Physics Informed Reinforcement Learning for Power Grid Control using Augmented Random Search

Kaveri Mahapatra, Xiaoyuan Fan, Xinya Li, Yunzhi Huang, Qiuhua Huang
Energy and Environment Directorate
Pacific Northwest National Laboratory
xiaoyuan.fan@pnnl.gov

Abstract

Wide adoption of deep reinforcement learning in energy system domain needs to overcome several challenges, including scalability, learning from limited samples, and high-dimensional continuous state and action spaces. In this paper, we integrated physics-based information from the generator operation state formula, also known as Swing Equation, into the reinforcement learning agent's neural network loss function, and applied an augmented random search agent to optimize the generator control under dynamic contingency. Simulation results demonstrated the reliability performance improvements in training speed, reward convergence, and future potentials in its transferability and scalability.

1. Introduction

Physics-informed machine learning (PIML) are getting increasing attentions from scientists and engineers, to alleviate the challenges in scientific discovery, such as high-cost data acquisition, uncertainty quantification, and robust control. Recent progress in physics informed neural network (PINN) and physics aware neural network (PANN) models has demonstrated that they could be universal function approximators, and are capable of encoding any underlying physical laws that govern a given dataset. PINN can be trained to solve supervised learning tasks while respecting the governing physical laws of system described by general nonlinear partial differential equations. In a nutshell, PINN-based problem formulations takes advantage from both data and physics of the system. It also utilizes the same formulation for both forward and inverse problems. Therefore, PINN can not only overcome the curse of dimensionality problem, but also be used in any scientific field for discovering dynamical system much faster than traditional methods.

In energy domain, dedicated research efforts involving PIML have been done, primarily for monitoring and predicting the behaviors of power system during steady state or transient conditions. [1] introduces a PINN-based framework, in which the rotor angle and frequency can be accurately predicted with 87 times speed up. In addition, [2] applies PINN with PMU measurements to monitor the system inertia in real-time, its performance has been verified through comparisons with Unscented Kalman Filter (UKF). [3, 4] discuss challenges and opportunities when adopting PINN for transient stability analysis and predicting states of a larger system, while [5] discusses PINN modeling in applications to state estimation (SE) under limited observability. Furthermore, [6] presents application of PINN with worst-case guarantees for the DC optimal power flow problem, and [7] applies PINN to estimate the power system dynamic behavior in presence of a converter based generation and voltage disturbances.

In addition, PANN has been utilized in the problem of distribution system state estimation [8], which resulted in the reduction of total number of coefficients needed to parameterize the mapping between the measurements and system state. Last but not least, based on the emerging Graph Neural Networks (GNN), [9] proposes a hybrid scheme to embed physics modeling of power systems into GNN, which provides a reliable and explainable parameter and state estimation framework, and [10] presents a physics-informed graph learning algorithm to estimate network parameters of 3-phase power distribution systems. It should be noted that most of these works demonstrated open loop operation from small to large power system, but none of them has utilized PINN in closed loop power system control scenario.

1.1. Opportunities and Challenges in Reinforcement Learning for Control

The ever growing penetration of renewable energy resources calls upon improvement of grid control and emergency responses. Recent methods on application of reinforcement learning (RL) in power system was demonstrated in L2RPN competition [11], in which many winning teams utilize power system agnostic approaches. When combined with deep neural network (DNN), deep reinforcement learning (DRL) has shown super-human performance without any human knowledge in areas that were intractable before [12]. In robotics [13], autonomous driving [14], power grid emergency responses [15], DRL agents and bots achieved promising results,

More specifically, in a partially observable environments such as power system networks, with control tasks occurring over a long-time horizon in highly dimensional states and action spaces, RL and DRL with agents can achieve promising results [16]. DRL is used to learn the environmental states and value of performing a set of control action at each state; once trained, DRL could suggest control actions that lead to optimal outcomes. In short, DRL is a set of learning algorithms that produce fully autonomous agents through interactions with their environments, and learn optimal behaviors for design or control optimization.

1.2. Enhance RL with Derivative-free Techniques

Existing RL frameworks for power system control [15, 17] focus on Q-learning, which has several challenges for larger system application, such as scaling up the solutions, time-consuming training, and hyper-parameter tuning. From the ML algorithm perspective, these core challenges still remain: 1) scalability, 2) learning from limited samples, and 3) high-dimensional continuous state and action spaces. Recent developments in RL with model-free, derivative-free algorithms and evolutionary search strategies have led to techniques such as basic random search (BRS) [18] and augmented random search (ARS) [19, 20] algorithms, which are highly competitive yet scalable alternatives to traditional gradient based methods.

BRS aims to pick a paramaterized policy π_θ, shock (or perturb) the parameters θ by applying $-\nu\delta$ and $+\nu\delta$, (where $\nu < 1$ is a constant noise and δ is a random number generate from a normal distribution). Then actions are applied based on $\pi_{(\theta+\nu\delta)}$ and $\pi_{(\theta-\nu\delta)}$, the rewards $r(\theta+\nu\delta)$ and $r(\theta-\nu\delta)$ are collected resulting from those actions. With the rewards of the perturbed θ, the average $\Delta = {}^{1}\!/_{N} \sum [r(\theta+\nu\delta) - r(\theta-\nu\delta)]\delta$ for all the δ is computed, then the parameters θ are updated using Δ and a learning rate α with $\theta^{j+1} = \theta^{j} + \alpha\Delta$.

ARS differs from BRS by utilizing three axes of enhancements, (1) division by standard deviation, (2) normalization of states, (3) utilization of top performing directions to influence the policy updates. ARS utilizes policy parameter space explorations, and the sampled roll-outs are performed for estimating the gradient of the returns. To achieve desired performance and eliminates the need of expensive hyper-parameter tuning, ARS only utilizes a small number of sensitive hyper-parameters. Application of ARS has been demonstrated in grid emergency control application [15], which has shown to have faster convergence performance in achieving better solutions, less sensitivity to hyper-parameters and random seeds. ARS has also been demonstrated in voltage stability control [21], which uses load shedding based continuous control actions obtained with a feedforward neural network (FNN) as well as Long Short Term Memory (LSTM) based policy model.

Since ARS can be scaled up for larger systems, we have considered ARS in this work. Different fault scenarios in power system can be used as multiple tasks to learn from during training. A large set of power system dynamic simulations can be performed with various perturbed policies under fault scenarios during training. These simulations are generally paralleled using parallel scheme for ARS (PARS) version and actor-based computations to speed up the training process. The key components in PARS are the ARS learner, workers, and the environment rollout actors, who exchange information during the process of learning. PARS performs search and exploitation to determine the optimal policies. However, implementations of PARS require multiple CPU cores to obtain the optimal results.

At the beginning of training, the ARS workers receive the policy weights distributed by ARS learner as well as statistical features of the observations from the previous iteration. Different policy network weight perturbations are then performed by workers. A set of slave actors receive these statistical features and perturbed policy weights from each worker. Single rollout is then performed by each slave actor for different tasks while inferring the perturbed policy. Observations and reward obtained by each rollout are sent back to the master worker, and statistical features of observations as well as the perturbation weights from each slave worker are sent back to the ARS learner, which then updates the weight policy. The basic algorithm to update these weight policies with ARS can be found in [20].

Since ARS is a model-free approach, the learning of all the control actions requires a large number of fault scenarios, and thus requires more policy update iterations during training. Therefore, the only way to influence ARS decisions based on system dynamics and to reduce the search space, is by introducing physics-informed algorithm in the loss calculation while updating the policy network parameters, which is the main focus of our work.

1.3. Methodology and Contribution

In addition to the derivative-free nature and scalability features of ARS, it is critical to leverage PINN to exploit the underlying physical laws governing power systems and further enhance the penetration of DRL applications in energy domain. In this paper, we propose a Physics Informed Reinforcement Learning framework, which aims to integrate generator operation state formula in the DRL agent's neural network loss function, as well as to apply an ARS agent to optimize the generator control under dynamic contingency.

The objective of a power system control framework is to ensure the generation and delivery of the electric power reliably, while maintaining voltage and frequency within allowable limits. The proposed method focuses on the control of individual generator in a decentralized manner; it can potentially be extended to a set of generators, which plays a crucial role in achieving the control objective in modern control centers by controlling the power output of generators. More specifically, training a RL agent for single generator node has the following benefits:

- minimize the observation state and action space and thus speed up the training process;
- integrate the physics-based generator model and be applied directly at RL agent's loss function;
- the trained RL agent has potential to be transfered to another generator with little or none further training thus improving both transferability and scalability.

To evaluate the performance of the proposed method, comparison with non-physics-informed RL agents in terms of its effectiveness, response time may be performed. A contribution summary of the paper is given as follows:

- A physics informed Neural Network model (PINN) in reinforcement learning-based control (RLC) framework has been developed;
- Application of the proposed RL framework for power system oscillation damping has been demonstrated;
- Conducted comprehensive performance evaluation of the proposed RL framework on system, demonstrated the benefits of introducing PINN, and assessed potential influence of random seed.

The paper is organized into six sections. Section 2 briefly discusses about power system dynamic equations and contingency modeling. Section 3 presents the proposed workflow for applying PINN in RLC framework. Section 4 provides details regarding PINN implementation in ARS algorithm. Section 5 presents discussion on the simulation results and performance evaluation. Section 6 concludes the paper.

2. Power System Simulation Environment for Synthesized Two-bus System

In general, the model of power systems can be expressed in the form of nonlinear differential and algebraic equations as follows.

$$\begin{aligned}\dot{\mathbf{x}} &= f(\mathbf{x}, \mathbf{u}, \mathbf{z}) \\ 0 &= \mathbf{g}(\mathbf{x}, \mathbf{u}, \mathbf{z})\end{aligned} \tag{1}$$

where, $\mathbf{x} \in \mathbf{R}^{N_x}$, $\mathbf{u} \in \mathbf{R}^{N_u}$ and $\mathbf{z} \in \mathbf{R}^{N_z}$ are the dynamic state variables, input variables, and algebraic variables, respectively. For example, in the single machine infinite bus (SMIB) model, $\mathbf{x}$ represents the rotor angle of the generator δ_1; $\mathbf{z}$ represents the bus voltage magnitudes $[v_1, v_2]$ and phase angle δ_2; and $\mathbf{u}$ represent the mechanical power input P_m of generator. The dynamics of the SMIB can be expressed as follows [1, 22] :

$$f(t, P_m) : M\ddot{\Delta\delta} + D\dot{\Delta\delta} + p_e - P_m = 0 \tag{2}$$

where, M represents inertia constant, $p_e = \frac{v_1 v_2}{X_{12}} \sin(\Delta\delta)$, X_{12} is transmission line impedance. Assuming $\delta_2 = 0$ for infinite bus, $\delta_1 - \delta_2 = \Delta\delta$, the damping coefficient $D = 0$ is considered.

To further simulate and explore the controllability of generator oscillations under severe bus faults, a synthesized two-bus model is created based on SMIB system, so that the voltage behavior at non-generator bus could be simulated correspondingly. Under steady state, machine is delivering power $p_e = P_m$ to the non-generator bus and the rotor angle of the machine is maintained at $\Delta\delta = 0$. Under a bus fault at non-generator bus, its bus voltage drops to 0 p.u. before clearing, then it is assumed to return to 1 p.u. for simulation simplicity; when P_m is assumed as constant without any action, it leads to undamped oscillations in the system when the fault was not cleared within critical clearing time (T_c).

Therefore, when the fault clearing time $\geq T_c$, P_m needs to be controlled in order to minimize the

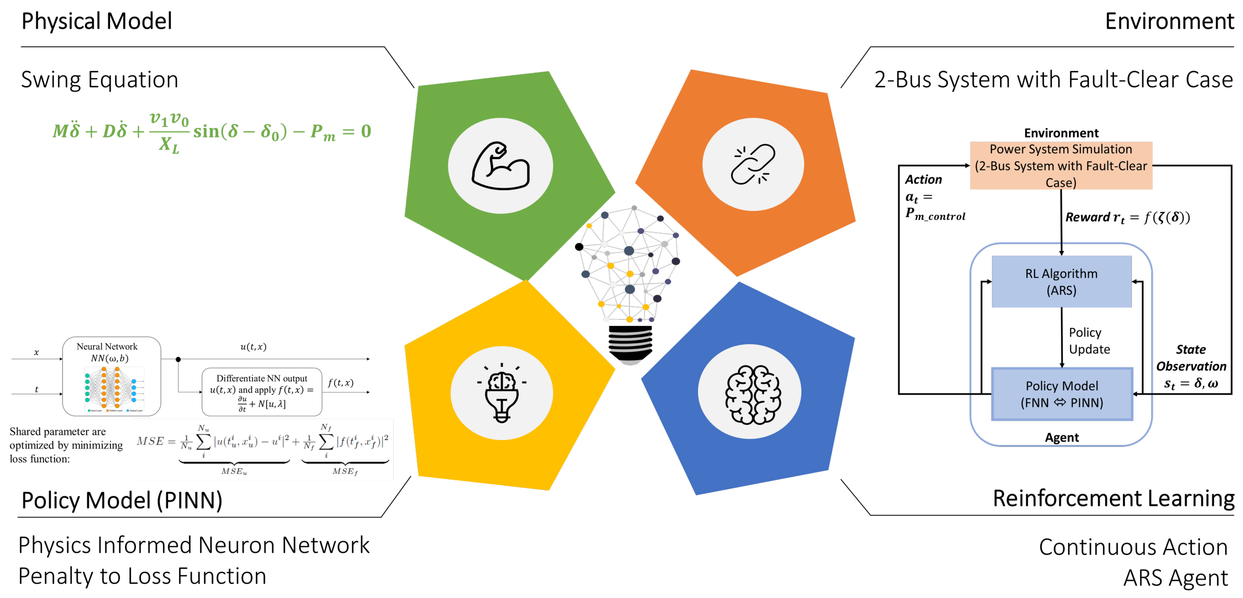
Physical Model
Swing Equation
$M\ddot{\delta} + D\dot{\delta} + \frac{v_1 v_0}{X_L}\sin(\delta - \delta_0) - P_m = 0$
Environment
2-Bus System with Fault-Clear Case
Environment
Power System Simulation (2-Bus System with Fault-Clear Case)
Action $a_t = P_{m\_control}$
Reward $r_t = f(\zeta(\delta))$
RL Algorithm (ARS)
Policy Update
State Observation $s_t = \delta, \omega$
Policy Model (FNN ⇔ PINN)
Agent
Neural Network $NN(\omega, b)$
x
t
$u(t,x)$
Differentiate NN output $u(t,x)$ and apply $f(t,x) = \frac{\partial u}{\partial t} + N[u,\lambda]$
$f(t,x)$
Shared parameter are optimized by minimizing loss function:
$MSE = \underbrace{\frac{1}{N_u}\sum_i^{N_u} |u(t_u^i, x_u^i) - u^i|^2}_{MSE_u} + \underbrace{\frac{1}{N_f}\sum_i^{N_f} |f(t_f^i, x_f^i)|^2}_{MSE_f}$
Policy Model (PINN)
Physics Informed Neuron Network
Penalty to Loss Function
Reinforcement Learning
Continuous Action
ARS Agent

Figure 1. A flow diagram for the proposed method.

oscillation of the rotor angle due to fault such that the states could be brought back to the pre-fault operating condition values. The following sections introduce different RL algorithms presented for this problem, as well as the proposed workflow for controlling of the generator mechanical power input for stabilizing rotor angle after disturbance using the proposed method.

3. Reinforcement Learning-based Control

A typical RLC framework consists of four components: physical model, environment, policy model (i.e., PINN), and RL algorithm as shown in Fig. 1. Physical model implies simulation model representing the power system behavior. Markov decision processes (MDPs) are generally used to model a fully or partially observable environment, key elements in a MDP are as follows: 1) state space S, 2) action space A, 3) environment transition function $P : S \times A \to S$, 4) reward $R : S \times A \to R$, 5) discount factor $\gamma \in [0, 1]$.

Moreover, the RL agent observes the state $o_t \in S$ at each time step, calculates reward $r_t \in R$ from the environment and then selects an action $a_t \in A$ based on its action policy to update the environment. The objective is to learn selecting the optimal action given a set of observations to maximize the rewards over time. An agent learns a control policy through interacting with the environment via trial-and-errors, executes the policy and adds randomness to the actions. If the result (based on the reward r_t) is better than expected, policy is updated via some optimization techniques (e.g., stochastic gradient descent, SGD) to do more of the same in the future; otherwise, do less.

In our study, a FNN is utilized as a policy model [23]. The policy model maps a set of observation states to space of control actions, and mimics the potential system response and following control actions. A fault event will trigger voltage drop on load bus and thus oscillation occurs on dynamic states. The generator bus's rotor angle and speed are considered as the observation states (S), while continuous action (A) is performed by adjusting the mechanical power. Practically, changing mechanical power input in a large amount requires adjusting the governor's parameters. Without loss of generality, the P_m input to a system can be adjusted between its minimum and maximum value $[P_m^{min}, P_m^{max}]$ without changing other settings. The continuous change in mechanical power ($P_m^{min} \leq P_m \leq P_m^{max}$) will be determined from the policy model.

Reward encourages the event of bringing the observation states back to equilibrium by modifying the policy network. Reward R consists of two terms: the first one is a function of damping ratio of system state, the second one represents the similarity of the network to the pre-disturbance system status. Typically for large dimensions of observation and action space, RL is applied for the policy model training, which learns to find a policy to maximize the expected reward when following this policy in the given environment.

4. Introducing PINN in RL-based Power System Control

In order to influence the ARS decisions by introducing physical system dynamics into learning, in this work a physics informed RLC framework has been formulated and shown on the right hand side of Fig. 1. It consists of three components: 1) a traditional RLC framework; 2) A PINN based policy network; 3) A physics based system model for inclusion of system dynamical equations into PINN training. This section discusses how PINN is introduced into the ARS algorithm.

4.1. PINN in ARS algorithm

To study physics-informed RL agent performance in terms of scalability and sampling efficiency, we propose to apply one ARS agent on single generator node in the test system. The generator's rotor angle and speed will remain as the observation states, which are the RL agent's neural network inputs.

In this work, the goal is to utilize PINN for policy network training. As shown in Fig. 1, the output of PINN and the associated physics-based information of generator operational states are provided as input to the RL agent's neural network loss function during its training. PINN-based ARS has been implemented as a modification of traditional ARS with surrogate gradients, and the modified algorithm is described in **Algorithm 1**.

The main difference between PINN and conventional neural network is the training process of the shared parameters, especially regarding the loss minimization with initial/boundary data on $\delta(t;x)$ and collocation points for $f(t,x)$ in an episode. Comparing to [1], the neural network predicting $f(t,x)$ has the same parameters for predicting $\delta(t;x)$, but now with different activation functions.

Certain disturbance events in the test system may trigger oscillations, which may require necessary control action for damping. Continuous actions P_m^c should be applied to bring back the system to stability, where reward encourages the system states δ and ω return to equilibrium,

$$P_m = P_m^c\left(P_{\max} - P_{\min}\right), P_m^c \in [0,1] \tag{3}$$

the proposed PINN-based ARS agent uses an additional penalty term MSE_f in the loss function, which represents the physic laws of Swing Equation,

$$loss = MSE_u + MSE_f, \quad MSE_f = f(t, P_m) \tag{4}$$

Episodic loss considers the mean squared error at a finite set of collocation points. Considering physical laws during model training allows to bound the space of admissible solutions to the neural network parameters, which translates to a lower requirement in both the amount of training data and neural network size. Those potential advantages of PINN are eliminating the need of large number of training set as well as utilizing a simple network structure.

Algorithm 1: Modified ARS with PINN

```
Result: Policy actor network, policy
        $f_{net}(\pi_\theta, s, t)$, $s$ is state vector at time $t$
  , $n$ is the number of directions
Initialization: initial parameters $\theta^{(0)}$ , loss
function $f_P(\theta)$, learning rate $\eta$, and hyper
parameters $\alpha, \beta, \sigma^2, P$;
for $t = 1$ to $T$ do
    Calculate the surrogate gradients $\nabla f\left(\theta^{(t)}\right)$ ;
    Update low-dimensional guiding subspace
      $U$ with surrogate gradient;
    Define search covariance
      $\Sigma = \frac{\alpha}{n} I + \frac{1-\alpha}{k} U U^T$ ;
    for $i = 1$ to $P$ do
        Sample perturbation $\varepsilon_i \sim N\left(0, \sigma^2 \sum\right)$;
        Compute antithetic pair of losses
          $f(\theta - \varepsilon_i)$ and $f(\theta + \varepsilon_i)$, where,
          $f = r(\pi(\theta + \varepsilon_i))$ is the episodic reward
          using policy model $\pi$ whose weights $\theta$
          have selected perturbations $\epsilon_i$
    end
    Compute guided evolutionary search
      gradient estimate
      $g = \frac{\beta}{2\sigma^2 P} \sum_{i=1:P} f\left(\theta + \varepsilon_i\right) - f\left(\theta - \varepsilon_i\right) -$
      $loss_{f_{PINN}}$, where $loss_{f_{PINN}} =$
      $f_P\left(\pi\left(\theta + \varepsilon_i\right)\right) - f_P\left(\pi\left(\theta - \varepsilon_i\right)\right)$;
    Update parameters using gradient descent
      $\theta \leftarrow \theta - \eta g$;
end
```

Loss is calculated at each instance of t in an episode of fault scenario as follows:

$$loss_t : f_{net}(\pi_\theta, s_i, t) : M\Delta\ddot{\delta} + p_e - \left(P_{\max} - P_{\min}\right) p_m^c(\pi_\theta) \tag{5}$$

loss calculated over an episode length N_e of a simulation duration ($0 \leq it_s \leq N_e t_s$) is given as

$$loss_e : f_P = \sum_{i=1:N_e} f_{net}\left(\pi, s_i, t\right) \tag{6}$$

where p_m^c is the controller output from policy network given the system states s_i as input, for any given instant $t = it_s$. Similarly, reward is calculated as a function of two terms for an episode length N_e is as follows:

$$r_t = r_{similarity} + 0.1\zeta_\delta \tag{7}$$

$$r_{similarity} = \mu_{s_i, i:it_s \le t} - s_0 \tag{8}$$

where $r_{similarity}$ indicates the similarity of the current system states s_i to the pre-fault values s_0, s_0 indicates the pre-fault observation state at instant $t = 0$; μ_{s_i} represents the mean of observation states s_i until instant t, ζ_δ is the damping ratio of δ calculated by considering samples until instant t during an episode.

5. Simulation and Results

A synthesized two-bus system with one generator has been used as a test system. At the non-generator bus, self clearing bus faults with different duration have been used in the training cases. For each scenario, fault was applied at 0.5s, Fig. 2 shows the bus voltage drop to 0, and then was cleared after 6 cycles. Time domain simulations were run for 10s with a time step of 0.1s. Fault events with different fault duration trigger voltage drop on load bus, as a result, equilibrium gets broken, and oscillation continues without any action. Both δ, ω are considered as observation states for RLC framework. A typical fault scenario without any control action is presented in Fig. 3-6, which shows undamped oscillations for both δ and ω, as well as P_e due to no responsive action in mechanical power P_m. Damping ratio of δ is then calculated for each fault simulation. Larger damping ratio indicates returning the system to equilibrium faster. Damping ratio can be improved by continuously changing the mechanical power input in Eq. (3) to the system, P_m is considered as a control action in this test system where P_m^c is the control decision coming from RLC framework.

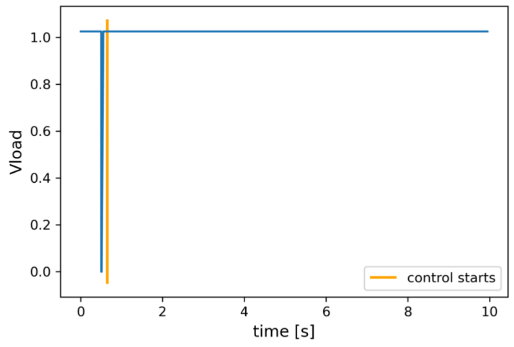

Figure 2. Synthesized load bus voltage considering bus fault in 10 second simulation.

We have tested two RLC implementations; (1) with traditional ARS (noPINN); (2) with PINN in ARS. This

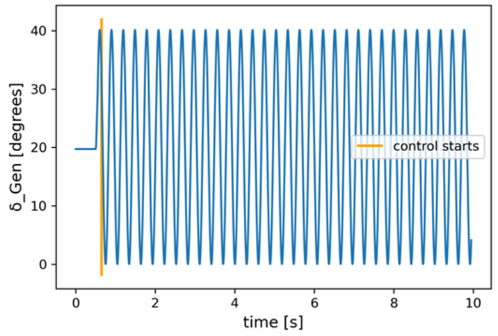

Figure 3. The simulated rotor angle with sustained oscillation when no control action was applied.

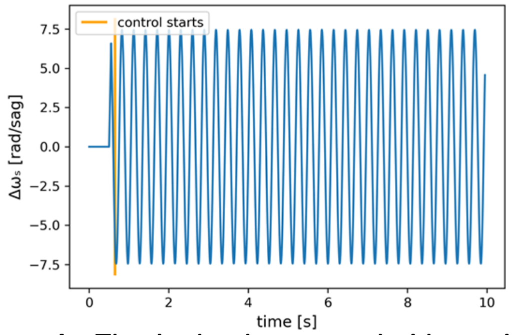

Figure 4. The simulated rotor speed with sustained oscillation when no control action was applied.

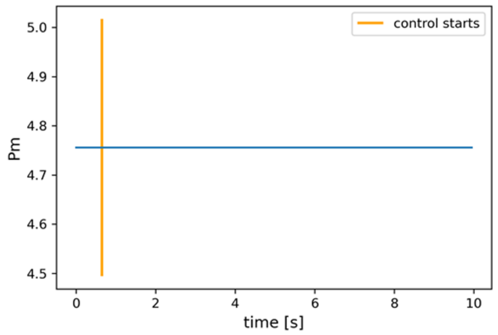

Figure 5. The simulated mechanical power when no control action was applied.

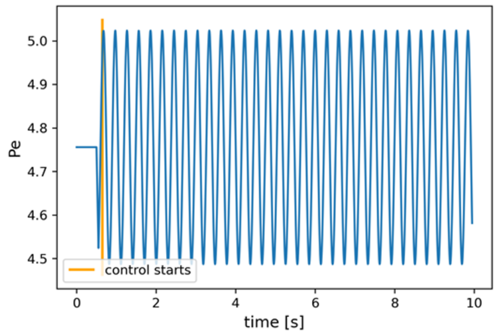

Figure 6. The simulated electrical power with sustained oscillation when no control action was applied.

section presents corresponding discussions on results from those two implementations. To evaluate the performance of these two methods during training, reward is calculated for each fault scenario as a function of the damping ratio of δ and a function of similarity to the pre-fault system operating condition. Average reward has been used for calculating rewards from all fault scenario episodes considered during training. Convergence of the RLC model is obtained when the average reward is positive and not improving over the following iterations.

Similarly, in testing stage, different fault scenarios which has not been used in the training were presented to the system, then the ability of the RLC framework to navigate the system back to normal state for each testing scenario. Detailed comparison between traditional ARS and PINN-based ARS application to power system control problem are presented below.

5.1. Traditional ARS

Traditional ARS algorithm has been tested with FNN architecture for policy model. The network size was set to [2,2]. For FNN only architectures, the last 10 recent observation states were stacked and used as input to ARS, and thus the dimension of the input was 20. From the state observations, damping ratio and the difference between pre-fault and temporal status were calculated and given as input to the ARS agent. Action dimension was considered to be 1 for P_m. A total of 24 perturbations were considered for ARS agents along with 24 cores for training. Total 8 top performing perturbations were used for gradient estimate. SGD step size was set to be 1. A neural network model decay ratio of 0.995 was considered. The results after testing the trained ARS model is presented in Fig. 7 - 10. Note that the yellow vertical line indicates the trigger time of ARS agent for control actions.

More specifically, Fig. 7 shows the damped oscillation of rotor angle due to the ARS agent, while Fig. 8 shows a well damped rotor speed. Moreover, it is observed that the continuously controlled mechanical power P_m in Fig. 9 was adjusted by ARS based policy model such that stability is achieved only after it has deviated from its pre-fault value. Similarly, the electrical power was also deviated from the pre-fault values as shown in Fig. 10. Due to this deviation, final system state δ is settling at a different value than its pre-fault value. This shows that although traditional ARS damps the oscillation, the trained model was not sufficient to bring the system back to pre-fault operating condition.

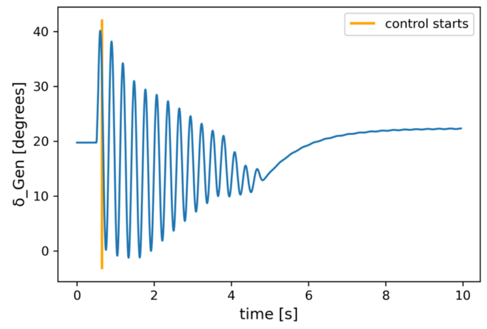

Figure 7. The simulated rotor angle with damped oscillation when traditional ARS was applied.

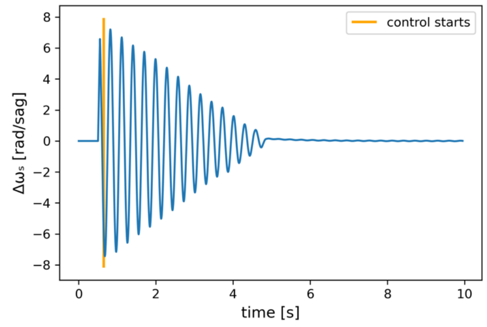

Figure 8. The simulated rotor speed with damped oscillation when traditional ARS was applied.

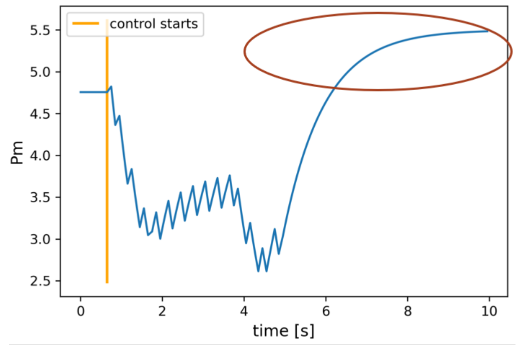

Figure 9. The simulated mechanical power when traditional ARS was applied.

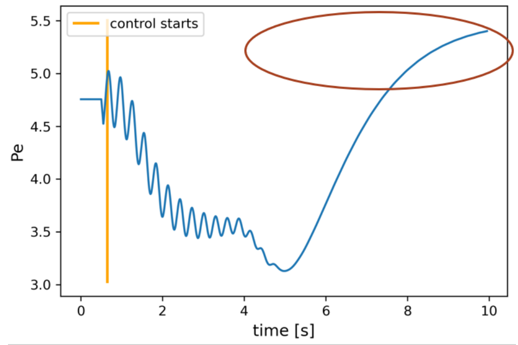

Figure 10. The simulated electrical power with damped oscillation when traditional ARS was applied.

5.2. With PINN in ARS

PINN-based ARS utilizes Swing Equation to influence the loss function, the results are presented in Fig. 11-14. Fig. 11 and 12 represents the oscillation in the observation states after the fault has been cleared. Both δ and ω are returning to the pre-fault operating condition due to the control action from PINN based ARS policy. Similarly, the mechanical power and electrical power outputs are also returning to the pre-fault values thus the controller brings back the pre-fault operating condition after the disturbance.

5.3. Performance Evaluation

The performance of both traditional ARS and PINN-based ARS were compared regarding the reward at the ending iteration, as well as the training cost to achieve those results. Fig. 15 shows the average reward obtained at the end of each iteration with PINN-based ARS stays higher after the 60-th iteration than traditional ARS method. The statistics of different simulations have also been presented in Table 1, which presents maximum rewards obtained at the end of 300 iterations, the number of iterations to reach stable reward and total training time of 300 iterations. As can be seen from the table, comparing PINN-based ARS to traditional ARS, maximum reward obtained is higher, the number of iterations to reach stable reward is 2.7 times faster, and the total time to complete all 300 iterations is lesser. This shows effectiveness of the PINN-based ARS in achieving desired result more efficiently.

Table 1. Performance comparison between noPINN and PINN implementations

	Max Reward	Iteration No. when stable reward > 0	300 ite. time cost (minute)
RL, noPINN	0.59	187	33.4
RL, w/ PINN	2.95	69	28.9

5.4. Influence of Seed

To properly capture the influence of random seed and further validate the benefits of the proposed PINN-based ARS, three groups of experiments with random seeds were run for 100 training iterations, and average rewards were collected and are shown in Fig. 16 for both the PINN-based ARS and traditional ARS. It can be seen that the average reward over those three groups is still higher for PINN-based ARS than the traditional ARS

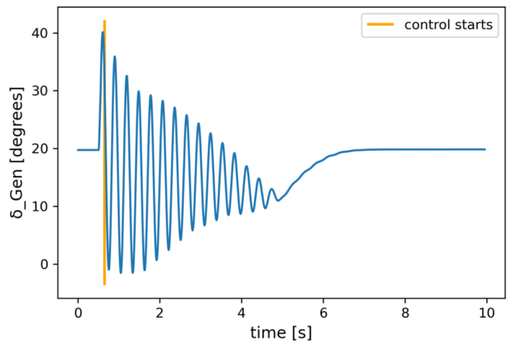

Figure 11. The simulated rotor angle with damped oscillation when PINN ARS was applied.

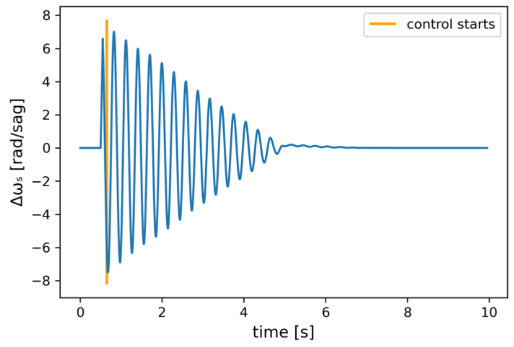

Figure 12. The simulated rotor speed with damped oscillation when PINN ARS was applied.

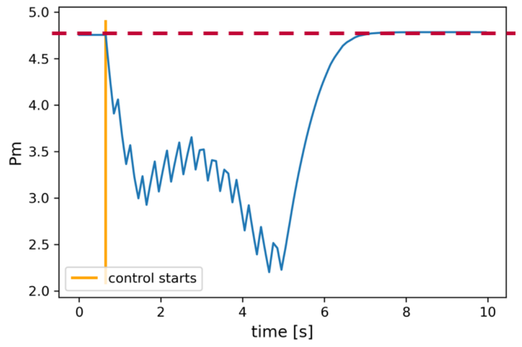

Figure 13. The simulated mechanical power when PINN ARS was applied.

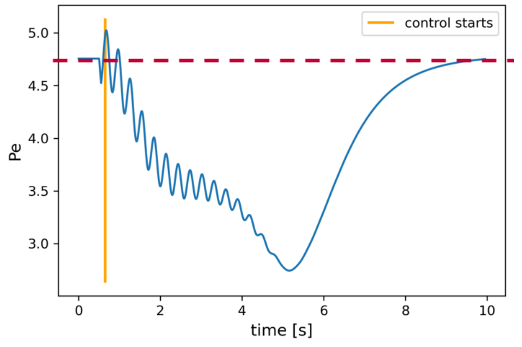

Figure 14. The simulated electrical power with damped oscillation when PINN ARS was applied.

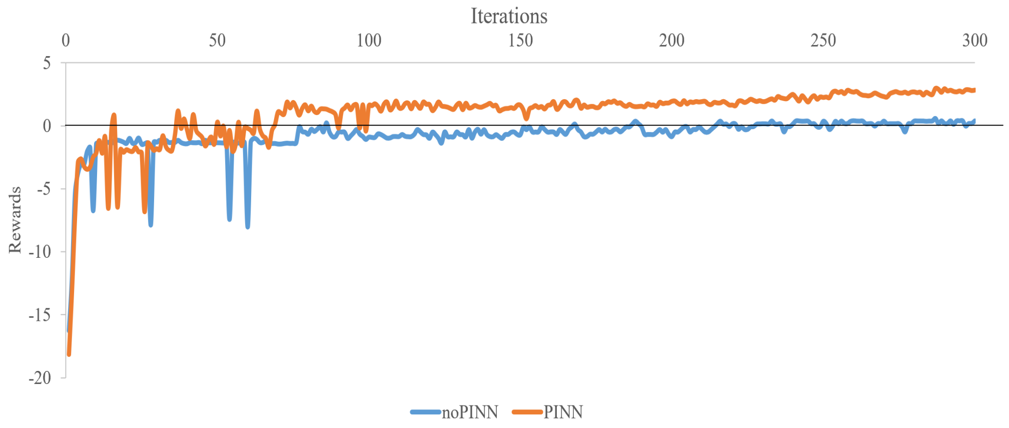

Figure 15. Rewards comparison between traditional ARS (noPINN) and PINN ARS.

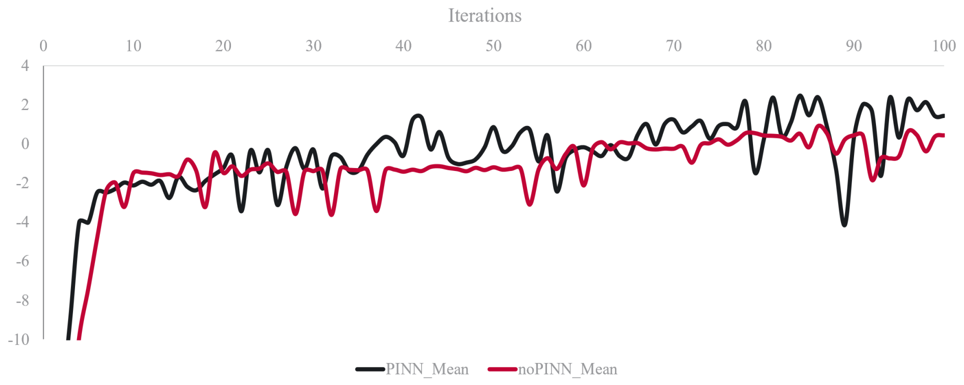

Figure 16. Influence of Seed when comparing the average reward of traditional ARS (noPINN) and PINN ARS with three random seeds.

method; therefore, it demonstrates the effectiveness of PINN-based ARS when applied in power system control.

6. Conclusion

By taking advantages of grid model physical priors and that of a model-free, derivative-free algorithm, we proposed and implemented a physics informed reinforced learning framework with ARS agent, which demonstrated superior performance comparing to traditional noPINN method. A synthesized two-bus system has been adopted for the simulation and validation of post-fault generator damping control. It shows 2.7 times faster in convergence speed, and the results are consistent in training sessions started with different random seeds. Potential application of the proposed PINN-based ARS can be further explored for larger power systems, and to achieve faster convergence without increasing the size of the policy network; more importantly, it can navigate the system back to pre-disturbance equilibrium once the fault was cleared.

For next step, the proposed methodology may be extended and evaluated in larger network model, such as IEEE 39-bus network [15]. On the other hand, more complex environments with different levels of difficulty can be evaluated, such as including additional physics equations for the controllers of generators and varying loads.

7. Acknowledgement

This work was funded by the Physics Informed Machine Learning (PIML) investment at the Pacific Northwest National Laboratory (PNNL), more information can be found at the webpage `https://www.pnnl.gov/projects/physics-informed-machine-learning-energy-and-environment` . The authors would also like to acknowledge PNNL colleagues George G. Muntean and Vikas Chandan for their support, comments in this project.

References

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

[2] J. Stiasny, G. S. Misyris, and S. Chatzivasileiadis, “Physics-informed neural networks for non-linear system identification applied to power system dynamics,” 2020.

[3] J. Stiasny, G. S. Misyris, and S. Chatzivasileiadis, “Transient stability analysis with physics-informed neural networks,” *arXiv preprint arXiv:2106.13638*, 2021.

[4] J. Stiasny, S. Chevalier, and S. Chatzivasileiadis, “Learning without data: Physics-informed neural networks for fast time-domain simulation,” *arXiv preprint arXiv:2106.15987*, 2021.

[5] J. Ostrometzky, K. Berestizshevsky, A. Bernstein, and G. Zussman, “Physics-informed deep neural network method for limited observability state estimation,” *arXiv preprint arXiv:1910.06401*, 2019.

[6] R. Nellikkath and S. Chatzivasileiadis, “Physics-informed neural networks for minimising worst-case violations in dc optimal power flow,” *arXiv preprint arXiv:2107.00465*, 2021.

[7] G. S. Misyris, J. Stiasny, and S. Chatzivasileiadis, “Capturing power system dynamics by physics-informed neural networks and optimization,” *arXiv preprint arXiv:2103.17004*, 2021.

[8] A. S. Zamzam and N. D. Sidiropoulos, “Physics-aware neural networks for distribution system state estimation,” *IEEE Transactions on Power Systems*, vol. 35, no. 6, pp. 4347–4356, 2020.

[9] L. Pagnier and M. Chertkov, “Physics-informed graphical neural network for parameter & state estimations in power systems,” *arXiv preprint arXiv:2102.06349*, 2021.

[10] W. Wang and N. Yu, “Estimate three-phase distribution line parameters with physics-informed graphical learning method,” 2021.

[11] A. Marot, B. Donnot, G. Dulac-Arnold, A. Kelly, A. O’Sullivan, J. Viebahn, M. Awad, I. Guyon, P. Panciatici, and C. Romero, “Learning to run a power network challenge: a retrospective analysis,” *arXiv preprint arXiv:2103.03104*, 2021.

[12] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, “Mastering the game of go without human knowledge,” *Nature*, no. 550, pp. 354–359, 2017.

[13] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, “Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation,” 2018.

[14] A. Sallab, M. Abdou, E. Perot, and S. Yogamani, “Deep reinforcement learning framework for autonomous driving,” *Electronic Imaging*, vol. 2017, p. 70–76, Jan 2017.

[15] Q. Huang, R. Huang, W. Hao, J. Tan, R. Fan, and Z. Huang, “Adaptive power system emergency control using deep reinforcement learning,” *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1171–1182, 2020.

[16] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[17] M. Glavic, R. Fonteneau, and D. Ernst, “Reinforcement learning for electric power system decision and control: Past considerations and perspectives,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6918–6927, 2017. 20th IFAC World Congress.

[18] A. D. Flaxman, A. T. Kalai, and H. B. McMahan, “Online convex optimization in the bandit setting: gradient descent without a gradient,” *arXiv preprint cs/0408007*, 2004.

[19] H. Mania, A. Guy, and B. Recht, “Simple random search provides a competitive approach to reinforcement learning,” *arXiv preprint arXiv:1803.07055*, 2018.

[20] H. Mania, A. Guy, and B. Recht, “Simple random search of static linear policies is competitive for reinforcement learning,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 1805–1814, 2018.

[21] R. Huang, Y. Chen, T. Yin, X. Li, A. Li, J. Tan, W. Yu, Y. Liu, and Q. Huang, “Accelerated deep reinforcement learning based load shedding for emergency voltage control,” *arXiv preprint arXiv:2006.12667*, 2020.

[22] P. Kundur, “Power system stability,” *Power system stability and control*, pp. 7–1, 2007.

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