A Verification and Validation Framework for Physics-Informed Machine Learning for Power System Dynamics

Petros Ellinas*†, Indrajit Chaudhuri†, Johanna Vorwerk, Spyros Chatzivasileiadis
Department of Wind and Energy Systems, Technical University of Denmark (DTU), Kgs. Lyngby, Denmark
†These authors contributed equally. *Corresponding author: petrel@dtu.dk

***Abstract*—Trusting Physics-informed Machine Learning Models (PIML) of power system components is a strong prerequisite for the wide deployment of PIML models and the drastic acceleration of power-system dynamic simulations. This paper presents a unified framework for certifying PIML surrogates against conventional solvers. We introduce an adjoint-based differentiable solver interface to efficiently optimize worst-case deviations, coupled with an active sampling strategy that targets the most informative initial conditions and disturbances. We further introduce and extend conformal prediction to a more robust variant, introducing an Upper Confidence Bound formulation, which ensures finite-sample coverage. Benchmarks on 2nd-, 4th- and 6th-order synchronous-machine models demonstrate that our methods reliably bound functional and solution errors while identifying critical operating scenarios. The proposed toolkit integrates seamlessly with existing PIML training pipelines, providing practitioners with practical guarantees for deploying learned operators in real-time and large-scale power-system studies.**

***Index Terms*—Active sampling, Adjoint methods, Conformal prediction, Error bounds, Physics-informed neural networks**

I. Introduction

Modern power systems have grown in size and complexity, incorporating renewable energy sources, power electronic converters, and flexible loads. Besides dramatically increasing computational challenges, such as increasing system stiffness, high-dimensionality, and the need for fine time resolution, these developments result in operational challenges, including new rapid transients, nonlinear interactions, and uncertainty in system behavior. As a result, conventional time-domain simulation tools struggle to provide accurate and efficient simulation results in an acceptable time frame [1], [2]. Numerical integrators, such as implicit Runge–Kutta and trapezoidal methods, solve the differential-algebraic equations that describe the power system's dynamic behavior by iterating on small time steps to ensure stability and accuracy. However, this approach leads to extensive runtimes for real-time and large-scale studies [1]. Even with parallelization and adaptive step-size control, the stiffness and multiscale nature of power-system dynamics can force step sizes into the microsecond range, rendering contingency assessment and hardware-in-the-loop tests computationally slow [3]. Thus, there is a pressing need for alternative solvers that can maintain accuracy while increasing computational speed [4].

Physics-Informed Machine Learning (PIML) has emerged as a compelling surrogate approach, embedding the residual of the governing Ordinary Differential Equations (ODEs) into neural-network training to approximate time-stepping operators without traditional discretization [5], [6]. Since [5], PIML models have been successfully applied across fluid dynamics, electro-magnetics, and recently to power system models, demonstrating speed-ups [5], [3]. Extensions, such as toolkits like PINNSim, operationalize PIML models for multi-component grid simulations [3].

However, two critical gaps must be addressed before PIML models can be reliably adopted in operational power-system analysis: rigorous verification of the learned operators and robust validation against real-world dynamics [7], [8]. Verification assesses whether a PIML model correctly represents the physical equations it is intended to approximate, through residual analysis or comparison with benchmark solutions. Validation, in contrast, evaluates the model's agreement with high-fidelity simulations or real measurements, ensuring reliable performance under realistic operating conditions. In this work, we propose a unified framework that systematically distinguishes and integrates both aspects: verification treats the mathematical model as the ground truth, while validation uses empirical data from real components as the reference. We further develop practical methods for each and define criteria for their consistent application in downstream tasks such as time-domain simulation.

To address the verification challenge, we follow two complementary approaches. First, we classify the sources of causality in physics-informed machine learning (PIML) models and decompose the resulting errors in a systematic and deterministic manner. Second, we develop a fully differentiable solver interface that represents high-fidelity time-domain integrators as continuous functions. This enables efficient gradient computation using the continuous adjoint method. By embedding this interface in a gradient-based optimization loop, we can directly search for the worst-case deviation between a learned PIML model and its ground truth. We further combine this with an active, trajectory-aware sampling strategy that selects initial conditions and disturbances adding the most new dynamical behavior, thereby focusing computational effort where it improves error estimation the most.

This work is supported by the European Research Council (ERC) Starting Grant VeriPhIED, Grant Agreement No. 949899.

On the validation side, we also employ two complementary techniques. For validation, we propose a probabilistic framework based on conformal Prediction (CP) to quantify uncertainty in PIML outputs and evaluate their reliability. Classical split conformal prediction ensures that the model's predictions include the true value at the desired rate on average, while the UCB extension strengthens this by ensuring—with high confidence—that this coverage level will hold even for a single dataset. We employ this UCB–CP framework in a playback validation procedure that compares predicted and recorded current trajectories, offering a rigorous and data-driven measure of model fidelity under realistic operating conditions. In addition, we propose the use of a zeroth-order black-box optimization method to identify the most informative disturbance scenarios for real-time, closed-loop testing. This approach extends validation beyond simulation, enabling experiments with physical components or general models without requiring differentiability of the underlying dynamics.

The main contributions of the presented work include:

1) **Verification toolkit:** An open-source toolkit that combines adjoint-based differentiable solvers, active trajectory sampling, and a black-box zeroth-order optimization algorithm to compute high-confidence error certificates for PIML models. The toolbox can be found on GitHub: https://github.com/elpetros99/PINNProof.
2) **Distribution-free validation:** We introduce Upper Confidence Bound Conformal Prediction (UCB-CP), the first conformal prediction method applied to power-system surrogates that delivers finite-sample, distribution-free uncertainty intervals and strengthens guarantees from average coverage to high-probability coverage.
3) **Systematic error analysis:** A thorough breakdown, description, and evaluation of 2nd-, 4th-, and 6th-order synchronous-machine models, comparing functional vs. solution errors and diagnosing interfacing-variable deviations using the proposed tool.

The remainder of this paper is organized as follows: Section II formulates the physics-informed time-stepping operator and contrasts its functional residual error with the classical ODE solver global error. Section III presents our deterministic and probabilistic error decompositions, including the adjoint verification method and active sampling for worst-case search. Furthermore, Section III-B details the adjoint-based solver interface and the active trajectory sampling technique. Section III-C details the conformal prediction framework and the proposed black-box optimization algorithm for validation. Section IV benchmarks all methods on 2nd-, 4th-, and 6th-order synchronous-machine models, while analyzing the errors on the state variables and the interfacing errors. Finally, Section V concludes and outlines directions for future work.

II. Physics-Informed Machine Learning for Power Systems Dynamics

In this section, we introduce the formulation of PIML models. Consider a system of ODEs written as:

$$\frac{d\mathbf{u}(t)}{dt} = f\big(\mathbf{u}(t), \mathbf{c}(t), t\big), \qquad \mathbf{u}(t_0) = \mathbf{u}_0, \tag{1}$$

where $\mathbf{u}(t) \in \mathbb{R}^n$ denotes the differential states and $\mathbf{c}(t) \in \mathbb{R}^m$ represents prescribed time-dependent inputs e.g. voltages or currents. The system dynamics are defined by $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$, and the vector $\mathbf{u}_0$ denotes the initial state at time t_0.

To approximate the solution operator $\big(\mathbf{u}_0, \mathbf{c}(\cdot), t_0, \xi\big) \mapsto \mathbf{u}(t_0+\xi)$ without explicit time-stepping, we introduce a neural network U whose inputs are the initial state $\mathbf{u}_0$, a set of discrete time samples of the input function $\mathbf{c}(\cdot)$, the start time t_0, and a time step ξ. For simplicity, we assume that $t_0 = 0$. The NN output is written as:

$$U\big(\mathbf{u}_0, \mathbf{c}(\cdot), t_0 = 0, \xi\big) \approx \mathbf{u}(t_0 + \xi) = \mathbf{u}(\xi). \tag{2}$$

Rather than relying only on labeled data, e.g., solution trajectories, PIML also imposes the ODE residual and the initial-condition constraint as soft penalties during training. Specifically, we sample the ODE residual at N collocation points ξ_i and M initial states $\mathbf{u}_{0,j}$, and minimize the training loss:

$$\begin{aligned}\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\Big\|\frac{\partial U_i}{\partial \xi} - f\big(U_i, \mathbf{c}(\xi_i)\big)\Big\|^2 + \frac{1}{N}\sum_{i=1}^{N}\|U_i - u_i\|^2 \\ + \frac{1}{M}\sum_{j=1}^{M}\|U_j - \mathbf{u}_{0,j}\|^2.\end{aligned} \tag{3}$$

Let $U_i \equiv U(\mathbf{u}_{0,i}, \mathbf{c}_i, \xi_i)$ and $f\big(U_i, \mathbf{c}(\xi_i)\big) \equiv f(U_i, \mathbf{c}_i(\xi_i), \xi_i)$. The first loss term enforces that the partial derivative $\partial_\xi U$ matches f at predefined collocation points ξ_i, while the second term encourages U to match labeled data u_i for given initial conditions $\mathbf{u}_{0,i}$. The third term ensures that $U(\mathbf{u}_0, \mathbf{c}, 0) = \mathbf{u}_0$, i.e., the operator reproduces the initial condition at $\xi = 0$.

Therefore, instead of solving ODEs in the usual way, we now treat the problem differently. Our operator $U(\mathbf{u}_0, \mathbf{c}, \xi)$ does not just take time as input; it also depends on the *initial conditions* $\mathbf{u}_0$, the *external controls* $\mathbf{c}$ applied, and the *simulation end time* ξ we want to check. As a result, the problem is no longer just about how states evolve over time. Now, the learned mapping depends on multiple variables, including the starting state, which effectively makes the learning problem a partial differential equation, even though the training data come from ODE trajectories. This makes the choice of training data especially important: it must cover a wide range of initial conditions and system behaviors.

PIML models and traditional ODE solvers evaluate accuracy based on different principles. PIML models minimize a functional error by embedding the differential equation into their loss function. This means they measure how closely the NN's output satisfies the differential operator, i.e., whether expressions like $\frac{\partial U}{\partial \xi} - f(U, \mathbf{c}(\xi)) = 0$ hold at sampled points. A small residual, or low operator error, implies that the NN approximately satisfies the equation, but it does *not* guarantee that the predicted solution $u_\theta(t)$ is close to the true solution $u(t)$, particularly in sensitive or ill-posed problems.

In contrast, classical ODE solvers, such as Runge-Kutta methods, control the global error, defined as the difference between the true solution $u(t_n)$ and the numerical estimate u_n at each time step t_n. This error is bounded by the timestep size and the solver's order.

The key distinction is that classical solvers are solution-driven, directly minimizing error in the solution itself, while

PIML models are residual-driven, minimizing how well the predicted solution satisfies the equation. Reliable use of PIML models thus requires validating the solution error through comparisons, convergence studies, or uncertainty quantification.

III. Error Analysis

PIML error analysis identifies the largest deviation between two operators of different accuracy across all valid inputs. Formally, we define:

$$E(\mathcal{U}) = \max_{(\mathbf{u}_0,\mathbf{c}(t),t)\in\mathcal{U}} \left| G(\mathbf{u}_0,\mathbf{c}(t),t) - U(\mathbf{u}_0,\mathbf{c}(t),t) \right| \quad (4)$$

Here, G and U represent two different operators, such as high-fidelity simulators, numerical solvers, experimental datasets, or ML models. The set $\mathcal{U}$ includes all admissible initial states $\mathbf{u}_0$, control trajectories $\mathbf{c}(t)$, and times t.

In this paper, we cast verification and validation as a maximization problem, systematically exploring initial conditions, disturbances, and control inputs to identify the worst-case scenario that yields the largest error. The main challenges arise from the heterogeneity of U and G, which may correspond to distinct numerical solvers or physical components, making optimization over them difficult. Along with the system's high dimensionality, nonlinearity, and nonconvexity, this complicates the search for the true worst-case scenario.

A. Error Decomposition

When validating or verifying PIML models of power system components or entire networks, it is crucial to interpret the total error through both deterministic and probabilistic lenses. If a closed-form ground truth is available, meaning the exact system behavior can be described analytically or computed using a high-fidelity numerical model, a deterministic error decomposition helps identify the main sources of discrepancy between the learned and true solutions. This decomposition helps identify which stage of the learning process, namely model design, data quality, or optimization, contributes most to the overall error and guides model refinement and uncertainty analysis. In this view, the total error consists of four main components: approximation error, training error, optimization error, and model error, as shown in Fig.1. Here, the approximation error quantifies the gap between the best neural network within the chosen architecture and the true solution, as governed by universal approximation theory and the selected hyperparameters. The training error arises when fitting that architecture to a finite dataset, and can be reduced by strategically choosing the number and placement of collocation points. The optimization error reflects the challenges of minimizing a highly non-convex PIML loss function. While optimizers like Adam or BFGS often reach acceptable minima, there is no guarantee of finding the global optimum. Together, approximation and training errors form the generalization error, which measures how close the network's learned solution is, to the exact PDE solution in the ideal scenario of perfect optimization and model correctness. Finally, the model error reflects physical misrepresentations, simplifications, or inherent stochasticity within the system under study.

When only real or simulated data are available and no exact model is known, particularly in the presence of measurement

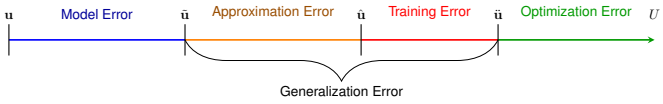


Fig. 1: Schematic representation of the total PINN-approximation discrepancy decomposed into its four main contributions.

noise or prediction uncertainty, a probabilistic framework is more appropriate. In this paradigm, the total error sources are reinterpreted as uncertainty contributions: reducible epistemic uncertainty arises from limited data, incomplete physical knowledge, finite network capacity, and imperfect optimization, whereas irreducible aleatoric uncertainty stems from inherent sensor noise or stochastic behavior in the physical process.

Bayesian methods (e.g., Bayesian neural networks, variational inference) model epistemic uncertainty by updating parameter priors into posteriors via Bayes' theorem. The resulting *posterior predictive distribution* directly captures *epistemic uncertainty*: where data are abundant, the posterior concentrates (narrow variance), indicating high confidence. Where data are sparse, the posterior remains broad, indicating higher uncertainty about the model. From this posterior we derive *credible intervals* which quantify the probability, given our data D and prior beliefs, that the true parameter or output lies within a determined interval. These methods have been used for uncertainty quantification for PIML models in the literature, [9], [10]. However, these techniques require some model assumptions, as they rely on explicit prior and likelihood assumptions tied to a particular model structure, which limits their generality.

In contrast, frequentist, distribution-free methods such as conformal prediction may be applied post hoc to any trained model, e.g. a PINN, ensemble predictor, or classical surrogate, to produce *prediction intervals*, or sets, that enjoy *finite-sample validity guarantees*, under suitable assumptions. The term "conformal" refers to the idea of conformity: the prediction set is chosen so that new data points are not unusually different from, i.e., they conform to, the patterns observed in the training data. By the term "validity guarantees" or "calibration", we mean that, assuming *exchangeability*, i.e., independent and identically distributed draws of (x, y), the constructed interval will—on average over new, unseen observations—contain the true outcome at least at a user-specified fraction of the time (e.g., 95%). Thus, these prediction bands provide *marginal* coverage guarantees without requiring strong assumptions about the noise distribution and the model used. Although conformal intervals do not by themselves decompose uncertainty into epistemic versus aleatoric components, their transparency, robustness, and auditability make them especially valuable in engineering and safety-critical domains, where one needs operational guarantees that the nominal coverage holds empirically under realistic conditions.

In the following, we bound total error against the solver's solution using deterministic methods (since comparator equations are available). For validation, where real-world equations are unknown, we adopt probabilistic methods, specif-

ically frequentist approaches, which are model-agnostic and assumption-free [11], [12].

B. Verification

When verifying one model against another, such as our PIML model versus a high-fidelity reference, we frame the error in purely deterministic terms, since both systems are fully specified. Generally, the ODE solution acquired from an ODE solver, through the lens of the previously introduced error decomposition, can be represented by $\hat{u}$ in Fig.1. However, in this section, we consider the solution produced by a high-accuracy conventional ODE solver as our "ground truth" $\tilde{\mathbf{u}}$, relying on its ability to approximate the true trajectory arbitrarily closely, since there is no mathematical way to acquire the closed-form true solution of the dynamical system. Under this assumption, this section first shows how to efficiently differentiate through the solver to tackle the verification optimization in (4). Then we introduce a method for constructing an informative, sample-based dataset, built on the same differentiability principles, to enable fast, targeted verification of individual components.

1) Adjoint Method for Differentiable Solvers: We assume that U is a PIML model, making it inherently differentiable with respect to its parameters and inputs. However, the ODE solver G, which evolves an initial state $\mathbf{u}_0$ under a time-varying control $\mathbf{c}(t)$ up to time T, is not necessarily efficiently differentiable. Given a scalar loss $\mathcal{L}(\mathbf{u}(T))$, such as in (4), we require its gradients with respect to $\mathbf{u}_0$ and $\mathbf{c}(t)$ to perform optimization. Instead of storing all intermediate states and backpropagating through every solver step, an approach that is both memory-intensive and numerically unstable, we use the *adjoint sensitivity method*, which computes these gradients by integrating an auxiliary adjoint ODE backward in time. This makes G effectively differentiable with constant memory cost, enabling efficient gradient-based verification of dynamic PIML models.

In contrast, the adjoint method avoids storing the full trajectory by introducing the adjoint variable:

$$\mathbf{a}(t) \;=\; \frac{d\mathcal{L}}{d\mathbf{u}(t)} \tag{5}$$

which quantifies how a perturbation in the state at time t affects the final loss. Differentiating the loss at T yields the terminal condition:

$$\mathbf{a}(T) \;=\; \frac{\partial \mathcal{L}}{\partial \mathbf{u}(T)}. \tag{6}$$

Using the chain rule, we obtain:

$$\frac{d\mathcal{L}}{d\mathbf{u}(t)} = \frac{d\mathcal{L}}{d\mathbf{u}(T)} \cdot \frac{d\mathbf{u}(T)}{d\mathbf{u}(t)} = \mathbf{a}(T) \cdot \frac{d\mathbf{u}(T)}{d\mathbf{u}(t)}. \tag{7}$$

To compute how $\mathbf{u}(T)$ changes with respect to a perturbation in $\mathbf{u}(t)$, we use the variational equation. If we perturb the state slightly, the variation $\delta\mathbf{u}(t)$ evolves under the linearized dynamics of (1):

$$\frac{d}{dt}\delta\mathbf{u}(t) = \frac{\partial f}{\partial \mathbf{u}}(t)\,\delta\mathbf{u}(t). \tag{8}$$

To derive the dynamics of the adjoint variable $\mathbf{a}(t)$, we differentiate its definition in time. Since $\mathbf{u}(T)$ is a function of earlier states through the forward ODE, we differentiate this expression using the variational equation:

$$\frac{d}{dt}\left(\frac{d\mathbf{u}(T)}{d\mathbf{u}(t)}\right) = -\frac{\partial f}{\partial \mathbf{u}}(t) \cdot \frac{d\mathbf{u}(T)}{d\mathbf{u}(t)}. \tag{9}$$

Substituting into the time derivative of $\mathbf{a}(t)$, and assuming the time derivative of $\frac{d\mathcal{L}}{d\mathbf{u}(T)}$ is zero, we obtain:

$$\frac{d\mathbf{a}}{dt} = -\left(\frac{\partial f}{\partial \mathbf{u}}\right)^{\top} \mathbf{a}(t), \tag{10}$$

which is the backward-in-time ODE governing the evolution of the adjoint state.

We integrate this from $t = T$ down to $t = 0$, requiring only the ability to evaluate $\partial f/\partial \mathbf{u}$ (and $\partial f/\partial \mathbf{c}$ if controls are to be differentiated) at the current state, without ever storing all past states.

Once the backward integration is complete, the gradient with respect to the initial state is obtained directly as $\frac{d\mathcal{L}}{d\mathbf{u}_0} = \mathbf{a}(0)$, while the gradient with respect to the control at each time t is given by $\frac{d\mathcal{L}}{d\mathbf{c}(t)} = \left(\frac{\partial f}{\partial \mathbf{c}}\right)^{\top} \mathbf{a}(t)$.

In this way, we treat the solvers G, U and their difference as a smooth function $\mathbf{u}_0, \mathbf{c}(\cdot) \mapsto \mathbf{u}(T)$ and perform reverse-mode differentiation with only two ODE solves, one forward for $\mathbf{u}$ and one backward for $\mathbf{a}$, rather than storing a long chain of intermediate Jacobians. This gives us the opportunity to directly solve the optimization (4), with any gradient-based solver such as Adam, L-BFGS, etc.

2) Sample-Based Methods: Practitioners often favor sample-based testing because it allows them to hand-pick realistic scenarios, directly observe where the model errs, and immediately discard any test cases that are not physically plausible. This hands-on approach not only highlights regions of strong and weak performance but also guarantees that every evaluation remains grounded in real-world conditions, enhancing both interpretability and trust. However, due to the curse of dimensionality, they cannot sample the space naively and search for the regions with the largest error. Therefore, they must have ways to set up an informative dataset. In order to generate representative solution trajectories of an ODE system, one must carefully select a set of initial conditions from which to integrate. The quality and diversity of these trajectories depend critically on how the initial conditions are sampled. Given a dynamical system governed by (1), we aim to construct a dataset by evaluating the flow map $U(\mathbf{u}_0, \mathbf{c}(t), t)$ for a set of initial conditions and external, discretized input functions $\{\mathbf{u}_0^{(i)}, \mathbf{c}(t)^{(i)}\}_{i=1}^{N}$. This paper investigates a new trajectory-based active sampling method, aiming to optimize dynamic diversity.

To address the limitations of naive input-space sampling, we introduce an active sampling strategy that selects initial conditions based on the novelty of their associated trajectories in the output space. The method is iterative and builds a dataset of trajectories $\mathcal{S}_k = \{\mathbf{u}^{(i)}(t)\}_{i=1}^{k}$ by sequentially selecting new initial conditions that yield solutions most dissimilar from those already observed.

Given the current set of trajectories, the $(k+1)$-th initial condition is obtained by solving the following optimization problem:

$$\mathbf{u}_0^{(k+1)} = \arg\max_{\mathbf{u}_0 \in \mathcal{U}_0} \mathcal{J}(\mathbf{u}_0), \tag{11}$$

where the objective function $\mathcal{J}(\mathbf{u}_0)$ quantifies the novelty of the trajectory originating from $\mathbf{u}_0$. We define this objective function using a smooth approximation of the minimum squared distance between trajectories:

$$\mathcal{J}(\mathbf{u}_0) = -\frac{1}{\alpha} \log \sum_{i=1}^{k} e^{-\alpha \left\| \mathbf{u}(\mathbf{u}_0^{(i)}, \mathbf{c}(t)^{(i)}) - \tilde{\mathbf{u}}^{(i)}(t) \right\|_2^2}, \tag{12}$$

where $\alpha > 0$ is a sharpness parameter that controls the softness of the minimum. As $\alpha \to \infty$, this expression approximates the true minimum squared distance. This formulation encourages the selection of trajectories that are far from all previously observed ones, promoting dynamic diversity in the dataset.

The optimization is performed using gradient-based methods, leveraging the differentiability of the ODE solver. Specifically, for each candidate $\mathbf{u}_0$, the trajectory $\mathbf{u}(t; \mathbf{u}_0)$ is computed using a neural ODE solver with backpropagation. To ensure the candidate remains within the admissible domain, the optimization enforces box constraints via projection or clamping at each iteration.

This approach is particularly valuable in systems where the dynamics exhibit bifurcations, multi-stability, or other localized phenomena that may be overlooked by non-adaptive sampling strategies. By focusing on trajectory space rather than just initial conditions, such active sampling facilitates efficient exploration of the solution manifold and reduces redundancy in the training data.

C. Validation

For model validation in power systems, we employ CP methods, starting with Split Conformal and extending to Upper UCB-CP, to provide statistically rigorous, distribution-free error bounds and reliability guarantees on model prediction errors. We also propose the Every Call is Precious (ECP) algorithm [13] for cases where the system is non-differentiable or gradient information is unavailable, such as in hardware-in-the-loop (HIL) [14], real-time digital simulation (RTDS) [15], and event playback [16]. ECP solves the verification problem in (4) by identifying worst-case disturbances that maximize discrepancies between a model and a reference without relying on gradients. Together, CP and ECP enable systematic, gradient-free validation of dynamic power system models across data-driven and experimental settings.

1) Conformal Prediction for Model Validation: Conformal prediction constructs intervals around the outputs of a trained model U by comparing each prediction to a held-out calibration dataset through a nonconformity score. These intervals are guaranteed to contain the true value with a specified probability, assuming the data samples are exchangeable, that is, drawn from the same distribution and invariant to reordering. Let $\{(x_i, y_i)\}_{i=1}^n$ denote the calibration set, where $f_\theta(x)$ is the trained PIML surrogate of a power-system component and $\hat{y}_i = f_\theta(x_i)$ its output. The nonconformity score measures the discrepancy between predictions and observations:

$$s_i = \frac{1}{\sigma(x_i)} |y_i - \hat{y}_i|, \tag{13}$$

where $\sigma(x) > 0$ is a scale function. If $\sigma(x) \equiv 1$, the score reduces to the absolute residual, leading to constant-width intervals.

Classical split conformal prediction guarantees that, under exchangeable data, the threshold q_α computed from a held-out calibration set ensures valid marginal coverage:

$$\Pr\{y \in C_\alpha(x)\} \geq 1 - \alpha. \tag{14}$$

The calibration set, disjoint from the training data, is used solely to evaluate prediction residuals and estimate q_α, preventing information leakage. The conformal interval is defined as

$$C_\alpha(x) = \left[\hat{y}(x) - q_\alpha\, \sigma(x),\ \hat{y}(x) + q_\alpha\, \sigma(x)\right], \tag{15}$$

where $\hat{y}(x)$ is the model prediction, $\sigma(x)$ is a scale function, and q_α is the $\lceil (n+1)(1-\alpha) \rceil$-th largest nonconformity score s_i among the n calibration samples. Intuitively, $C_\alpha(x)$ represents the range of outputs statistically consistent with both the model and the calibration data, containing the true response with a probability of at least $1 - \alpha$. This relies on exchangeability: the rank of a new score among calibration scores is uniformly distributed. Consequently, split conformal prediction provides finite-sample, distribution-free coverage guarantees without assumptions on the error distribution. In this work, we use split conformal prediction as a baseline for validating PIML models of component dynamics.

Classical split conformal prediction guarantees valid marginal coverage under exchangeable data, but this guarantee holds only on average across calibration draws. In safety- or cost-critical settings, even rare violations beyond the target level α can be unacceptable. To achieve stronger reliability, we employ a high-confidence calibration procedure that controls miscoverage with probability at least $1-\delta$ over the calibration sample set.

Let the standardized residuals on the calibration set be defined as in (13), and assign uniform weights $w_i = 1/n$. For a candidate threshold λ (analogous to q_α in classical conformal prediction), the empirical miscoverage rate is

$$\hat{p}(\lambda) = \sum_{i=1}^{n} w_i\, \mathbf{1}\{v_i > \lambda\}, \tag{16}$$

where $\mathbf{1}\{\cdot\}$ equals 1 if the condition inside the braces is true and 0 otherwise, marking whether the interval associated with λ fails to cover the corresponding calibration target.

An upper confidence bound (UCB) on the true miscoverage is given by

$$\mathrm{UCB}(\lambda) = \hat{p}(\lambda) + \sqrt{\tfrac{1}{2} \log \tfrac{1}{\delta} \sum_{i=1}^{n} w_i^2}, \tag{17}$$

and the final threshold is chosen as

$$\lambda^* = \inf\{\lambda : \mathrm{UCB}(\lambda) \leq \alpha\}. \tag{18}$$

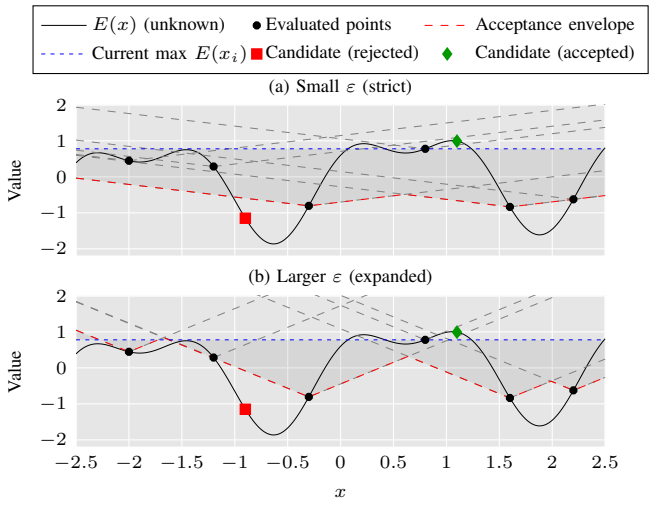


Fig. 2: ECP acceptance region for a 1D function. Each evaluated point $(x_i, E(x_i))$ forms a cone–shaped upper bound. Candidates below all cones are rejected; those outside at least one cone are accepted. Smaller ε_t yields wider acceptance, while larger ε_t focuses the search.

The resulting intervals $C_{\lambda^*}(x)$ satisfy

$$\Pr_{\text{calibration}}\big(R(\lambda^*) \le \alpha\big) \ge 1 - \delta, \tag{19}$$

where

$$R(\lambda^*) = \Pr_{\text{test}}(y \notin C_{\lambda^*}(x)) \tag{20}$$

is the true miscoverage risk on test data. This high-probability guarantee extends classical conformal prediction by ensuring the realized miscoverage rarely exceeds α.

2) Worst–Case Disturbance Search via Global Optimization: We aim to identify the worst–case input $x = (\mathbf{u}_0, \mathbf{c}(\cdot), t) \in \mathcal{U}$ that maximizes the error (4). This method is an extension of the Lipschitz method presented in [17]. However, the proposed algorithm ECP treats every function evaluation as costly, and therefore only evaluates candidates that *could plausibly improve the current maximum*.

a) Algorithmic principle: At iteration t, suppose we have already evaluated $\{(x_i, E(x_i))\}_{i=1}^{t}$. ECP proposes a new candidate $x' \sim \mathrm{Uniform}(\mathcal{U})$, but accepts it for evaluation only if it lies in the *acceptance region*:

$$\min_{1\le i\le t}\Big(E(x_i) + \varepsilon_t \,\|x' - x_i\|\Big) \;\ge\; \max_{1\le j\le t} E(x_j), \tag{21}$$

where $\|\cdot\|$ is the metric on $\mathcal{U}$, and $\varepsilon_t > 0$ is a tolerance parameter that expands over time.

b) Geometric view: Each past evaluation $(x_i, E(x_i))$ generates a Lipschitz-style upper bound

$$E(x) \;\le\; E(x_i) + \varepsilon_t \|x - x_i\|,$$

which can be visualized as a cone rooted at x_i (see Fig. 2). In one dimension, these cones appear as dashed lines forming a V-shaped envelope above each evaluated point. In higher dimensions, they become conical surfaces or circular pyramids enclosing the region that could still contain the worst-case error. If a new proposal x' lies *below all cones*, then even in the best case, its value cannot exceed the current maximum, and it is rejected. Conversely, if x' lies *outside at least one cone*, it remains a plausible maximizer and is accepted for evaluation. This geometric test ensures that expensive solver calls are concentrated on the subset of $\mathcal{U}$ that may still contain the maximizer. This mechanism acts as a spatial filter, concentrating evaluations on the portions of the input space where the error might still increase, thereby avoiding unnecessary computations in regions already deemed safe.

c) Adaptive relaxation: The strictness of this filter is governed by ε_t. The algorithm begins with a small ε_1, enforcing tight cones and aggressive rejection of unlikely points. Whenever too many consecutive proposals fail the acceptance test, or once an evaluation is made, ε_t is updated multiplicatively:

$$\varepsilon_{t+1} = \tau\,\varepsilon_t, \qquad \tau > 1. \tag{22}$$

This relaxation gradually widens the cones, enlarging the acceptance region until, in the limit, $\mathcal{A}_{\varepsilon_t} = \mathcal{U}$. Thus, the true maximizer $x^\star$ is guaranteed not to be permanently excluded. Despite never requiring knowledge of a Lipschitz constant, ECP achieves minimax–optimal rates of global optimization [13].

Beyond the Lipschitz setting, ECP can be extended by replacing the global constant with a semi-metric ℓ that models local smoothness near the optimum. The acceptance rule then becomes $\max_i f(X_i) \le \min_i\big(f(X_i)+\ell(X_{t+1}, X_i)\big)$, enabling ECP to operate under weaker regularity assumptions [13].

IV. Results

To demonstrate the proposed verification and validation methods and the insights they offer into PIML models' performance, we apply them to three Synchronous Machine (SM) models of increasing complexity. Verification uses adjoint-based sensitivity analysis with multiple optimizers and active sampling, while validation employs the UCB-CP and ECP frameworks to assess model accuracy and robustness.

A. Benchmark Models and Evaluation Setup

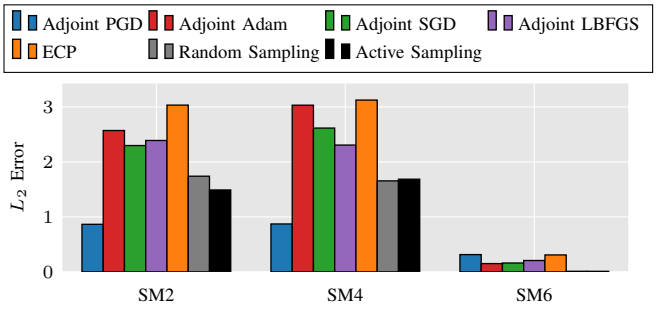
PINNs are trained for three SMs with different level of model complexity: the 2-state swing equation (SM2), the 4th-order model (SM4), and the 6th-order model (SM6). All model parameters, including inertia constants, damping ratios, etc., are taken from [18] and are reported in [19]. In addition, the NN training framework from [19] is used. All PINNs share the same architecture: three hidden layers with 64 neurons each. All employ the L-BFGS optimizer. Unless otherwise stated, training is performed to approximate the solution of each model over a time horizon of $t_{\max} = 0.2\,\mathrm{s}$. For the SM6 case with $t_{\max} = 0.2\,\mathrm{s}$, we used 600000 simulated and collocation points during training, compared with the 60000 points used in other cases, to achieve the highest possible accuracy for subsequent analyses.

For each PIML, we first solve the worst-case error optimization (4). For the SM6, we then apply weighted conformal prediction to bound the error on the injected currents and compare the resulting intervals against the true trajectories. Finally, we examine how the interface current and voltage errors evolve over time and across various operating conditions,

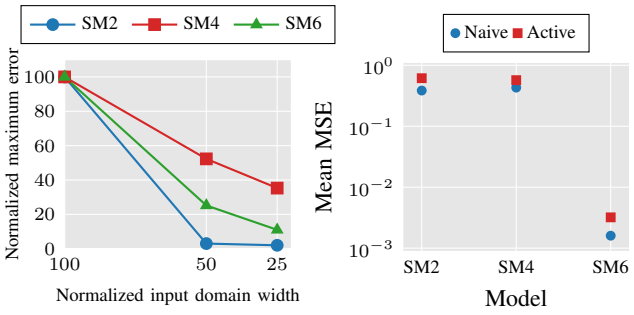
and we contrast the PIML model's residual-based functional error with its solution approximation error. In the remainder of this section, we refer to the d,q-axis stator currents in the synchronous reference frame as I_D and I_Q.

B. Methods Comparison

In this section, we evaluate the effectiveness of several optimization algorithms that will be used together with the proposed adjoint method to compute the worst-case error: Adjoint with (i) Projected Gradient Descent (PGD) [20], (ii) Stochastic Gradient Descent (SGD) [21], (iii) Adam [22], and (iv) L-BFGS [23], as well as the validation method, (v) ECP, to assess their overall performance.



(a) Comparison of GA, ECP, and batch sampling maximum L_2 errors.



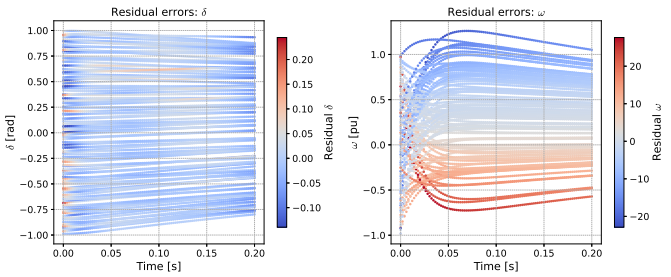
(b) Normalized maximum error of ECP as a function of domain width.

(c) Mean MSE across all state variables for Naive and Active Sampling approaches.

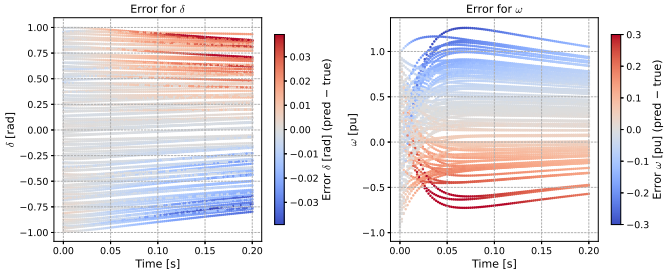
Fig. 3: Overview of error behaviors across different sampling strategies, domain widths, and method comparisons.

Fig.3a contrasts worst-case L_2 errors found by each of the proposed methods across the three SM models. For SM2 and SM4, ECP returns the largest maxima (3.03 and 3.12), with adjoint–Adam close behind, followed by adjoint–SGD/LBFGS, while adjoint–PGD yields the lowest value. For SM6, the absolute maxima are small, and the adjoint–PGD value slightly exceeds the ECP value. In all cases, random and active sampling report substantially smaller maxima, highlighting that sampling captures distributional behavior but underestimates adversarial extremes. Moreover, we notice that the gradient-free method consistently finds better maxima, indicating it navigates the complex loss landscape more effectively.

Fig.3b shows how the worst–case error identified by the ECP method evolves as the admissible range of the rotor angle δ is progressively reduced. Both axes are normalized and expressed as percentages. For example, the PINN error for SM2 is eliminated (i.e. achieving almost 100% accuracy) if we train the PINN for a certain range of angle δ but deploy it only in the middle 50% of that range. Especially for SM2 and SM6, the errors reduce superlinearly, which means that the worst-case errors are concentrated closer to the boundaries of the training domain. The chart in Fig. 3c compares the mean MSE obtained with naive, random sampling, and active sampling across the three benchmark SM models. The log scale on the y-axis highlights the large differences in error magnitudes between the SM6 case and the other models. For SM2 and SM4, both methods yield errors of similar order, with active sampling giving slightly larger mean MSE than naive sampling, reflecting the higher concentration of points in difficult regions. In the SM6 case, the absolute errors are generally lower overall, primarily because the models were trained with a larger number of labeled samples. Nevertheless, the same trend is observed: active sampling yields higher mean error values compared to random sampling. This suggests that the active sampling strategy successfully identified initial conditions and disturbances that trigger distinct dynamical system behaviors, thereby enriching the testing set with a greater variety of system responses. However, this focus on "difficult" or atypical scenarios also tends to inflate the average error observed. In contrast, naive sampling spreads the training points more evenly across the entire domain. This provides a more balanced overall performance estimate, but is less focused on reflecting the most challenging regions.



(a) Residual errors for δ [rad] and ω [pu].



(b) Solution errors for δ [rad] and ω [pu].

Fig. 4: Comparison of error for the rotor angle δ and speed deviation ω in the SM6. (a) Residual errors indicate the mismatch of the physics-informed equations. (b) Solution errors show the deviation of the predicted trajectories from ground truth.

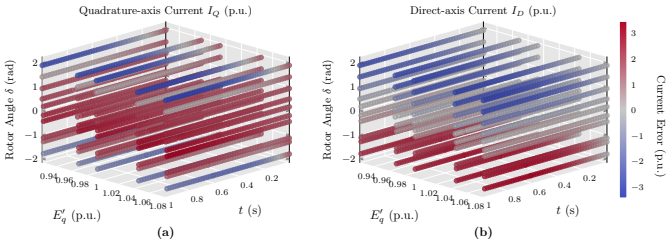


Fig. 5: Current error scatter plots. (a) Quadrature-axis current I_Q; (b) Direct-axis current I_D.

C. Comparison between Functional Error and Solution Error

This section compares two notions of error, described in Section II. The *functional error*, shown in Fig.4a, measures the residuals of the governing equations and reflects physics consistency, while the *solution error*, depicted in Fig.4b, quantifies the deviation of predicted trajectories from the solution of the ODE solver. Although correlated large residuals often align with large solution errors, the two are not equivalent: Small residuals do not guarantee accurate trajectories, and large residuals do not always indicate poor accuracy, as evident from the functional and solution errors of δ. Residual errors appear as bursts around transients, while solution errors accumulate over time, leading to visible drifts in δ and ω. Both error measures increase toward the edges of the time horizon, likely due to boundary effects and the accumulation of the residual error. In addition, the errors are larger in regions where the solution varies more rapidly (i.e., where gradients are higher). This indicates that the neural network struggles in regimes with high gradient variance, where residual errors accumulate more strongly. This limitation suggests that the current architecture cannot adequately capture rapidly changing dynamics, motivating the exploration of alternative architectures tailored for such variability.

D. Interfacing Error

Figure 5 presents the interfacing variable (current) error, for the quadrature- and direct-axis currents as 3D scatter plots over time t, internal EMF E'_q (p.u.), and rotor angle δ (rad), with point colors indicating the current error (p.u.). The q-axis current, depicted in Fig. 5(a), exhibits large deviations, and errors extend to the highest magnitudes, approaching the colorbar extrema. Errors concentrate during the late-time window and at larger $|\delta|$, where the dynamics are most nonlinear. Moreover, the I_Q errors maintain a largely consistent sign across broad regions of the trajectory, indicating a directional mismatch rather than mere random scatter. In contrast, the d-axis current in Fig. 5(b) remains comparatively bounded over most of the trajectory, with smaller dispersion in both magnitude and state-space coverage.

The interfacing error is mainly driven by the quadrature-axis channel during transients and large rotor-angle swings, where the approximation is least reliable. This underscores the need to refine the I_Q pathway under stressed conditions, while the I_D channel remains sufficiently accurate for nominal operation.

E. Conformal Prediction

This section performs conformal prediction of the interfacing error by comparing solver data with data generated by the PIML SM6 model. Since the current errors can be large, attention is restricted to the central region of the state space, where interfacing errors are smaller. Specifically, we select the subset with errors below 0.05 p.u. and run conformal prediction only in this region. This gives us control over the maximum error, enabling a more rigorous assessment of the method.

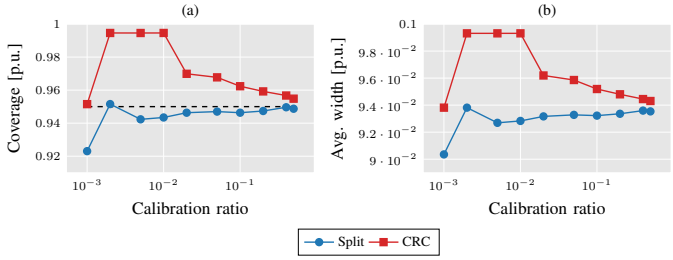


Fig. 6: Performance of Split and UCB-CP across calibration ratios: (a) empirical coverage relative to the target level 1=0.95 (dashed black line); (b) average prediction-interval width (p.u.).

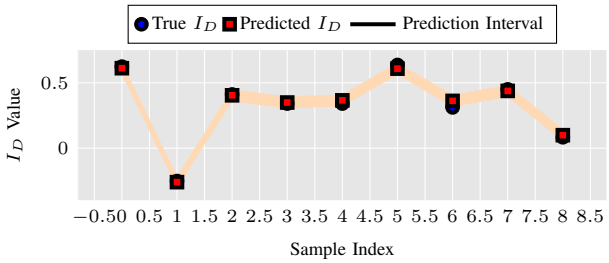


Fig. 7: True vs. predicted drain current I_D for 10 test samples. Blue circles denote ground truth, red squares model predictions, and the shaded band conformal intervals.

Figure 6 compares Split Conformal Prediction (Split) and the proposed UCB-CP method across calibration ratios. Panel (a) shows that Split under-covers when calibration data are scarce, with empirical coverage falling below the nominal target $1\alpha = 0.95$ for ratios below roughly 0.5%. In contrast, UCB-CP maintains coverage close to or slightly above the target even with very few calibration points, highlighting its strong finite-sample reliability and built-in risk control. As the calibration set grows, both methods converge toward the desired coverage, but UCB-CP remains more stable across all regimes.

Panel (b) presents the corresponding average interval widths. UCB-CP intervals are slightly wider—particularly in low-data regimes—reflecting their conservative calibration. However, this conservativeness diminishes with larger calibration ratios, where UCB-CP achieves near-Split efficiency while retaining full coverage guarantees. Overall, the figure illustrates a consistent trade-off between efficiency and reliability: Split produces narrower but occasionally under-confident intervals, whereas UCB-CP yields more reliable and

well-calibrated uncertainty bounds, ensuring valid coverage even under limited calibration data.

F. Results Discussion

This section analyzes the key characteristics of existing PIML models and their corresponding training procedures. We observed that training and testing errors stabilized at around 10^{-3} and 10^{-4}, respectively. This plateau indicates a complex, non-linear, and non-convex loss landscape resulting from governing equations and NN designs, offering opportunities for innovation. Recent research shows that tailoring optimizers specifically for PIML can help overcome these challenges [24].

Additionally, we found that capturing the high variance of gradients required for accurate operator learning remains a challenging task. This insight suggests promising directions for model design that leverage gradient information more effectively, thereby enhancing robustness for iterative evaluations. Moreover, the conducted experiments highlight the important role of data quality and coverage: strategies such as active trajectory-based sampling can provide richer collocation points, enabling the networks to learn more effectively.

Overall, these findings highlight the achieved progress, while also indicating several opportunities that lie ahead. With advances in optimization, sampling, and architecture design, PIML models are well-positioned to achieve higher accuracy and reliability, ultimately supporting their use in downstream processes such as time-domain simulations of power systems.

V. Conclusion

In this work, we introduced a unified framework for certifying PIML models for power system dynamics. Verification combines an adjoint-based differentiable solver with active trajectory sampling for efficient worst-case error search, while validation utilizes weighted conformal prediction to provide finite-sample, distribution-free uncertainty intervals for interface variables. Benchmarks on 2nd-, 4th-, and 6th-order SM models show that the approach reliably identifies critical scenarios, bounds residual and solution errors, and produces calibrated uncertainty bands. Future work will extend the framework to multi-machine networks with full DAE structure, develop solver-agnostic adjoint interfaces for hybrid differential–algebraic models, and explore hierarchical conformal calibration for large systems with varying noise levels. We also plan to integrate the toolkit into hardware-in-the-loop and real-time digital simulation platforms, and to design adaptive horizon selection strategies that tighten certificates in long-duration and contingency analyses.

During the preparation of this work, the author, Petros Ellinas, used ChatGPT (OpenAI) to improve readability and check grammar. After using this tool, the author reviewed and edited the content as needed and takes full responsibility for the content of the published article.

References

[1] J. Stiasny and S. Chatzivasileiadis, "Physics-informed neural networks for time-domain simulations: Accuracy, computational cost, and flexibility," *Electric Power Systems Research*, vol. 224, p. 109748, 2023.

[2] G. S. Misyris, A. Venzke, and S. Chatzivasileiadis, "A comprehensive analysis of pinns for power system transient stability," *Electronics*, vol. 13, no. 2, p. 391, 2023.

[3] J. Stiasny, B. Zhang, and S. Chatzivasileiadis, "Pinnsim: A simulator for power system dynamics based on physics-informed neural networks," https://arxiv.org/abs/2303.10256, 2023.

[4] B. Zhang, J. Stiasny, and S. Chatzivasileiadis, "Integrating physics-informed neural networks into power system simulation frameworks," https://arxiv.org/abs/2404.13325, 2024.

[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] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "fPINNs: Fractional physics-informed neural networks," *SIAM Journal on Scientific Computing*, vol. 41, no. 4, pp. A2603–A2626, 2020.

[7] S. Mishra and R. Molinaro, "Estimates on the generalization error of physics informed neural networks (pinns) for approximating pdes," *arXiv preprint arXiv:2006.16144*, 2020. [Online]. Available: https://arxiv.org/abs/2006.16144

[8] L. Podina, M. Torabi Rad, and M. Kohandel, "Conformalized physics-informed neural networks," in *ICLR 2024 Workshop on AI4DiffEqtnsInSci*, 2024.

[9] A. F. Psaros, X. Meng, Z. Zou, L. Guo, and G. E. Karniadakis, "Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons," *Journal of Computational Physics*, vol. 477, p. 111902, Mar. 2023. [Online]. Available: http://dx.doi.org/10.1016/j.jcp.2022.111902

[10] L. Yang, X. Meng, and G. E. Karniadakis, "B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data," *Journal of Computational Physics*, vol. 425, p. 109913, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999120306872

[11] V. Gopakumar, A. Gray, L. Zanisi, T. Nunn, D. Giles, M. J. Kusner, S. Pamela, and M. P. Deisenroth, "Calibrated physics-informed uncertainty quantification," 2025. [Online]. Available: https://arxiv.org/abs/2502.04406

[12] F. Eiras, A. Bibi, R. Bunel, K. D. Dvijotham, P. Torr, and M. P. Kumar, "Efficient error certification for physics-informed neural networks," 2024. [Online]. Available: https://arxiv.org/abs/2305.10157

[13] F. Fourati, S. Kharrat, V. Aggarwal, and M.-S. Alouini, "Every call is precious: Global optimization of black-box functions with unknown lipschitz constants," in *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 258. PMLR, 03–05 May 2025, pp. 5176–5184. [Online]. Available: https://proceedings.mlr.press/v258/fourati25a.html

[14] *IEEE Standard for Hardware–in–the–Loop Testing of Real–Time Electric Power Simulators*, IEEE Std. 1646-2004, 2004.

[15] R. Mackiewicz, W. Wu, and Z. Yang, "Real-time digital simulation of hvdc and facts controllers," *IEEE Transactions on Power Delivery*, vol. 30, no. 3, pp. 1121–1128, 2015.

[16] J. Lu, G. Joos, and J.-S. Thorp, "Application of event playback in power system disturbance analysis," *IEEE Transactions on Power Systems*, vol. 29, no. 1, pp. 143–151, 2014.

[17] P. Ellinas, I. Karampinis, I. Ventura Nadal, R. Nellikkath, J. Vorwerk, and S. Chatzivasileiadis, "Physics-informed machine learning for power system dynamics: A framework incorporating trustworthiness," *Sustainable Energy, Grids and Networks*, p. 101818, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352467725002000

[18] P. W. Sauer and M. A. Pai, *Power system dynamics and stability*, 1st ed. Upper Saddle River, N.J: Prentice Hall,, 1998.

[19] I. Karampinis, P. Ellinas, I. V. Nadal, R. Nellikkath, and S. Chatzivasileiadis, "Toolbox for developing physics informed neural networks for power systems components," 2025. [Online]. Available: https://arxiv.org/abs/2502.06412

[20] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017. [Online]. Available: https://arxiv.org/abs/1706.06083

[21] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, Sep. 1951. [Online]. Available: https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-22/issue-3/A-Stochastic-Approximation-Method/10.1214/aoms/1177729586.full

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015, arXiv:1412.6980. [Online]. Available: https://arxiv.org/abs/1412.6980

[23] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*,

vol. 45, no. 3, pp. 503–528, 1989. [Online]. Available: https://link.springer.com/article/10.1007/BF01589116

[24] E. Kiyani, K. Shukla, J. F. Urbán, J. Darbon, and G. E. Karniadakis, "Optimizing the optimizer for physics-informed neural networks and kolmogorov-arnold networks," 2025. [Online]. Available: https://arxiv.org/abs/2501.16371