based algorithm in trajectory prediction, demonstrating the generalization capability across fault durations and locations. Furthermore, the addition of a noise removal block significantly enhances the robustness of the model against measurement noise, which is a common issue in real-world power systems.

In practice, the number of sensor measurements and fault events are limited and the operation condition varies greatly over time. To address these challenges, a promising approach is to improve the neural network's generalization capability by incorporating physics-based priors, such as network topology and functional form of the underlying DAEs. Finally, examining a broader range of electrical variables and controllers, such as voltage trajectories and voltage regulation could be beneficial. These extensions will be explored in the future.

References

[1] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[2] C. Ren, Y. Xu, and R. Zhang, "An interpretable deep learning method for power system transient stability assessment via tree regularization," *IEEE Trans. on Power Syst.*, vol. 37, no. 5, pp. 3359–3369, 2021.

[3] L. Zhu, D. J. Hill, and C. Lu, "Intelligent short-term voltage stability assessment via spatial attention rectified rnn learning," *IEEE Trans. on Ind. Inform.*, vol. 17, no. 10, pp. 7005–7016, 2020.

[4] R. Yan, G. Geng, Q. Jiang, and Y. Li, "Fast transient stability batch assessment using cascaded convolutional neural networks," *IEEE Trans. on Power Syst.*, vol. 34, no. 4, pp. 2802–2813, 2019.

[5] M. Cui, F. Li, H. Cui, S. Bu, and D. Shi, "Data-driven joint voltage stability assessment considering load uncertainty: A variational bayes inference integrated with multi-cnns," *IEEE Trans. on Power Syst.*, vol. 37, no. 3, pp. 1904–1915, 2021.

[6] V. Miranda, P. A. Cardoso, R. J. Bessa, and I. Decker, "Through the looking glass: Seeing events in power systems dynamics," *Int. J. Electr. Power Energy Syst.*, vol. 106, pp. 411–419, 2019.

[7] X. Kong, K. Yamashita, B. Foggo, and N. Yu, "Dynamic parameter estimation with physics-based neural ordinary differential equations," in *2022 IEEE Power & Energy Society General Meeting*, pp. 1–5.

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

[9] S. Zhang and N. Yu, "Learning power system dynamics with nearly-Hamiltonian neural network," in *2023 IEEE Power & Energy Society General Meeting*, pp. 1–5.

[10] J. Li, M. Yue, Y. Zhao, and G. Lin, "Machine-learning-based online transient analysis via iterative computation of generator dynamics," in *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids*, pp. 1–6.

[11] H. Yang, W. Zhang, F. Shi, J. Xie, and W. Ju, "PMU-based model-free method for transient instability prediction and emergency generator-shedding control," *Int. J. of Electr. Power Energy Syst.*, vol. 105, pp. 381–393, 2019.

[12] S. Zhang, 2023. [Online]. Available: https://github.com/szhan311/Neural_ODE_Power_System_Dynamic.

[13] R. T. Q. Chen, *Torchdiffeq*, 2018. [Online]. Available: https://github.com/rtqichen/torchdiffeq.

[14] G. Wanner and E. Hairer, *Solving ordinary differential equations II*. Springer Berlin Heidelberg New York, 1996, vol. 375.

[15] P. S. T. C. Archive. "118 bus power flow test case." (Aug., 1993), [Online]. Available: https://labs.ece.uw.edu/pstca/pf118/pg_tca118bus.htm.

[16] Power System Stability Study Group, *Integrated Analysis Software for Bulk Power System Stability*. CRIEPI Report: ET90002, Jul. 1991.